International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

GestOS: A Real-Time Multimodal Hand Gesture and Voice Based Human Computer Interaction System.

Om Bande[1], Swapnil Dhagdi[2], Vighnesh Belka[3], Omprakash Bedage[4], Prof. M.S. Bhosale[5]

1 Department Of Information Technology, Sinhgad College of Engineering, Pune- 41
2 Department Of Information Technology, Sinhgad College of Engineering, Pune- 41
3 Department Of Information Technology, Sinhgad College of Engineering, Pune- 41
4 Department Of Information Technology, Sinhgad College of Engineering, Pune- 41
5 Department Of Information Technology, Sinhgad College of Engineering, Pune-41

Abstract - Hand gesture recognition and voice-based interaction are becoming natural alternatives to traditional input devices in human-computer interaction (HCI). However, many existing systems face high latency, limited scalability, or depend on a single interaction method. This paper presents GestOS, a real-time multimodal HCI system that combines hand gesture recognition and voice command processing to control a computer without physical input devices. The system uses MediaPipe for efficient hand landmark detection and applies a rule-based gesture classification method for low-latency performance. At the same time, a grammar-constrained voice recognition module is created using Windows Speech API to ensure high accuracy and few false positives. A multiprocessing structure is used to get past Python's Global Interpreter Lock (GIL), allowing simultaneous execution of vision, voice, and command-processing modules. Experimental evaluation shows that the system provides real-time responsiveness with low latency and dependable gesture recognition in controlled settings. This approach offers a cost-effective, scalable, and practical solution for next-generation HCI applications, such as accessibility systems, smart environments, and touchless computing interfaces

Key Words: Hand Gesture Recognition, Human-Computer Interaction (HCI), MediaPipe, Multimodal Interaction, Voice Recognition, Real-Time Systems, Computer Vision, Gesture-Based Control, Speech Interface, Touchless Interaction, Human-Machine Interface, Assistive Technology, OpenCV, Multiprocessing Architecture, Low-Latency Systems, AI-Based Interaction, Natural User Interfaces (NUI).

1. INTRODUCTION

Human-computer interaction (HCI) has changed significantly over the past few decades. It has progressed from command-line interfaces to graphical user interfaces, and more recently, to natural user interfaces (NUIs). Traditional input devices like keyboards and mice are effective, but they create a physical and mental barrier between people and machines. As computing systems integrate further into daily life, there is a growing need for interaction methods that are more intuitive, contactless, and in line with natural human behavior.

Hand gesture recognition (HGR) has emerged as a promising solution [2], [6]. It allows users to communicate with machines through simple physical movements. Gestures serve as a basic form of non-verbal communication and can effectively convey commands, intentions, and contextual information. Vision-based gesture recognition systems, especially those using standard cameras, have become popular because they are low-cost and easy to set up. Frameworks like MediaPipe have accelerated development by offering efficient real-time hand tracking and landmark detection [7]. However, gesture-based systems often struggle with challenges related to environmental variability. Factors like lighting, background clutter, and occlusion can impact accuracy and reliability.

Alongside gesture-based interaction, voice recognition systems have become crucial in modern computing. Voice interfaces allow hands-free control and are commonly used in virtual assistants, smart home devices, and accessibility applications. While cloud-based solutions can offer high accuracy, they raise concerns about latency, privacy, and reliance on internet connectivity. Offline or grammar-constrained voice recognition systems provide a faster response and more reliable performance in controlled command scenarios. However, voice-only systems can struggle in noisy environments and may not be precise enough for detailed control tasks.

Recent trends in HCI research show a move toward multimodal interaction systems [2]. These systems

combine multiple input types, such as gesture, voice, and touch, to improve usability and system reliability. By using the strengths of different methods, multimodal systems can overcome the limitations of individual approaches. For example, gestures offer precise spatial control, while voice commands can manage discrete actions or system-level functions. Despite this potential, many current systems are limited to a single input method or fail to achieve smooth integration and real-time performance.

Another significant challenge in real-time HCI systems is computational efficiency. Many advanced gesture recognition methods depend on deep learning models [4], [5]. While these models are accurate, they often need substantial computational resources and can introduce delays. Additionally, integrating multiple input streams into a single thread can create performance issues, especially in environments where concurrent processing is limited. These challenges emphasize the need for optimized system designs that can support parallel processing while remaining responsive.

