AI Virtual Mouse

Raju Ranjan
*rajur.ug*18*.ec@nitp.ac.in*
B.Tech., Department of
Electronics and Communication Engineering
National Institute of Technology Patna, India

Shruti Sinha
*shrutis.ug*18*.ec@nitp.ac.in*
B.Tech., Department of
Electronics and Communication Engineering
National Institute of Technology Patna, India

Abstract

In today's world, mobiles are using touch screen technology. But this can not be used in desktops because of its high cost. Making a virtual human computer interaction device such as mouse using a webcam and computer vision techniques can be an alternative options for the touch screen. In this we are designing a finger tracking based virtual mouse application and implementing using webcam. The main motive was to create an object tracking application to interact with the computer device, and finally develop a virtual human computer interaction device

1 Introduction

The most important challenges in Human Computer Interactions is to design and develop more natural interfaces. The Computation of environments are strongly dependent on the availability of high resolution pointing device with a single, discrete 2-D cursor. The modern GUI (Graphical User Interface) is current standard interface on personal computers (PCs) ,which provides an efficient interface for users to use various applications and is well defined.
In this project, we are going to create an AI based Mouse Controller. We will first detect the hand landmarks and then track and click based on these points. We will also apply smoothing techniques to make it more usable. The software's that will be required to implement the proposed system are OpenCV and python. The output of the camera will be displayed on the system's screen so that it can be further calibrated by the user.
This paper proposes a way to control the position of the cursor with the bare hands without using any electronic device. While the operations like clicking and dragging of objects will be performed with different hand gestures.

.

2 Methodology

In this design, We will first detect the hand landmarks and then track and click based on these points. We will also apply smoothing techniques to make it more usable.

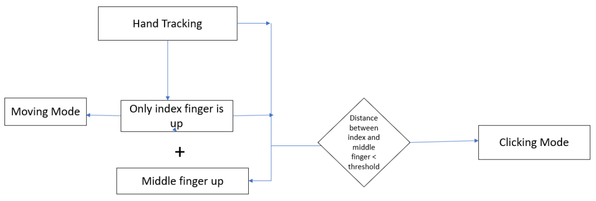


Figure 1. Flow Chart

2.1 Building a Hand Tracking System using OpenCV

OpenCV is a library used for computer vision applications. With help of OpenCV, we can build an enormous number of applications that work better in real-time. Mainly it is used for image and video processing

2.1.1 MediaPipe

MediaPipe is a framework mainly used for building audio, video, or any time series data. With the help of the MediaPipe framework, we can build very impressive pipelines for different media processing functions
Some of the major applications of MediaPipe

- Multi-hand Tracking
- Face Detection
- Object Detection and Tracking
- Objectron: 3D Object Detection and Tracking
- AutoFlip: Automatic video cropping pipeline etc

2.1.2 Hand Landmark Model

Basically, the MediaPipe uses a single-shot palm detection model and once that is done it performs precise key point localization of 21 3D palm coordinates in the detected hand region.
Hand Keypoint detection is the process of finding the joints on the fingers as well as the finger-tips in a given image. It is similar to finding keypoints on Face (a.k.a Facial Landmark Detection) or Body (a.k.a Human Body

Pose Estimation), but, different from Hand Detection since in that case, we treat the whole hand as one object.

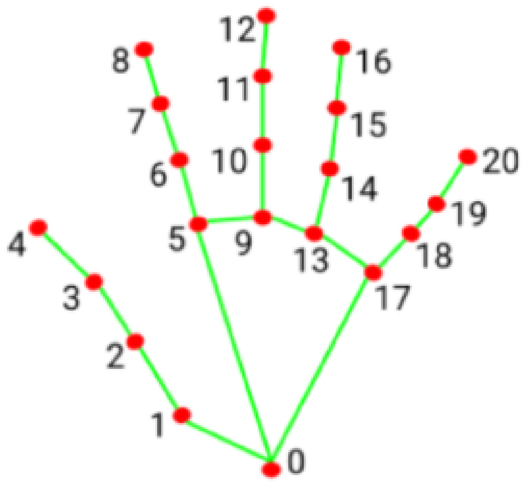
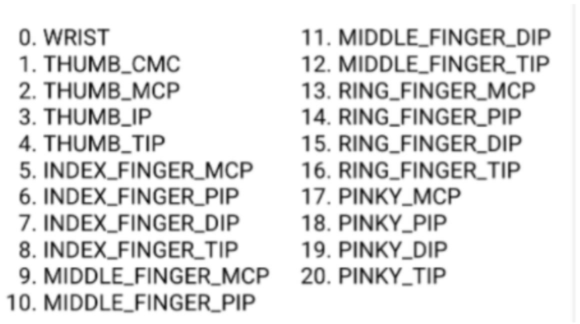


Figure 2. Hand Landmark map

The MediaPipe pipeline utilizes multiple models like, a palm detection model that returns an oriented hand bounding box from the full image. The cropped image region is fed to a hand landmark model defined by the palm detector and returns high-fidelity 3D hand key points.



