Real-Time AI Virtual Mouse System Using Computer Vision

Submitted by

Bedant Patnaik	–	20BCE0568
Arpan Kumar Samal	–	20BEC0191
Ankit Kishan	–	20BEC0193
Akepati Akash Reddy	–	20BEC0593

1. Introduction

1.1 Overview

Using a hand as a virtual mouse allows you to accomplish all mouse-related tasks with little any system involvement. Users can identify hands using their system's webcam. Then, it will create a frame around the hand and draw attention to the forefinger and middle finger. The forefinger will serve as the cursor, and by moving it, we may move the pointer. Now, it is determining the distance between the forefinger and the middle finger in order to effectively click utilising hand tracking. When they are coupled, it will perform a click.

The AI virtual mouse system was developed using OpenCV, a computer vision toolkit, and the Python programming language. Along with the Autopy packages for browsing the computer's window screen and carrying out actions like left click, right click, and scrolling, the MediaPipe package is used to track hands and fingers.

1.2 Purpose

The primary goal of our artificial intelligence (AI) virtual mouse system is to create a replacement for the standard and conventional mouse system to perform and control the mouse functions. This can be done with the use of a web camera that records hand gestures and hand movements.

2. Literature Survey

2.1 Purpose

To move the mouse pointer, Chen-Chiung Hsieh et al. [1] suggested "A Real Time Hand Gesture Recognition System Using Motion History Image". To reduce misclassifications, the suggested solution uses an adaptive skin colour detection model. They employed a C++ software platform with the image processing package open cv loaded to create these techniques. To manage the cursor and carry out clicking activities, Kamran Niyazi et al. [2] suggested "Mouse Simulation Using Two Coloured Tapes," which made use of the Background Subtraction method, Skin Detection method, and HSV Colour Model. The clicking actions were guided by the distance between the tape colours. Java software was used to build this model.

A "Mouse Control Using a Web Camera Based on Colour Detection" was proposed by Abhik Banerjee et al. [3] to control cursor motions and click events by detecting camera colour. The simultaneous detection of the hues is used to carry out clicking actions. Each colour represents a different cursor control. The MATLAB image processing tool box and software were used to develop this solution. Sandeep Thakur et

al. [4] presented "Vision-based Computer Mouse Control Using Hand Gestures". This technique uses a vision-based system to manage various mouse operations, such as left and right clicking, with hand gestures in order to increase the efficiency and dependability of the interface. Different coloured caps are put on fingers to identify hand gestures in order to increase the system's effectiveness and performance.

The MATLAB environment was used to implement this method. To control the mouse cursor, Horatiu-stefan Grif et al [5] proposed "Mouse Cursor Control Based on Hand Gesture". They used an external camera attached to a hand pad and colour strips attached to the fingers in the proposed method. To implement this methodology, they used C programming software along with an image processing library called OpenCV. For using camera-captured hand gestures to control a mouse, Pooja Kumari et al. [6] presented "Cursor Control Using Hand Gestures". In this technique, the camera serves as a sensor, catching and identifying coloured tips that are affixed to the hand. This technique is often referred to as the marker-based approach method since it requires the user to have coloured tips on his hand in order to operate the mouse. They employed the MATLAB environment, the MATLAB Image Processing Tool box, and the OpenCV library to put this concept into practise.

Danling Lu et al. [7] suggested a "Gesture Recognition Using Data Glove" approach for directing the mouse pointer and carrying out clicking activities. This method makes use of an Extreme Learning Machine. They used a cutting-edge data glove in this technique to gather data for gesture recognition. This glove costs little. This approach was developed using cutting-edge machine learning. The "Design of Intangible Interface for Mouseless Computer Handling Using Hand Gestures" was put up by Alisha Pradhana et al. [8] to manage mouse cursor and click actions. Microsoft Virtual Studio, a Microsoft Integrated Development Environment, was used to construct this approach, which makes use of the convex hull algorithm.

