

Virtual Mouse Using Hand Gesture with Voice Assistant

Arya Prashant Ghadge, Tanvi Prasad Chavan, Prof. Mohan Bonde

^{1,2} Dept. of Data Science and Engineering Usha Mittal Institute of Technology Mumbai, India
Assistant Prof Dept. of Data Science and Engineering Usha Mittal Institute of Technology Mumbai, India

ABSTRACT-Human Computer Interaction (HCI) is evolving toward more natural and contactless interaction methods to improve usability, accessibility, and safety. Traditional input devices like mouse and keyboard require continuous physical contact, which may not be suitable in certain environments or for users with physical limitations. This paper proposes a multimodal system, Virtual Mouse Using Hand Gesture with Voice Assistant that enables hands-free computer control using real-time hand gesture recognition and voice commands. The system uses MediaPipe and OpenCV to detect hand movements for cursor control, clicking, and system operations, while a voice assistant allows users to perform tasks such as opening applications and searching information.

By integrating gesture and voice interaction into a unified framework, the system ensures efficient and seamless operation. The proposed solution is cost-effective, easy to implement, and suitable for practical real-world applications.

Keywords – virtual mouse, hand gesture recognition, voice assistant, human computer interaction, mediapipe, opencv, multimodal interface, accessibility.

I. INTRODUCTION

Human Computer Interaction is central to the design of contemporary computing systems, due to its direct influence on how efficiently and comfortably users interact with digital environments. While the classical mouse and keyboard are still the most popular input devices, they also involve continuous physical contact and fine grained motor action. Such a requirement is limiting for patients who are motor impaired and may be less appropriate in situations (hospitals, laboratories, public kiosks) where maintaining physical hygiene is paramount.

Latest developments in computer vision and speech processing have made it possible to consider alternative interaction paradigms based on hand gestures or voice commands. Gestures provide an intuitive way to manipulate visual elements, while voice interfaces allow users to give high-level commands or access information. Most available solutions, however, treat gesture-based and

voice-based interaction as separate modalities, limiting their real-world applicability. A multimodal interaction mechanism can address this limitation by integrating continuous gesture control with command-driven voice interaction.

In this paper, a consolidated multimodal system called *Virtual Mouse Using Hand Gesture with Voice Assistant* is introduced. The system enables real-time cursor manipulation through hand movements while also allowing system-level and application-level operations through synchronized voice commands.

II. RELATED WORK

A lot of studies look at how gestures can control virtual mice, some relying on older image processing tools while others apply newer machine learning ideas.[4][5] Lately, using MediaPipe for hand landmark detection has gained attention due to its efficiency and ability to track hands accurately in real time on everyday hardware.[1] Still, certain projects rely on different deep learning setups like CNNs or RNNs to recognize moving hand patterns.[1][4] Even so, ways like these often lead to sharper recognition results. Still, they tend to drain system resources.[1]

Nowhere is the shift clearer than in tools shaped by talk and touch. Instead of relying on just one method, people explore combinations like moving hands while speaking to create smoother experiences.[2][3] Yet here's a catch, most setups today still depend on bulky hardware such as depth cameras.[4] That gear adds expense and makes installation tougher. So, while early versions work, they often come at a steep price in setup hassle.

What makes the suggested system different is how it combines gestures and voice control within one smooth interface, all powered by an ordinary webcam. Rather than keeping these inputs apart, it brings them together into a unified experience that feels natural and immediate. A broad set of hand movements allows fine-grained interaction, while voice commands are understood to trigger more complex tasks. At its core, a smart coordination system keeps everything aligned so both inputs behave as one logical input. This setup blocks

clashing commands while keeping hand gestures and spoken control working together making things feel smoother, more human.

III. PROPOSED SYSTEM

A. SYSTEM OVERVIEW

The proposed system consists of three main modules:

1. Hand gesture recognition based virtual mouse module.
2. Voice assistant and command interpretation module.
3. Control and coordination module.

A camera continuously captures live input to track hand movements and perform cursor operations such as movement, clicking, and scrolling. Simultaneously, voice commands are processed to perform tasks such as opening applications and browsing. A centralized coordination system ensures synchronization between gesture and voice inputs, preventing conflicting actions and enabling seamless interaction.

B. System Architecture

The proposed Virtual mouse using hand gesture with voice assistant follows a modular layered architecture to support real-time processing and multimodal interaction.

