Virtual Mouse Using Hand Gesture Recognition

PAVAN M, ISIRI BHAT S M, MANASA K, SHREYA A P, AKSHAY M J

Department of Information Science and Engineering, Jawaharlal Nehru New College of Engineering, Shimoga, Karnataka, India

ABSTRACT

The traditional input devices like physical mice, which face accessibility issues, hygiene problems, and ergonomics. They are based on physical contact and mobility, sometimes limiting people with physical impairments or in very hygienic environments. This problem needs an alternative interaction method that is more inclusive and contact-free. All the above problems will be overcome with this proposed approach by integrating a gesture-based interaction system. This method avoids dependency on the physical input device of the system as a webcam is used to identify and interpret the hand movement in real-time. Computer vision and machine learning algorithms deployed in gesture recognition result in high accuracy and efficiency and make it highly responsive and user-friendly. The results of this system will be a touch-free system that will better access users with reduced physical abilities, hygiene in shared environments, and an ergonomically comfortable alternative to the traditional devices. Its applications will range from the most basic control of the computer to its integration into the fields of virtual reality, gaming, and assistive technologies, which are the possible applications of gesture recognition as a transformative step in human-computer interaction.

Keywords: Human-computer interaction, hand gesture recognition, virtual mouse technology, computer vision, MediaPipe, gesture-based control, machine learning, assistive technology.

I. INTRODUCTION

Human Computer Interaction (HCI) is the research of how interaction happens between computers and users and to what extent computers are or are not developed for efficient and seamless interaction with human beings [1]. As Human-Computer Interaction evolved, traditional input devices such as keyboards and mice are gradually being supplemented or replaced by more intuitive and natural interfaces. As technology advances, many aspects of lives are becoming virtualized. For example, speech recognition, it allows spoken language to be understood and translated into text. This innovation holds the potential to serve as a future replacement for traditional keyboards. Similarly, eye-tracking technology, which enables control of the mouse pointer using just our eyes, could eventually take the place of physical mice. These advancements are paving the way for more intuitive and immersive interactions with technology [2]. Among these, gesture-based virtual input systems offer a revolutionary shift by utilizing computer vision and natural hand gestures for interaction. Gestures used for recognition by devices can take forms such as hand images, simplified pixel-based representations, or specific human poses, with the aim of being designed to reduce computational complexity and resource requirements, enabling devices to recognize and process them more efficiently and effectively [2]. These systems allow user computer interaction without relying on physical devices and creating new possibilities for seamless, touch-free control in diverse environments.

Recent developments in gesture recognition technologies leverage cameras to detect and interpret user gestures in real time. This approach finds application in virtual mice, multimedia control, and even robotic systems. While utilizing a remote or a Bluetooth mouse, a few gadgets particularly like the mouse requires battery to drive it, to control a computer, using virtual mouse the client utilizes his/her inborn camera or visual camera and utilizations his/her hand signs to deal with the PC mouse works out [3]. Using frameworks like OpenCV, MediaPipe, and Python, researchers have designed virtual input solutions that track hand movements to perform essential functions such as mouse navigation, clicking, and other functions, fostering a more engaging and accessible user experience.

Gesture-based virtual mouse not only simplify interaction but also offer accessibility for physically limited people and enable efficient device control in critical settings such as surgeries or hazardous work environments. These solutions highlight the importance of non-contact interaction, especially in a post-pandemic world, where minimizing touch has become a priority. As technology advances, the integration of such interfaces in VR, AR, and daily computing holds tremendous potential to redefine human-computer interactions.

II. RELATED WORKS

In AI Virtual Mouse Using Hand Gesture Recognition [1] and Artificial Intelligence Virtual Mouse using Hand Gesture [7] employs a webcam to capture live video input using Open CV, which is processed using the MediaPipe framework to detect hand landmarks. Using Some techniques interpret these gestures to enable virtual mouse control. In Human Computer Interaction Using Hand Gesture [5] and A Survey on Hand Gesture Recognition for Simple Mouse Control [6] author Sharma R.P and at all uses detection of skin color to isolate the hand from the background in video feeds. It extracts features like shape, size, and position for gesture recognition but Skin-color detection is sensitive to lighting and background which reduces accuracy and preprocessing steps like noise removal and face recognition. Utilization of OpenCV to process images and employs HSV-based color detection to identify regions of interest, like colored caps or markers on fingertips or colored markers on hands are discussed in Virtual Mouse Implementation Using OpenCV [3] and Design and Development of Hand Gesture Based Virtual Mouse [9]. Hand-Mouse Interface Using Virtual Monitor Concept for Natural Interaction [2] uses Kinect to track hand movements and map them onto a virtual monitor space aligned with a physical monitor. It is effective but relies on costly Kinect hardware. Noraini

