WWW.IJRPR.COM

International Journal of Research Publication and Reviews

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

Virtual mouse using hand gestures

[1] Sameer Sahu, [2] Shubham Patel, [3] Yashu Baghel, [4] Vikram Sahu

[1 2 3 4] Department of Computer Science and Engineering, Shri Shankaracharya Technical Campus, Bhilai(C.G)

ABSTRACT:

In this study, a virtual mouse system that uses hand motions recorded by a webcam is shown. To identify and decipher hand landmarks, the system makes use of computer vision techniques, including Python's OpenCV and MediaPipe packages. This removes the need for conventional input devices and offers a touchless, accessible substitute that is especially helpful for individuals with physical limitations or in sterile settings.

Keywords: Virtual Mouse, Hand Gesture Recognition, OpenCV, MediaPipe, Python, Computer Vision, Human- Computer Interaction.

INTRODUCTION

Touchless systems have drawn attention due to their accessibility and ease of use as human-computer interaction (HCI) has grown. Conventional input devices, such as keyboards and mouse, have drawbacks in terms of physical accessibility and hygiene. An easy way to communicate with digital systems is through gesture-based technologies. In order to carry out fundamental mouse activities, this paper suggests a virtual mouse that uses Python, OpenCV, and MediaPipe to recognize hand gestures in real time.

LITRATURE SURVEY

Over the past 20 years, there has been a considerable evolution in the development of gesture-based human-computer interaction (HCI) systems. Particularly, hand gesture recognition has drawn a lot of interest since it offers a natural and straightforward way to communicate with machines without making physical touch. Sensor-based systems and vision-based systems are the two main categories into which the literature on gesture recognition and virtual input devices can be generally divided.

Sensor-Based Gesture Recognition

In order to track motion, sensor-based systems use technology such depth cameras, gyroscopes, accelerometers, and infrared sensors. Prominent examples include gadgets like the Leap Motion controller and Microsoft Kinect. These devices are very good at tracking motion and offer precise 3D positional data.

For instance, the Leap Motion gadget uses infrared sensors to precisely follow the movements of the hands and fingers. Nevertheless, sensor-based methods are less portable, frequently expensive, and require extra technology. For gesture recognition, studies like [Molchanov et al., 2015] investigated deep learning techniques in combination with 3D sensors; the results showed good accuracy, but at the expense of more complexity and hardware dependence.

Vision-Based Gesture Recognition

Vision-based systems monitor and analyze hand motions using ordinary RGB or depth cameras. The widespread use of cameras and the accessibility of open-source computer vision libraries make these techniques more widely available. The accuracy and resilience of vision-based systems have been significantly enhanced by the application of machine learning and deep learning approaches.

The difficulties of vision-based hand gesture recognition, such as occlusion, background variation, and computational complexity, were covered in a 2007 study by Erol et al. Despite these difficulties, feature extraction and classification accuracy have significantly increased thanks to developments in computer vision, especially the advent of convolutional neural networks (CNNs).

MediaPipe and Real-Time Landmark Detection

Google's launch of MediaPipe marked a significant breakthrough in real-time hand tracking. A lightweight and effective system called MediaPipe Hands employs machine learning to identify 21 hand landmarks from a single RGB video stream in real time. In contrast to previous systems that used special backdrops or color gloves, MediaPipe works effectively without the requirement for specialized equipment in a range of settings. Research using MediaPipe, such Zhang et al. (2020), showed how useful it is for real-time applications like gesture-controlled interfaces and sign language

translation. It is a well-liked option for both commercial and academic applications because of its cross-platform compatibility and connection with platforms like OpenCV.

Gesture Recognition for Virtual Input Devices

A number of studies have suggested gesture-based virtual mouse systems. For example, Shah et al. (2019) used color segmentation to create a hand gesture detection system to control mouse activities. Nevertheless, these devices frequently have real-time performance issues and are constrained by lighting conditions. Python's PyAutoGUI has simplified the process of emulating keyboard and mouse movements, allowing for the quick development of gesture-controlled systems.
More recently, dynamic gesture recognition has been investigated using neural networks and transfer learning, allowing for more intricate interactions than just clicks or movements. Large datasets and intensive training are necessary for these techniques, though.