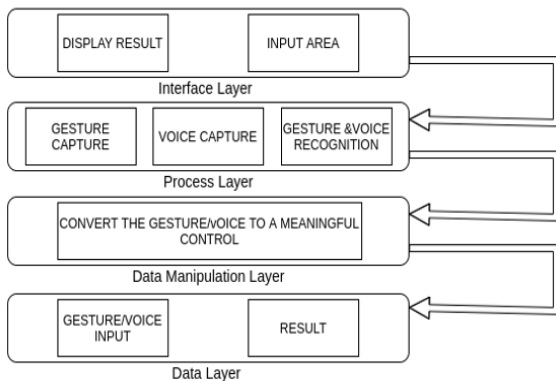


Fig 3: System architecture

1. User Interface Layer:

The user interface provides a graphical interface that displays user commands, assistant responses, and system status. Commands are forwarded to the Python backend using Eel functions, acting as a communication bridge between the user and the system.

2. Interaction Processing Layer:

This layer consists of two major subsystems.

a) Hand Gesture Processing Subsystem
The system captures video frames using OpenCV and processes them using MediaPipe to detect hand landmarks. A geometric approach is used to recognize gestures based

on landmark positions. A stabilization mechanism is applied to reduce noise and improve accuracy.

b) Command Interpretation Subsystem
The system processes user commands using rule-based matching. It supports application control, web navigation, conversational responses, and system operations. Knowledge queries are handled through a lightweight interface.

3. Control and Coordination Layer:

This layer manages synchronization between gesture and voice inputs using a shared execution flag. It controls the activation of the gesture engine and ensures that actions are executed only when enabled.

4. Execution Layer:

This layer performs system actions such as cursor movement, mouse operations, drag-and-drop, system control, application launching, web browsing, and keyboard automation. It ensures proper mapping between recognized inputs and system responses.

IV. Research Methodology

This section is about how we designed and implemented the multimodal interaction framework that we proposed.

A. Software Libraries and Tools:

We used the following libraries and tools to make this work:

- OpenCV to capture video and process frames.
- MediaPipe to detect hand landmarks in time.
- PyAutoGUI to automate the mouse and control the cursor.
- NumPy and math functions to do geometric calculations.
- pyttsx3 to give text-to-speech feedback.
- Eel framework to make the user interface and communicate with the backend.
- Wikipedia API to answer knowledge-based questions.
- OS and web browser libraries to do system and web things.
- Multithreading to run modules at the same time.

B. HAND GESTURE RECOGNITION METHOD:

The hand gesture recognition module enables users to interact with the system using hand movements. A webcam continuously captures live video frames, which are processed using OpenCV. The frames are flipped horizontally for natural interaction and converted to RGB format for processing using MediaPipe Hands.

MediaPipe detects hand landmarks in real time and identifies the presence of one or two hands. A geometric analysis is applied to recognize gestures based on the relative positions and distances between landmarks.

To improve accuracy and reduce false detections, a stabilization mechanism is applied where a gesture is considered valid only if it is consistently detected across multiple frames.

The system supports the following operations:

- Cursor movement .
- Left click.
- Right click.
- Double click.
- Drag and drop.
- Volume control.
- Brightness control.

Each recognized gesture is mapped to a corresponding system action using automation functions. A neutral hand state is maintained to prevent unintended operations.

C. Voice Assistant and Command Processing Method:

The voice assistant operates as a text-based command processing system that receives user input through a graphical chatbot interface and processes it using a Python backend via the Eel framework.

Commands are parsed using a rule-based intent recognition approach. Each input is preprocessed and matched with predefined patterns to determine the appropriate action.

The assistant supports multiple categories of operations, including system control, application launching, web navigation, and information retrieval. Once a command is recognized, the corresponding task is executed, and feedback is provided using text-to-speech synthesis.

D. Mechanism of Multimodal Coordination:

There is a shared execution state variable, which indicates whether the gesture recognition engine is running. This is controlled by a central coordination mechanism issuing commands through the user interface of the chatbot.

The gesture controller checks whether it is executing before every frame and will not perform any cursor action if gesture recognition is disabled. This technique allows switching between interaction modes without accidentally moving when switching the mode.

E. WORKING:

The Virtual Mouse Using Hand Gesture with Voice Assistant system operates in real time by processing both hand gestures and user commands simultaneously. The system integrates gesture recognition and command

processing to perform system operations with low latency and smooth interaction.

