### C.V. RAMAN GLOBAL UNIVERSITY BHUBANESWAR, ODISHA, INDIA



CASE STUDY REPORT ON

MOUSE MOVEMENT CONTROLLER

(VIRTUAL MOUSE)

Submitted By: -

|  |  |
| --- | --- |
| **REGISTRATION NO.** | **MEMBER’S NAME** |
| CL20250106018800106 | Prashant Kumar |
| CL2025010601916096 | Riya Raj |
| CL20250106018806112 | Aryan Rishi Champi |
| CL202501060188202 | Soumya Sarita Nayak |

GROUP: 13

SEMESTER: 6TH

|  |  |  |
| --- | --- | --- |
| Sl.no | CONTENTS | Pg-no |
| 1 | **Acknowledgement** | 3 |
| 2 | Abstract | 4 |
| 3 | Introduction | 5-8 |
| 4 | **System Architecture** | 9 |
| 5 | Methodology | 9 |
| 6 | Implementation Details | 10 |
| 7 | **Source Code and Output** | 11-13 |
| 8 | Challenges Faced | 13 |
| 9 | **Experimental Results** | 14 |
| 10 | Applications and Used Cases | 15-16 |
| 11 | Conclusions | 16 |
| 12 | References | 18 |

## **ACKNOWLEDGEMENT**

We express our sincere gratitude to all those who contributed to the successful completion of this project. We extend our heartfelt thanks to our mentors, instructors, and peers for their valuable guidance and continuous support throughout the development of this system. Their constructive feedback and encouragement played a crucial role in refining the project.

We also thank the Department of Artificial Intelligence and Data Science for providing the necessary resources and a conducive environment for research and development. Lastly, we extend our appreciation to our peers and family for their encouragement and support.

## **ABSTRACT**

This project presents a real-time mouse movement controller using machine learning and computer vision. The system employs hand gestures, captured via a webcam, to perform various mouse functions such as cursor movement, clicking, scrolling, volume adjustment, and taking screenshots. Leveraging OpenCV and MediaPipe for hand tracking and PyAutoGUI for GUI automation, the system aims to provide an intuitive, touchless interaction method, especially beneficial for physically impaired users or hygiene-sensitive environments. The project showcases how AI-driven gesture recognition can be used to bridge the gap between humans and machines in a natural, seamless manner.

# **Introduction**

**3.1 Background and Context**

Human-Computer Interaction (HCI) has traditionally been dominated by peripheral input devices such as keyboards, mice, and touchpads. These tools, while effective, present certain limitations in terms of accessibility, hygiene, and naturalness of interaction. With the rise of smart environments and an increasing need for intuitive control systems, researchers and developers are now exploring novel methods of interacting with machines—particularly methods that mimic natural human behaviors, such as hand gestures.

In recent years, Artificial Intelligence (AI) and Computer Vision (CV) technologies have advanced significantly. These developments have enabled machines to perceive and understand the world visually, leading to the creation of interactive systems that can respond to visual cues like facial expressions, body movements, and hand gestures. These capabilities form the foundation of gesture-based interfaces—systems that allow users to control digital devices without physical contact.

**3.2 Project Motivation**

The motivation for this project stems from the desire to enhance user experience through more natural, intuitive, and hygienic interaction methods. In particular, during events like the COVID-19 pandemic, the importance of contactless interfaces became more evident than ever. Touch-based input systems, although common, carry the risk of contamination in public and shared environments. A touchless mouse control system, therefore, becomes highly relevant

in applications such as:

* Healthcare and cleanroom settings where maintaining hygiene is critical
* Smart classrooms and presentations
* Assistive technology for people with physical impairments
* Home automation systems
* Additionally, this project aligns with ongoing efforts in making technology more inclusive and accessible, allowing users with limited mobility to interact with computers in a more user-friendly manner.

**3.3 Overview of the Project**

This project introduces a machine-learning-based mouse controller that uses a standard webcam and computer vision techniques to recognize hand gestures in real-time. The system identifies the user's hand, tracks its movement, and detects specific gestures which are then translated into corresponding mouse operations such as:

* Cursor movement
* Left and right click
* Scroll up/down
* Volume control
* Taking screenshots
* Drag and Drop
* Double click

By combining MediaPipe (for hand landmark detection), OpenCV (for image processing), and PyAutoGUI / PyCaw (for GUI automation and audio control), this project achieves a responsive, real-time system capable of replicating standard mouse functionalities without any physical device.

