Virtual Mouse: Gesture Based Mouse Controls

Mayuresh Anand Pingale, *Stony Brook University*

Abstract

The computer mouse is the prevalent input device used for interacting with systems. While the Wireless Mouse was the next invention that improved mobility and experience, Touch-Screen laptops provided mouse free experience. However, they are costly. A gesture-based virtual mouse will eliminate these disadvantages. A virtual mouse can imitate the operations of a regular hardware mouse like clicks, moving, pointing, etc. I developed a system that uses finger gestures for interaction. Gestures captured through an inbuilt webcam will be processed using computer vision and appropriate function will take place. The system does not require any additional hardware and works in real time. Furthermore, a small study was conducted to determine the goodness of the gesture.

Keywords

Virtual Mouse, Open CV, Media Pipe, Gesture Control

1. INTRODUCTION

In the past recent years, the development of the gesture-based interaction system has increased. This progress has further been encouraged due to freely available software and the high computational power available. Gestures are intuitive in nature and a powerful interhuman communication modality. In [1], they have developed a perception pipeline that recognizes different types of gestures efficiently.

Since the 1980s, we have been using Graphical User Interfaces for interacting with computers. A mouse is the first choice of hardware device used for interacting with GUIs. A mouse is capable of performing various functions on a computer like a move cursor, selecting, dragging and dropping, hovering, etc. Mouse technology has evolved over several years. It includes wireless, touchpad, Laser, and Trackball technologies. Though these technologies are promising, they are hardware-based and involve an additional cost.

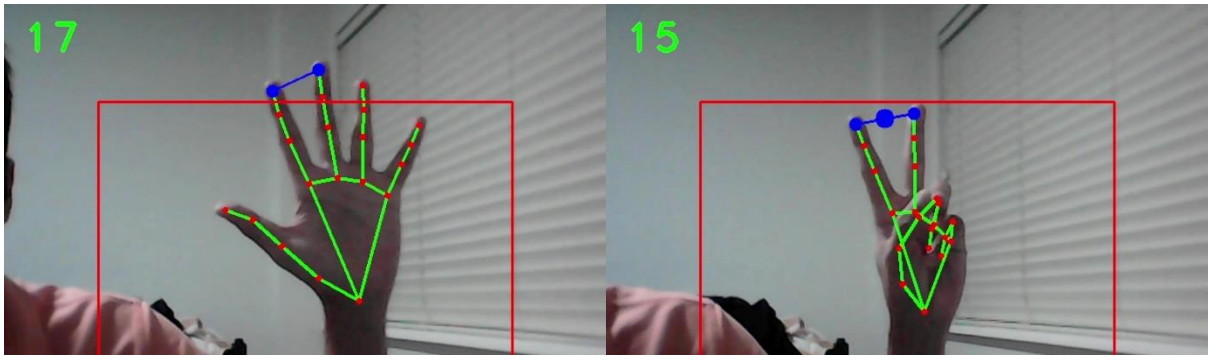


Figure 1

Gesture-based mouse control is a good substitute for these technologies. A virtual mouse will take human gestures as input and produces an appropriate output. I have designed a system that captures images from the in-built camera eliminating the need for additional hardware. Instead of building a new hand recognition model that will require thousands of annotated datasets, I used a media-pipe project which is open source. It provides hand recognition in real time. To recognize gestures, I have used OpenCV. The system developed works in real-time and delivers reliable output. Figure 1 shows some of the gestures supported by the system.

2. RELATED WORK

Many HCI researchers have tried to develop a system that controls mouse movement using camera devices. These techniques differ in hand motions to control the mouse.

Horatiu-Stefan et al [2] has implemented a human hand gesture system to control the mouse cursor. Their setup comprises of Microsoft LifeCam HD-3000 webcam, a tripod to support the camera, and a hand pad. The system was able to capture images under low illuminance as well. While their approach gives reliable results, they have just supplied the gestures to control the mouse cursor. They have not provided gestures for other mouse operations like scrolling, zooming, etc. Additionally, their method requires a need of an additional camera.

Chu-Feng Lien [3] designed a real-time hand-mouse system on Handheld Devices. They used two approaches for palm recognition. The first approach involved the use of the Ada-Boost algorithm, whereas the second method involved the usage of Motion History Images (MHI). The image density was used to track the motion of the hand. Their system performed well in a controlled environment. However, in the case of fast-moving motion, their system had a high error rate. Furthermore, the environmental light or the shadow of the projector provided misleading results.

