AIROCURSOR

AiroCursor is a smart virtual mouse powered by hand gestures and computer vision. Using just a webcam, it tracks your hand movements and interprets natural gestures to control your computer — no physical mouse needed! With AiroCursor, you can move the cursor, click, scroll, take screenshots, and even record your screen, all by performing simple hand gestures in the air. It's built using Python, OpenCV, and MediaPipe for real-time gesture recognition.

Key Features:

- 🖱️ Cursor movement with your index finger
- 👆 Left, right, and double click using hand poses
- 📜 Scroll using thumbs-up or thumbs-down gestures
- 📸 Take screenshots with a three-finger sign
- 🎥 Start/stop screen recording with hand gestures

Program :

```
import cv2
import numpy as np
import mediapipe as mp
import pyautogui
import random
import os
import util
from pynput.mouse import Button, Controller
from datetime import datetime

mouse = Controller()
screen_width, screen_height = pyautogui.size()

mpHands = mp.solutions.hands
hands = mpHands.Hands(
    static_image_mode=False,
    model_complexity=1,
    min_detection_confidence=0.7,
    min_tracking_confidence=0.7,
    max_num_hands=1
)

# Create folders for screenshots and recordings if they don't exist
if not os.path.exists("screenshots"):
```

```
    os.makedirs("screenshots")
if not os.path.exists("recordings"):
    os.makedirs("recordings")

recording = False
out = None

def find_finger_tip(processed):
    if processed.multi_hand_landmarks:
        hand_landmarks = processed.multi_hand_landmarks[0]
        index_finger_tip =
hand_landmarks.landmark[mpHands.HandLandmark.INDEX_FINGER_TIP]
        return index_finger_tip
    return None, None

def count_raised_fingers(landmark_list):
    if len(landmark_list) < 21:
        return 0
    fingers = [8, 12, 16, 20]
    count = sum(landmark_list[f][1] < landmark_list[f - 2][1] for f in fingers)
    return count

def move_mouse(index_finger_tip):
    if index_finger_tip is not None:
        x = int(index_finger_tip.x * screen_width)
        y = int(index_finger_tip.y / 2 * screen_height)
        pyautogui.moveTo(x, y)

def is_left_click(landmark_list, thumb_index_dist):
    return (
        util.get_angle(landmark_list[5], landmark_list[6], landmark_list[8]) < 50 and
        util.get_angle(landmark_list[9], landmark_list[10], landmark_list[12]) > 90 and
        thumb_index_dist > 50
    )

def is_right_click(landmark_list, thumb_index_dist):
    return (
        util.get_angle(landmark_list[9], landmark_list[10], landmark_list[12]) < 50 and
        util.get_angle(landmark_list[5], landmark_list[6], landmark_list[8]) > 90 and
        thumb_index_dist > 50
    )

def is_double_click(landmark_list, thumb_index_dist):
    return (
        util.get_angle(landmark_list[5], landmark_list[6], landmark_list[8]) < 50 and
        util.get_angle(landmark_list[9], landmark_list[10], landmark_list[12]) < 50 and
        thumb_index_dist > 50
    )
```

```
def is_screenshot(landmark_list):
    return count_raised_fingers(landmark_list) == 3

def is_start_recording(landmark_list):
    return count_raised_fingers(landmark_list) == 4

def is_stop_recording(landmark_list):
    return count_raised_fingers(landmark_list) == 0

def is_scroll_up(landmark_list):
    if len(landmark_list) < 21:
        return False
    # Thumb pointing up and all other fingers closed
    thumb_up = landmark_list[4][1] < landmark_list[3][1]  # Thumb tip above joint
    fingers_closed = all(landmark_list[tip][1] > landmark_list[tip - 2][1] for tip in [8, 12, 16, 20])
    return thumb_up and fingers_closed

def is_scroll_down(landmark_list):
    if len(landmark_list) < 21:
        return False
    # Thumb pointing down and all other fingers closed
    thumb_down = landmark_list[4][1] > landmark_list[3][1]  # Thumb tip below joint
    fingers_closed = all(landmark_list[tip][1] > landmark_list[tip - 2][1] for tip in [8, 12, 16, 20])
    return thumb_down and fingers_closed

def detect_gesture(frame, landmark_list, processed):
    global recording, out
    if len(landmark_list) >= 21:
        index_finger_tip = find_finger_tip(processed)
        thumb_index_dist = util.get_distance([landmark_list[4], landmark_list[5]])

        if util.get_distance([landmark_list[4], landmark_list[5]]) < 50 and
util.get_angle(landmark_list[5], landmark_list[6], landmark_list[8]) > 90:
            move_mouse(index_finger_tip)
        elif is_left_click(landmark_list, thumb_index_dist):
            mouse.press(Button.left)
            mouse.release(Button.left)
            cv2.putText(frame, "Left Click", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        elif is_right_click(landmark_list, thumb_index_dist):
            mouse.press(Button.right)
            mouse.release(Button.right)
            cv2.putText(frame, "Right Click", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
        elif is_double_click(landmark_list, thumb_index_dist):
            pyautogui.doubleClick()
            cv2.putText(frame, "Double Click", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255,
0), 2)
        elif is_scroll_up(landmark_list):
```

```
        pyautogui.scroll(30)
        cv2.putText(frame, "Scroll Up", (50, 150), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 200, 255),
2)
    elif is_scroll_down(landmark_list):
        pyautogui.scroll(-30)
        cv2.putText(frame, "Scroll Down", (50, 150), cv2.FONT_HERSHEY_SIMPLEX, 1, (200, 100,
255), 2)
    elif is_screenshot(landmark_list):
        im1 = pyautogui.screenshot()
        label = random.randint(1, 1000)
        im1.save(f'screenshots/my_screenshot_{label}.png')
        cv2.putText(frame, "Screenshot Taken", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,
255, 0), 2)
    elif is_start_recording(landmark_list) and not recording:
        filename = f"recordings/recording_{datetime.now().strftime('%Y%m%d_%H%M%S')}.avi"
        fourcc = cv2.VideoWriter_fourcc(*"XVID")
        out = cv2.VideoWriter(filename, fourcc, 20.0, (screen_width, screen_height))
        recording = True
        cv2.putText(frame, "Recording Started", (50, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,
255, 255), 2)
    elif is_stop_recording(landmark_list) and recording:
        recording = False
        out.release()
        out = None
        cv2.putText(frame, "Recording Stopped", (50, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0,
255), 2)

def main():
  global recording, out
  draw = mp.solutions.drawing_utils
  cap = cv2.VideoCapture(0)
  try:
    while cap.isOpened():
      ret, frame = cap.read()
      if not ret:
        break
      frame = cv2.flip(frame, 1)
      frameRGB = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
      processed = hands.process(frameRGB)

      landmark_list = []
      if processed.multi_hand_landmarks:
        hand_landmarks = processed.multi_hand_landmarks[0]
        draw.draw_landmarks(frame, hand_landmarks, mpHands.HAND_CONNECTIONS)
        for lm in hand_landmarks.landmark:
          landmark_list.append((lm.x, lm.y))

      detect_gesture(frame, landmark_list, processed)
```

```
        if recording and out is not None:
            screenshot = pyautogui.screenshot()
            frame_np = cv2.cvtColor(np.array(screenshot), cv2.COLOR_RGB2BGR)
            out.write(frame_np)

        cv2.imshow('Frame', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    finally:
        cap.release()
        cv2.destroyAllWindows()
        if recording and out is not None:
            out.release()

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

CODE EXPLANATION :

What this code does:

- Uses your webcam to see your hand.
- Recognizes different hand gestures (like thumbs up, fingers up, closed fist).
- Depending on the gesture, it:
    - Moves the mouse
    - Clicks (left/right/double)
    - Scrolls up/down
    - Takes a screenshot
    - Starts/stops screen recording

Libraries used and why:

Library	What it does
cv2 (OpenCV)	Opens the webcam and handles image/video processing
numpy	Works with image data in number format

Library	What it does
mediapipe	Detects hand landmarks (like finger tips and joints)
pyautogui	Controls the mouse, takes screenshots, scrolls, etc.
random	Creates random labels for screenshot filenames
os	Handles folders and files (like creating screenshot folder)
util	A custom helper file that calculates distances and angles
pynput.mouse	Clicks left or right mouse buttons
datetime	Gives current date/time to name recordings properly

Key Parts of the Code

1. Setup and initialization

mouse = Controller()

screen_width, screen_height = pyautogui.size()

- This sets up the mouse and gets your screen size to help with cursor movement.

---

2. MediaPipe hands setup

hands = mpHands.Hands(...)

- Tells MediaPipe to detect only one hand, and sets the detection and tracking confidence.

---

3. Create folders