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

Figure 3. 20 Hand Landmark points

Now to implement the Hand tracking model.
Install the required modules

- pip install opencv-python
- pip install mediapipe

The pop up a window if any webcam is connected to your PC and also shows the frames per second (fps) on the top left corner of the output w The number of hands to detect is set to 2, minimum detection confidence is set to 0.5 and the minimum tracking confidence is set to 0.5

we read the frames from the webcam and convert the image to RGB. Then we detect hands in the frame. Once the hands get detected we will locate the key points and then we highlight the dots in the keypoints using cv2.circle, and connect the key points.

2.2 Design virtual Mouse

In this section now we have built our Hand Tracker module, we can use that to create our virtual mouse.

First we will do video capturing using webcam available with the help of openCV Library The steps Involved in this are as follow:

1. Find hand Landmarks
   - We get the landmarks using the findlandmark function that is defined in the "HandTracking-Module" module just by calling it.

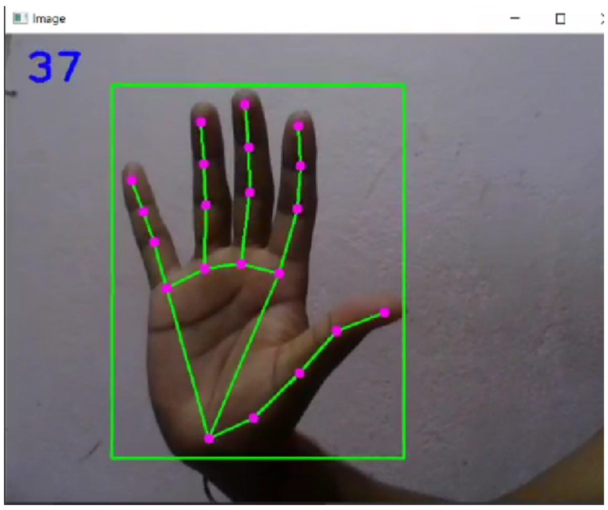
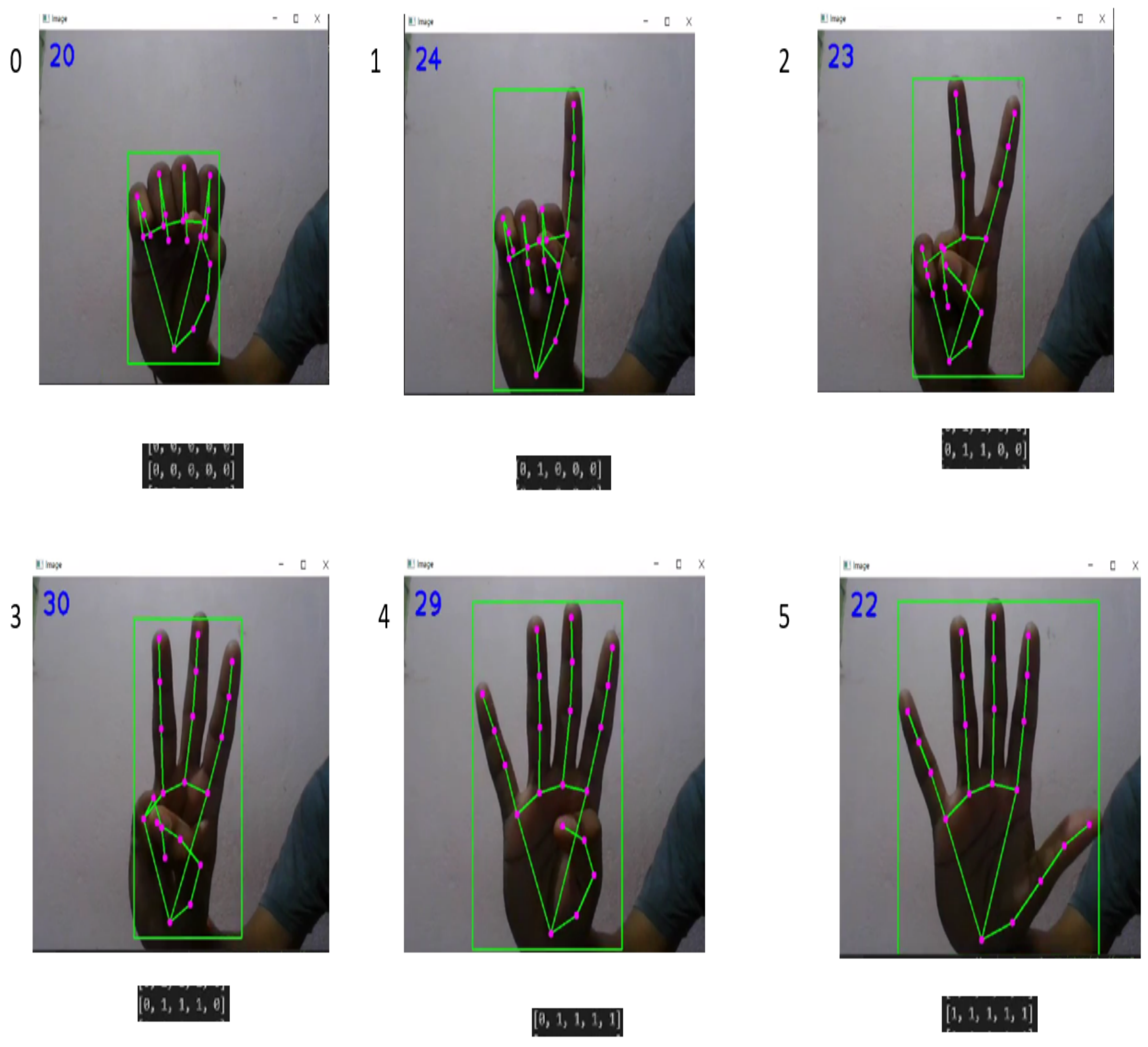


