Real-Time Hand Gesture Recognition

This project implements a real-time static hand gesture recognition system using a standard webcam. It leverages a rule-based approach on top of a powerful hand-tracking library to accurately identify a vocabulary of seven distinct gestures. The application also includes detection smoothing to provide a stable and user-friendly experience.

The recognized gestures are:

- 👍 Thumbs Up
- ✊ Fist
- ✌□ Peace
- 👌 OK
- 🖐□ Open Palm
- ☝ Pointing
- □ Crossed Fingers

Your Full Name

[Your Full Name Here]

Technology Justification

The core technologies chosen for this project were **MediaPipe**, **OpenCV**, and **NumPy**. This combination provides a robust, high-performance, and accessible framework for solving real-time computer vision problems.

- **MediaPipe**: Google's MediaPipe framework was selected for the critical task of hand detection and landmark extraction.
  - **State-of-the-Art Accuracy**: It provides 21 high-fidelity 3D landmarks for each detected hand, forming a detailed skeletal model. This completely eliminates the need to gather data and train a custom deep learning model.
  - **Exceptional Performance**: MediaPipe is highly optimized for real-time inference on standard CPU hardware, making the application fast and responsive without requiring a dedicated GPU.
  - **Robustness**: The model is pre-trained on a vast and diverse dataset, allowing it to perform reliably under various lighting conditions and with different hand sizes and skin tones.
- **OpenCV (Open Source Computer Vision Library)**: OpenCV serves as the backbone for all video I/O and image manipulation.
  - **Camera Interfacing**: It provides the essential cv2.VideoCapture class to effortlessly capture live video frames from a webcam.
  - **Image Processing & Rendering**: It is used for necessary pre-processing steps (like converting image color spaces from BGR to RGB for MediaPipe) and for rendering the final output, including drawing the hand landmarks, bounding boxes, and overlaying the recognized gesture text onto the video feed.
- **NumPy**: This library is fundamental for performing efficient numerical computations. In this project, it is used to convert landmark data into arrays for easy manipulation and to calculate Euclidean distances between key points, which is crucial for gestures like "OK" and "Crossed Fingers".

This technology stack represents an industry-standard approach, allowing the project to focus entirely on the gesture recognition logic itself, rather than the complex underlying task of hand tracking.

Gesture Logic Explanation

The system identifies gestures using a set of rules based on the geometric relationships between the 21 hand landmarks provided by MediaPipe. The core logic for most gestures is built upon a simple helper function that determines if each finger is **extended** or **curled**.

A finger is considered **"extended"** if its tip's vertical coordinate is higher on the screen than its middle (PIP) joint's coordinate. This simple but effective heuristic works well for a front-facing, upright hand.

Here is the specific methodology for four of the required gestures:

👍 Thumbs Up

1. **Thumb Position**: The thumb tip (landmark #4) must be pointing upwards, meaning its Y-coordinate is less than the Y-coordinate of the joint below it (landmark #3).
2. **Thumb Extension**: A custom check confirms the thumb is extended outwards from the palm by comparing its distance to the wrist.
3. **Fingers Curled**: The other four fingers (index, middle, ring, and pinky) must all be curled. This is verified by checking that none of them meet the criteria for being "extended."

✊ Fist

1. **All Fingers Curled**: This gesture is recognized when the logic determines that **all five fingers** (including the thumb) are in a curled or non-extended state. It is a foundational check that is given high priority in the classification sequence.

✌□ Peace Sign

1. **Index & Middle Fingers Extended**: The index finger (tip #8) and middle finger (tip #12) must both be extended.
2. **Ring & Pinky Fingers Curled**: The ring finger (tip #16) and pinky finger (tip #20) must both be curled. The state of the thumb does not affect this gesture's detection.

👌 OK Sign

1. **Tip Proximity**: This gesture is uniquely identified by the proximity of the thumb tip (#4) and the index fingertip (#8). The **Euclidean distance** between these two points is calculated in normalized coordinates.
2. **Dynamic Threshold**: This distance is compared against a dynamic threshold. Instead of a fixed pixel value, the threshold is a fraction of the hand's bounding box diagonal size (0.18 * hand_diagonal). This clever approach makes the detection robustly scale with the hand's distance from the camera. If the distance is below this threshold, the gesture is confirmed.

Detection Smoothing

To prevent the output text from flickering rapidly between different classifications, the application uses a smoothing buffer (deque). It stores the last 6 detected gestures and only updates the final displayed gesture if a single gesture becomes the majority in that buffer. This results in a much more stable and user-friendly output.

Setup and Execution Instructions

Follow these steps to set up the environment and run the hand gesture recognition application.

1. Prerequisites

- Python 3.8 or newer installed on your system.
- A webcam connected to your computer.

2. Clone or Download the Code

If using Git, clone the repository to your local machine:

```
git clone <your-repository-url>
cd <repository-directory>
```

Alternatively, just download the main.py script and save it in a new folder.

3. Create a Virtual Environment (Recommended)

Using a virtual environment is the best practice for managing project dependencies.

- **On macOS / Linux:**
  ```
  python3 -m venv venv
  source venv/bin/activate
  ```

- **On Windows:**
  ```
  python -m venv venv
  .\venv\Scripts\activate
  ```

4. Install Dependencies

Install the required Python libraries using pip:

```
pip install opencv-python mediapipe numpy
```

5. Run the Application

Execute the script from your terminal:

```
python main.py
```

A window will pop up showing your webcam feed. Position your hand in the frame to see the detected landmarks and the recognized gesture displayed in the top-left corner.

To quit the application, press the ESC key.