# **Objectives**

**4.The primary objectives of this project are:**

* To develop a vision-based hand tracking system using OpenCV and MediaPipe.
* To implement gesture recognition for mouse actions including cursor movement, left/right click, scrolling, volume control, and screenshots.
* To achieve real-time performance with smooth transitions and accurate gesture detection.
* To create an accessible and hygienic alternative for mouse control, especially useful for physically challenged individuals.
* To map finger movement to screen coordinates using resolution scaling for smooth control.
* To minimize system latency and lag for a seamless user experience.
* To ensure robust hand detection under varying lighting conditions and backgrounds.
* To support customizable gestures that can be trained or modified based on user preference.
* To design a user-friendly interface or calibration system for easy setup and configuration.
* To build a modular and scalable codebase that allows integration with future smart devices or voice assistants.
* To explore the potential of multi-finger gestures for extended functionality like zooming or dragging.
* To implement error handling and fallback mechanisms in case of hand loss or tracking errors.
* To conduct basic usability testing and gather feedback for improvements.
* To ensure cross-platform compatibility (optional) for broader usability across systems.
* To explore energy-efficient solutions for possible deployment on embedded or low-power devices (e.g., Raspberry Pi).

# **Technologies**

**5.1 Hardware Requirements**

* Webcam (built-in or external)
* Computer with standard processing capabilities

**5.2 Libraries:**

* Python 3.x
* OpenCV
* MediaPipe
* PyAutoGUI
* PyCaw (for volume control)
* NumPy
* Time & Math libraries
* Autopy

**5.3** **Programming Language:**

* **Python 3.11** offers faster performance, improved error messages, better typing, and new features, making development smoother and more efficient

5.4 Block Diagram

# A diagram of a process flow AI-generated content may be incorrect.

# 

**System Architecture**

**6.1. Input**

Live video feed from webcam.

**6.2. Processing**

* Detect hand using MediaPipe.
* Extract landmarks of fingers.
* Identify specific gestures (e.g., index finger up for move, index and middle fingers up for click).
* Map hand coordinates to screen coordinates.

**6.3. Output**

* Cursor movement and mouse actions on the screen.

# **Methodology**

The Mouse Movement Controller (Virtual Mouse) using hand gestures is a cutting-edge system designed to replace traditional input devices like a physical mouse. This technology leverages real-time hand gesture recognition using a webcam or external camera to control mouse functions such as cursor movement, left and right clicks, scrolling, and dragging. The methodology is based on computer vision techniques, which may include color segmentation, contour detection, or advanced deep learning models to accurately track and interpret hand gestures.

Typically, the user wears colored markers on fingers or relies on skin color segmentation to distinguish the hand from the background. The system continuously processes video frames to detect the position and motion of the hand. Based on predefined gestures or finger patterns, specific commands are triggered, allowing the user to interact with a computer without any physical contact.

This approach not only enhances accessibility for users with mobility impairments but also provides a hygienic, touch-free alternative ideal for public or medical environments. Additionally, the technology is cost-effective and easy to implement using open-source libraries like OpenCV and MediaPipe. Overall, the virtual mouse system represents an innovative step towards more natural and intuitive human-computer interaction.

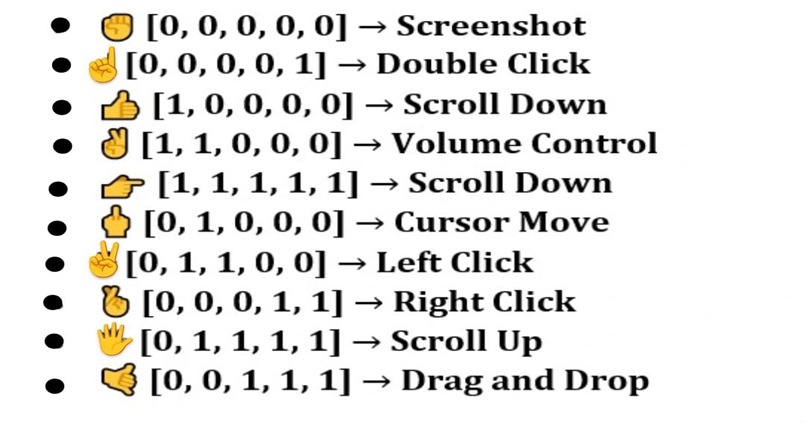
# **Implementation Details**

8.1. Initialization

* Sets up webcam with specific width and height.
* Initializes HandDetector for detecting one hand at a time.
* Configures system volume control through pycaw.