A virtual mouse based on color pointer tracking was developed by Bhavana [4]. In their setup, users wear color tape on their fingers. The in-built Web Camera captures input images which are further analyzed to extract meaningful information. The method supports mouse control and clicking operations. One of the major drawbacks of this methodology is the presence of other colored objects in the background might cause an incorrect response. Also, the fixed frame rate of the video causes the system to perform slower.

None of the related work provides gestures for operations like scrolling, zooming, dragging, and dropping. Moreover, all of these techniques require an additional component or hardware. The system proposed does not require any extra elements and works in real-time.

3. SYSTEM IMPLEMENTATION

Figure 2 depicts the system's architecture. The laptop's in-built camera captures the video. Here real-time images obtained by the camera are passed to the media pipe for hand-tracking detection. The Media pipe's Hand Landmark Model produces twenty-one 3D coordinates of different indicators of the hand.

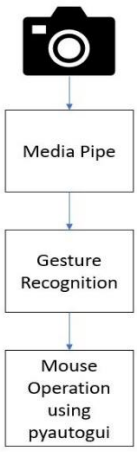
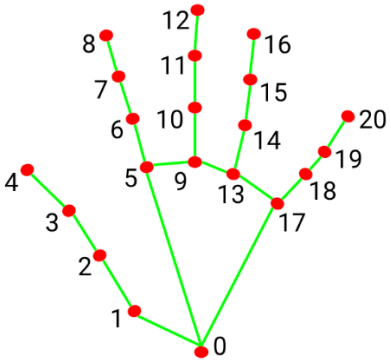


Figure 2

Figure 3 illustrates the handmarks detected by the model. The Gesture Recognition model takes the coordinates as the input and based on the position of the fingers, tries to predict the gesture. The orientation of fingers tells us the type of gesture made. To automate the mouse operation, I used the pyautogui library.



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

A. Media Pipe

Media Pipe Hand Model is a highly reliable hand and finger tracking solution. It utilizes deep learning to extrapolate 21 3D junctures of a hand from just a single frame. The API provides some configuration options. While I have not changed the parameters, I have modified the hand detection confidence to 0.8. This has reduced false detection of the hand. I have set maximum hand detection as 2.

B. Gesture Recognition Model

Figure 4 demonstrates the working of the gesture recognition model. It takes the input of the coordinates generated by the media pipe model. The proposed system supports both one-hand and two-hand gestures. Media pipe provides the number of hands detected in the images.

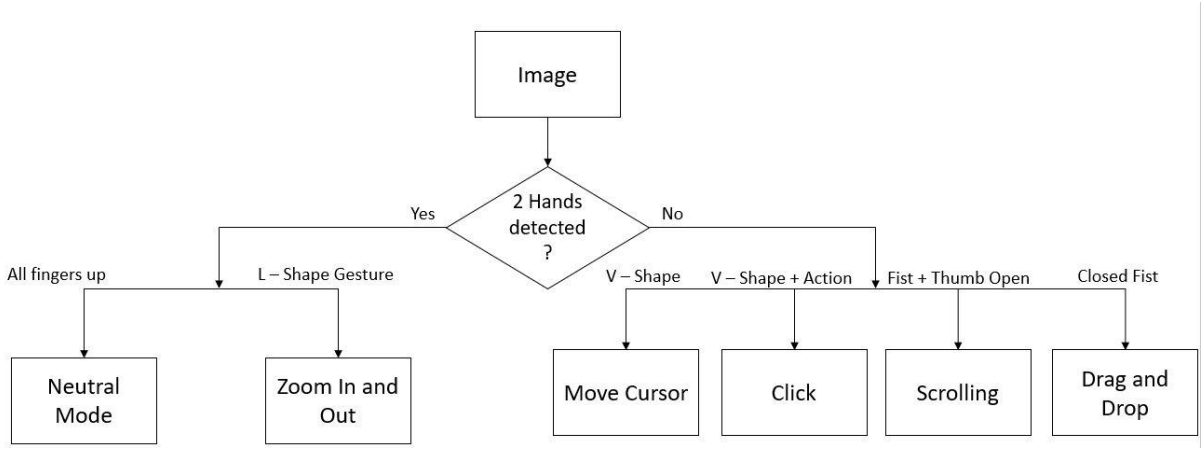


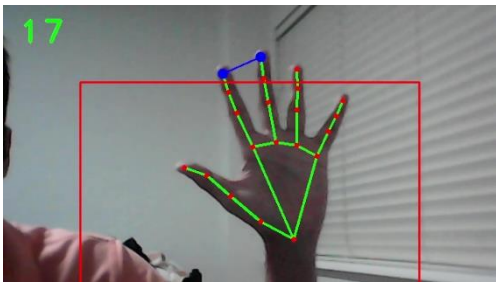
Figure 4

To identify different gestures, I have used the x and y coordinates of the hand landmark given by the Media Pipe Model. The orientation of fingers provides us with the necessary information for determining the gestures. The system tries to locate the direction of each finger. Referring to Figure 3, an index finger orientation is considered in an upward direction if landmark 8's y coordinate is greater than landmark 6's y coordinates. Similar logic is applied to determine other fingers' orientation. Once the orientation is determined, it is easy to classify the gesture.

