

Gesture-Based Multi-Parameter Controller for Mac using Machine Learning and Computer Vision

Vrushabh Sachin Shah

Computer Science and Engineering, MIT World Peace University, Pune, India

Abstract - The proposed system uses a lightweight framework called Media pipe which not only allows for detection of hands but can classify whether it is the left or right hand and represents this data as 21 data points representing the various parts of the hand. With these uniquely identified 21 data points for each hand, we can identify different gestures and assign some functionality for them. The system checks the position of fingers, which of them are raised, and which are not to uniquely identify the hand gestures. It is capable of controlling multiple basic functionalities like increasing or decreasing the volume level, controlling the brightness intensity, a virtual mouse, zooming in and out, and scrolling in all four directions. These functionalities are divided between the gestures created by the left and right hand. This particularly comes in handy when watching a movie or video where the laptop is kept at a distance or connected to a TV and reaching the keyboard to control all these functionalities continuously becomes an irritating task.

Index Terms - Machine Learning, Computer Vision, Mac, Media Pipe, Gesture Detection, AppleScript.

I. INTRODUCTION

With the advancements in the field of machine learning and computer vision, it has become possible to perform tasks like face recognition, object detection, pose estimation, etc. The proposed system uses a lightweight, high fidelity machine learning framework MediaPipe Hands which allows for hand detection and tracking and is also capable of classifying the left hand from the right.

It represents the detected hands in the form of 21 different data points which allows identifying the fingers of the hand and their position uniquely. This forms the basis for the working of the system, wherein the position of the fingers is continuously monitored to check for the different gestures, and when a registered gesture is encountered the respective

functionality is performed. The system is capable of controlling various basic functionalities without requiring the use of the keyboard like controlling the volume, brightness, mouse with clicking functionality, zooming in and out on a particular screen, and scrolling in all four directions. Since the number of functionalities to be controlled is more these are split up between gestures created by the left and right hand. The left-hand gestures control volume and brightness and the right-hand gestures control the mouse, zoom, and scroll.

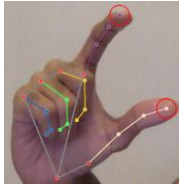
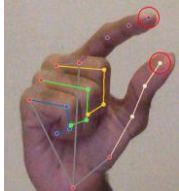

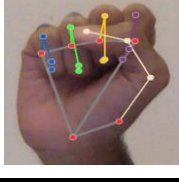
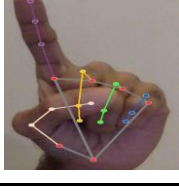
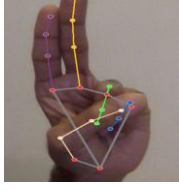
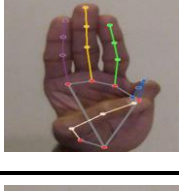
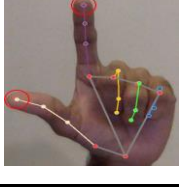
II. OBJECTIVES

The idea is to ease the use of basic functionalities which we use frequently in our daily life without requiring to always be near the keyboard but control them using hand gestures as long as the user is in the field of view of the webcam or can use a USB webcam to operate these functionalities from a long distance.

1. Control volume level using hand gestures without using the keyboard.
2. Increase or decrease the screen brightness.
3. Use the mouse functionality without using the actual trackpad or mouse along with the click functionality.
4. Zoom in or out on a particular screen or photo using hand gestures.
5. Scrolling left, right, up, or down as required without the need for a keyboard or mouse.

III. HAND GESTURES AND THEIR RESPECTIVE FUNCTIONALITIES

The proposed system uses a variety of hand gestures created by the left and right hands to control the above-mentioned functionalities. The webcam is continuously monitoring for these gestures and when encountered the respective activity is performed. The below table shows the various hand gestures and the functionalities mapped to them.

