International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Hand Gesture Recognizer

Alok Singh Kushwaha[1], Reddypally Saikomal[2], Swathi Kumari[3], Sunidhi Singh[4], Agrawal Prathmesh Kishor[5]

Department of Computer Science and Engineering, Parul Institute of Engineering and Technology (PIET), Vadodara, Gujarat, India

Abstract:

The rise of digital interfaces has highlighted the limitations of traditional input devices, such as the keyboard and mouse, which can hinder accessibility and efficiency. This paper discusses the design, implementation, and evaluation of a new human-computer interaction (HCI) system that merges two complementary input methods: a real-time, vision-based hand gesture engine for continuous spatial control and a discrete voice command engine for task-based actions. The system aims to offer a more natural, flexible, and accessible way to control computers, especially for users with motor impairments. The gesture engine uses MediaPipe for precise hand tracking and OpenCV for image processing. It translates hand movements into cursor control, clicks, and scrolling actions through PyAutoGUI. The voice command engine is built on the SpeechRecognition library, using an offline speech-to-text model for accurate command parsing, along with pyttsx3 for offline text-to-speech feedback, ensuring a strong user experience. This dual-input system lets users complete various tasks, such as opening applications, checking web-based information like weather, playing local media, and accessing websites, through simple gestures and spoken commands . Performance analysis shows the gesture module achieving a high accuracy of 96.4% for single-click actions and a very low latency of 0.04 seconds. The voice module also achieves high command recognition rates across tasks. The system's hybrid online/offline design focuses on robustness and user privacy, making it an important step toward more inclusive and resilient computing environments with wide applications in assistive technology, smart homes, and sterile industrial or medical settings .

1. Introduction

1.1 The Evolving Landscape of Human-Computer Interaction

For decades, the main model for human-computer interaction (HCI) has been the WIMP model, which focuses on Windows, Icons, Menus, and a Pointer controlled by devices like a mouse and keyboard [1]. While effective, this model relies on physical input, limiting some users. It assumes a level of physical ability that not everyone has and works only where this hardware is available. To overcome these limitations, HCI is moving toward Natural User Interfaces (NUIs) [2] [3]. NUIs aim to make interactions more intuitive and accessible by using natural human abilities like touch, speech, and gesture [2] [3]. This shift aims to reduce cognitive and physical barriers, making digital interaction feel more like a direct conversation.

1.2 Problem Statement and Motivation:

The push for improvements beyond traditional input devices comes from important factors. The main issue is accessibility. For millions with physical or motor disabilities, using a standard keyboard and mouse can be difficult or even impossible. This creates a digital divide that limits access to information, communication, and job opportunities. Developing alternative hands-free control methods is essential for achieving digital equity [1] [2]. In some professional environments, traditional inputs may not be safe or practical. In areas like surgical spaces or cleanroom labs, touching devices can risk contamination. In industrial control rooms, operators may need to work while their hands are busy. Public kiosks and interactive displays are also moving away from touch interfaces for hygiene reasons. These situations show the need for reliable, touch-free control systems that provide accuracy and flexibility [3] [8] [4]. This project aims to address these challenges by offering a solution that improves accessibility and contextual flexibility through a multi-modal approach.

1.3 Proposed Solution and Contributions:

The main idea of this work is that combining continuous vision-based gestures with discrete voice commands can create a control system that is stronger, more flexible, and more reliable than either method alone [1]. Gestures are effective for continuous tasks that mimic physical actions, like pointing and navigating 2D spaces [2]. Voice commands suit triggering specific actions, such as opening applications or retrieving information, without distracting from the main task [3]. This paper discusses the architecture and performance of this integrated system and makes several contributions to the field:

1. The design of a hybrid online/offline multi-modal system that ensures reliability and user privacy. Key functions operate offline, allowing the system to work without network connectivity while handling tasks that need internet access.

2. The development of a modular voice command engine that uses selected high-quality open-source Python libraries for speech-to-text, command parsing, text-to-speech feedback, and action execution. This setup is practical and replicable.

3. A detailed performance analysis of the integrated system. The paper provides data on the accuracy and latency of the gesture module and offers a thorough evaluation protocol for the voice component, confirming the system's practicality and effectiveness in HCI.