1. Hand Gesture Recognition Working:

The system captures live video input using a webcam and detects hand movements using MediaPipe. Hand landmarks are tracked and analyzed to recognize gestures, which are mapped to corresponding system actions such as cursor movement, clicking, and scrolling. The system ensures real-time response by continuously processing frames and executing actions without delay.

1.1 Landmark extraction and hand detection:

Video frames are captured and processed using MediaPipe Hands to detect hand landmarks. The system identifies multiple landmark points and determines hand orientation, enabling accurate tracking using a standard RGB camera without requiring specialized hardware.

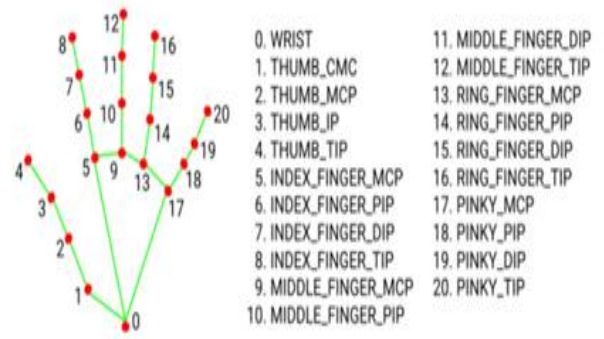


Fig 1: Hand landmarks detected using mediapipe hands model [6]

1.2. Gesture Classification:

Gestures are recognized using geometric relationships between hand landmarks, including distances and relative positions. To improve reliability, a gesture is confirmed only when it is consistently detected across multiple frames, reducing the effect of noise and unintended movements.

1.3. Action Mapping and Execution:

Recognized gestures are mapped to predefined system actions using automation functions. The system performs cursor control, clicking, dragging, and system adjustments such as volume and brightness. Execution occurs only when the gesture system is active, ensuring controlled and stable interaction.

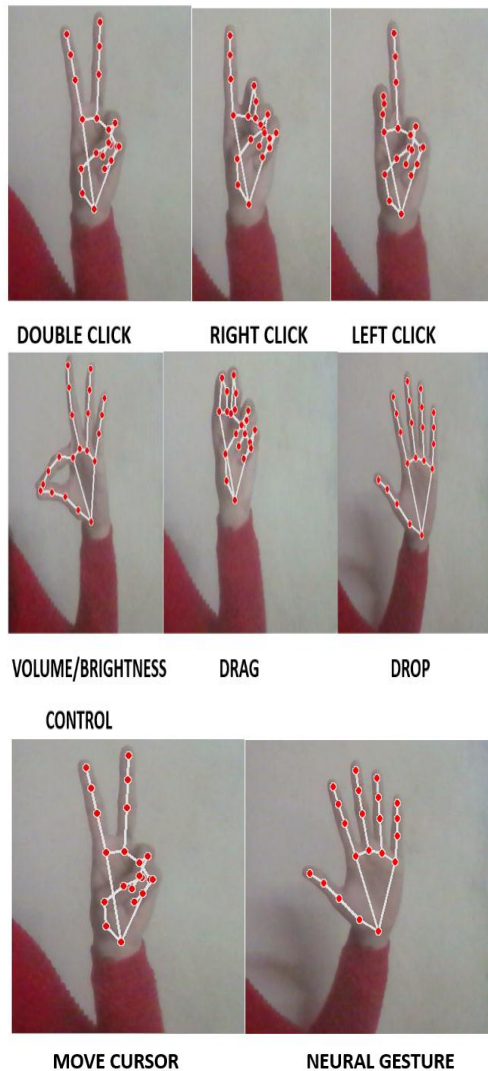


Fig 2: Supported hand gestures for mouse click operations, cursor movement and system control .

2. Voice Assistant Working:

The voice assistant works alongside gesture control by processing user commands through a chatbot interface. Commands are sent to the backend, preprocessed, and matched using rule-based logic. Based on the identified intent, the system performs actions such as application control, web navigation, and information retrieval. Responses are generated using text-to-speech and displayed on the interface to provide feedback to the user.

3. Dataset Description:

Our proposed system does not rely on custom collected dataset as it utilizes pre trained MediaPipe Hands model. The underlying training dataset is a large-scale proprietary dataset composed of real-world and synthetic hand images. The dataset covers diverse hand poses, skin tones, lighting conditions and backgrounds.