Mohamed and at all reviews as 98 articles, summarizing challenges in gesture recognition, including environmental factors and gesture representation but it only focuses on controlled environment neglecting the real-world applicability in A Review of the Hand Gesture Recognition System: Current Progress and Future Directions [4]**.** In Vision-Based Hand Pose Estimation: A Review [10] highlights the complexities in hand pose estimation, such as variability in hand shapes and movements, occlusions, and lighting conditions. It discusses the computational challenges of model-based methods. It faces the challenge of difference in shape and size of hand. Hand Gesture Recognition Based on Computer Vision: A Review of Techniques [8] discusses about segmentation of hand region from the background using techniques like background subtraction and skin segmentation. Contour extraction and fingertip position analysis map gestures to computer actions.

III. METHODOLOGY

This research focuses on developing an intuitive and efficient gesture-based control system for enhancing user and computer interaction. The methodology involves several key steps, each aimed at ensuring robust gesture detection and seamless system control.

1. Initialization of Libraries and Webcam

The implementation begins by importing essential libraries such as OpenCV for video processing and MediaPipe for detecting hand landmarks. The webcam is initialized using OpenCV's Video Capture to enable real-time video capture. For detecting and tracking hand landmarks efficiently MediaPipe Hands is set up.

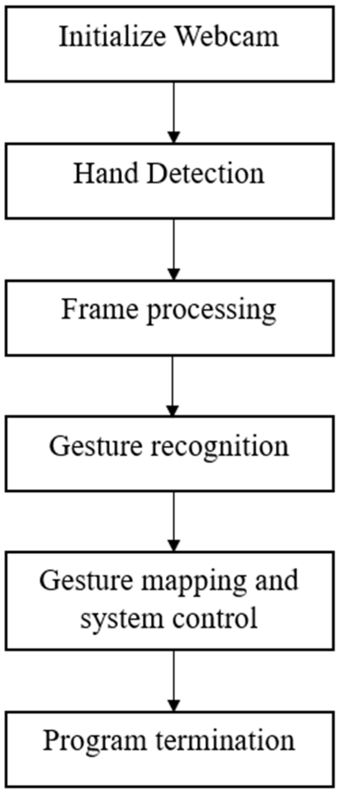


Figure 1: Block diagram for a mouse control based on hand gesture recognition.

2. Frame Processing and Detection of Hand Landmark

Captured frames from the webcam are preprocessed by converting them into an RGB format suitable for MediaPipe. The MediaPipe Hands module analyzes each frame and detect if there is a presence of hand and identify key landmarks such as fingertips, knuckles, and the wrist. These landmarks are extracted and serve as the foundation for gesture recognition.

Here's an overview of what such an image would depict:

- 21 Key Points (Landmarks): Each hand has 21 landmarks detected by Mediapipe, which include points on the tips, joints, and base of the fingers and the palm.
- Landmark Positions (from base to tip):
    - Thumb: 4 landmarks
    - Index Finger: 4 landmarks
    - Middle Finger: 4 landmarks
    - Ring Finger: 4 landmarks
    - Pinky Finger: 4 landmarks
    - Palm: 1 landmark (center)

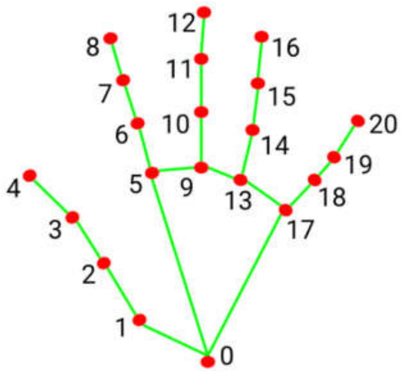
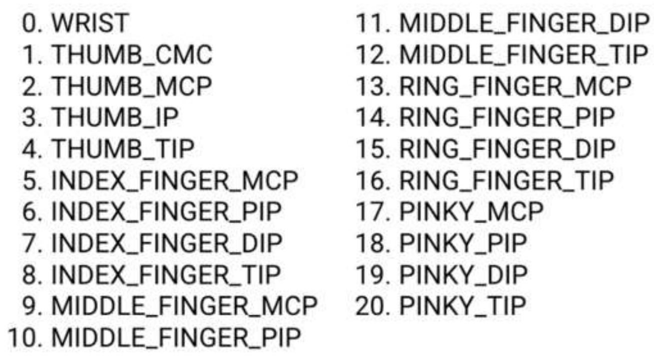