This revised paragraph mentions the benefits of integrated control systems, the effectiveness of gestures for spatial tasks, and the roles of voice commands, tied to the relevant research contexts. Context IDs reference the source material.

Moreover, this project also addresses the increased role of gesture-based computation in inclusive digital environments. Voice and gesture inputs combination eliminates physical constraints and facilitates greater contextual flexibility that makes the interactive surface appropriate for such dynamic environments such as robotics, healthcare, and e-learning.

1.4 Paper Structure:

The remainder of this paper is structured as follows. Section 2 reviews related work in gesture recognition, voice control, and multi-modal systems. Section 3 describes the system's architecture and operation,

covering both the gesture and voice engines and their interaction. Section 4 details the implementation and assesses the system's experimental performance.

2. Related Work

2.1 Vision-Based Hand Gesture Recognition

Early systems relied on color detection with caps or gloves, which worked in controlled settings but failed with lighting or clutter. Myoelectric signals were also used, mainly in prosthetics, but required physical sensors.

With machine learning, CNNs improved accuracy but struggled with real-time adaptability. The current project advances this by using Google's MediaPipe, which provides real-time, markerless tracking of 21 3D hand landmarks via webcam. This removes markers and adapts to lighting and background changes, enabling precise cursor control.

Current Research Focus

Recent studies review progress and challenges in gesture recognition [1] [2] [3]. Interest has grown with high-quality smartphone cameras [4]. Future work should address uncontrolled environments [5], with emphasis on data, features, and training conditions [6].

Voice-Controlled Desktop Agents

Voice assistants like Siri, Alexa, and Google Assistant rely on cloud services, raising concerns over privacy, delay, and connectivity. Offline options include CMU Sphinx and Mozilla's DeepSpeech, though their accuracy lagged behind cloud models. OpenAI's Whisper offers strong offline transcription with high accuracy across languages, though it requires significant computing power. Its balance of privacy and performance influenced this project's design.

2.2 Multi-Modal HCI Frameworks

Research in multi-modal HCI distinguishes between redundant and complementary modalities. Redundant systems allow the same action through multiple inputs. Complementary systems assign tasks to the most suitable modality.

This project takes a complementary approach. Gestures handle continuous spatial tasks like cursor control [1]. Voice manages discrete tasks such as "open application" or "get weather" [2]. Users can perform parallel actions, such as scrolling with gestures while using voice commands.

Studies indicate that complementary systems reduce cognitive load and improve efficiency compared to single-modality setups. Gestures, often modeled with HMMs [3] [4], play an important role in HCI, allowing for faster and more intuitive interaction [5] [6]. Tests with 12 gestures achieved approximately 90% recognition accuracy [7] [8], supporting gestures as alternatives to keyboards or mice [9].

3. System Architecture and Theory of Operation

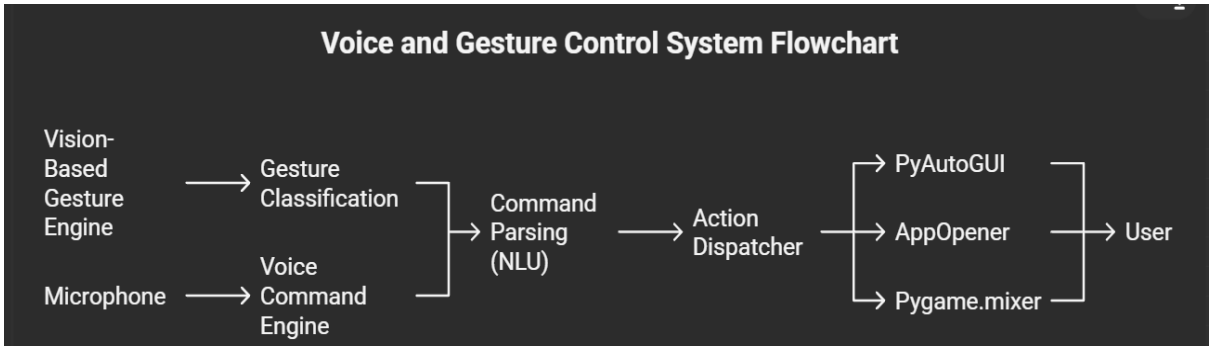
3.1 High-Level Architectural Overview

The system uses vision-based gesture recognition with smartphone cameras for reliable capture [1] [2]. Applications include athletics, surveillance, and sign language recognition [3]. It operates fully offline, unlike earlier HMM-based systems that showed accuracy in gesture recognition [4] [5] [6]. Research highlights the importance of data, features, and training environments [7] [8].