METHODOLOGY

It usually takes a mix of machine learning algorithms and computer vision techniques to create a virtual mouse using hand motions. Here are some common algorithms and a broad methodology:

1. Hand Recognition:

Identify and follow the user's hand in the video stream using computer vision algorithms.
YOLO (You Only Look Once), SSD (Single Shot Multi Box Detector), Mobile Net, and other more recent architectures are examples of algorithms that use Convolutional Neural Networks (CNNs) that have been trained using commercial datasets such as Open CV's Haar cascades.

2. Gesture Recognition:

Acknowledge particular hand motions and associate them with mouse operations such as scrolling, clicking, and movement.
Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), or more modern architectures like Convolutional Long Short-Term Memory (ConvLSTM) networks are examples of machine learning models. To train these models on gesture datasets, you can utilize libraries such as Py Torch or Tensor Flow.

1. Monitoring Hand Motions:
To convert hand movements into screen cursor movements, track the hand's movements.
Algorithm: The hand's motion between successive frames can be estimated using optical flow algorithms like Lucas-Kanade or feature tracking methods like KLT (Kanade-Lucas-Tomasi) algorithm.

User Interface Integration: Include the virtual mouse feature in the system's user interface.
Algorithm: Relies on the programming language and platform. For instance, when creating a desktop application, you may utilize Python modules such as PyAutoGUI to programmatically control mouse clicks and motions.

1. Real-Time Performance Optimization: To guarantee real-time performance, optimize methods and algorithms.
Algorithm: To speed up hand detection, gesture recognition, and cursor movement, strategies including parallelization, neural network architecture optimization, and hardware acceleration (such as GPUs and TPUs) can be used.

2. Testing and Evaluation: Make sure the system is accurate, reliable, and easy to use by conducting a thorough testing process.
There isn't a particular procedure, but methods like user input gathering, integration testing, and unit testing are crucial for assessment.

