AI VIRTUAL MOUSE

1) INTRODUCTION :

1.1 Overview:

All contemporary computer systems employ the mouse, which is a crucial input device. We frequently utilise input devices throughout the day because they have high contact points. The mouse is therefore covered in bacteria and germs. There is still a need to touch the gadget even though wireless mice have eliminated the necessity for a tangled mess of connections. In light of the epidemic, this proposed system makes advantage of the built-in camera or a separate web camera to record hand movements and detect fingertip movements, which may be used to perform standard mouse actions including left- and right-clicking, scrolling, and pointer operations. Based on machine learning, the algorithm. The programme is taught using deep learning so that hands can be recognised by the camera. So, by removing the necessity for human intervention and reliance on physical devices to manage the computer system, the suggested solution will stop the spread of Covid-19.

1.2 Purpose:

A lot has changed in the peripheral gadgets we use every day, including emerging fields of AR (augmented reality), many of which are becoming more portable and compact thanks to technologies like Bluetooth. This paper makes the claim that we should employ a mouse system based on artificial intelligence that can recognise hand motions and identify finger tips in order to perform mouse functions on a computer using computer vision. The primary goal of the suggested system is to replace a traditional mouse device with a built-in camera or a web camera accessory to achieve common mouse tasks like clicking and scrolling. In order to connect with the computer, hand gestures and fingertip detection are used. The web camera on the computer can be used with this system to track the fingertip to show cursor movement and conduct scrolling and cursor tasks.

2) LITERATURE SURVEY :

2.1 Existing problem :

This AI virtual mouse can be made use of in places where using a physical mouse is not feasible such as moist or wet conditions, to aid people who can’t handle a device and where there isn’t ample space for using a mouse and to eliminate the need of mouse and cable altogether. In view of the pandemic, there arises a need to avoid high-contact surfaces, out of

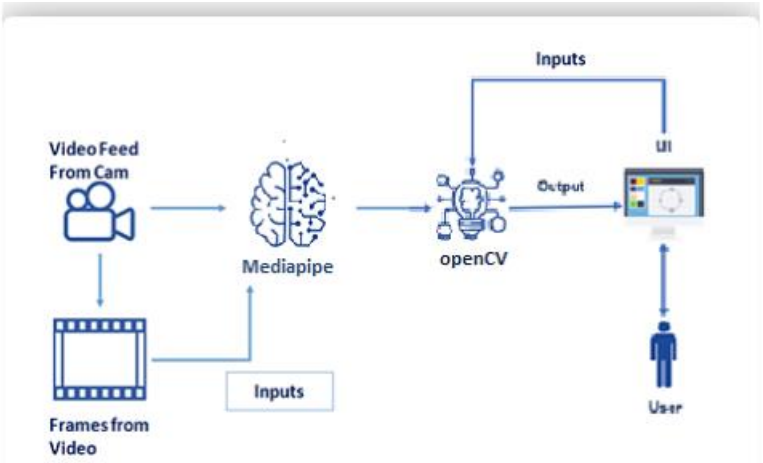
which the mouse is one. In order to eliminate the need to avoid contact, the system provides an intuitive way to interact with the computer system with the use of hand-gestures and finger tips to emulate mouse-like functions such as scrolling, clicking, and pointing.

2.2 Proposed solution :

The primary goal of the proposed virtual AI mouse is to provide a replacement for the traditional physical mouse that performs mouse functions with the aid of a computer vision enabled device that is equipped with a web camera and recognises fingers and hand gestures and processes the captured frames before using a machine learning algorithm to carry out the defined mouse functions, such as moving the cursor, right clicking, left clicking, and scrolling function. In addition, we are working on this project with other libraries.

3) THEORITICAL ANALYSIS :