Each hand sample is annotated with twenty-one landmark points representing the wrist and finger joints. Synthetic hand images generated using three-dimensional modelling are included to improve robustness against occlusion, extreme poses and perspective variations.

Data augmentation techniques such as rotation, scaling, flipping and illumination adjustment are applied during training of the model. The availability of a robust pre-trained model allows the proposed system to perform accurate real-time hand tracking without collecting or training a custom dataset.

V. IMPLEMENTATION

This section describes the implementation of the proposed system and how the architecture is realized in practice.

A. Backend Framework and Execution Environment:

The entire system is implemented in Python and executed on a Windows platform. Independent modules handle gesture recognition, assistant logic and user interface communication. Multithreading enables concurrent execution of the gesture engine and chatbot interface.

B. Graphical User Interface Integration:

The graphical interface is embedded into the application using the Eel framework. The backend exposes Python functions to the interface, allowing bidirectional communication.

User messages are added to the chat window and processed by the backend. System responses and execution feedback are displayed in real time.

C. Voice Assistant and Command Handling Implementation:

The assistant operates as a text-based intelligent command processor. Incoming commands are stored in a queue and processed sequentially.

Each command is analyzed using rule-based matching. Supported commands include greeting, assistant identification, date and time queries, web access, application launching, gesture engine control, copy and paste operations and program termination.

Spoken feedback is generated using an offline text-to-speech engine.

D. Gesture Recognition Engine Implementation:

The gesture recognition engine runs as an independent thread. It continuously captures frames from the webcam and processes them using MediaPipe.

For each detected hand, landmark positions are extracted and passed to the gesture recognition logic. The system supports both single-hand and dual-hand interaction using major and minor hand identification.

E. Cursor Control and Motion Stabilization:

Cursor position is computed using a stable landmark point on the hand. A dampening mechanism is applied using the difference between current and previous positions to reduce jitter and sudden cursor jumps. The scaling factor is adjusted dynamically to ensure smooth motion.

F. Gesture-to-Action Mapping:

The following gesture mappings are implemented:

- V-shaped finger posture for cursor movement.
- Fist gesture for drag-and-drop.
- Two-finger closed gesture for double click.
- Index finger gesture for right click.
- Middle finger gesture for left click.
- Pinch gesture using the major hand for system volume control.
- Pinch gesture using the major hand for screen brightness control.

Multiple item selection is supported through continuous drag-based interaction.

G. System Brightness and Volume Control:

System brightness is controlled using a dedicated brightness control interface. Pinch displacement magnitude is converted into incremental brightness changes.

System volume control is implemented using the operating system audio endpoint interface. The volume level is updated smoothly based on pinch gesture movement.

H. Application and System Control Implementation:

The assistant supports direct execution of system applications through operating system calls. The supported applications include:

- Notepad.
- Calculator.
- Command Prompt.
- Microsoft Word.

- Microsoft Excel.

- WhatsApp.

Web browsing and search operations are executed through the default system browser.

I. Keyboard Automation:

Keyboard automation is implemented to support copy and paste operations using simulated control key combinations. This enables text manipulation without physical input devices.

J. Concurrent Execution and Thread Management:

The chatbot interface and gesture recognition engine operate in separate threads. A shared execution flag ensures that the gesture engine can be started and stopped safely without interrupting the chatbot process.

This design improves responsiveness and ensures stable system behaviour.

VI. RESULTS AND DISCUSSION

1. Testing and Validation :

The proposed system was tested under varying conditions, including different lighting environments, backgrounds, and hand positions, to evaluate its performance and adaptability. The gesture recognition module demonstrated smooth cursor movement and accurate detection of actions such as clicking and dragging under stable lighting conditions. However, slight accuracy degradation was observed in low-light conditions or when hand occlusion occurred.

The voice assistant module showed reliable performance with fast response times and accurate command execution. The integration of gesture and voice inputs ensured smooth interaction, even when one modality was temporarily unavailable. The system was further evaluated based on responsiveness and consistency during continuous usage. Gesture-based actions exhibited low latency, enabling near real-time interaction, while command processing remained efficient. The system maintained stable performance over extended periods, indicating its reliability. Additionally, the multimodal approach improved usability, flexibility, and user comfort by reducing dependency on physical input devices.

