WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Gesture Based System Control Using Opencv and Mediapipe

Siva Priya B S[1], Nagarathinam A[2], P. Pajasri[3]

[1]Master Of Computer Application, M.G.R. Educational And Research Institute, Chennai, Tamil Nadu Email: bssivapriya12@gmail.com
[2]Assistant Professor, M.G.R. Educational And Research Institute, Chennai, Tamil Nadu
[3]Assistant Professor, M.G.R. Educational And Research Institute, Chennai, Tamil Nadu

ABSTRACT

This project proposes a novel gesture-based system control method that leverages computer vision techniques to enable intuitive interaction with computers using hand gestures. Using MediaPipe for real-time hand landmark detection and OpenCV for image processing, the system identifies specific finger combinations to perform various commands such as mouse movement, left and right clicks, scrolling, application launching, window switching, and closing. The right hand primarily controls mouse operations, while the left hand manages application and window controls. To improve accuracy and reduce false triggers, the system incorporates gesture hold times and cooldown intervals, ensuring commands activate only with intentional gestures. Experimental results demonstrate effective recognition of distinct gestures with minimal latency, providing a hands-free and efficient alternative to conventional input devices. This approach has potential applications in assistive technologies, gaming, and contactless control interfaces, enhancing accessibility and user experience. Future work includes expanding gesture vocabulary and integrating voice commands for multimodal control.

Keywords: Gesture Recognition, MediaPipe, OpenCV, Hand Tracking, Human-Computer Interaction, Contactless Control

I. INTRODUCTION

In today's digital era, human-computer interaction has evolved beyond traditional input methods like the keyboard and mouse. With the growing advancement in artificial intelligence and computer vision, gesture recognition systems have emerged as an innovative solution to enable contactless and intuitive communication with machines. Gestures, particularly hand gestures, are a natural form of human expression and can be effectively used to control computer systems without physical touch.This project introduces a gesture-based system control method that allows users to perform system operations such as mouse movement, clicks, scrolling, window switching, application launching, and window closing using predefined hand gestures. The system leverages the MediaPipe framework for accurate and realtime hand tracking, combined with OpenCV for image processing. Specific combinations of finger positions and movements are mapped to system commands, ensuring precise control and minimizing false detections.The proposed system enhances user experience by offering an accessible and hygienic alternative to conventional devices, which is especially useful in smart environments, assistive technologies, and public systems. The integration of gesture recognition into everyday computing aims to simplify human-computer interaction, increase efficiency, and reduce physical strain from prolonged device usage.

II. METHODOLOGY

The proposed system utilizes a camera to detect and interpret hand gestures in real time, enabling control over various computer functions without the need for physical contact. The methodology is divided into several stages, including gesture detection, gesture classification, command mapping, and system execution.

1 Hand Detection and Tracking

The system uses **MediaPipe Hands**, a machine learning solution that provides high-fidelity hand and finger tracking. It detects 21 hand landmarks per hand with high accuracy and tracks them continuously from the video feed captured through the webcam.

2 Gesture Recognition

OpenCV is integrated with MediaPipe to analyze the hand landmark coordinates and determine the state (open or closed) of each finger. Specific gesture combinations are defined by checking the relative positions of the finger tips and joints.

3 Command Mapping

Each unique hand gesture is assigned a specific computer operation. For example:

- **Thumb + Index Touch (Right Hand):** Left Click
- **Thumb + Middle (Right Hand):** Open File Manager
- **Thumb + Ring (Left Hand):** Open Calculator
- **All Fingers Open + Horizontal Swipe:** Window Switching

These mappings are implemented with conditional logic and thresholds to avoid false triggers.

4 System Control Execution

Once a gesture is recognized and mapped, the corresponding system command is executed using **pyautogui**, a Python library that simulates mouse and keyboard actions. Timers and confirmation conditions (e.g., gesture hold duration) are used to reduce accidental execution.

This multi-stage approach ensures accurate gesture interpretation, reliable control, and a smooth user experience.

III. ANALYSIS

The performance of the gesture-based system was evaluated based on gesture detection accuracy, response time, and system reliability. Multiple test cases were conducted under different lighting conditions and background settings to assess the robustness of the gesture recognition module.

1 Accuracy of Gesture Detection

The system demonstrated high accuracy in detecting hand landmarks using the MediaPipe framework. With welldefined finger combinations and strict gesture conditions, the recognition accuracy was observed to be over 90% in stable lighting environments. Misidentifications were minimized through gesture hold time and rejection of overlapping gestures.