3.1 Block diagram:



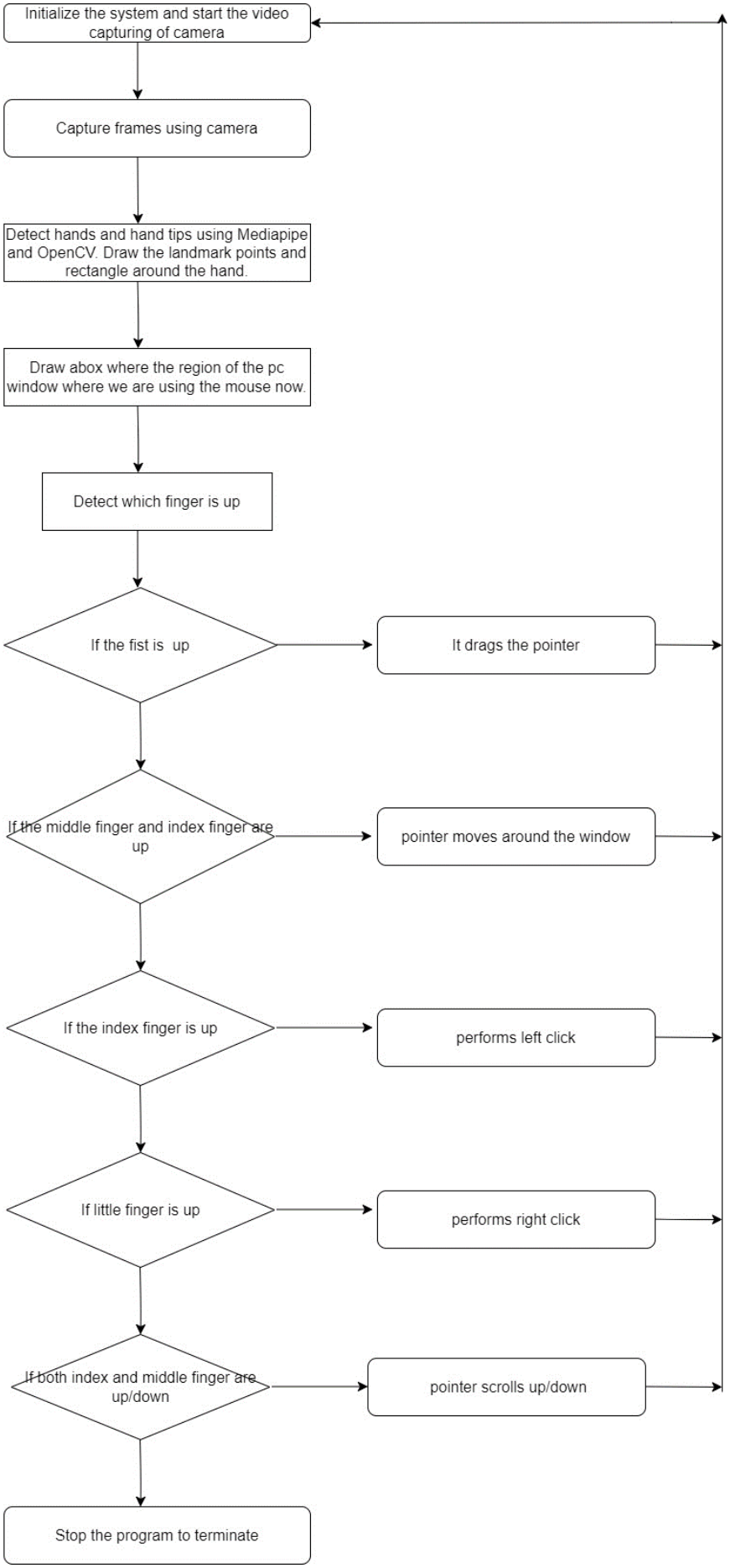
3.2 Hardware / Software designing :

1. PyCharm
2. Python(OpenCV)
3. AutoPy
4. Mediapipe0.8.3.1
5. Flask
6. Requests
7. Camera

4) EXPERIMENTAL INVESTIGATIONS :

Although such systems are not practical for explicitly performing mouse operations, work related to virtual mice has been done where the user wears a glove for the system to recognise and collect data. Another system has also been done where coloured pieces of paper are attached to hands for gesture recognition. Many users don't want to wear gloves or the gloves may not fit properly, making it difficult to recognise gloves in some situations. In contrast, employing coloured tips for gesture processing and detection may not always be accurate enough but might give the required results more efficient than the gloves.