Hand Gesture	Hand	Functionality
	Left	Increase Volume
	Left	Decrease Volume
	Left	Increase Brightness
	Left	Decrease Brightness
	Right	Mouse
	Right	Left Click
	Right	Right Click
	Right	Zoom In

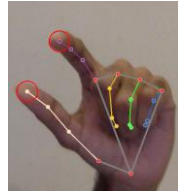
Hand Gesture	Hand	Functionality
	Right	Zoom Out

Table1. Hand Gestures and their Mapped Functionalities

IV. METHODOLOGY

This section explains in detail the different components in the system, their working, and the technologies used. The system uses computer vision to continuously monitor the user for hand gestures and when detected perform the respective task.

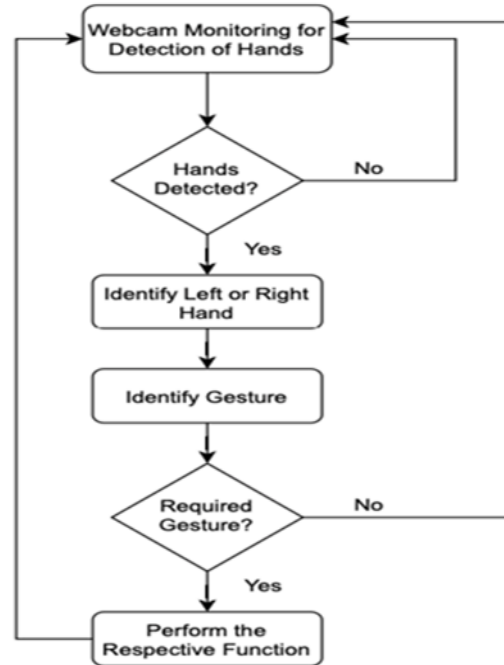


Fig. 1. System Flowchart

The MediaPipe Hands framework plays the most important role in detecting hand gestures. It utilizes machine learning to capture 21 data points of a hand in the frame. It uses various ML models working together like palm detection and a hand landmark model which works on the data returned by the palm detection model to capture and return the 21 data points representing hand landmarks. The 21 data points referring to the various hand landmarks are shown in Figure 2.

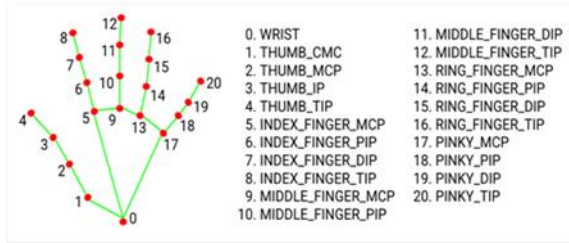


Fig. 2. 21 Hand Landmarks Returned by the Hand Landmark Model

It returns a set of detected hands as a collection of 21 landmarks, where each landmark contains the data about x, y, and z coordinates which are normalized in the range [0.0, 1.0] with respect to image height and width. To get the actual coordinates, these normalized values are multiplied by the height and width of the screen which can be obtained by using the size() method of the pyautogui library. The x coordinates are multiplied by the width and y coordinates are multiplied by the height of the screen respectively. The above ML pipeline also allows identifying left and right hands uniquely, which allows us to split the functionalities between the two hands.

The hand gestures can be uniquely identified by checking which of the fingers of the hand are raised and not, which can be inferred in either of the two ways, firstly by calculating the Euclidean distance between the fingertips represented by the 4th, 8th, 12th, 16th, 20th hand landmark from the wrist which is represented by the 0th landmark, as seen in Figure 2. If the distance is greater than a specific threshold, then the finger can be considered as raised otherwise not. Another method is to check the y-coordinate at points 6, 8, 10, 12, 14, 16, 18, and 20 as seen in figure 2. If the y-coordinate of the fingertips represented by points 8, 12, 16, 20 is lesser than the y-coordinate at points 6, 10, 14, 18, we can infer that the finger is raised otherwise not. However, to detect whether the thumb is raised or not we consider the x-coordinate. For the right thumb, if the x-coordinate of point 4 is less than that of point 0 as seen in Figure 2, we can consider it to be raised otherwise not. For the left thumb, however, if the x-coordinate of the 4th point is greater than that of the 0th point it is considered to be raised.