Figure 2: Hand landmarks detected by Mediapipe

3. Gesture Recognition Based on Hand Landmarks

Using the detected landmarks, distances between specific points on the hand are calculated to determine finger positions. Gestures like pinching, pointing are identified based on these positions. Detected gestures are pinch, index and pinky, v gesture (victory), mid gesture (middle finger), index finger gesture, two finger closed gesture.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 3: Hand landmarks points

4. Gesture mapping and system control

Predefined gestures are mapped to system actions such as clicks, navigating, or adjusting system settings.

Identified gestures are mapped to specific system actions:

- **Pinch Gesture:** Controls system brightness or volume. When the index finger and thumb come close together (pinching), the system interprets it as a command to adjust brightness or volume. If the movement is horizontal then it is brightness control, if the movement is vertical then it is volume control.
- **Index and Pinky gesture:** Minimizes the active window. This gesture is identified when the index finger and pinky finger is extended, signaling the software to diminish the current window.
- **First Three Finger gesture:** Maximizes the active window. This gesture is interpreted when the index, mid and ring fingers are extended

upwards, signaling the system to maximize the current window.

- **V Gesture (Victory):** Moves the cursor. When the middle finger and index finger are straightened in a V shape, the system moves the cursor to the specified position.
- **Mid Gesture (Middle Finger):** Single click action. This gesture is mapped for performing left click at the cursor's position.
- **Index Finger Gesture:** Right-click action. When the index finger is extended, it triggers a right-click action.
- **Two Finger Closed Gesture:** Double-click action. When the middle and index fingers are positioned close together, the system interprets it as a double-click command.
- **Pinky Gesture:** Closes the current window. When the pinky finger is extended, the system uses the alt + f4 shortcut to close the active window.

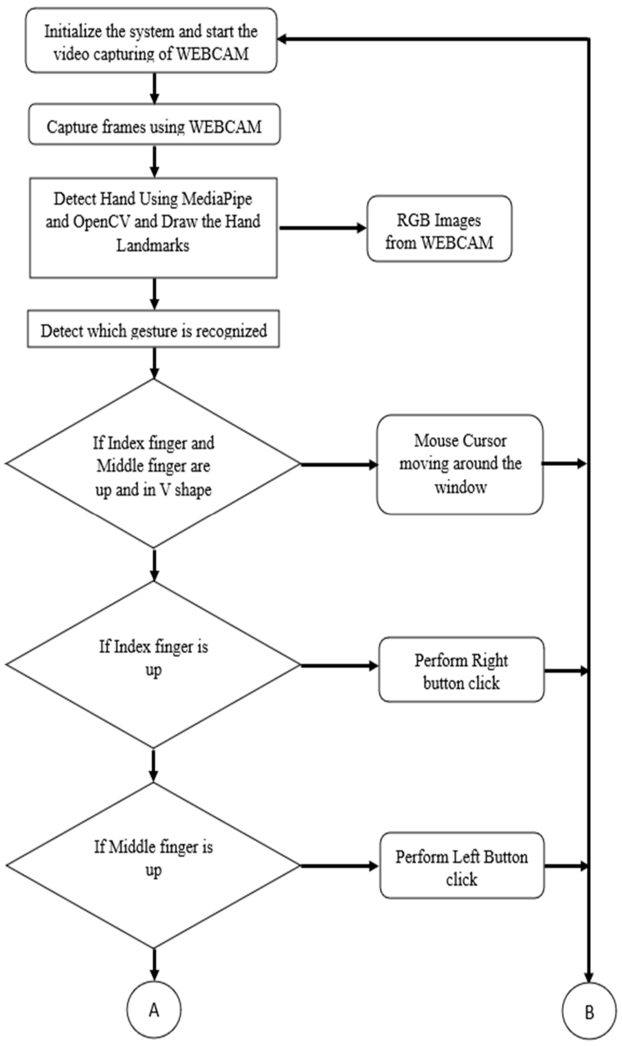
Each mapped action is executed using PyAutoGUI, which simulates keyboard shortcuts and mouse movements to enable smooth and efficient computer control.

5. Program Termination

The system continuously processes frames until a specific key, such as 'q', is pressed. At this point, the webcam feed is released, and all display windows are closed, ensuring a clean program exit. This step-by-step methodology integrates gesture recognition and computer vision to construct an intuitive and responsive virtual mouse system.

