Hand Gesture Virtual Mouse

Python Computer Vision Implementation using MediaPipe, OpenCV & PyAutoGUI

Language: Python 3.8+ **Libraries:** OpenCV, MediaPipe, PyAutoGUI, NumPy **Category:** Computer Vision / HCI

System Architecture & Dependencies

OpenCV

Captures video feed from the webcam and handles image transformations.

MediaPipe

Extracts 21 3D hand landmarks in real-time with high efficiency.

PyAutoGUI

Maps landmark positions to screen coordinates and triggers mouse clicks.

NumPy & Math

Performs linear interpolation (`interp`) and distance calculations.

Gesture Control Mapping

Action	Gesture Trigger	Visual Indicator
Move Cursor	Move Index Finger Tip (Landmark 8)	Blue Boundary Box
Left Click	Pinch Index Finger Tip & Thumb Tip (< 30 px)	Green Filled Circle
Right Click	Pinch Middle Finger Tip & Thumb Tip (< 30 px)	Red Filled Circle
Quit Program	Press key 'q' on active OpenCV window	Program Exits

Complete Source Code

```
import cv2
import mediapipe as mp
import pyautogui
import numpy as np
import math

# Configure PyAutoGUI
pyautogui.FAILSAFE = False
screen_w, screen_h = pyautogui.size()

# Initialize MediaPipe Hands
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(
    max_num_hands=1,
    min_detection_confidence=0.7,
    min_tracking_confidence=0.7
)
mp_draw = mp.solutions.drawing_utils

# Setup Webcam
cap = cv2.VideoCapture(0)
```

```python
# Smoothness & Mapping Variables
prev_x, prev_y = 0, 0
curr_x, curr_y = 0, 0
smoothing = 5
margin = 100

def distance(p1, p2):
    return math.hypot(p2[0] - p1[0], p2[1] - p1[1])

while cap.isOpened():
    success, frame = cap.read()
    if not success:
        break

    frame = cv2.flip(frame, 1)
    h, w, _ = frame.shape

    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = hands.process(rgb_frame)

    if results.multi_hand_landmarks:
        for hand_landmarks in results.multi_hand_landmarks:
            mp_draw.draw_landmarks(frame, hand_landmarks, mp_hands.HAND_CONNECTIONS)

            landmarks = [(int(lm.x * w), int(lm.y * h)) for lm in hand_landmarks.landmark]

            index_tip = landmarks[8]
            thumb_tip = landmarks[4]
            middle_tip = landmarks[12]

            # Coordinate mapping to screen dimensions
            x_mapped = np.interp(index_tip[0], (margin, w - margin), (0, screen_w))
            y_mapped = np.interp(index_tip[1], (margin, h - margin), (0, screen_h))

            # Smoothing motion using Exponential Moving Average
            curr_x = prev_x + (x_mapped - prev_x) / smoothing
            curr_y = prev_y + (y_mapped - prev_y) / smoothing

            pyautogui.moveTo(curr_x, curr_y)
            prev_x, prev_y = curr_x, curr_y

            # Left Click Trigger (Index + Thumb Pinch)
            if distance(index_tip, thumb_tip) < 30:
                cv2.circle(frame, index_tip, 12, (0, 255, 0), cv2.FILLED)
                pyautogui.click()
                pyautogui.sleep(0.15)

            # Right Click Trigger (Middle + Thumb Pinch)
            if distance(middle_tip, thumb_tip) < 30:
                cv2.circle(frame, middle_tip, 12, (0, 0, 255), cv2.FILLED)
                pyautogui.rightClick()
                pyautogui.sleep(0.2)

    cv2.rectangle(frame, (margin, margin), (w - margin, h - margin), (255, 255, 0), 2)
    cv2.imshow("Hand Mouse", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Installation & Execution

```
# Step 1: Install required dependencies
pip install opencv-python mediapipe pyautogui numpy

# Step 2: Run the python script
python virtual_mouse.py
```