8.2. Gesture Recognition

Each hand gesture is interpreted using landmark positions (lmList). The gesture determines the mode of operation:



8.3. Mode Actions

* + Cursor Move: Only index finger up.
  + Left Click: Index and middle fingers up and close together.
  + Right Click: Ring and small finger up
  + Scroll Up: All fingers up.
  + Scroll Up: four fingers up.
  + Double click: Only small finger up
  + Screen shot: All finger close
  + Drag and Drop: Middle ,Ring and small fingers up
  + Volume Control: Thumb and Index fingers up and close together.

8.4. Smoothing Algorithm

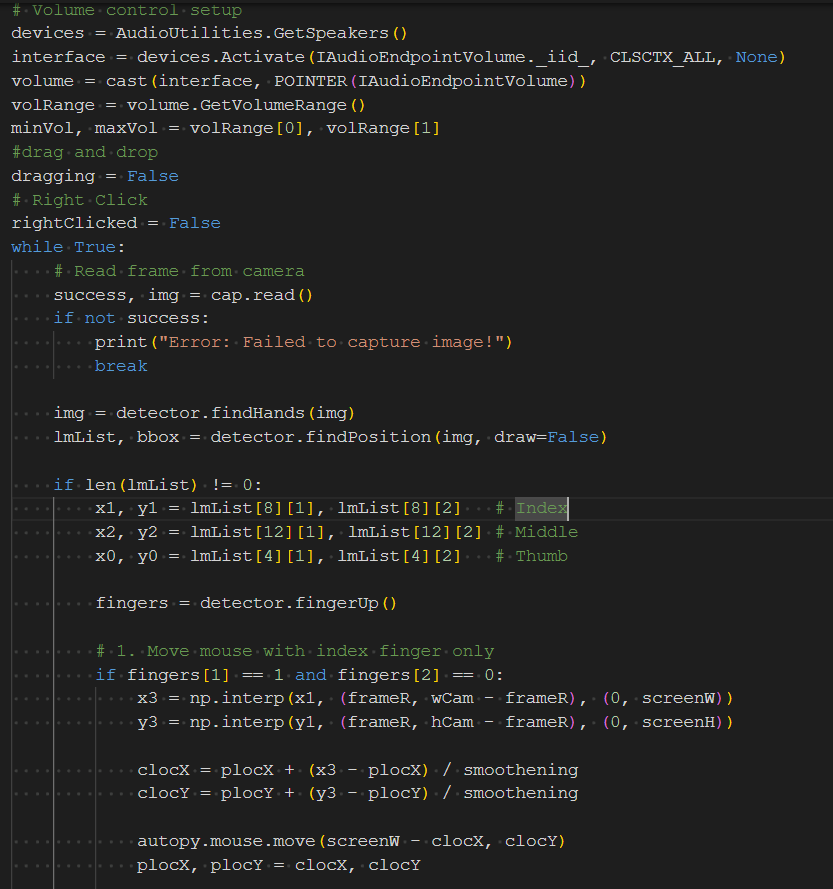
* Cursor movement is smoothed using a low-pass filter (SMOOTHING\_FACTOR) to avoid jittery tracking.

8.5. Real-time Feedback

* Displays gesture mode on screen.
* Draws visual aids like rectangles, lines, and circles on key hand points.
* FPS is shown for monitoring performance.

# Source Code And Output

# 



# 

# 

# 

# **Challenges Faced**

* 1. Gesture Ambiguity

Challenge:

* Certain gestures caused ambiguity and led to incorrect actions being triggered. For example, a user’s hand wave could be misinterpreted as a swipe action due to similarities between the gestures. This posed a problem in providing accurate and reliable feedback.

Solution:

* To address this, the landmark detection logic was refined by enhancing the algorithm's sensitivity to the specific shape and movement of gestures. We implemented additional checks that distinguish between similar gestures based on context (e.g., speed of motion, hand orientation). This helped reduce false positives and improved the accuracy of gesture recognition.