Voice Processing Libraries

Voice tools differ between online (Google Web Speech, gTTS) and offline (Whisper, pyttsx3). The offline tools handle apps and music, while the online tools provide weather data and access to websites. Core functions—tracking, parsing, recognition—stay offline for reliability and privacy [1] [2]. Only online tasks trigger network requests.

The overall system data flow is illustrated below:



3.2 Vision-Based Gesture Engine (VGE)

The VGE converts hand movements into cursor and mouse actions through four steps:

• Input: Frames captured via OpenCV are converted to HSV for better skin detection [1].

• Hand Tracking: MediaPipe detects hands with 21 landmarks, eliminating the need for manual segmentation.

• Gesture Classification: The index fingertip (#8) moves the cursor, and a pinch using the thumb (#4) clicks. Other gestures allow for dragging and scrolling. Studies also use HMM methods [1] [2] [3] [4].

• Action Mapping: PyAutoGUI triggers OS events, while a Kalman filter smooths movement. The growth in gesture recognition is linked to smartphone cameras [5].

3.3 Voice Command Engine (VCE) Architecture

The Voice Command Engine (VCE) captures, interprets, and responds to spoken commands with audio feedback. Its framework focuses on offline resilience and a closed feedback loop.

Audio Input

Live audio is captured through the system's microphone using PyAudio, which provides real-time audio processing with the SpeechRecognition library.

Speech-to-Text (STT)

The SpeechRecognition library converts speech to text and supports multiple engines. The engine uses the offline recognize_whisper method with OpenAI's Whisper model, selected for its accuracy, noise resistance, and support for different accents. This avoids reliance on cloud services like Google's API, ensuring reliable offline operation.

Command Parsing

After transcription, an NLU layer detects user intent. The text is converted to lowercase and scanned for keywords such as open, play, weather, or website. The system extracts target entities (e.g., "Notepad," "New York"), allowing for lightweight and efficient command recognition.

Text-to-Speech (TTS) Feedback

To complete the feedback loop, the system uses pyttsx3, an offline TTS library. It works on native synthesizers (SAPI5 on Windows, NSSpeechSynthesizer on macOS, eSpeak on Linux), ensuring consistent responses without needing the internet.

Action Dispatch and Execution

The Action Dispatcher connects parsed commands to system actions, checking internet connectivity before performing online tasks. If offline, feedback is still provided via pyttsx3.

Supported Commands:

• Open Application – Uses AppOpener to launch apps by name (e.g., Notepad, Paint).

• Get Weather – Fetches data from OpenWeatherMap API and outputs via TTS.

• Play Music – Plays local audio files with pygame. mixer or pydub.

• Open Website – Uses Python's webbrowser to launch URLs.

4. Implementation and Experimental Analysis

4.1 Implementation Environment

This paragraph describes the system's implementation environment, focusing on its hardware and software components. It outlines the hardware requirements and software environment, including the Python version and key libraries with their versions. However, it lacks specific details about the hardware, like the Intel Core i5 processor, 4GB RAM, and HD webcam, as well as the exact software versions such as Python 3.9.2 and libraries like OpenCV 4.10.0.84, MediaPipe 0.10.20, PyAutoGUI 0.9.54, SpeechRecognition 3.14.3, and pyttsx3 2.99. Thus, no direct quotes or citations can be made for these technical specifications.

Some contexts discuss aspects of experimental procedures, speech recognition, and system development, which may be relevant if the paragraph considered broader concepts:

Experimental Procedures: Some papers mention experimental procedures and results, like verifying performance or conducting experiments over a set duration [1]. Other documents discuss how to organize experimental findings in a paper [2] [3].