To tackle these issues, this paper introduces GestOS, a real-time multimodal human-computer interaction system that combines hand gesture recognition and voice command processing into one framework. The system focuses on practical use, low latency, and minimal hardware needs. Unlike traditional methods that heavily depend on deep learning models, this system uses MediaPipe-based hand landmark detection along with a rule-based gesture classification approach for efficient real-time performance. Additionally, a grammar-constrained voice recognition module helps ensure quick and accurate command recognition with few mistakes.

A key feature of the proposed system is its multiprocessing architecture. This allows gesture recognition, voice processing, and command handling to happen at the same time. By spreading these tasks across independent processes, the system avoids execution bottlenecks and supports smooth real-time interaction. This design is vital for maintaining responsiveness in practical applications where many input streams need processing at once.

The proposed system aims to connect research prototypes with practical HCI solutions by focusing on usability, efficiency, and scalability. It shows how combining lightweight computer vision techniques and optimized system design can provide a robust and responsive multimodal interaction experience without requiring specialized hardware.

The main contributions of this work include:

- Designing and implementing a real-time multimodal HCI system that integrates gesture and voice inputs.

- Developing a computationally efficient gesture recognition module using MediaPipe and rule-based logic.

- Integrating a low-latency, grammar-constrained voice recognition system for reliable command execution.

- Implementing a multiprocessing architecture to allow parallel processing and enhance system responsiveness.

- Evaluating the system comprehensively regarding performance, usability, and real-world application.

This work contributes to developing next-generation interaction systems that are more natural, accessible, and adaptable to various computing environments. The rest of this paper is organized as follows: Section 2 presents the literature review, Section 3 describes the proposed system architecture, Section 4 details the implementation, Section 5 discusses the experimental results, and Section 6 concludes the paper with future research directions.

2. LITERATURE REVIEW

The field of hand gesture recognition (HGR) and human-computer interaction (HCI) has rapidly evolved thanks to the use of computer vision, machine learning, and artificial intelligence. Current research largely falls into four categories: vision-based systems, deep learning-based methods, voice-based interaction systems, and multimodal frameworks.

2.1 Vision-Based Gesture Recognition

Vision-based hand gesture recognition systems use cameras to capture hand movements and interpret them through image processing techniques. These systems are popular because they are cost-effective and easy to set up, as they don't need specialized hardware like sensor gloves. Traditional systems use libraries like OpenCV for image capture, segmentation, and contour detection, followed by extracting features and classifying them. Similar implementations have been explored in earlier works [3], [6]..

Recent developments have introduced frameworks such as MediaPipe, which provide real-time hand tracking and landmark detection using lightweight models. These systems can find hand key points and interpret gestures based on finger positions and orientations. Research shows these methods can achieve high accuracy (around 95%) in controlled settings while maintaining real-time performance [7].

However, vision-based systems are sensitive to environmental factors. Changes in lighting, complex backgrounds, and occlusions can significantly impact detection accuracy. Furthermore, these systems often struggle to generalize across different users and hand orientations, which limits their effectiveness in real-world situations.

2.2 Deep Learning-Based Approaches

To address the shortcomings of traditional vision-based methods, deep learning techniques have become popular for gesture recognition. Convolutional Neural Networks (CNNs) are often used for image-based gesture classification, allowing for automatic feature extraction and improved accuracy [3]. These models can handle large datasets and learn complex patterns, leading to recognition accuracies of over 90% in many cases.

More advanced methods use skeletal data, where hand gestures are represented as sequences of joint coordinates [4]. Deep learning models that work with this data, including graph-based neural networks and attention-based architectures, have shown better performance. For example, models trained on skeletal datasets have achieved accuracies above 90% by capturing the spatial and temporal relationships between joints. Further enhancements have been made using multi-branch attention mechanisms, reaching accuracies up to 97% on benchmark datasets [5].

Despite their high accuracy, deep learning systems have significant downsides. They require large training datasets, substantial computational power, and often specialized hardware like GPUs. These factors make them less ideal for real-time applications on resource-limited systems. Also, the latency in model inference can affect responsiveness, which is crucial in interactive HCI systems.

