**Cursor Movement with Hand Tracking Using AI**

Enable control of the mouse cursor using **hand gestures**, especially tracking the **index fingertip**, through AI-based hand tracking in real time.

**Python**

the complete Python implementation of the AI-based hand tracking cursor movement project, incorporating software engineering best practices — clean structure, modularity, comments, and error handling.

**Python Variables and Functions – Basics + Project Context**

**1. Variable Declaration in Python**

**General Syntax:**

variable\_name = value

**Best Practices:**

* Use camelCase for variable and function names
* Use UPPER\_CASE for constants
* Prefix booleans with is, has, can, etc.
* Avoid reserved words like list, str, input

**Examples from the project:**

screenWidth, screenHeight = pyautogui.size() # camelCase

isTrackingEnabled = True # boolean naming

FRAME\_WIDTH = 640 # constant (UPPER\_CASE)

**2. Function Declaration in Python**

**General Syntax:**

def function\_name(parameters):

# function logic

return result

**3. Technologies Used**

* **Python**: Programming language for the implementation.
* **MediaPipe**: An AI-based framework by Google for real-time hand tracking.
* **OpenCV**: For capturing video frames from the webcam and image processing.
* **PyAutoGUI**: For controlling the system mouse (cursor) with Python.

**Key Concepts Involved**

**a. Hand Landmark Detection**

* MediaPipe detects **21 landmark points** on each hand.
* Each point has **x, y, z coordinates** (x and y are relative to the video frame).
* Landmark id = 8 refers to the **index finger tip**.

**b. Coordinate Mapping**

* Camera frame coordinates are different from screen resolution.
* We **scale** the hand coordinates from camera space to screen space to match the actual cursor position.

**c. Real-Time Video Processing**

* Video is captured frame-by-frame using the webcam.
* Each frame is processed to:
  + Detect the hand
  + Identify landmarks
  + Track the movement of the fingertip

**4. Step-by-Step Workflow**

**Step 1: Capture webcam video**

* Use OpenCV (cv2.VideoCapture) to access the webcam.

**Step 2: Detect hand using MediaPipe**

* Convert frame to RGB (required by MediaPipe).
* Run MediaPipe’s hand detection model.

**Step 3: Track index finger**

* Access landmark #8 (index fingertip).
* Extract its coordinates from the frame.

**Step 4: Map camera coordinates to screen coordinates**

* Use screen resolution to map the (x, y) values.
* Cursor should move to the equivalent point on the actual screen.

**Step 5: Move mouse cursor**

* Use PyAutoGUI to programmatically move the mouse to the calculated screen position.

**5. Benefits of This Approach**

* Touchless control – hygienic and futuristic.
* Useful for accessibility and gesture-based interfaces.
* Can be integrated with kiosks, presentations, games, or AR apps.

**7. Limitations**

* Dependent on **good lighting** and camera quality.
* May have **latency** on low-end systems.
* **Accuracy** may vary based on hand position, background, and frame resolution.

**4. Error Handling**

**Why:** Prevents unexpected crashes.

* Add try-except blocks, especially around:
  + Webcam initialization
  + Landmark extraction
  + Mouse movement

**9. Scalable Architecture**

**Why:** Prepares codebase for future features (e.g., gestures, UI, multi-hand tracking).

* Follow a loosely MVC-like structure:
  + **Model:** Hand detection and tracking logic
  + **View:** Webcam window display
  + **Controller:** Maps finger to cursor

# **10.Use of Software Engineering**

* Structured Code Organization
* Readability & Maintainability
* Error Handling
* Reusability
* Testing and Debugging
* Performance Optimization
* Scalability
* Documentation
* Professionalism