A. Volume Control

A variety of libraries are available to control the volume for Windows and Linux systems but not for

Mac systems. So we have to make use of osascript python library which allows us to run

Applescripts which allows automated control over scriptable Mac applications and facilitate us to automate key press or sending keystrokes.

Running the “get volume settings” command using osascript gives us the current volume which can be rounded to the nearest multiple of 10 and increase and decrease in volume can be obtained by increasing or decreasing the current volume in the increments or decrements of 10 by using the osascript command “set volume output volume x” where x values between 0 to 100.

For volume control to be activated the webcam checks if the hand is left and its middle, ring, and the little ringer are not raised and the thumb and index finger are raised. Then the Euclidean distance between the tips of index and thumb is calculated, if it is greater than a threshold the volume increases in increments of 10, otherwise decreases in the decrements of 10.

B. Brightness Control

This functionality also makes use of the osascript library and Applescripts as libraries to control the brightness for Mac are not available directly. For this to be activated the webcam checks if the hand detected is the left hand and all its fingers are raised, if yes it increases the brightness and if none are raised, then it turns down the brightness. The brightness is controlled by automating a keypress using AppleScript where each keyboard key and functionality is assigned a key code and the key code for increasing the brightness is 144 and for decreasing the brightness is 145. The AppleScript for automating a keypress is as follows:

```
tell application "System Events"
    key code x
end tell
```

Where x is the key code of the key to be pressed or functionality to be performed.

C. Mouse

The webcam looks for the right hand and checks if only the index finger is raised if so then it enters the mouse mode and the mouse cursor is moved along with the movement of the hand. This is achieved using pynput library which allows us to control the mouse by setting its x and y coordinate. The problem however with this setup is that while trying to reach the lower

part of the screen the hand goes out of the field of view of the camera and tracking fails, so we define a rectangular region smaller than the actual screen size and interpolate the coordinates of this region to match the actual screen size.

For the left click functionality, the webcam checks whether the index and middle finger of the right hand are raised, and for the right-click, it checks whether the index, middle, and ring fingers are raised. The scrolling functionality makes use of the mouse pointer functionality itself, when the cursor is near any of the screen boundaries like left, right, up, or down, it scrolls in the respective direction using the keyboard control available with pynput library.

D. Zooming Function

The webcam looks for the right hand and checks if only the index finger and thumb are raised, and then by calculating the Euclidean distance between the tips of the index finger and thumb and by comparing it with a threshold, it decides whether to zoom in or out. If the distance is more than that of the threshold set, it zooms in onto the current screen and zooms out otherwise. The zoom is again controlled using AppleScript and we automate the keypress “Command” and “+” for zoom in and “Command” and “-” for zoom out, the key code for which is 24 and 27 respectively.

V. CONCLUSION

The system allows for easy control over the basic and commonly used functionalities like volume control, brightness, mouse control, and zoom. This comes in handy when the device is kept at a distance like when watching a movie or connected to a TV etc. We don’t need to reach the keyboard and mouse to control these functionalities as long as we are in the field of view of the camera. An extended USB Camera allows controlling these functionalities from even larger distances. The system also proves to be beneficial if the keyboard and mouse stop working as we can still control everything using this virtual mouse and an on-screen keyboard.

REFERNCES

- [1] <https://google.github.io/mediapipe/solutions/hands>
- [2] <https://www.makeuseof.com/tag/learn-automate-mac-applescript-part-1-introduction/>

- [3] <https://eastmanreference.com/complete-list-of-applescript-key-codes>
- [4] <https://ss64.com/osx/osascript.html>
- [5] <https://pynput.readthedocs.io/en/latest/>