Figure 4. Hand Landmarks Result

2. Get the tip of the index and middle fingers if len(lmList) != 0:
   - We will also get the fingerTip from the Hand-TrackingModule. The idea is if we have only the index finger, the mouse will move ,if we have the middle finger as well then it is in clicking mode
3. Check which fingers are up:
   - We need to check which fingers are up. Based on that we will look after the case in which the virtual mouse will be in different modes.We have the fingers array in which we have the values as 0 or 1. The index of the fingers are from 0 to 4 ,as we are considering here a hand with 5 fingers. If a finger is up then the corresponding element at that finger index will be 1, otherwise 0.



```
[0, 0, 0, 0, 0]
[0, 1, 0, 0, 0]
[0, 1, 1, 0, 0]
[0, 1, 1, 1, 0]
[0, 1, 1, 1, 1]
[1, 1, 1, 1, 1]
```

Figure 5. Fingers count using the fingersup method

4. Only Index Finger : Moving Mode
    - If we have only the index finger then the mouse will be in moving mode, but if the middle finger is up as well then the mouse will be in clicking mode.
5. Convert Coordinates
    - convert the coordinates as per the Screem height and width.
6. Smoothen Values
    - Smoothen the values of the smoothing factor and tune the frame reduction in a manner so that we will get the better result.
7. Move Mouse
    - We are using the **autopy** library to use the functionality of virtual mouse.If we have only the index finger then the mouse will be in moving mode.

mouse — autopy module for working with the mouse:

- This module contains functions for getting the current state of and controlling the mouse cursor.
- Unless otherwise stated, coordinates are those of a screen coordinate system, where the origin is at the top left.

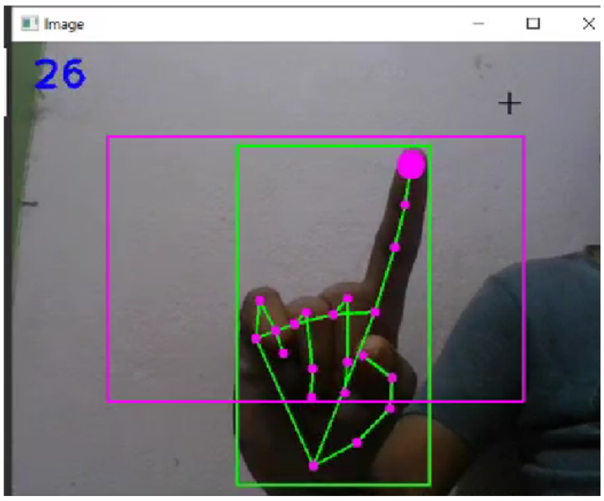


Figure 6. Moving Mode

- Functions
  - autopy.mouse.location() -> (float, float): Returns a tuple (x, y) of the current mouse position
  - autopy.mouse.toggle(button: Button=None, down: bool): Holds down or releases the given mouse button in the current position. Button can be LEFT, RIGHT, MIDDLE, or None to default to the left button.
  - autopy.mouse.click(button: Button=None, delay: float=None) :Convenience wrapper around toggle() that holds down and then releases the given mouse button. By default, the left button is pressed.
  - autopy.mouse.move(x: float, y: float) : Moves the mouse to the given (x, y) coordinate.