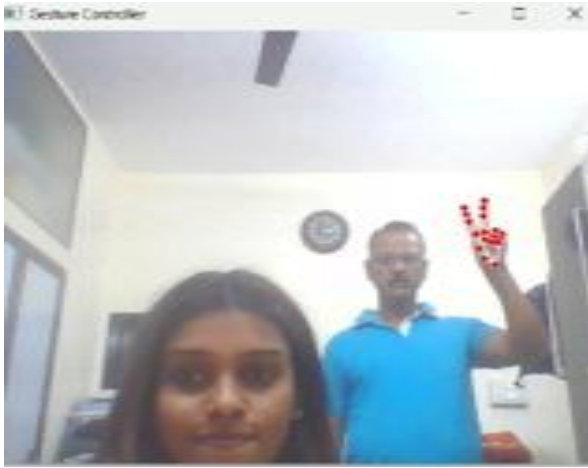


Fig 4: Gesture captured from a distance.

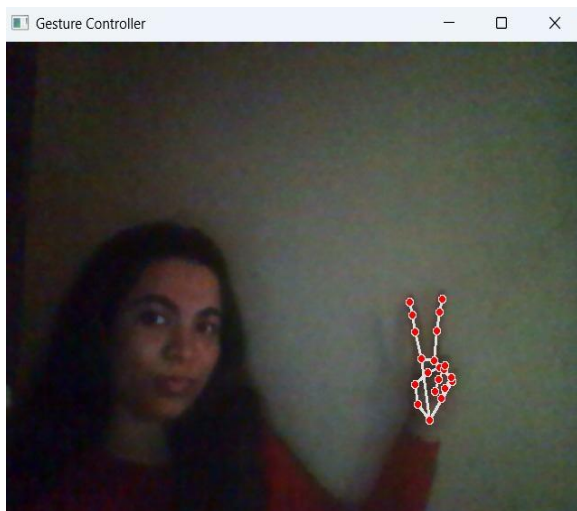


Fig 5: Gesture captured in dim light.

2. Results:

The results demonstrate that the proposed system achieves low-latency and reliable real-time interaction. The gesture recognition module provides accurate cursor control and stable execution of actions, while voice commands enable efficient system-level operations without interrupting gesture input.

The integration of gesture and voice interaction, supported by a coordination mechanism, prevents conflicting actions and ensures organized execution. The system operates effectively on standard hardware with low computational requirements, making it cost-effective and practical.

Although the system performs well under normal lighting conditions, its performance is affected in low-light or noisy

environments, indicating scope for further improvement in robustness.