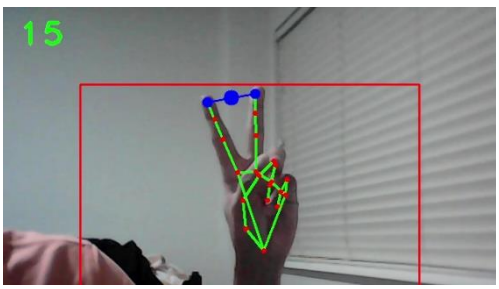
1) *One Hand Gestures:*

Different one-hand gestures implemented are

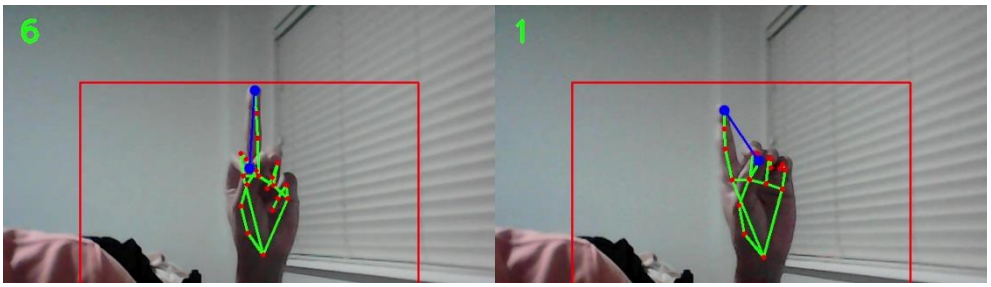
- Neutral Gesture: A neutral gesture is recognized by the system when all fingers point upward direction. The purpose of the Neutral Gesture is to initialize the system.



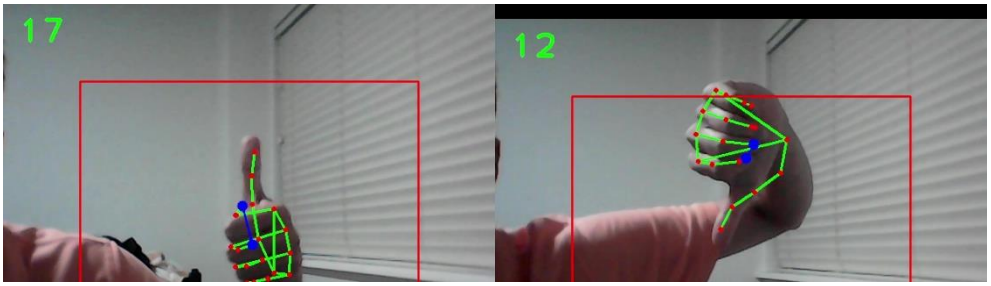
- Move Cursor: A V-shape made with the index and middle fingers provides control over the mouse. The system identifies this gesture when the index and middle fingers have an upward orientation while the other three have a downward direction. Continuing the same gesture, if the hand moves in the left direction, the mouse cursor will follow the same path.



- Clicks: To perform a left click, in a V-shape gesture, a user has to close his index finger. Similarly, a right click is executed when a user closes his middle finger in a V-shape gesture. These motions are very intuitive in nature.

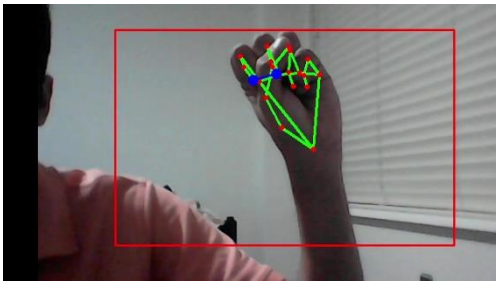


- Scroll: Scrolling operations are accomplished with the help of a Fist gesture and thumb orientation. The direction of the thumb decides its motion. The downward thumb orientation indicates scrolling in the down direction. Likewise, if the thumb's tip point towards the left, the left scroll is executed



