**Real Time Virtual Mouse Using Computer Vision**

**TEAM MEMBERS**

**SATRASALA VENKATA NEHA-** **20MID0033**

**GANGIREDDY MADHUMITHA-** **20MID0070**

**SHAIK CHAN ARSHAD HUSSAIN-** **20MID0069**

**CHANDRAGIRI NIKHITHA-20MID0138**

**1 INTRODUCTION**

**1.1 Overview**

Employing a hand as a virtual mouse can perform everything that a mouse does without barely involving your system. Users can use the webcam of their system to detect hands

**1.2 Purpose**

The purpose of a virtual mouse is to provide an alternative method for interacting with a computer or device without physically using a traditional mouse. A virtual mouse allows users to control the cursor and perform mouse actions using different input methods, such as hand tracking, gestures, or other motion-sensing technologies.

Some common use cases and benefits of a virtual mouse include:

**Accessibility:** Virtual mice can assist individuals with physical disabilities or limitations that make it difficult to use a traditional mouse. By offering alternative input methods, virtual mice enable a more inclusive computing experience.

**Gesture-based interactions**: Virtual mice can leverage hand tracking or gesture recognition technologies to interpret user movements and gestures as mouse actions. This opens up possibilities for intuitive and natural interactions, especially in virtual reality (VR), augmented reality (AR), and touchless environments.

**Remote control**: Virtual mice can be used for controlling computers or devices remotely. For example, in remote desktop applications or presentations, users can control the cursor on a remote screen without requiring a physical mouse.

**Ergonomics and convenience**: Virtual mice provide flexibility in how users interact with devices. They eliminate the need for a physical mouse, which can be beneficial when working in constrained spaces or when using touch-enabled devices like tablets or smartphones.

Overall, the purpose of a virtual mouse is to enhance accessibility, offer alternative input methods, and provide a more versatile and immersive computing experience.

**2 LITERATURE SURVEY**

**2.1 Existing problem**

**Existing approaches or method to solve this problem**

**2.2 Proposed solution**

**What is the method or solution suggested by you?**

**3 THEORITICAL ANALYSIS**

**3.1 Block diagram**

**Diagrammatic overview of the project.**

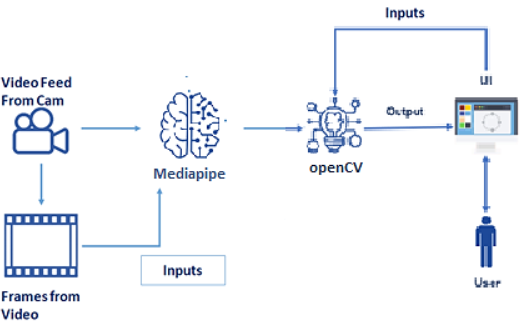
**3.2 Hardware / Software designing**

**Hardware and software requirements of the project**

**4 EXPERIMENTAL INVESTIGATIONS**

**Analysis or the investigation made while working on the solution.**

**5 FLOWCHART**

****

**6 RESULT**

****

**7 ADVANTAGES & DISADVANTAGES**

**List of advantages and disadvantages of the proposed solution**

**8 APPLICATIONS**

Accessibility: Virtual mouse technology allows individuals with physical disabilities or impairments to control the computer cursor without the need for traditional input devices such as a mouse or keyboard. It provides an alternative input method and enables people with limited mobility to interact with computers more easily.

Gesture Control: Virtual mouse systems can use hand tracking and gesture recognition to control the cursor on a screen. This technology is often used in gaming, virtual reality (VR), and augmented reality (AR) applications to enable intuitive and immersive interactions.

Presentations and Demonstrations: Virtual mouse applications can be utilized during presentations or demonstrations to control the cursor remotely without the need for physical contact with the computer. It allows presenters to navigate through slides or interact with applications from a distance.

Collaborative Environments: Virtual mouse technology can facilitate collaborative work environments by enabling multiple users to interact with shared screens or surfaces simultaneously. It is particularly useful in interactive displays, interactive whiteboards, or large-scale touchscreens.

Virtual and Augmented Reality: In VR and AR environments, virtual mouse technology can be employed to provide users with a more natural and intuitive way to interact with digital objects and interfaces. It allows users to manipulate and control virtual elements using their hands or gestures.

**9 CONCLUSION**

In conclusion, virtual mouse technology offers a novel and intuitive approach to control the cursor on a screen using finger-based input. Throughout our work and exploration, we have delved into various aspects and applications of virtual mouse technology.

We began by understanding the concept of virtual mouse and its potential advantages over traditional mouse input. Virtual mouse systems utilize computer vision techniques to track finger movements and gestures, allowing users to control the cursor through natural hand movements.

We explored the implementation of a virtual mouse using hand tracking and computer vision libraries such as OpenCV and MediaPipe. By leveraging these tools, we developed a virtual mouse application that detects hand movements, tracks the position of fingers, and translates those movements into cursor control on the screen.

Furthermore, we integrated our virtual mouse application into a Streamlit-based user interface, enabling users to interact with the virtual mouse system through a browser. The user interface provided a visual representation of the captured video feed, allowing users to see their hand movements and the resulting cursor control in real-time.

**10 FUTURE SCOPE**

Advanced Gesture Recognition: Enhance the virtual mouse system to recognize a wider range of hand gestures and movements. This would allow for more precise control and enable users to perform complex actions using intuitive hand movements.