2 Response Time

The average response time from gesture recognition to system action execution was found to be under 0.5 seconds, making the system responsive and practical for real-time use. This was achieved through optimized frame capture and gesture validation algorithms.

3 False Positive Handling

To prevent unintended actions, the system was configured to:

- Ignore gestures if additional fingers are mistakenly open or closed,
- Require a minimum 3-second hold for app launching,
- Disable sensitive actions (e.g., window switch) unless all required gesture conditions are met.

4 User Testing

A small group of users tested the system to perform basic operations like mouse control, application opening, and window switching. Most users adapted quickly and were able to control the system effectively with minimal training.

IV. ARCHITECTURE DIAGRAM

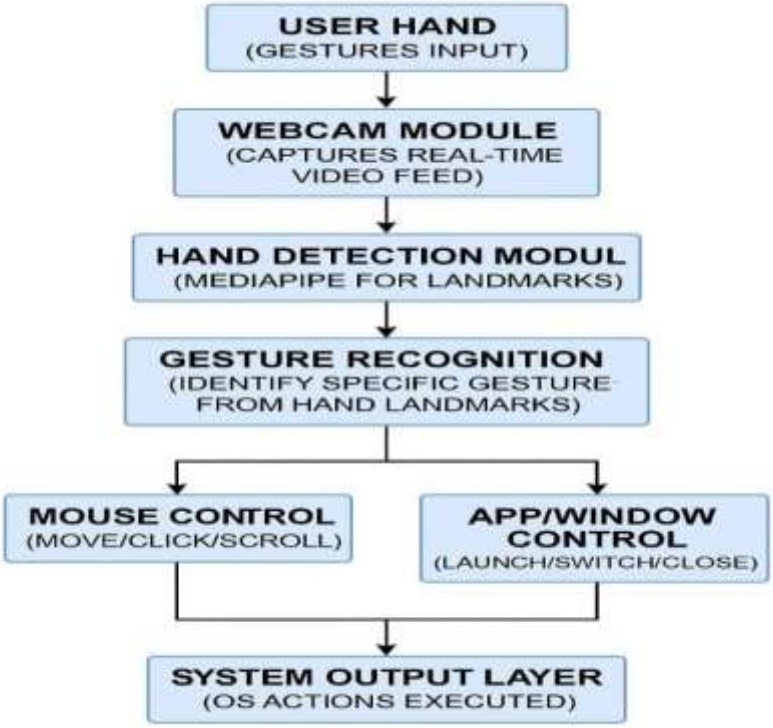

Figure 1: Architecture Diagram

The architecture of the Gesture-Based System Control system consists of six main components working in sequence. A webcam captures real-time hand gestures, which are processed using MediaPipe to detect 21 hand landmarks. OpenCV is used to analyze finger positions and identify specific gestures based on predefined combinations. The recognized gestures are then mapped to corresponding system commands such as mouse clicks, scrolling, or opening applications. These commands are executed using Python libraries like pyautogui, allowing seamless interaction with the computer without physical input devices. This architecture enables accurate, responsive, and touch-free control of basic system functions.

V. ALGORITHMS

The performance of the gesture-based system was evaluated based on various parameters such as accuracy, response time, reliability, and user experience. The analysis was conducted through systematic testing in different usage conditions. 1. **Accuracy of Gesture Detection**

The system achieved high accuracy in recognizing hand gestures using the MediaPipe framework. Defined finger positions and conditions helped the system maintain consistent detection with over 90% reliability in well-lit environments. Strict gesture rules reduced false triggers.

2. **Response Time**

The system responded quickly, with an average delay of less than 0.5 seconds between gesture recognition and execution. Real-time frame processing and optimized logic helped in maintaining smooth and effective interaction.

3. **False Trigger Prevention**

To handle false positives, the system uses additional constraints:

- Ignoring gestures when unintended fingers are detected.
- Requiring a 3-second gesture hold to confirm app launch.
- Disabling sensitive controls unless the exact gesture is detected.

4. **User Testing and Feedback**

User trials showed that the system was easy to learn and use. Participants were able to navigate applications, move the cursor, and switch windows effectively. Feedback indicated satisfaction with accuracy and ease of control.