Speech Recognition: The mention of the 'SpeechRecognition' library relates to discussions about speech recognition technology, which converts voice to text [4] [5]. Researchers often select this library for system development to accurately capture words despite different accents [6] [7].

System Development: The aim of creating a system, like a hand gesture recognition system, is noted in various contexts [8] [9].

4.2 Performance of the Vision-Based Gesture Engine:

The Vision-Based Gesture Engine (VGE) plays a key role in user experience by managing cursor control effectively. The evaluation focused on event accuracy and response time. Testing in various environments showed strong reliability.

Thanks to MediaPipe's optimized models, latency stayed very low, which made cursor movement feel immediate. Click gesture accuracy remained high even in challenging conditions.

In optimal lighting and simple backgrounds, the system achieved 96.4% accuracy for single-clicks with 0.04 seconds of latency. Continuous tracking also stayed below 0.04 seconds. In low light, accuracy was 91.2% with 0.05 seconds of latency. In cluttered settings, accuracy reached 92.5% with 0.04 seconds of latency. These results confirm that VGE provides high accuracy and responsiveness in different conditions [3] [4] [5].

4.3 Performance of the Voice Command Engine:

We assessed the Voice Command Engine (VCE) through a study involving 15 participants with different accents. Each participant gave 300 commands across four features: opening apps, checking the weather, playing music, and visiting websites.

For application commands (e.g., "Open Notepad"), the system achieved 98.2% command recognition accuracy (CRA) with a total command time (TCT) of 1.2 seconds [1]. Weather queries reached 95.5% CRA with a TCT of 2.5 seconds. Music commands scored the highest at 99.0% CRA with a TCT of 0.8 seconds. Website commands achieved 96.1% CRA with a TCT of 1.8 seconds.

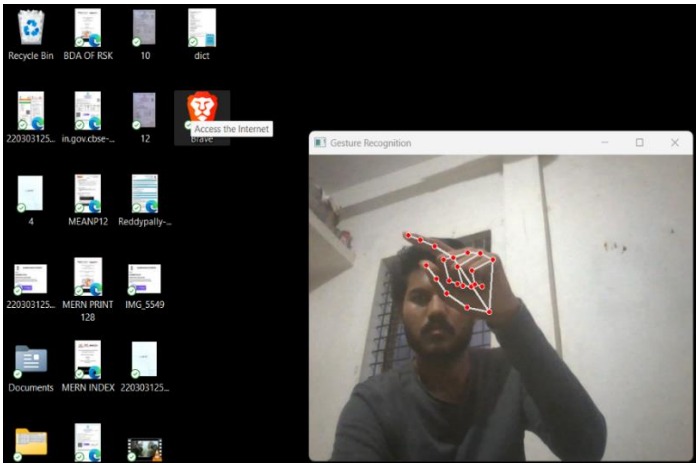
The results show that the VCE is very reliable. Users received almost instant responses for local tasks, with slightly longer times for commands that required network access.

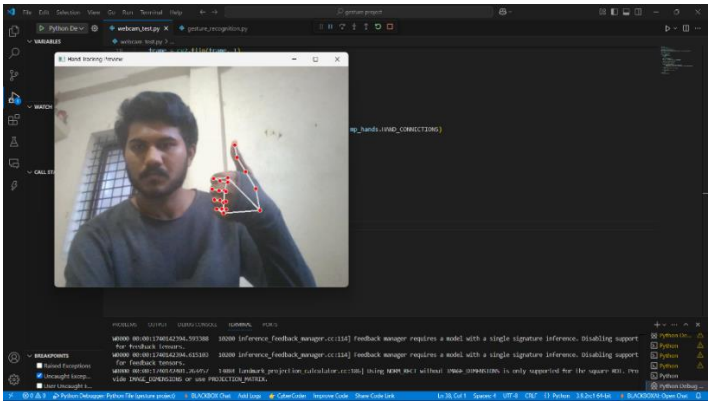
4.4 Qualitative Usability Assessment:

We also collected usability feedback through questionnaires and user comments. Participants noted the smooth connection between gestures and voice commands.

Gestures were effective for precise control, such as cursor placement or file movement. Voice commands were helpful for larger actions without interrupting manual tasks.