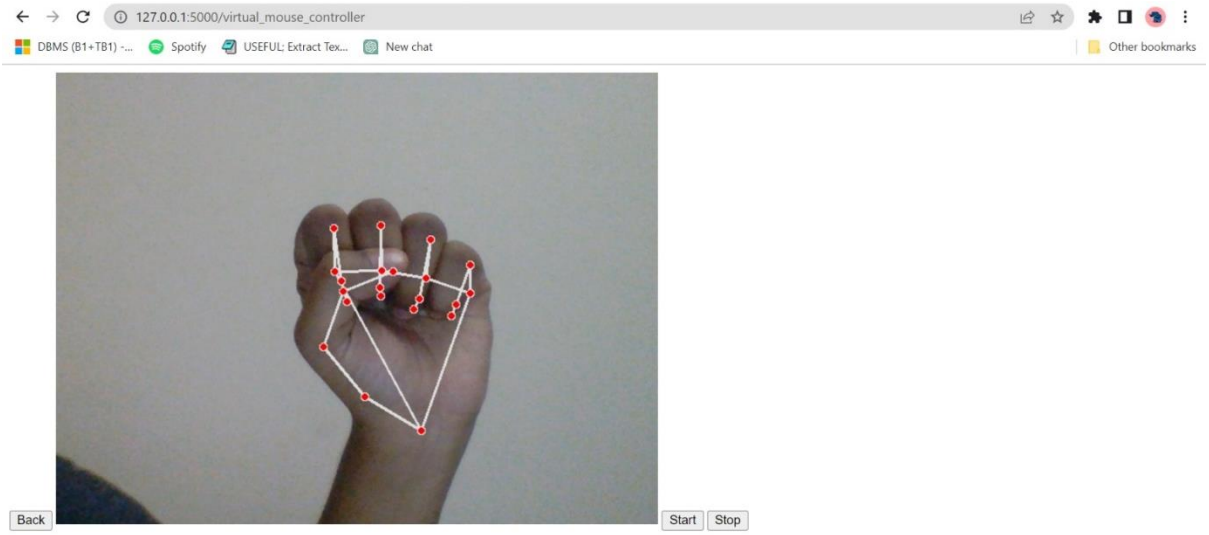
5) FLOWCHART :


Initialize the system and start the video capturing of camera
Capture frames using camera
Detect hands and hand tips using Mediapipe and OpenCV. Draw the landmark points and rectangle around the hand.
Draw abox where the region of the pc window where we are using the mouse now.
Detect which finger is up
If the fist is up
It drags the pointer
If the middle finger and index finger are up
pointer moves around the window
If the index finger is up
performs left click
If little finger is up
performs right click
If both index and middle finger are up/down
pointer scrolls up/down
Stop the program to terminate

6) RESULT:

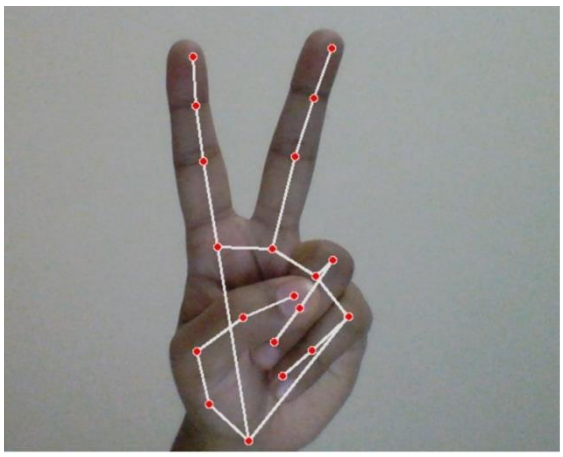
RECOGNIZING OF THE GESTURE

The hand is currently being tracked, and if any finger is being held pointed, MediaPipe recognises the finger and the tip using the 21 coordinates on the fingers. After processing the gesture, the corresponding mouse function is then handled.



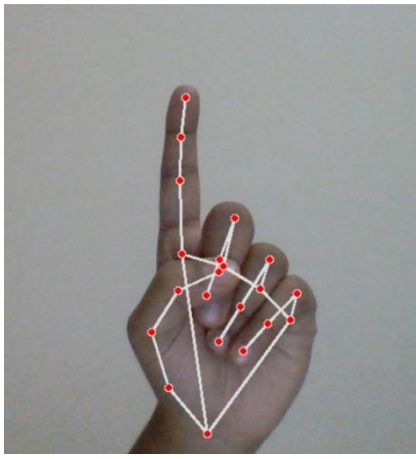
MOVING AROUND USING THE VIRTUAL MOUSE

In order to be able to utilise the computer system and operate the pointer of the virtual mouse, OpenCV recognises the hand, draws a rectangle window around the hand, and then employs a transformation method to determine the co-ordinates of the fingertips of index and middle fingers from the screen capture window. The box is painted around the fingertip when it is recognised as being associated to a specific gesture, enabling it to function as a pointer and execute basic moving functionality, as shown in the figure below.



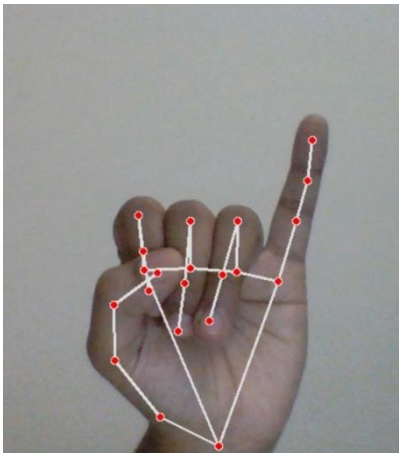
LEFT CLICK OPERATION

If the tip of the index finger is held up such that a left click is performed as shown in below figure.