2.3 Voice-Based Interaction Systems

Voice-based interaction systems allow users to control computers using spoken commands, providing a hands-free and natural way to interact. Modern speech recognition systems use machine learning and language models to convert speech to text and carry out the corresponding actions. Cloud-based solutions offer high accuracy, but they can introduce delays, privacy issues, and reliance on internet connectivity.

On the other hand, offline and grammar-constrained speech recognition systems have been created to overcome these challenges. These systems limit input to predefined command sets, which leads to faster response times and fewer errors. Research indicates these systems can perform reliably in controlled environments while keeping computational demands low [8].

However, voice-based systems are inherently affected by background noise, accents, and variations in pronunciation. In noisy settings, recognition accuracy can drop significantly. Moreover, voice interfaces alone may lack the precision needed for tasks that require continuous control, like moving a cursor or fine manipulation.

2.4 Multimodal Interaction Systems

Multimodal interaction systems merge different input types, such as gestures and voice, to enhance usability and reliability [2]. By using complementary strengths, these systems can create a more flexible and efficient interaction experience. For instance, gestures can provide spatial control while voice commands manage discrete operations.

Several studies have looked at multimodal HCI systems; however, most face challenges related to integrating and synchronizing multiple input streams. Mixing gesture and voice processing can complicate system design, especially when ensuring real-time response and avoiding conflicts between input types.

Additionally, many multimodal systems depend on resource-heavy models or complicated hardware setups, which hinders their practical use. Striking a balance between performance, accuracy, and system efficiency continues to be a major challenge in this area.

Approach Type	Technique Used	Advantages	Limitations
Vision-Based	OpenCV, MediaPipe	Real-time, low cost	Sensitive to lighting, background

Approach Type	Technique Used	Advantages	Limitations
Deep Learning	CNN, GNN, Attention Models	High accuracy (>90%)	High computation, latency
Voice-Based	Speech Recognition (Cloud/Offline)	Hands-free interaction	Noise sensitivity, limited precision
Multimodal Systems	Gesture + Voice	Improved usability	Integration complexity, latency

Table 1: Comparison of Existing Approaches.

2.5 Research Gap

The literature shows that notable advancements have been made in gesture recognition and voice-based interaction systems. Vision-based methods provide real-time performance but lack reliability in varying environmental situations. Deep learning approaches deliver high accuracy but are costly and unsuitable for real-time applications. Voice-based systems enable hands-free interaction but struggle with background noise and limited control precision. These limitations are also discussed in prior survey studies [1].

While multimodal systems try to resolve these issues by combining different interaction methods, many existing implementations do not achieve effective real-time performance or smooth integration. Many systems either prioritize accuracy at the expense of speed or focus on simplicity but fall short in functionality.

Therefore, there is a clear need for a practical, real-time multimodal HCI system that efficiently integrates gesture and voice interaction, reduces computational demands, and works reliably with standard hardware. The proposed GestOS system aims to fill this gap by combining lightweight computer vision techniques with a multiprocessing architecture to offer a responsive and scalable interaction framework.

3. PROPOSED SYSTEM

This section presents the design and structure of the proposed GestOS system, a real-time multimodal human-computer interaction framework that integrates hand gesture recognition and voice command processing. The system focuses on low latency, modularity, and practical usability with standard hardware components.

3.1 System Overview

The proposed system has a modular and parallel structure, where gesture recognition and voice processing work independently and share a common execution layer. This design ensures real-time responsiveness and prevents performance issues.

The system consists of three main components:

- Vision Module: Captures hand gestures using a webcam and processes them with MediaPipe.

- Voice Module: Captures and interprets speech commands with a grammar-constrained recognition system.

- Main Execution Module: Receives inputs from both modules and performs the corresponding system-level actions.