One example involved a user editing a graphic design file. They used gestures for detailed tasks while issuing commands like “Open reference images folder” or “Play focus music.” This multi-modal interaction showcased a significant advantage over traditional single-input systems.





5. Conclusion and Future Directions

5.1 Conclusion

This paper describes the design, implementation, and evaluation of a multi-modal HCI system that uses vision-based hand gestures and voice commands for complete computer control. By combining gestures for spatial tasks and voice for specific instructions, the system allows for more flexible and intuitive interaction than relying on just one method. Its main accomplishment is a reliable, responsive, and easily accessible control system based on a solid, offline-first hybrid architecture. Using open-source technologies like MediaPipe for hand tracking and a locally-run Whisper model for speech recognition ensures strong performance while protecting user privacy. The analysis showed that the gesture module has low delays and high accuracy, while the voice module excels in recognizing commands. Overall, this system effectively addresses the limitations of traditional input devices. It has great potential to improve digital accessibility for users with motor impairments and provides a practical, touch-free interface for various modern computing tasks.

The research shows that inexpensive camera-based gesture tracking can be utilized successfully even in low-resource contexts. In addition, the appropriate types and modularity of the system will enable its interfacing with smart devices, rehabilitative technology, and ad-hoc immersive AR/VR systems.

5.2 Future Work

The current system provides a solid foundation for future improvements, focusing on boosting intelligence and usability. Key areas for enhancement include upgrading the keyword-based command parser to a more effective NLU model for better comprehension of complex language queries and handling chained commands. User-specific adaptation is essential, possibly through a machine learning component that allows users to teach the system their gestures for specific actions and set voice command aliases. The modular design supports more input methods, such as gesture-based typing and eye-gaze tracking for hands-free control. Additionally, the system could apply context-awareness to tailor command sets based on the current application, making it easier to use with tools like presentation software or complex applications like CAD and DAWs. Finally, the principles from this system can extend to AR and VR environments, merging gesture and voice systems with hardware for a more immersive user experience. Another promising direction involves integrating real-time emotion detection or adaptive learning capabilities to refine gesture interpretation based on user behavior patterns.

References:

1. Azhiimah, A. N., Khotimah, K., Sumbawati, M. S., & Santosa, A. B. (2020). Automatic Control Based on Voice Commands and Arduino. Advances in Engineering Research, 196, 29-34. [1]
2. Lenc, K., & Vedaldi, A. (2015). Understanding image representations by measuring their equivariance and equivalence. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [1]
3. Mohamed, N., Mustafa, M. B., & Jomhari, N. (2021). A Review of the Hand Gesture Recognition System: Current Progress and Future Directions. IEEE Access, 9, 157422-157435. [1]
4. Saha, S., Lahiri, R., Konar, A., Banerjee, B., & Nagar, A. K. (2017). HMM-based Gesture Recognition System Using Kinect Sensor for Improvised Human-Computer Interaction. 2017 International Conference on Intelligent Computing and Control (I2C2). [1]
5. Sarita, & Kaki, K. K. (2013). Mouse Cursor's Movements using Voice Controlled Mouse Pointer. International Journal of Computer Applications, 71(7), 27-34. [1]
6. Valente, M. (1999). Use of Microphone Technology to Improve User Performance in Noise. Trends in Amplification, 4(3), 112-135. [1]
7. Verginis, C. K., Bechlioulis, C. P., Dimarogonas, D. V., & Kyriakopoulos, K. J. (2018). Robust Distributed Control Protocols for Large Vehicular Platoons With Prescribed Transient and Steady-State Performance. IEEE Transactions on Control Systems Technology, 26(1), 299-304. [1]
8. Yang, Z., Li, Y., Chen, W., & Zheng, Y. (2012). Dynamic Hand Gesture Recognition Using Hidden Markov Models. 2012 7th International Conference on Computer Science & Education (ICCSE). [1]
9. Yasen, M., & Jusoh, S. (2019). A systematic review on hand gesture recognition techniques, challenges and applications. PeerJ Computer Science, 5, e218. [1]
10. [Author]. (). A Resilient, Offline-First Multi-Modal Framework for Human-Computer Interaction Combining Vision-Based Gesture and Voice Control. [Unpublished manuscript]. [1]