2.2 Proposed solution

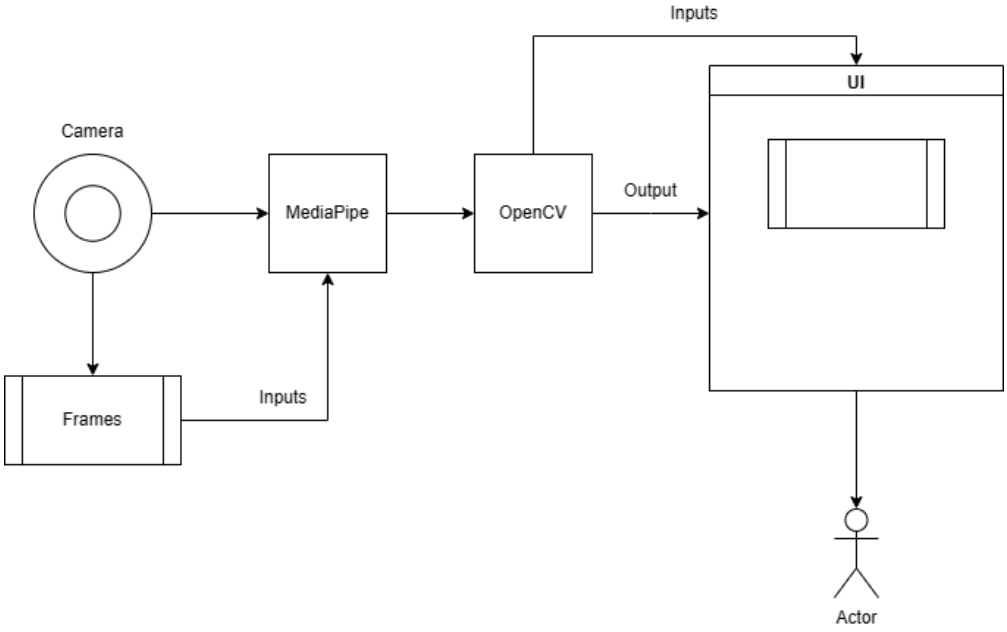
We had used a hand as a virtual mouse, which had allowed us to accomplish all mouse-related tasks with little system involvement. Users had been able to identify hands using their system's webcam. Then, a frame had been created around the hand and attention had been drawn to the forefinger and middle finger. The forefinger had served as the cursor, and by moving it, we had been able to move the pointer. At that time, we had been determining the distance between the forefinger and the middle finger in order to effectively click utilizing hand tracking.

When they had been coupled, a click had been performed. The AI virtual mouse system had been developed using OpenCV, a computer vision toolkit, and the Python programming language. Along with

the Autopy packages for browsing the computer's window screen and carrying out actions like left click, right click, and scrolling, the MediaPipe package had been used to track hands and fingers.

3. Theoretical Analysis

3.1 Block Diagram



3.2 Hardware / Software designing:

This project requires the following to work:

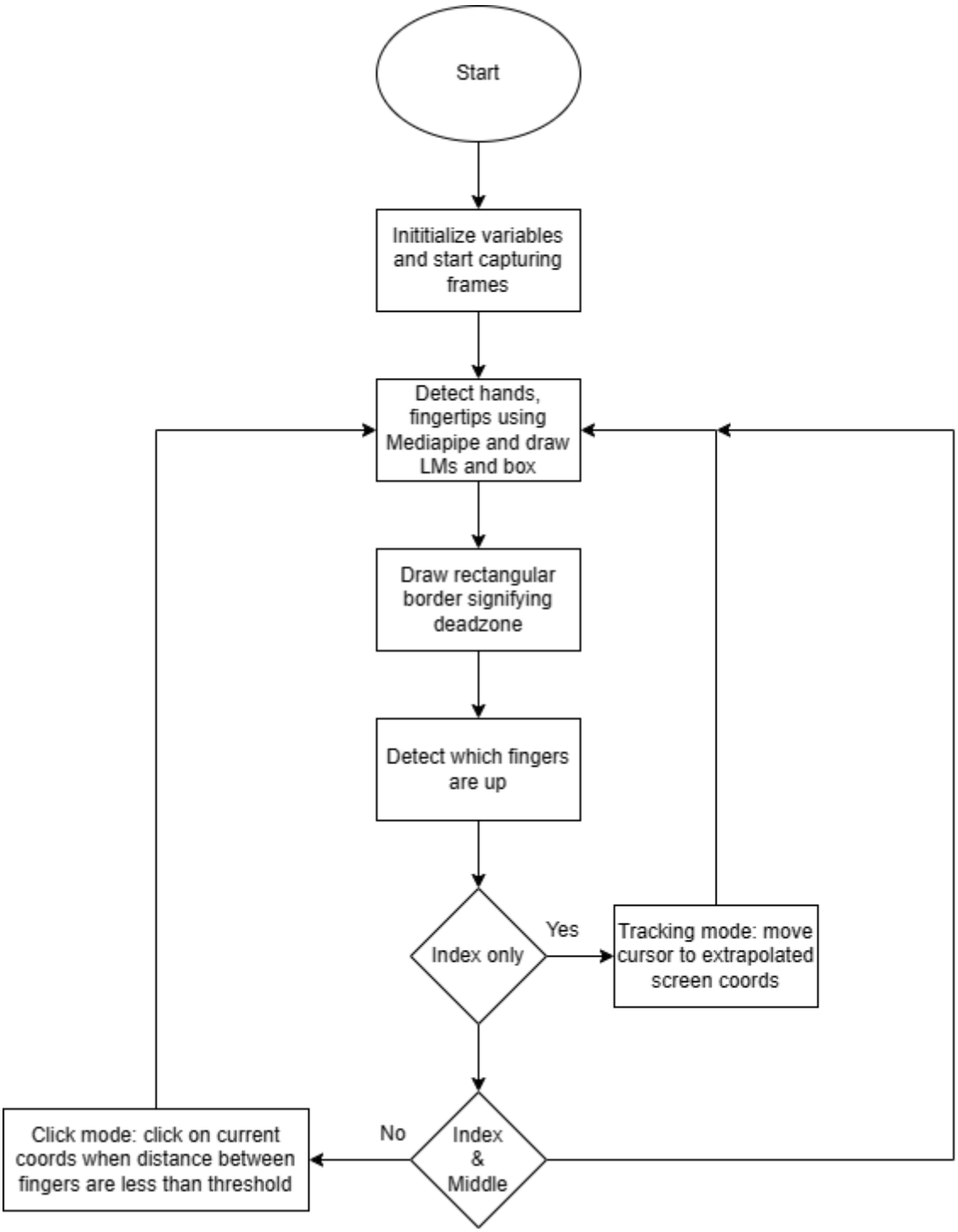
- Python 3.7
- AutoPy
- OpenCV
- MediaPipe API