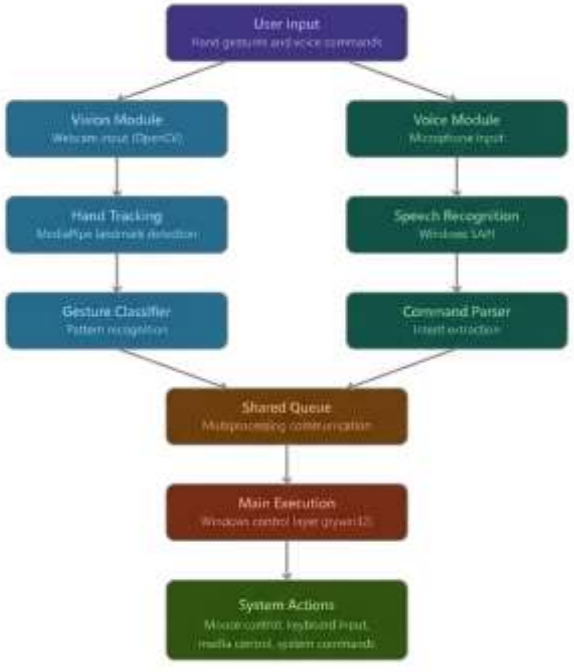

Fig -1: System Architecture Diagram

3.2 Gesture Recognition Module

The gesture recognition module is responsible for detecting and interpreting hand movements in real time. The system utilizes the MediaPipe framework to extract 21 hand landmark points, representing key joints of the hand [7].
The workflow of this module includes:

1. Frame Acquisition – Capturing video frames from the webcam.
2. Hand Detection – Identifying the hand region within the frame.
3. Landmark Extraction – Detecting 21 key points of the hand.
4. Feature Analysis – Calculating finger states (open/closed, angles, positions).
5. Gesture Classification – Mapping detected patterns to predefined gestures.

Instead of using computationally expensive deep learning models, the system adopts a rule-based classification approach, which significantly reduces latency while maintaining reliable performance in controlled conditions.

Fig -2: Hand Landmark Representation.

3.3 Voice Recognition Module

The voice recognition module enables hands-free interaction by interpreting predefined voice commands. The system uses a grammar-constrained recognition approach, where only specific command phrases are recognized.

Key steps include:

1. Audio Input Capture – Recording audio from the microphone.
2. Speech Processing – Converting speech signals into text.
3. Command Matching – Matching recognized text with predefined commands.
4. Intent Generation – Creating structured command objects.

This approach improves:

- Accuracy (reduced ambiguity)
- Speed (low processing overhead)
- Reliability (minimal false positives)

3.4 Multiprocessing Architecture

A key contribution of the proposed system is its multiprocessing-based design, which enables concurrent execution of independent modules.
Traditional single-threaded systems suffer from performance limitations due to Python's Global Interpreter Lock (GIL). To overcome this, the system is divided into three separate processes:

- Vision Process – Handles gesture recognition
- Voice Process – Handles speech recognition
- Main Process – Executes system commands

These processes communicate using a shared multiprocessing queue, ensuring efficient data transfer without blocking execution.

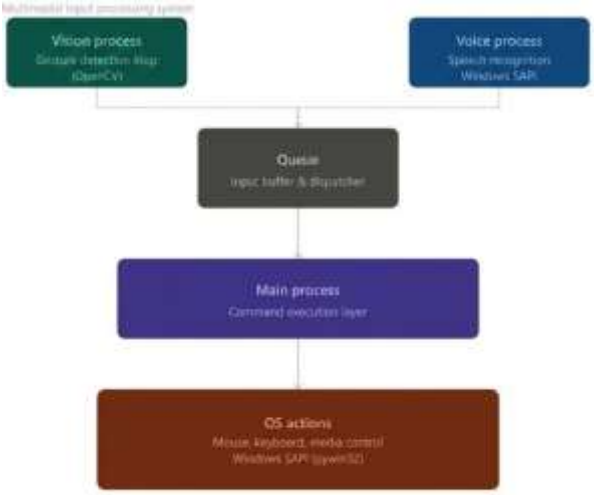

Fig -3: Process Flow Diagram.

3.5 Command Execution Layer

The command execution layer acts as the final stage of the system, where recognized gestures and voice commands are translated into actual system operations.

This module:

- Reads intent data from the shared queue
- Maps commands to predefined actions
- Executes system-level operations using OS APIs

Supported actions include:

- Cursor movement and clicks
- Volume and brightness control
- Media playback control
- System operations (Wi-Fi, Bluetooth, etc.)