DIAGRAMS

Figure 1 : Flowchart

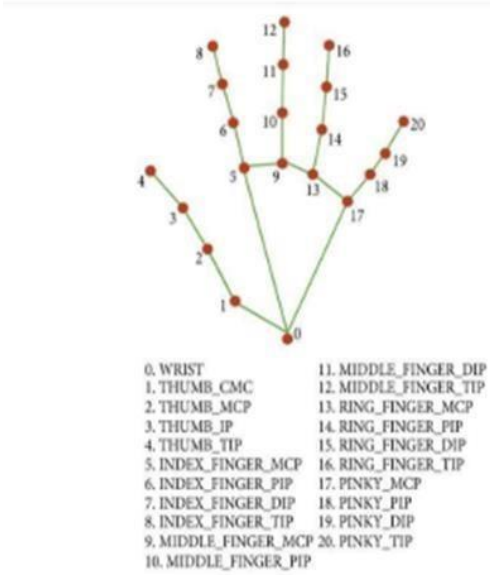

Figure 2: Model Graph of Media Pipe

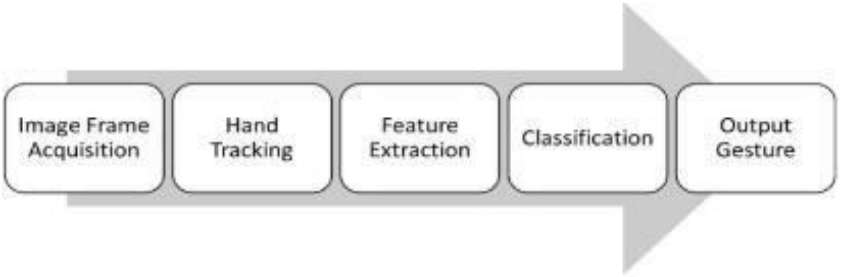

Figure 3: Workflow of hand gesture recognition

RESULT

Figure : Cursor move

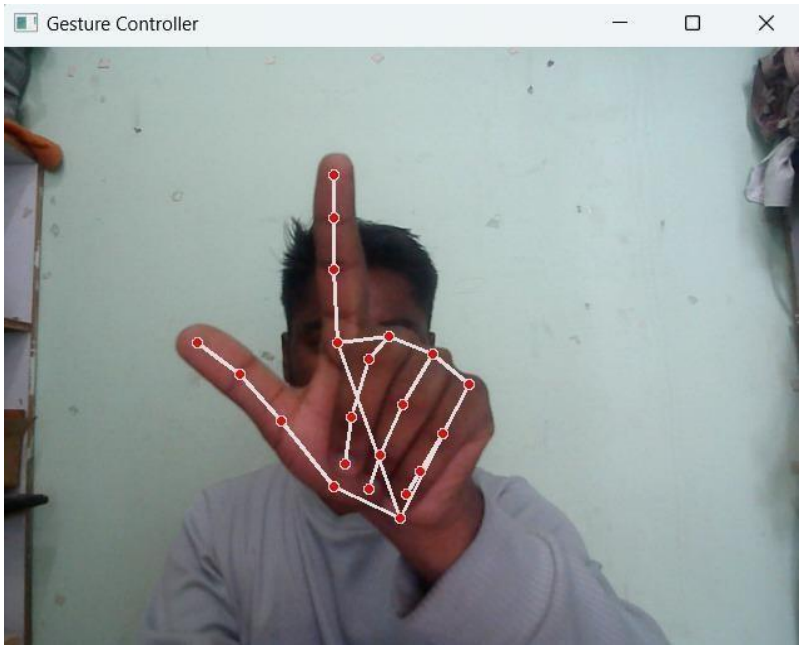

Figure : right click

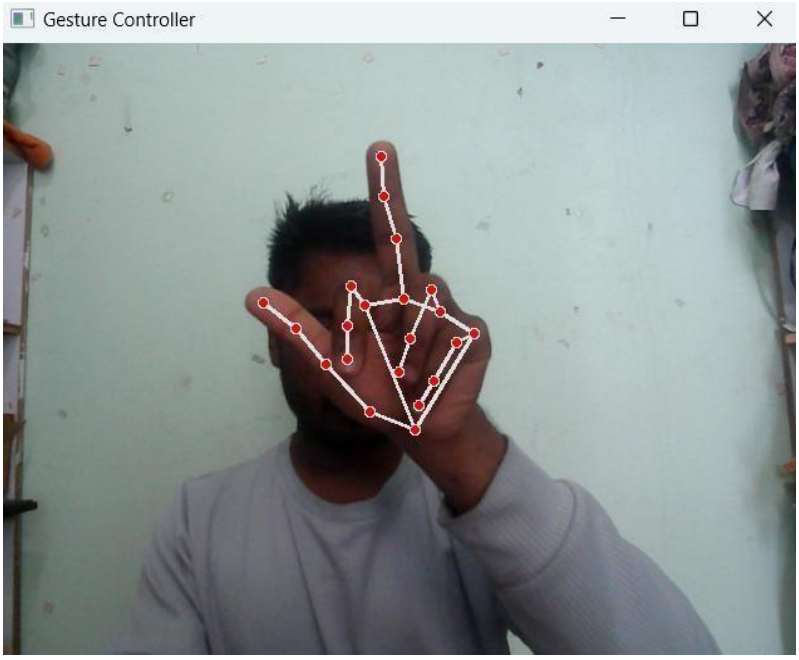

Figure : left click

CONCLUSION

This study offers a workable and affordable virtual mouse system that makes use of computer vision algorithms to recognize hand gestures. In order to control mouse cursor motions and simulate click occurrences, we developed a system that uses OpenCV, MediaPipe, and PyAutoGUI to recognize hand landmarks in real-time and map them to screen coordinates.

The experiment shows that touchless interaction systems can successfully replace conventional input devices, particularly in settings where physical contact is awkward or unhygienic—such as public spaces, medical assistive technology and information kiosks for those with physical limitations. With the help of MediaPipe's real-time hand tracking, important hand motions could be reliably and efficiently detected without the need for specialist gear, providing a solution that a variety of users could employ.

According to experimental testing, the device achieved an average frame rate of 15–20 frames per second in well-lit, controlled conditions. Simple motions such as clicking and moving the pointer were consistently identified. However, in situations when the hand was partially obscured, the background was cluttered, or the lighting was bad, the system's performance suffered. This demonstrates the need for more effort to improve robustness and the accuracy of gesture recognition in a variety of environmental settings.