4. Experimental Investigations

The idea of employing computer vision to advance human-computer interaction is presented in the suggested AI virtual mouse system.

Because there are so few datasets available, it is difficult to compare testing of the AI virtual mouse system. For tracking of the hand gestures and hand tip detection, the webcam has been placed at various distances from the user in order to evaluate the hand gestures and fingertip detection in various lighting situations.

5. Flowchart



6. Result

For testing this program, the computer is set to not perform any mouse actions on the screen.

If only the index finger is up, then the program enters tracking mode, and the mouse cursor moves along with the user's index finger. The coordinates are extrapolated by scaling the webcam coordinates into screen coordinates.

If the index finger and middle finger are up and closer than 40px, then the program triggers a single click at the current cursor position.

For all other cases, no action is performed. However, more actions can be assigned to every possible finger up/down combination.

7. Advantages and Disadvantages

Advantages:

- Reduced hardware cost since there is less reliability on the physical devices like mouse or keyboard
- Main advantage of using hand gestures is to interact with computer as a non-contact human computer input modality
- Time savvy
- Convenient for users not comfortable with touchpad
- Framework may be useful for controlling different types of games and other applications dependant on the controlled through user defined gestures

Disadvantages:

- Any fault in software can lead to system inefficiency.
- High level of coding
- Present application seems to be more feasible and user friendly
- Initial cost of system can be high

8. Applications

The AI virtual mouse system can be utilised for a variety of purposes, including instances when we cannot use a physical mouse and those where it would be inconvenient to do so. The technology reduces the need for gadgets while enhancing human-computer connection. Some of the major applications include:

i. The proposed model's accuracy of 99% is far higher than that of other virtual mouse models that have also been proposed, and it has various applications.
ii. The proposed AI virtual mouse may be used to control the PC mouse functions without using the physical mouse during the COVID-19 condition because it is not safe to use the devices by touching them because it may result in a situation where the virus is propagated by touching the devices.
iii. Without the need of gadgets, the system can be utilised to control robots and automation systems.
iv. Persons with problems in their hands can use this system to control the mouse functions in the computer
v. In designing and architecture, the proposed system can be used for designing virtually for prototyping

9. Conclusion

The primary goal of the AI virtual mouse system is to replace the use of a physical mouse with hand gestures for controlling mouse cursor functionalities. The suggested system can be implemented utilising a webcam or an integrated camera that recognises hand motions and hand tips and processes these frames to carry out certain mouse actions.

We may infer from the model's findings that the suggested AI virtual mouse system has done very well, has better accuracy than the existing models, and also gets beyond the majority of the drawbacks of the latter. The AI virtual mouse can be utilised for real-world applications since the suggested model is more accurate, and it can also be used to stop the spread of COVID-19 because the proposed mouse system can be operated virtually using hand gestures rather than the conventional physical mouse.

The model has some drawbacks, including a slight loss of accuracy in right-click mouse functionality and some challenges with dragging and clicking to pick text. In order to get over these restrictions, we will now work on making the finger tip detection algorithm more accurate.