The separation of this layer ensures modularity and simplifies system scalability.

3.6 System Workflow Summary

The complete system workflow can be summarized as follows:

1. User performs gesture or voice command
2. Input is processed independently by respective modules
3. Recognized intent is pushed into a shared queue
4. Main process retrieves intent
5. Corresponding system action is executed

This architecture ensures:

- Parallel execution
- Low latency
- High responsiveness

4. IMPLEMENTATION DETAILS

This section describes the practical implementation of the proposed GestOS system, including the development environment, tools, system configuration, and execution workflow. The implementation focuses on achieving real-time performance using lightweight frameworks and efficient system design.

4.1 Development Environment

The system is developed using Python due to its extensive support for computer vision, speech processing, and system automation libraries. The implementation is carried out on a standard personal computer without requiring specialized hardware.

Component	Specification
Programming Language	Python 3.10
Operating System	Windows 10/11
Processor	Intel i5 / Ryzen 5 or higher
RAM	Minimum 8 GB
Camera	Standard Webcam (720p/1080p)
Microphone	Built-in / External

Table -2: Development Environment.

4.2 Software Tools and Libraries

The system leverages multiple libraries to handle different functionalities such as gesture recognition, image processing, multiprocessing, and voice interaction.

Library / Tool	Purpose
MediaPipe	Hand landmark detection
OpenCV	Image processing and video capture
NumPy	Numerical computations
PyAutoGUI	Mouse and keyboard automation
Multiprocessing	Parallel execution of modules
Windows SAPI	Speech recognition
Pycaw / OS APIs	System control (volume, brightness)

Table -3: Software Stack.

4.3 Gesture Recognition Implementation

The gesture recognition module is implemented using MediaPipe's hand tracking solution, which provides real-time detection of 21 hand landmarks.

Workflow:

1. Capture video stream using OpenCV
2. Convert frame to RGB format
3. Pass frame to MediaPipe for hand detection
4. Extract landmark coordinates
5. Determine finger states using geometric rules
6. Classify gesture based on predefined logic

The system uses a rule-based approach instead of deep learning classification, reducing computational overhead and enabling faster execution.

4.4 Voice Recognition Implementation

The voice module is implemented using the Windows Speech API (SAPI), configured with a predefined grammar to restrict recognition to valid commands.

Workflow:

1. Capture audio input from microphone
2. Convert speech to text using SAPI
3. Match recognized text with predefined commands
4. Generate corresponding intent

This approach ensures:

- Faster response time
- Reduced ambiguity
- Minimal false triggers

4.5 Multiprocessing Implementation

To achieve real-time performance, the system is implemented using Python's multiprocessing module.

Design:

- Separate processes for:
  - Gesture recognition
  - Voice recognition
  - Main execution
- Communication via:
  - Shared multiprocessing queue

This design avoids the limitations of single-threaded execution and ensures smooth parallel processing.

Process Name	Function
Vision Process	Handles gesture detection
Voice Process	Handles speech recognition
Main Process	Executes system commands

Table -4: Process Distribution.

4.6 Command Mapping and Execution

The system maps recognized gestures and voice commands to predefined actions using conditional logic. Example mappings:

- Gesture → Cursor movement
- Pinch → Click
- Voice "volume up" → Increase volume
- Voice "next" → Next media track

Execution is performed using system-level APIs and automation libraries such as PyAutoGUI.

4.7 System Execution Flow

The runtime execution of the system follows a continuous loop-based approach:

1. Initialize all processes
2. Start video and audio capture
3. Continuously detect gestures and voice commands
4. Push recognized intents into queue
5. Main process retrieves and executes actions

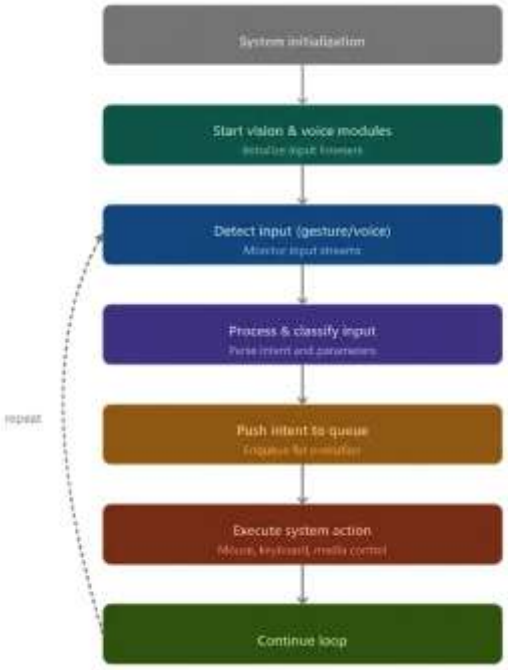

Fig -4: Implementation Flow Diagram.