IV. ALGORITHM

The gesture-controlled mouse uses a complex algorithm, to recognize and track the hands of users and their finger movements and translate them into cursor movements. The algorithm relies on recognition of hand gesture technology, to recognize the user's hand and track their hand movements.


Initialize the system and start the video capturing of WEBCAM
Capture frames using WEBCAM
Detect Hand Using MediaPipe and OpenCV and Draw the Hand Landmarks
RGB Images from WEBCAM
Detect which gesture is recognized
If Index finger and Middle finger are up and in V shape
Mouse Cursor moving around the window
If Index finger is up
Perform Right button click
If Middle finger is up
Perform Left Button click
A
B

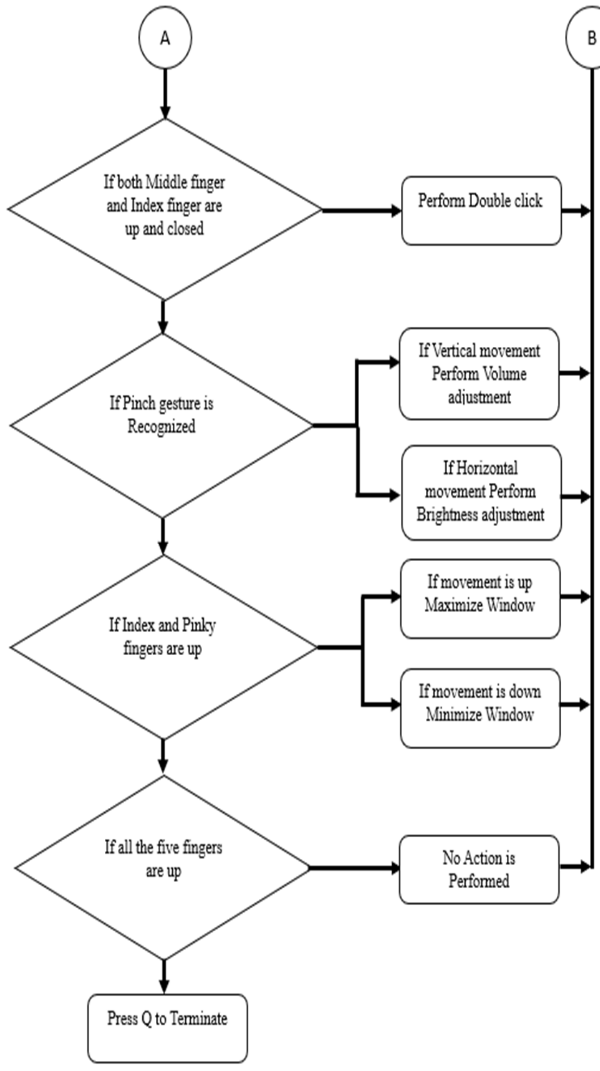

A
B
If both Middle finger and Index finger are up and closed
Perform Double click
If Pinch gesture is Recognized
If Vertical movement Perform Volume adjustment
If Horizontal movement Perform Brightness adjustment
If Index and Pinky fingers are up
If movement is up Maximize Window
If movement is down Minimize Window
If all the five fingers are up
No Action is Performed
Press Q to Terminate

Figure 4: Flow chart of recognition of hand gestures for virtual mouse

Steps followed:

Step 1. Import required libraries.
Step 2. Open webcam using cv2.VideoCapture.
Step 3. Initialize MediaPipe hands for detecting hand landmarks.
Step 4. Continuously capture frames from the webcam feed.
Step 5. Process the frame for hand sensing using MediaPipe.
Step 6. Detect landmarks in the captured frame and extract them.
Step 7. Measure distances between specific hand landmarks to determine finger states.
Step 8. Use predefined gestures based on finger positions (e.g., pinch, index).
Step 9. Use hand position to move the mouse on the screen.
Step 10. Use pinch gesture to control system brightness and volume.
Step 11. Change system brightness or volume based on pinch gesture direction.
Step 12. Draw hand landmarks on the frame using mediapipe.
Step 13. Display the processed frame in a window.
Step 14. Exit the loop when 'q' key is pressed.

OpenCV (Open-Source Computer Vision Library): OpenCV is a versatile and widely adopted open-source library for computer vision, commonly utilized for image and video processing tasks. In this paper, OpenCV is incorporated to access the webcam feed, process video frames, and display the processed output. It also helps with operations like frame conversion (e.g., BGR to RGB) and drawing visual overlays, such as hand landmarks, on the video feed.