10. Future Scope

The suggested AI virtual mouse has some drawbacks, including a little reduction in right click precision and certain issues with the model's ability to click, drag, and scroll while selecting text. These are some of the

drawbacks of the suggested AI virtual mouse technology, which we will try to address in the forthcoming study.

2D and 3D images can be drawn using the AI virtual system using the hand gestures. AI virtual mouse can be used to play virtual reality- and augmented reality-based games without the wireless or wired mouse devices. Furthermore, the proposed method can be developed to handle the keyboard functionalities along with the mouse functionalities virtually which is another future scope of Human-Computer Interaction (HCI).

11. Bibliography

[1] Chen-Chiung Hsieh ,Dung-Hua Liou and David Lee "A real time hand gesture recognition system using motion history image" Proc.IEEE, 2010, pp. V2-394-V2-398.

[2] Kamran Niyazi, Vikram Kumar , Swapnil Mahe and Swapnil Vyawahare "Mouse Simulation Using Two Coloured Tapes",IJIST 2012, Vol.2, No.2, DOI : 10.5121.

[3] Abhik Banerjee , Abhirup Ghosh , Koustuvmoni Bharadwaj , Hemanta Saikia "Mouse Control using a Web Camera based on Colour Detection"IJCTT ,March 2014, volume 9 number 1, ISSN: 2231-2803

[4] S. Thakur, R. Mehra and B. Prakash, "Vision based computer mouse control using hand gestures," International Conference on Soft Computing Techniques and Implementations (ICSCTI), 2015, pp. 85-89.

[5] Horatiu-stefan Grif, Train turc "Human hand gesture based system for mouse cursor control" INTERENG, October 2017.

[6] Pooja Kumari ,Ghaziabad Saurabh Singh, Ghaziabad Vinay and Kr. Pasi "Cursor Control using Hand Gestures" International Journal of Computer Applications (0975 – 8887),2016.

[7] Danling Lu, Yuanlong Yu, and Huaping Liu "Gesture Recognition Using Data Glove: An Extreme Learning Machine Method"Proc.IEEE,December 2016, pp. 1349-1354.

[8] Alisha Pradhan , B.B.V.L. Deepak "Design of Intangible Interface for Mouseless Computer Handling using Hand Gestures" ICCCV(International Conference on Communication, Computing and Virtualization), 2016, oi: 10.1016/j.procs.2016.03.037.

- SmartInternz
- https://ieeexplore.ieee.org/document/9935944
- www.irjet.net

APPENDIX

Code Snippet:

HandTrackModule.py

```
import cv2  # Can be installed using "pip install opencv-python"
import mediapipe as mp  # Can be installed using "pip install mediapipe"
import time
import math
```

```
import numpy as np


class handDetector():
    def __init__(self, mode=False, maxHands=2, detectionCon=False, trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.detectionCon = detectionCon
        self.trackCon = trackCon

        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode, self.maxHands,
                                        self.detectionCon, self.trackCon)
        self.mpDraw = mp.solutions.drawing_utils
        self.tipIds = [4, 8, 12, 16, 20]

    def findHands(self, img, draw=True):    # Finds all hands in a frame
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)

        if self.results.multi_hand_landmarks:
            for handLms in self.results.multi_hand_landmarks:
                if draw:
                    self.mpDraw.draw_landmarks(img, handLms,
                                               self.mpHands.HAND_CONNECTIONS)

        return img

    def findPosition(self, img, handNo=0, draw=True):   # Fetches the position of
hands
        xList = []
        yList = []
        bbox = []
        self.lmList = []
        if self.results.multi_hand_landmarks:
            myHand = self.results.multi_hand_landmarks[handNo]
            for id, lm in enumerate(myHand.landmark):
                h, w, c = img.shape
                cx, cy = int(lm.x * w), int(lm.y * h)
                xList.append(cx)
                yList.append(cy)
                self.lmList.append([id, cx, cy])
                if draw:
```

```
                    cv2.circle(img, (cx, cy), 5, (255, 0, 255), cv2.FILLED)

            xmin, xmax = min(xList), max(xList)
            ymin, ymax = min(yList), max(yList)
            bbox = xmin, ymin, xmax, ymax

            if draw:
                cv2.rectangle(img, (xmin - 20, ymin - 20), (xmax + 20, ymax +
20),
                              (0, 255, 0), 2)

        return self.lmList, bbox

    def fingersUp(self):    # Checks which fingers are up
        fingers = []
        # Thumb
        if self.lmList[self.tipIds[0]][1] > self.lmList[self.tipIds[0] - 1][1]:
            fingers.append(1)
        else:
            fingers.append(0)

        # Fingers
        for id in range(1, 5):

            if self.lmList[self.tipIds[id]][2] < self.lmList[self.tipIds[id] -
2][2]:
                fingers.append(1)
            else:
                fingers.append(0)

        # totalFingers = fingers.count(1)

        return fingers

    def findDistance(self, p1, p2, img, draw=True,r=15, t=3):   # Finds distance
between two fingers
        x1, y1 = self.lmList[p1][1:]
        x2, y2 = self.lmList[p2][1:]
        cx, cy = (x1 + x2) // 2, (y1 + y2) // 2

        if draw:
            cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), t)
            cv2.circle(img, (x1, y1), r, (255, 0, 255), cv2.FILLED)
```

```
            cv2.circle(img, (x2, y2), r, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (cx, cy), r, (0, 0, 255), cv2.FILLED)
        length = math.hypot(x2 - x1, y2 - y1)

        return length, img, [x1, y1, x2, y2, cx, cy]


def main():
    pTime = 0
    cTime = 0
    cap = cv2.VideoCapture(1)
    detector = handDetector()
    while True:
        success, img = cap.read()
        img = detector.findHands(img)
        lmList, bbox = detector.findPosition(img)
        if len(lmList) != 0:
            print(lmList[4])

        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime

        cv2.putText(img, str(int(fps)), (10, 70), cv2.FONT_HERSHEY_PLAIN, 3,
                    (255, 0, 255), 3)

        cv2.imshow("Image", img)
        cv2.waitKey(1)


if __name__ == "__main__":
    main()
```