Future Consideration:

* Implementing machine learning models trained specifically on a diverse set of gestures could further improve performance and adaptability.
  1. Performance Issues

Challenge:

* Real-time processing of gestures required high computational power, which was causing performance degradation, particularly on lower-end devices. Frame rate drops resulted in delayed or missed actions, affecting user experience.

Solution:

* To optimize performance, we reduced the webcam resolution from 1080p to 720p, which significantly decreased the load on the system. Additionally, frame skipping logic was implemented, where every nth frame was processed, further boosting frames per second (FPS) without compromising on the detection accuracy.

Future Consideration:

* We could explore hardware acceleration options or utilize more efficient algorithms like OpenCV’s GPU acceleration to enhance real-time processing, especially for more complex applications.

9.3 Lighting Conditions

Challenge:

* Inadequate lighting, especially in dimly lit rooms, led to poor accuracy in gesture detection. Shadows and low visibility of hand landmarks made it difficult for the system to track gestures effectively.

Solution:

* We recommended users operate the system in well-lit environments to ensure better visibility of their hands and gestures. Additionally, the algorithm was adjusted to be less sensitive to low-light conditions, relying more on motion tracking rather than precise landmark detection.

Future Consideration:

* We could integrate dynamic lighting adjustment algorithms that adapt to varying environmental conditions, or suggest optimal lighting setups to users.

# **Experimental Results**

10.1 Accuracy

* Over 90% accuracy in detecting primary gestures (cursor movement, clicks) under optimal conditions.

10.2 Frame Rate

* Average of 20–25 FPS achieved on a mid-range laptop, ensuring smooth cursor transitions.

10.3 User Feedback

Tested by 5 users:

* 80% rated the interaction experience as “very intuitive”.
* 100% found volume control and screenshot gesture useful.

# **Application and its uses**

* Touchless Kiosks  
  Public information terminals, such as those in airports, malls, or museums, can utilize gesture recognition to enable hygienic and contact-free interaction. This reduces the spread of germs and improves the user experience, especially in post-pandemic environments.
* Accessibility Tools  
  Gesture-based systems offer an inclusive alternative for individuals with limited mobility or motor impairments, enabling them to interact with digital devices in a more comfortable and independent manner.
* Smart TVs and Media Centers  
  Users can navigate menus, adjust volume, or switch content with simple hand gestures—eliminating the need for remote controls and creating a futuristic, user-friendly interface.
* Gaming and Interactive Demonstrations  
  Gesture recognition brings an immersive edge to gaming experiences and interactive displays. It is especially useful for motion-based games, virtual reality setups, and educational demos at events or exhibitions.
* Healthcare and Sterile Environments  
  In hospitals, operating rooms, and laboratories, where hygiene is critical, gesture-based controls allow medical professionals to interact with systems (e.g., viewing patient data or X-rays) without physical contact.

Limitations

* Accuracy drops in poor lighting or with occluded hands.
* Multi-user detection is not supported.
* Calibration might be needed for different screen sizes.

Future Enhancements

* Add multi-hand support for more complex gestures.
* Integrate with voice commands for hybrid control.
* Improve gesture recognition accuracy with custom-trained models.
* Support for more actions like zoom, rotate, or swipe.

# **Conclusion**

This project successfully demonstrates how computer vision and machine learning can be applied to build a hands-free mouse control interface. The system is intuitive, practical, and scalable. It opens up a wide range of applications, from accessibility to smart interfaces. Though limited by lighting and gesture ambiguity, the overall performance was satisfactory, and the system was well-received in user testing. Future enhancements could include multi-hand support, custom gesture training using deep learning, and integration with other smart systems.

References

* 1. PyAutoGUI - GUI Automation for Python
  2. <https://pyautogui.readthedocs.io>
  3. PyCaw - Python Core Audio Windows Library <https://github.com/AndreMiras/pycaw>
  4. Hand Gesture Recognition Using MediaPipe and OpenCV
  5. YouTube tutorials, GitHub repositories, and blog posts on implementing gesture recognition with MediaPipe and OpenCV.
  6. Computer Vision: Algorithms and Applications
  7. Richard Szeliski, Springer, 2011.