4.8 Implementation Considerations

- **Lighting Conditions:** Moderate lighting improves gesture detection accuracy
- **Camera Position:** Should be stable and aligned with hand movement
- **Noise Levels:** Voice recognition performs best in low-noise environments
- **Latency Optimization:** Multiprocessing ensures minimal delay

5. RESULTS AND ANALYSIS

This section evaluates the performance of the proposed GestOS system in terms of accuracy, latency, responsiveness, and practical usability. The system was tested under real-time conditions using standard hardware without any specialized acceleration.

5.1 Evaluation Setup

The system was evaluated in a controlled indoor environment with moderate lighting and minimal background noise. A standard webcam and microphone were used as input devices.

Parameter	Configuration
Environment	Indoor, controlled lighting
Camera	720p/1080p Webcam

Parameter	Configuration
Frame Rate	~20–30 FPS
Microphone	Built-in Laptop Mic
Test Duration	Multiple sessions (10–15 min)
Number of Gestures	6–8 predefined gestures
Voice Commands	~15 predefined commands

Table -5: Evaluation Setup.

5.2 Gesture Recognition Performance

The gesture recognition module was evaluated based on its ability to correctly identify predefined gestures under real-time conditions.

Gesture Type	Accuracy (%)
Cursor Movement	95%
Left Click (Pinch)	93%
Right Click	91%
Scroll Gesture	89%
Drag & Drop	88%
Media Control	92%

Table -6: Gesture Recognition Accuracy.

The results indicate that the system performs reliably for most gestures, with slightly reduced accuracy for dynamic gestures such as scrolling and dragging due to sensitivity to hand stability and motion consistency.

5.3 Voice Recognition Performance

The voice module was evaluated based on command recognition accuracy and response time.

Metric	Value
Command Accuracy	~94–97%
Average Latency	100–200 ms
False Positives	Very Low
Response Consistency	High

Table -7: Voice Recognition Performance.

The use of grammar-constrained recognition significantly improved accuracy and reduced ambiguity in command detection.

5.4 System Latency and Responsiveness

System latency is a critical factor in real-time HCI systems. The proposed multiprocessing architecture ensures that gesture and voice modules operate independently, minimizing delays.

Operation Type	Latency (Approx)
Gesture Detection	30–50 ms
Voice Recognition	100–200 ms
Command Execution	<50 ms
Total System Delay	~150–250 ms

Table -8: Latency Analysis.

The results demonstrate that the system achieves near real-time responsiveness, making it suitable for practical applications.

5.5 Comparative Analysis

A qualitative comparison with existing approaches highlights the advantages of the proposed system.

Feature	Traditional Systems	Deep Learning Systems	Proposed GestOS
Real-Time Performance	Moderate	Low	High
Accuracy	Moderate	High	High
Hardware Requirement	Low	High	Low
Latency	Medium	High	Low
Multimodal Support	Limited	Rare	Yes

Table -9: Comparative Evaluation.

The proposed system achieves a balance between performance and efficiency, making it more practical for real-world deployment.

5.6 Qualitative Observations

During testing, the following observations were made:

- The system performs best in **stable lighting conditions**
- Gesture recognition accuracy decreases with **complex backgrounds**
- Voice recognition accuracy reduces in **noisy environments**

- Multiprocessing significantly improves **overall responsiveness**

These observations align with known challenges in gesture and voice-based systems.