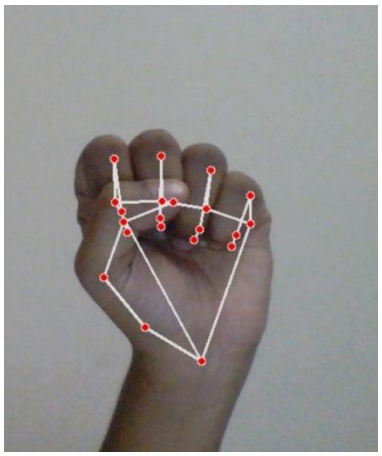
RIGHT CLICK OPERATION

If the tip of the little finger is held up such that a right click is performed as shown in below figure.



DRAG OPERATION

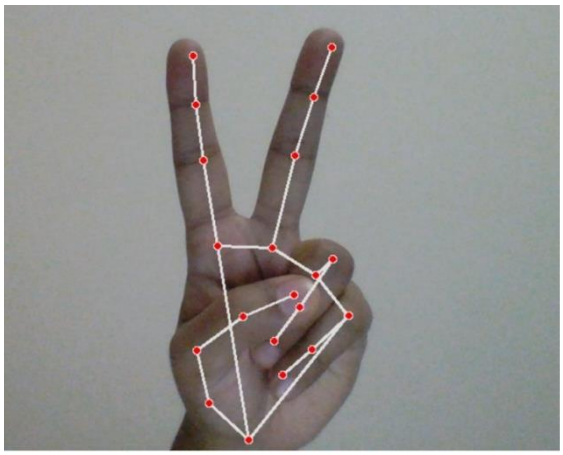
If the fist is held up such that a drag is performed as shown in below figure.



SCROLLING OPERATION

Scrolling up/down

For scrolling up/down the tips of the index and the middle finger gesture as to move from the bending position to straightening position or straightening to bending position for scroll up/down respectively as shown in below figure.



7) ADVANTAGES & DISADVANTAGES :

The AI virtual mouse system is useful for many applications; it can be used to reduce the space for using the physical mouse, and it can be used in situations where we cannot use the physical mouse. The system eliminates the usage of devices, and it improves the human-computer interaction. This AI virtual mouse has some drawbacks such as drop in accuracy of some functions like right click operation and inability to perform other mouse functions such as dragging and dropping, and selecting text. Another major limitation is that this model cannot function in the dark or low light settings. These drawbacks can be addressed in the future and can be overcome.

8) APPLICATIONS :

There are numerous application use cases for the virtual mouse powered by artificial intelligence. It can be used in situations where using a regular mouse is not an option, such as in areas where there is not enough room for one. This method eliminates the requirement for a physical mouse and allows for use without one.

Application examples

i. The precision of the suggested AI virtual mouse is about 98%, which is higher than that of other systems that have already been put into use.

ii. Without the need for a real object, it can be used in automation for robots and other systems.

iii. Drawings in two dimensions and three dimensions can be produced using the same technique.

iv. This technology can be used by those who are unable to physically operate a mouse to carry out mouse-like tasks.

v. It is advised to avoid touching high contact surfaces in this Covid-19 environment because doing so could cause the virus to spread. To control personal computers and tablet computers that have a web camera attached and can execute nearly all mouse activities, the proposed model therefore offers a workable alternative.

vi. The existing system can be expanded to function with such emerging technologies in the fields of virtual reality and augmented reality.

9) CONCLUSION :

The primary goal of the proposed virtual AI mouse is to provide an alternative to the traditional physical mouse that performs mouse functions with the aid of a computer vision enabled device that is equipped with a web camera and recognises fingers and hand gestures. This device then processes the captured frames and uses a machine learning algorithm to perform the defined mouse functions, such as moving the cursor, right clicking, left clicking, and scrolling function.

After extensive testing, we have concluded that the suggested virtual mouse system has performed remarkably well and with more precision when compared to earlier proposed models discussed in the relevant work. The present system has also outperformed the shortcomings of the earlier systems. This means that the suggested AI-based virtual mouse technology can be applied in real-world settings and in real-time. The method also helps to prevent the spread of the Covid-19 virus by removing the necessity for users to make physical contact with high-touch surfaces and gadgets by employing hand gestures rather than a traditional mouse device.