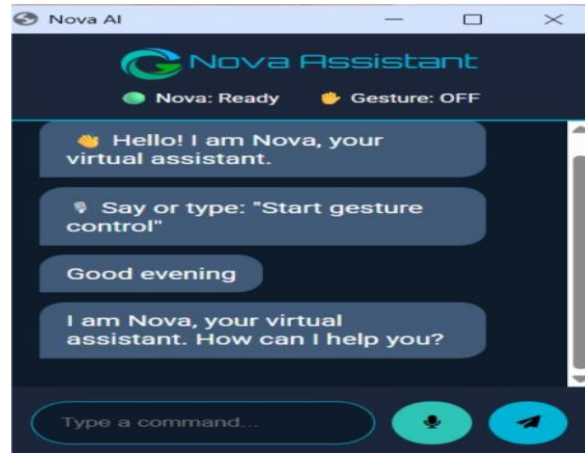


Fig 6: Graphical User Interface of Nova Assistant

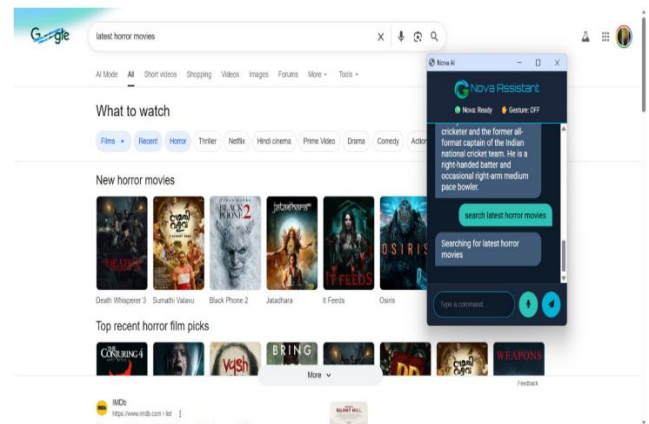
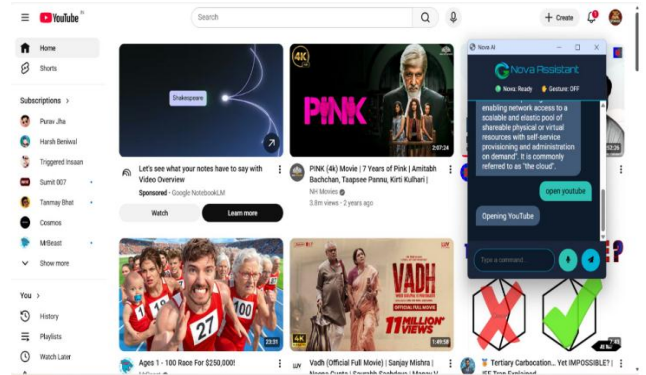


Fig 7: Illustrates the successful execution of voice based commands for web navigation and application control during system testin

VII. LIMITATIONS

There are shortcomings to the proposed approach that provide room for improvement: the current gesture recognition module is constrained to the finite set of predetermined hand gestures that limit the range of possible interactions for the user. Environmental conditions, such as poor lighting or complex background environments, have been reported to affect system performance, as they impact gesture detection accuracy and reliability.

VIII. CONCLUSION

This research presented a virtual mouse using hand gestures with a voice assistant which aimed at multimodal human-computer interaction system that allows full interaction with a computer without physical contact and combines a real-time hand gesture recognition engine with a smart voice-based assistant. The interface uses a gesture recognition engine based on a webcam and a conversational assistant.

Using a centralized coordinator to synchronize gesture and voice input and resolve conflicts, this system has been shown to exhibit stable, low-latency performance, and provide, gesture and voice, input as a reliable, low-cost and accessible alternative to desktop input mechanisms such as mice and keyboards.

In sum, the system has proven to be a solid platform for future clever interaction systems that are able to exploit visual and auditory information in a lightweight manner using off-the-shelf hardware.

Future iterations of the system could potentially be made smarter and more dynamic by using a machine learning based adaptive gesture learning that allows the system to learn user specific gestures and improve the system accuracy in the process. Expanding the voice assistant to support multiple languages and personalized commands would considerably increase its usability for diverse users.

Future extensions of this work could include multi-user gesture tracking for multi-user interfaces, the use of clever personal assistants and smart devices, and the application of the system to IoT-based systems for home automation, assistive technologies and smart work environments. This would improve the robustness, scalability and usability of the system in different application domains.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to the Department of Data Science, Usha Mittal Institute of Technology and SNDT Women's University for providing

the necessary facilities and academic support. The authors also thank their project guide for continuous guidance and encouragement during the development of this work.

REFERENCES

- [1] O.-J. Yaseen et al., "Next-Gen Dynamic Hand Gesture Recognition: Media Pipe, Inception-v3 and LSTM-Based Enhanced Deep Learning Model," *Electronics*, 2024.
- [2] Y. Bawa et al., "Virtual Mouse using Gesture Recognition and Voice Control," *IRE Journals*, 2024.
- [3] K. Kavyasree et al., "Hand Glide: Gesture-Controlled Virtual Mouse with Voice Assistant," *IJRASET*, 2024.
- [4] M. R. Visavarapu et al., "Gesture Control Mouse," *EasyChair Preprint*, 2024.
- [5] K. A. Jacob et al., "Gesture Recognition Based Virtual Mouse Using OpenCV and Python," *AIP Conference Proceedings*, 2024.
- [6] Google, "MediaPipe Hands," *Google Developers*, 2023.
- [7] F. Chollet, "Deep Learning with Python," *Manning Publications*, 2018.
- [8] D. Zhang, "Hand Gesture Recognition Using Computer Vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
- [9] S. Mitra and T. Acharya, "Gesture Recognition: A Survey," *IEEE Transactions on Systems, Man, and Cybernetics*, 2007.
- [10] R. W. Picard, "Affective Computing," *MIT Press*, 1997.
- [11] Kendon, "Gesture: Visible Action as Utterance," *Cambridge University Press*, 2004.
- [12] T. Starner and A. Pentland, "Real-Time American Sign Language Recognition from Video Using Hidden Markov Models," *MIT Media Lab*, 1995.