Notwithstanding its drawbacks, the suggested approach offers a solid basis for advancement. It makes it possible to include more intricate interactions including multi-finger gestures, drag-and-drop, right-click, and gesture-based authentication. Furthermore, the user experience might be greatly improved by using machine learning methods for gesture classification and using adaptive filtering for motion smoothing.

To sum up, this study effectively illustrates how hand gesture-based systems can be used as workable substitutes for actual input devices. Gesture-based HCI is anticipated to become a crucial component of commonplace user interfaces as computer vision continues to progress and computational resources become more widely available.

FUTURE RESEARCH DIRECTION

Although the virtual mouse system's current implementation offers a practical and effective way to accomplish simple gesture-based interaction, there are a number of areas that could use more investigation and improvement:

9.1 Enhanced Classification of Gestures

Future research can use deep learning or machine learning models to more accurately identify a wider variety of motions. To identify dynamic gestures like scrolling, dragging, zooming, and swiping, methods like transformer-based architectures, recurrent neural networks (RNNs), and convolutional neural networks (CNNs) can be trained on hand gesture datasets.

9.2 Adaptability to Environmental Factors

Current performance is impacted by occlusion, background clutter, and lighting changes. Research can concentrate on enhancing robustness by using adaptive illumination normalization, for example.

• Subtraction of background
• Monocular depth prediction or stereo vision for depth estimate

This would enable the system to function dependably in a variety of real-world scenarios.

9.3 Combining Virtual and Augmented Reality (AR/VR)

Gesture-based input can improve the user experience in immersive environments as AR/VR systems proliferate. Future studies could examine how this virtual mouse system might be modified for 3D interaction in virtual reality environments, enabling users to move virtual objects with hand gestures in a natural way.

9.4 Support for Multiple Hands and Users

Complex operations like two-handed item rotation or resizing will be made possible by expanding the system to accommodate multi-hand input. Furthermore, allowing multi-user engagement in shared digital places (such collaborative design platforms and smart schools) may lead to new application areas.

9.5 Interfaces for Accessibility Based on Gestures

The creation of assistive devices for those with restricted mobility can be the subject of future research. The system could be a useful tool for inclusive computing if voice feedback is integrated and gesture sets are modified to meet the demands of people with disabilities.

REFERENCES :

1. Zhang, Z., Cao, L., & Xu, C. (2020).
Real-time Hand Gesture Recognition Based on Deep Learning.
Journal of Visual Communication and Image Representation, 64, 102623. https://doi.org/10.1016/j.jvcir.2019.102623

2 Molchanov, P., Gupta, S., Kim, K., & Kautz, J. (2015).
Hand Gesture Recognition with 3D Convolutional Neural Networks.
Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 1–7. https://doi.org/10.1109/CVPRW.2015.7301347
Erol, A., Bebis, G., Nicolescu, M., Boyle, R. D., & Twombly, X. (2007).
Vision-Based Hand Pose Estimation: A Review.

3. Computer Vision and Image Understanding, 108(1–2), 52–73. https://doi.org/10.1016/j.cviu.2006.10.012
Shah, M. H., Shaikh, M. F., & Memon, Z. A. (2019).
Virtual Mouse Implementation Using Hand Gesture Recognition.
International Journal of Advanced Computer Science and Applications (IJACSA), 10(6), 149–155. https://doi.org/10.14569/IJACSA.2019.0100619

4. Google MediaPipe Documentation.
MediaPipe Hands.
Available at: https://google.github.io/mediapipe/solutions/hands.html
Bradski, G. (2000).
The OpenCV Library.
Dr. Dobb's Journal of Software Tools. Available at: https://opencv.org

5. PyAutoGUI Documentation.
PyAutoGUI - Programmatically Control the Mouse and Keyboard.
Available at: https://pyautogui.readthedocs.io

6. Szeliski, R. (2022).
Computer Vision: Algorithms and Applications (2nd ed.).Springer.
ISBN: 978-3-031-05632-3

7. OpenCV-Python Tutorials.
OpenCV-Python Tutorials Documentation.
Available at: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html

8. Tayal, A., Goel, R., & Verma, A. (2021).
Touchless Human-Computer Interaction Using Hand Gestures for Smart Environments. International Journal of Interactive Multimedia and Artificial Intelligence, 6(7), 128–134. https://doi.org/10.9781/ijimai.2021.03.006