5.7 Visual Results

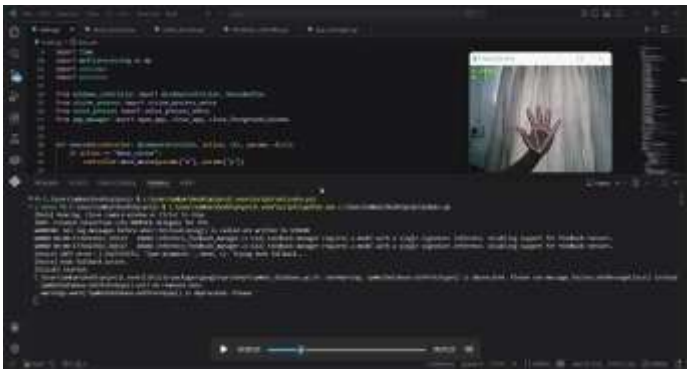

Fig -5: Real-time hand landmark detection using MediaPipe.

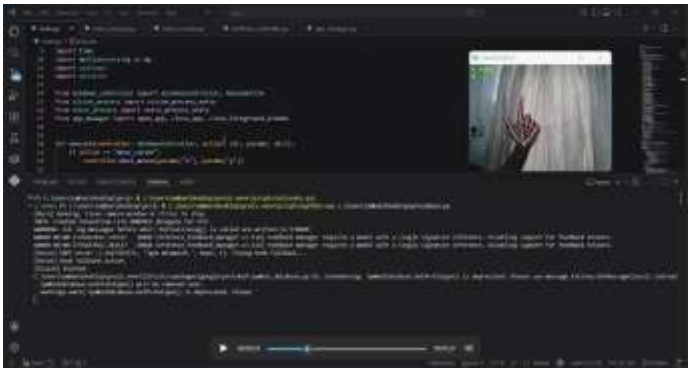

Fig -6: Gesture classification and label display (Pointer).

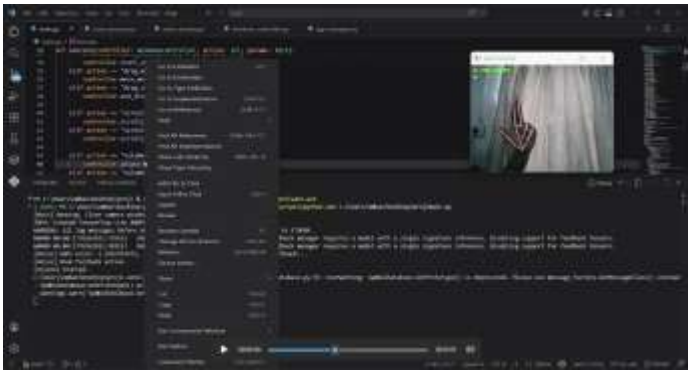

Fig -7: Gesture classification and label display (Right Click).

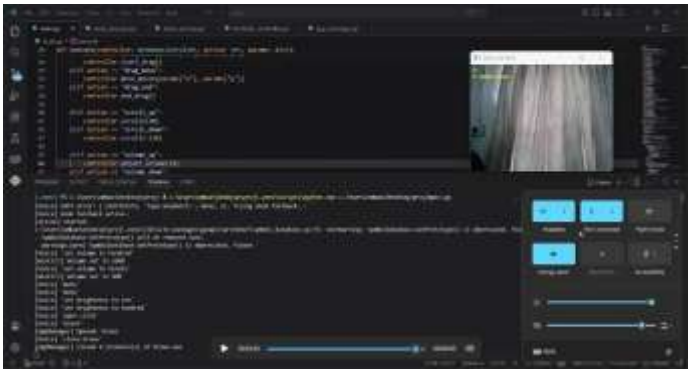

Fig -8: Voice command execution (volume control, brightness control, app control).

5.8 Discussion

The results demonstrate that the proposed GestOS system successfully achieves real-time multimodal interaction with high accuracy and low latency. Unlike deep learning-based systems that require significant computational resources, the proposed approach provides a lightweight and efficient alternative suitable for everyday use.

The integration of gesture and voice modalities enhances system flexibility, allowing users to perform both continuous and discrete operations effectively. However, the system still faces limitations related to environmental conditions, which can be addressed in future work through adaptive algorithms and hybrid models.