```
if not os.path.exists("screenshots"):
    os.makedirs("screenshots")
```

- Makes folders named screenshots and recordings to store output files.

---

4. Hand gesture recognition

find_finger_tip(processed)

- Finds where your index finger tip is.

count_raised_fingers(landmark_list)

- Counts how many fingers (except the thumb) are raised.

move_mouse(index_finger_tip)

- Moves your mouse cursor to match where your index finger is pointing.

---

5. Gesture functions

These functions recognize specific gestures based on finger positions and angles.

Function	What it detects
is_left_click()	Index finger bent in, middle straight (click)
is_right_click()	Middle finger bent in, index straight (right click)
is_double_click()	Both index and middle bent in
is_screenshot()	Three fingers raised
is_start_recording()	Four fingers raised
is_stop_recording()	Fist (no fingers up)
is_scroll_up()	Only thumb up, rest closed
is_scroll_down()	Only thumb down, rest closed

These use finger landmark coordinates and angles to detect what gesture you're making.

---

6. Gesture handling

def detect_gesture(frame, landmark_list, processed):

This function checks what gesture you're making and takes an action:

- Move the mouse
- Left/right/double click
- Scroll
- Screenshot
- Start/stop recording

It also writes a label on the webcam window to show what it’s doing (like “Scroll Down”).

---

7. Main loop

def main():

- Turns on your webcam
- Keeps checking each frame (image from webcam)
- If a hand is detected, it draws the hand and calls detect_gesture()
- If recording is on, it saves screenshots as video frames
- Ends when you press ‘q’