MediaPipe: MediaPipe is a cross-platform framework created by Google that delivers effective solutions for building machine learning pipelines. Specifically, the MediaPipe Hands module is adopted to recognize and track hand landmarks. It provides pre-trained models that recognize 21 key hand points in real-time, making it ideal for gesture recognition and hand pose estimation tasks.

PyAutoGUI: PyAutoGUI is a library in Python that facilitates programmatic control of the keyboard and mouse. In this paper, it is used to carry out system actions for example moving the mouse cursor, clicking, and other user interface interactions triggered by detection of specific hand gestures.

Pycaw: Pycaw is a library in Python that allows programmatic control of Windows audio devices with the help of Core Audio APIs. It allows functionalities like adjusting system volume, muting the audio, or retrieving active audio sessions. This has several classes and interfaces that enables audio control. Among these AudioUtilities class fetches audio endpoints, and the IAudioEndpointVolume interface adjusts the volume. It seamlessly integrates the gesture-based interactions for volume control.

Screen Brightness Control: This library enables adjustment of screen brightness programmatically on most systems. It supports multiple displays and provides functions to retrieve current brightness level and modify it. In this study, the recognized hand gestures are mapped to raise or reduce the brightness level using the set_brightness () and get_brightness () functions.

V. CONCLUSION

This research successfully demonstrates the advancement of an intuitive and efficient gesture-based mouse control system for improving interaction of users with computers. By combining Mediapipe for hand landmark detection in real time and PyAutoGUI for controlling mouse actions, the system enables users to perform various computer operations by performing natural hand gestures.

The system successfully maps specified hand gestures to corresponding system actions, such as adjusting system brightness or volume with a pinch gesture, closing current window using pinky gesture respectively, and perform other mouse actions with finger gestures. The system ensures smooth and accurate operation and guarantees that each recognized gesture is execute only per cycle offering reliable and user-friendly experience.

Continuous real-time processing along with immediate feedback mechanism, make sure that the system responds quickly to user inputs, enhancing the overall efficiency of interaction. The optimization and refinement of the system based on user feedback contribute to the system's reliability and ease of use.

REFERENCES

[1]. R. Suriya and V. Vijayachamundeeswari, "A survey on hand gesture recognition for simple mouse control," *International Conference on Information Communication and Embedded Systems (ICICES2014)*, Chennai, India, pp.1-52014.

[2]. K. S. Varun, I. Puneeth and T. P. Jacob, "Virtual Mouse Implementation using Open CV," *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, Tirunelveli, India, 2019, pp. 435-438.

[3]. Guha, J., Kumari, S. and Verma, S.K*, "AI Virtual Mouse Using Hand Gesture Recognition"* April 2022 International Journal for Research in Applied Science and Engineering Technology 10(4):3070-3076.

[4]. Sharma, R.P. and Verma, G.K*, "Human computer interaction using hand gesture",* Procedia Computer Science,2015.

[5]. N. Mohamed, M. B. Mustafa and N. Jomhari, "A Review of the Hand Gesture Recognition System: Current Progress and Future Directions," in *IEEE Access*, vol. 9, pp. 157422-157436, 2021.

[6]. C. Jeon, O. -J. Kwon, D. Shin and D. Shin, "Hand-Mouse Interface Using Virtual Monitor Concept for Natural Interaction," in *IEEE Access*, vol. 5, pp. 25181-25188, 2017.

[7]. V. S. Sangtani, A. Porwal, A. Kumar, A. Sharma, and A. Kaushik, "Artificial Intelligence Virtual Mouse using Hand Gesture", *IJMDES*, vol. 2, no. 5, pp. 26–30, May 2023.

[8]. Oudah, Munir, Ali Al-Naji, and Javaan Chahl. 2020. "Hand Gesture Recognition Based on Computer Vision: A Review of Techniques" *Journal of Imaging* 6, no. 8: 73.

[9]. K. H. Shibly, S. Kumar Dey, M. A. Islam and S. Iftekhar Showrav, "Design and Development of Hand Gesture Based Virtual Mouse," *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, pp. 1-5, Dhaka, Bangladesh, 2019.

[10]. Ali Erol, George Bebis, Mircea Nicolescu, Richard D. Boyle, Xander Twombly, "Vision-based hand pose estimation: A review", *Computer Vision and Image Understanding* 108(1-2):52-73, October 2007.