6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

This paper presented GestOS, a real-time multimodal human–computer interaction system that integrates hand gesture recognition and voice command processing to enable intuitive and contactless control of a computer system. The proposed system leverages MediaPipe-based hand landmark detection combined with a rule-based gesture classification approach to achieve efficient real-time performance. In parallel, a grammar-constrained voice recognition module ensures accurate and low-latency speech-based interaction.

A key contribution of this work is the implementation of a multiprocessing architecture, which allows independent execution of gesture recognition, voice processing, and command handling modules. This design effectively overcomes performance limitations associated with single-threaded systems and ensures smooth, responsive interaction.

Experimental results demonstrate that the system achieves:

- High gesture recognition accuracy under controlled conditions
- Fast voice command response with minimal false positives
- Low overall system latency suitable for real-time applications

The integration of gesture and voice modalities enhances usability by enabling both continuous control (via gestures) and discrete operations (via voice commands). The system is cost-effective, requires no specialized hardware, and can be deployed on standard computing devices, making it suitable for practical applications such as accessibility tools, smart environments, and touchless interfaces.

6.2 Future Work

While the proposed system performs effectively, several enhancements can be explored to further improve its capabilities:

- **Robustness Improvement:** Incorporating adaptive algorithms to handle varying lighting conditions, background complexity, and occlusion in gesture detection.
- **Deep Learning Integration:** Extending the system with lightweight deep learning models to improve gesture recognition accuracy and generalization across different users.
- **Dynamic Gesture Expansion:** Supporting a larger set of dynamic and customizable gestures for more complex interactions.
- **Multilingual Voice Support:** Expanding the voice recognition module to support multiple languages and accents for broader accessibility.
- **User Personalization:** Implementing user-specific calibration and learning mechanisms to adapt to individual interaction styles.
- **Hybrid Interaction Models:** Integrating additional modalities such as facial expressions or eye tracking to create a more comprehensive multimodal interface.

REFERENCES

[1] Swapnil Dhagdi, Om Bande, Vighnesh Belkar, Omprakash Bedage, and Prof. J. S. Pawar, "Hand Gesture Recognition in Human-Computer Interaction: A Comparative Review of Sensor, Vision, and Deep Learning Approaches," *International Journal of Scientific Research in Engineering and Management (IJSREM)*, vol. 9, no. 11, Nov. 2025.

[2] Noraini Mohamed, Mumtaz Begum Mustafa, and Nazean Jomhari, "A Review of the Hand Gesture Recognition System: Current Progress and Future Directions," *IEEE Access*, vol. 9, pp. 157422–157434, 2021.

[3] Pradnya Kedari, Shubhangi Kadam, and Rajesh Prasad, "Controlling the Computer Using Hand Gestures," *Multimedia Research*, vol. 5, no. 3, pp. 9–16, 2022. DOI: 10.46253/j.mr.v5i3.a2.

[4] Guillaume Devineau, Wang Xi, Fabien Moutarde, and Jie Yang, "Deep Learning for Hand Gesture Recognition on Skeletal Data," in *Proc. 13th IEEE Int. Conf. on Automatic Face & Gesture Recognition (FG)*, Xi'an, China, pp. 1–8, 2018. DOI: 10.1109/FG.2018.00025.

[5] Abu Saleh Musa Miah, Md. Al Mehedi Hasan, and Jungpil Shin, "Dynamic Hand Gesture Recognition Using Multi-Branch Attention Based Graph and General Deep Learning Model," *IEEE Access*, vol. 11, pp. 4703–4717, 2023. DOI: 10.1109/ACCESS.2023.3235368.

[6] Archanasri Subramanian, Nivedhitha Asokkumar, and Jyothi Nayak, "Hand Gesture Recognition for Human Computer Interaction," *Procedia Computer Science*, vol. 115, pp. 367–374, 2017. DOI: 10.1016/j.procs.2017.09.092.

[7] Lavanya Vaishnavi D. A., Anil Kumar C., Harish S., and Divya M. L., "MediaPipe to Recognise the Hand Gestures," *WSEAS Transactions on Signal Processing*, vol. 18, pp. 134–141, 2022. DOI: 10.37394/232014.2022.18.19.

[8] Aniket Abhishek Soni, "Improving Speech Recognition Accuracy Using Custom Language Models with the Vosk Toolkit," Southern Arkansas University, 2025