app.py

```
import cv2
import numpy as np
import time
import HandTrackModule as htm
import autopy   # Install using "pip install autopy"
from flask import Flask, render_template, Response
app = Flask(__name__)
### Variables Declaration



# Getting the screen size
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/mouse')
def mouse():
    return render_template('Mouse.html')


def gen_frames():
    pTime = 0  # Used to calculate frame rate
    width = 640  # Width of Camera
    height = 480  # Height of Camera
    frameR = 100  # Frame Rate
    smoothening = 8  # Smoothening Factor
    prev_x, prev_y = 0, 0  # Previous coordinates
    curr_x, curr_y = 0, 0  # Current coordinates

    cap = cv2.VideoCapture(0)  # Getting video feed from the webcam
    cap.set(3, width)  # Adjusting size
    cap.set(4, height)

    detector = htm.handDetector(maxHands=1)  # Detecting one hand at max
    screen_width, screen_height = autopy.screen.size()
    while True:
        success, img = cap.read()
        img = detector.findHands(img)  # Finding the hand
        lmlist, bbox = detector.findPosition(img)  # Getting position of hand

        if len(lmlist) != 0:
```

```
            x1, y1 = lmlist[8][1:]
            x2, y2 = lmlist[12][1:]

            fingers = detector.fingersUp()  # Checking if fingers are upwards
            cv2.rectangle(img, (frameR, frameR), (width - frameR, height -
frameR), (255, 0, 255),
                          2)  # Creating boundary box
            if fingers[1] == 1 and fingers[2] == 0:  # If fore finger is up and
middle finger is down
                x3 = np.interp(x1, (frameR, width - frameR), (0, screen_width))
                y3 = np.interp(y1, (frameR, height - frameR), (0, screen_height))

                curr_x = prev_x + (x3 - prev_x) / smoothening
                curr_y = prev_y + (y3 - prev_y) / smoothening

                autopy.mouse.move(screen_width - curr_x, curr_y)  # Moving the
cursor
                cv2.circle(img, (x1, y1), 7, (255, 0, 255), cv2.FILLED)
                prev_x, prev_y = curr_x, curr_y

            if fingers[1] == 1 and fingers[2] == 1:  # If fore finger & middle
finger both are up
                length, img, lineInfo = detector.findDistance(8, 12, img)

                if length < 40:  # If both fingers are really close to each other
                    cv2.circle(img, (lineInfo[4], lineInfo[5]), 15, (0, 255, 0),
cv2.FILLED)
                    autopy.mouse.click()  # Perform Click

        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime
        cv2.putText(img, str(int(fps)), (20, 50), cv2.FONT_HERSHEY_PLAIN, 3,
(255, 0, 0), 3)
        cv2.imshow("Image", img)
        cv2.waitKey(1)
        ret, buffer = cv2.imencode('.jpg', img)
        frame = buffer.tobytes()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
```

```
@app.route('/video_feed')
def video_feed():
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace;
boundary=frame')


if __name__ == "__main__":
    app.run(debug=True)
```