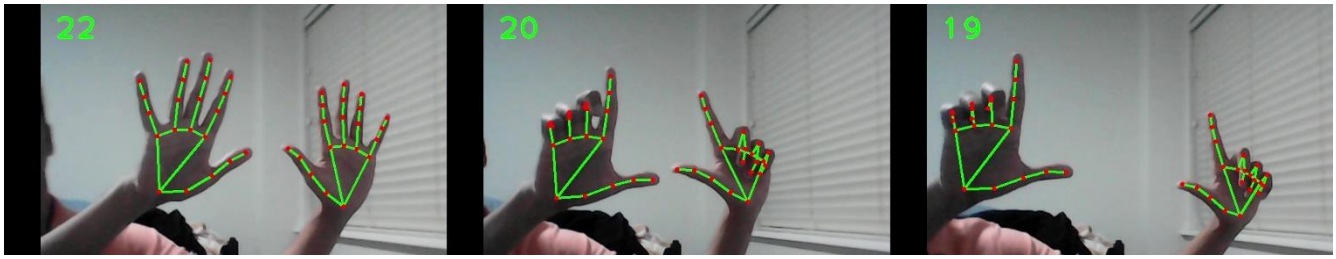
.

- Drag and Drop: A closed fist gesture is identified as a drag operation. When all fingers have a downward motion, the system recognizes it as a fist. It remains in the dragging mode until a user makes the neutral gesture.



2) Two Hand Gestures

The same logic is applied to two-hand gesture recognition. Both hands' finger motion is determined using the coordinates of the landmark provided by the model. Zooming in and out of the screen is achievable using two hand gestures. An L-shape gesture is the starting point of this action. In consecutive frames, if the distance between two thumbs reduces, then the screen zooms out (Resolution decreases). Likewise, continuing the L-shape motion, if the distance between thumbs increases then, the screen zooms in. The following figure depicts the two hand gesture operations.



From Left to Right Neutral, Zoom In and Zoom out

C. Pyautogui

Pyautogui gives python scripts to automate the interactions with the keyboard and mouse. The API design is simple and powerful to use. All functions developed by the system are simulated using pyautogui.

4. VALIDATION

The application is developed in python and uses media pipe and OpenCV libraries. To evaluate the gestures, I conducted a small study with a group of 5 people. Each participant used a virtual mouse and gave ratings about the goodness of the gesture. This study aimed to evaluate whether the gestures are easy and intuitive to use. Each user gave a rating on a scale of 0 to 5 where 0 being the lowest and 5 being the highest. Table I summarizes the average ratings of different gestures.

Table 1: Average Ratings

Gesture	Average Ratings (0 – 5)
Move Cursor	4.8
Click	4
Scrolling	4
Drag and Drop	4.8
Zoom	3.8

According to the feedback acquired, Mouse Cursor Gesture and Drag and Drop Gestures was the easiest and most intuitive. The zooming operation received the lowest average rating because it involved using two hands.

5. FUTURE SCOPE

The application has performed well, and most of the gestures are intuitive. The future aspect of the work involves building a more robust system that has few false positives. Instead of two-hand zooming gestures, the same operation can be accomplished using one hand. Additionally, in the future, the system can support more intuitive gestures.

6. CONCLUSION

A virtual gesture-based mouse that uses an inbuilt camera and requires no interaction with the hardware is designed. The approach has been implemented in python and uses Media Pipe and OpenCV. The system supported both one-hand and two-hand gestures. A small

study conducted has validated the usage of gestures.

The technology has broad applications in the fields like computer gaming, augmented reality, etc. Furthermore, it can also help people with motor disabilities as gestures are the easiest way of communication. Thus gesture-based mouse is created that does not require any additional device.

REFERENCES

[1] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan," MediaPipe: A Framework for Building Perception Pipelines", Google Research

[2] Horatiu-Stefan Grif, Train True," Human Hand gesture-based system for mouse control", 11th International Conference Interdisciplinarity in Engineering, 5-6 October 2017, Romania

[3] Chu-Feng Lien, "Portable Vision-Based HCI – A Realtime Hand Mouse System on Handheld Devices", National Taiwan University, Computer Science and Information Engineering Department.

[4] Bhavana I V," Virtual Mouse Based on Colour Pointer Tracking", International Journal of Science and Research (IJSR)

[5] Hojoon Park, "A Method for Controlling the Mouse Movement using a Real Time Camera", 2008, Brown University, Providence, RI, USA, Department of computer science

[6] https://pyautogui.readthedocs.io/en/latest/

[7] https://google.github.io/mediapipe/solutions/hands.html

[8] https://docs.opencv.org/4.x/

[9] https://www.autopy.org/documentation/api-reference/mouse.html