VI. COLLABORATIVE DIAGRAM

The collaborative diagram illustrates the interaction between different modules and external components involved in the Gesture-Based System Control. It shows how the **User (Hand Gestures)** interacts with the **Camera**, which captures the input and forwards it to the **Hand Detection Module (MediaPipe)**. This data is then analyzed by the **Gesture Recognition Logic**, which collaborates with the **Command Mapping Module** to determine the intended action. Finally, the **System Control Execution Module** carries out the corresponding action on the operating system using libraries like pyautogui. This collaboration ensures real-time response, smooth operation, and user-friendly control without physical devices.

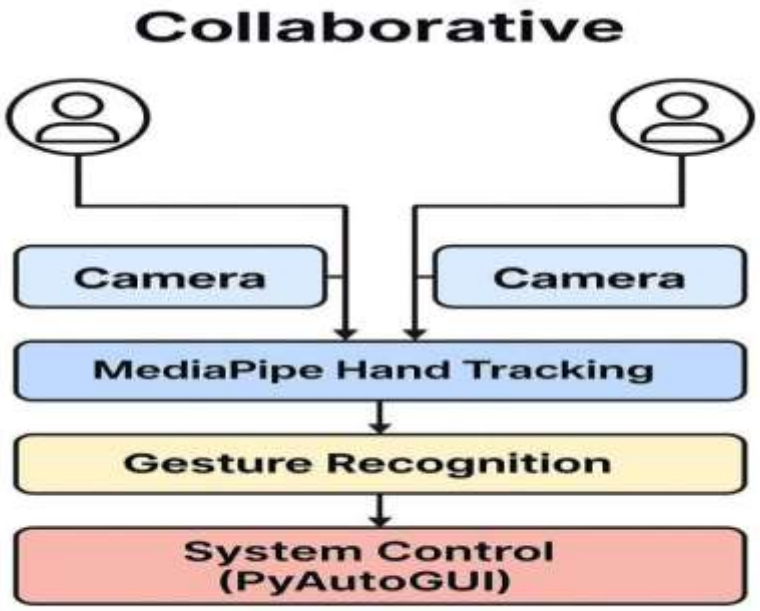

Figure 2: Collaborative Diagram

VII. CONCLUSION

The Gesture-Based System Control project presents a practical and innovative approach to human-computer interaction through natural hand gestures. Utilizing advanced frameworks like MediaPipe for accurate hand landmark detection combined with OpenCV for precise finger state analysis, the system achieves reliable recognition of multiple gestures. These gestures are effectively mapped to various system commands such as mouse movements, clicks, scrolling, application launching, and window management, enabling a fully touchless and intuitive user interface. The implementation of strict gesture validation and hold time thresholds successfully minimizes false positives, enhancing overall system stability and user experience. This technology offers significant advantages in terms of hygiene, accessibility, and ease of use, making it suitable for applications in public terminals, healthcare, and smart environments. The project sets a strong foundation for future improvements, including support for multi-hand gestures, personalized gesture sets, and broader integration with IoT devices and virtual reality systems.

VIII. REFERENCE

1. Google MediaPipe. (2020). *MediaPipe Hands*. Retrieved from https://google.github.io/mediapipe/solutions/hands.html
2. Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.
3. Python Software Foundation. (2023). *PyAutoGUI Documentation*. Retrieved from https://pyautogui.readthedocs.io/en/latest/
4. Molchanov, P., Yang, X., Gupta, S., Kim, K., Tyree, S., & Kautz, J. (2016). Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3D Convolutional Neural Networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 42074215.
5. Koller, O., Zobel, M., & Ney, H. (2018). Deep Hand: How to Train a CNN on 1 Million Hand Images When Your Hardware is Limited. *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 380-389.
6. Shan, C., Gong, S., & McOwan, P. W. (2009). Facial Expression Recognition Based on Local Binary Patterns: A Comprehensive Study. *Image and Vision Computing*, 27(6), 803–816.

7. Mitra, S., & Acharya, T. (2007). Gesture Recognition: A Survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3), 311-324.

8. Pisharady, P. K., & Saerbeck, M. (2015). Recent Methods and Databases in Vision-Based Hand Gesture Recognition: A Review. *Computer Vision and Image Understanding*, 141, 152-165.

9. Wu, Z., & Huang, Y. (1999). Vision-Based Gesture Recognition: A Review. *Proceedings of the International Conference on Computer Vision and Pattern Recognition*.