8. Both Index and middle fingers are up : Clicking Mode

   - The clicking mode of the mouse is only applicable if Both index and middle fingers are up.In this mode mouse will be restricted to its movement and the clicking is only allowed in that case.

9. Find distance between fingers

   - Find the distance between the fingers using the finddistance function defined in the "HandTrackingModule" .we will be focusing on the distance between the index finger and the middle finger, and based on this distance we will do the clicking operation.

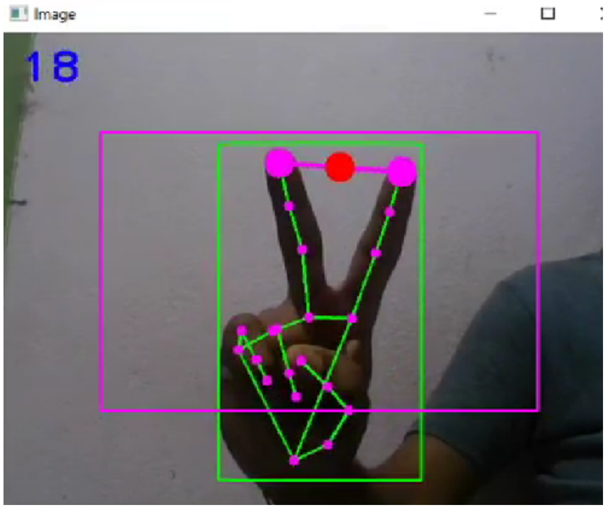
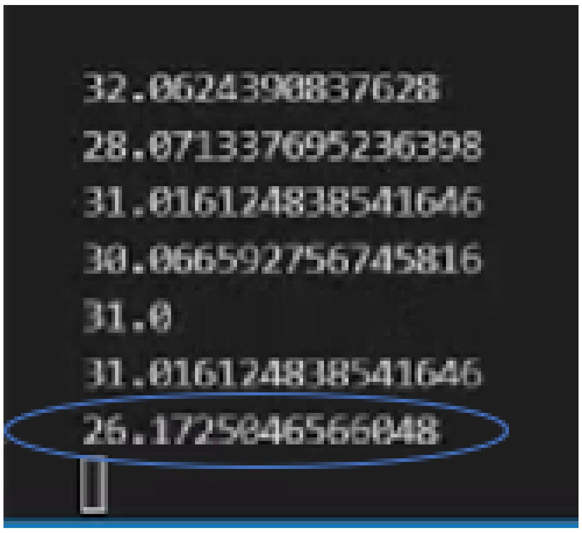


Figure 7. Clicking mode

10. Click mouse if distance short

    - If the Mouse is in clicking Mode then , if the distance between the index finger and the middle finger will become less than a defined threshold (we have tekn 28 in our case) then it is considered as a click.We will apply the click using the autopy click method



```
32.062439080837628
28.071337695236398
31.016124838541646
30.066592756745816
31.0
31.016124838541646
26.1725046566048
```

Figure 8. Clicking distance

11. Frame Rate

    - Frame rate, then, is the speed at which those images are shown, or how fast you "flip" through the book. It's usually expressed as "frames per second," or FPS. So if a video is captured and played back at 24fps, that means each second of video shows 24 distinct still images.

- We will set the Frame reduction to 100 in our case , this can be varied based on the conditions.

12. Display

3 Result and Conclusion

In this paper we Have designed a Artificial Virtual Mouse which worked in two mode i.e. clicking mode and moving mode.This designed has been used to click and to scroll pages and several standard uses of a mouse.This paper represents concept of image processing which is having wider scope in recent years. In this study, an object tracking based virtual mouse application has been developed and implemented using a webcam. Virtual mouse is capable of performing ac-curate control of remote display and simulating mouse.

4 Acknowledgement

We would like to Thank prof. Ritesh Kumar Mishra (Prof. NIT Patna) who had been constantly supporting and mentorinig us and also helped shape our report which helped in giving us much deeper insights into the course which eventually helped us in the completion of this report.

References

[1] https://www.ijarcce.com/upload/2016/may-16/IJARCCE2011.pdf

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

[3] https://www.computervision.zone/lessons/ai-virtual-mouse-video-lesson/

[4] https://www.analyticsvidhya.com/blog/2021/07/building-a-hand-tracking-system-using-opencv/

[5] https://www.techsmith.com/blog/frame-rate-beginners-guide/