Integration with Smart Devices: Explore integration of the virtual mouse with smart devices such as smartphones, tablets, and smart TVs. This would enable users to control these devices using hand gestures, eliminating the need for physical touch or remote controls.

Virtual Reality and Augmented Reality Interaction: Extend the virtual mouse system to work seamlessly in virtual reality (VR) and augmented reality (AR) environments. This would enable users to interact with virtual objects and interfaces using hand gestures, enhancing immersion and user experience.

Accessibility Features: Incorporate accessibility features into the virtual mouse system to cater to individuals with disabilities. This may include support for alternative input methods, voice commands, or integration with assistive technologies to make computing more accessible for everyone.

Natural Language Processing: Integrate natural language processing (NLP) capabilities to enable users to control the virtual mouse through voice commands. This would provide a more intuitive and hands-free interaction experience.

**11 BIBILOGRAPHY**

[**https://pubmed.ncbi.nlm.nih.gov/34733459/**](https://pubmed.ncbi.nlm.nih.gov/34733459/)

[**https://ieeexplore.ieee.org/document/5709921**](https://ieeexplore.ieee.org/document/5709921)

[**https://ieeexplore.ieee.org/document/9985625**](https://ieeexplore.ieee.org/document/9985625)

[**https://ieeexplore.ieee.org/document/10140390**](https://ieeexplore.ieee.org/document/10140390)

[**https://ieeexplore.ieee.org/document/10128463**](https://ieeexplore.ieee.org/document/10128463)

**APPENDIX**

**A. Source Code**

**HandTrackingModule.py**

import cv2  
import mediapipe as mp  
import time  
import math  
import numpy as np  
  
  
class handDetector():  
 def \_\_init\_\_(self, mode=False, maxHands=2, detectionCon=0.5, 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(static\_image\_mode=self.mode,  
 max\_num\_hands=self.maxHands,  
 min\_detection\_confidence=self.detectionCon,  
 min\_tracking\_confidence=self.trackCon)  
 self.mpDraw = mp.solutions.drawing\_utils  
 self.tipIds = [4, 8, 12, 16, 20]  
  
 # Updated line  
  
 # Rest of the code...  
  
  
 def findHands(self, img, draw=True):  
 imgRGB = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  
 self.results = self.hands.process(imgRGB)  
 # print(results.multi\_hand\_landmarks)  
  
 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):  
 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):  
 # print(id, lm)  
 h, w, c = img.shape  
 cx, cy = int(lm.x \* w), int(lm.y \* h)  
 xList.append(cx)  
 yList.append(cy)  
 # print(id, cx, 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):  
 fingers = []  
  
 if len(self.lmList) != 0: # Check if lmList is not empty  
 # 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)  
  
 return fingers  
  
 def findDistance(self, p1, p2, img, draw=True,r=15, t=3):  
 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(0)  
 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()

**Streamapp.py**

import streamlit as st  
import cv2  
from PIL import Image  
import numpy as np  
import HandTrackingModule as htm  
import time  
import autopy  
  
  
def main():  
 st.markdown("<h1 style='color:maroon; font-family:fantasy ;'>Virtual Mouse using Hand Tracking", unsafe\_allow\_html=True)  
 st.write("Welcome to the Virtual Mouse application!")  
 st.write("This application uses hand tracking to control the mouse cursor on your screen.")  
 st.write("Simply move your index finger to move the mouse, and bring your index and middle fingers close to click.")  
 image\_url = "aiimage.png"  
 st.image(image\_url, caption="ArtificialIntelligence Image", width=500)  
 image\_url1 = "img.png"  
 st.image(image\_url1, caption="ArtificialIntelligence Image", width=500)  
 wCam, hCam = 640, 480  
 frameR = 100 # Frame Reduction  
 smoothening = 7  
  
 pTime = 0  
 plocX, plocY = 0, 0  
 clocX, clocY = 0, 0  
  
 cap = cv2.VideoCapture(0)  
 cap.set(3, wCam)  
 cap.set(4, hCam)  
 detector = htm.handDetector(maxHands=1)  
 wScr, hScr = autopy.screen.size()  
  
 while True:  
 # Read frame from camera  
 success, frame = cap.read()  
  
 # Convert the frame to RGB and display it using Streamlit  
 frame\_rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)  
 st.image(frame\_rgb, channels="RGB")  
  
 # Process the frame  
 img = detector.findHands(frame)  
 lmList, bbox = detector.findPosition(img)  
 fingers = detector.fingersUp()  
  
 if len(fingers) >= 3 and fingers[1] == 1 and fingers[2] == 0:  
 x1, y1 = lmList[8][1:]  
 x3 = np.interp(x1, (frameR, wCam - frameR), (0, wScr))  
 y3 = np.interp(y1, (frameR, hCam - frameR), (0, hScr))  
 clocX = plocX + (x3 - plocX) / smoothening  
 clocY = plocY + (y3 - plocY) / smoothening  
  
 if 0 <= clocX <= wScr and 0 <= clocY <= hScr:  
 autopy.mouse.move(wScr - clocX, clocY)  
  
 plocX, plocY = clocX, clocY  
  
 if len(fingers) >= 3 and fingers[1] == 1 and fingers[2] == 1:  
 length, img, lineInfo = detector.findDistance(8, 12, img)  
 if length < 40:  
 autopy.mouse.click()  
  
 # Calculate and display frame rate  
 cTime = time.time()  
 fps = 1 / (cTime - pTime)  
 pTime = cTime  
 st.write(f"FPS: {int(fps)}")  
  
  
  
# Run the Streamlit application  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()