10) FUTURE SCOPE :

This AI virtual mouse has various shortcomings, such as a decrease in precision while performing specific actions, such as right-clicking, and the inability to carry out other mouse actions, including dragging and dropping objects or choosing text. This model's inability to operate in dim or low light conditions is another significant drawback. These issues can be resolved and overcame in the future.

In addition to the aforementioned, key board functionality may be added to imitate keyboard and mouse operations simultaneously, which shows the potential for the future.

11) BIBILOGRAPHY :

https://medium.com/analytics-vidhya/create-anywhere-use-flask-web-app-with-pycharm-fa67719df4d7
https://github.com/The-Assembly/Build-An-AI-Virtual-Mouse-With-OpenCV

https://github.com/j2eeexpert2015/flask-sample

APPENDIX

a)app.py :

```
import Flask, render_template, Response
from flask import request, redirect, url_for, flash
import cv2
from gesture_detection import GestureController
from threading import Thread, Event
import mediapipe as mp

mp_drawing = mp.solutions.drawing_utils
mp_hands = mp.solutions.hands
gesture_detection_active = Event()
gesture_detection_active.clear()

app = Flask(_name_)

def capture_frames(gc):
    global gesture_detection_active
    while True:
        ret, frame = gc.cap.read()
        if not ret:
            break

        frame = gc.process_frame(frame, gesture_detection_active)
        gc.frame = frame


def gen(gc):
    while True:
```

```
        ret, frame = gc.cap.read()  # getattr(gc, 'frame', None)
        frame = cv2.flip(frame, 1)

        # Draw hand landmarks on the frame
        if gc.hr_major or gc.hr_minor:
            if gc.hr_major:
                mp_drawing.draw_landmarks(frame, gc.hr_major, mp.solutions.hands.HAND_CONNECTIONS)
            if gc.hr_minor:
                mp_drawing.draw_landmarks(frame, gc.hr_minor, mp.solutions.hands.HAND_CONNECTIONS)

        ret, jpeg = cv2.imencode('.jpg', frame)
        if not ret:
            print("Failed to encode the frame")

        frame_bytes = jpeg.tobytes()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')


# Create the camera object
camera = cv2.VideoCapture(0)
gc = GestureController()
Thread(target=capture_frames, args=(gc,), daemon=True).start()


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

```
@app.route('/')
def index():
    return render_template('homepage.html')


@app.route('/settings')
def settings():
    return render_template('settings.html')  # Create a settings.html file inside the templates folder


@app.route('/virtual_mouse_controller')
def virtual_mouse_controller():
    return render_template("virtual_mouse_controller.html") # Create a virtual_mouse_controller.html file inside the templates folder


@app.route('/update_gesture_mappings', methods=['POST'])
def update_gesture_mappings():
    form_data = request.form
    print("Form data:", form_data)

    with open('mappings.txt', 'r') as f:
        lines = f.readlines()

    updated_lines = []
    used_gestures = set()
    for line in lines:
        if line.strip():
            gesture, action = line.strip().split(':')
```

```
            if action in form_data:
                updated_gesture = form_data[action]
                if updated_gesture in used_gestures:
                    return f"Error: Gesture '{updated_gesture}' is already mapped to another action. Please go back & select a different gesture.", 400
                used_gestures.add(updated_gesture)
                updated_lines.append(f"{updated_gesture}:{action}\n")
            else:
                updated_lines.append(line)
        else:
            updated_lines.append(line)

    with open('mappings.txt', 'w') as f:
        f.writelines(updated_lines)

    return redirect(url_for('settings'))


@app.route('/start_gesture_detection')
def start_gesture_detection():
    global gesture_detection_active
    gesture_detection_active.set()
    return '', 204


@app.route('/stop_gesture_detection')
def stop_gesture_detection():
    global gesture_detection_active
    gesture_detection_active.clear()
    return '', 204
```

```
if _name_ == '_main_':
    app.run()
```

b)webpage.html:

virtual_mouse_controller.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>AI Virtual Mouse</title>

</head>
<body>
    <button onclick="window.history.back();">Back</button>
    <img alt="Live Video Stream" src="{{ url_for('video_feed') }}">
    <button id="start">Start</button>
    <button id="stop">Stop</button>

    <script>
        const startButton = document.getElementById("start");
        const stopButton = document.getElementById("stop");

        startButton.addEventListener("click", () => {
            // Send a request to the server to start gesture detection
            fetch("/start_gesture_detection");
        });

        stopButton.addEventListener("click", () => {
            // Send a request to the server to stop gesture detection
```

```
            fetch("/stop_gesture_detection");
        });
    </script>
</body>
</html>
```