Implementing a Gesture-Control and Math-Solving Project

Description: The project aims to build an intelligent system that uses hand gestures to control a PC and simultaneously recognizes and solves handwritten math problems. The system will integrate two main functionalities:

Gesture-Based PC Control: Use hand gestures to interact with the computer, performing actions like scrolling, clicking, zooming, and adjusting system volume or brightness.

Handwritten Math Problem Recognition and Solving: Recognize handwritten mathematical problems and solve them using symbolic computation.

Key Functionalities:

1. **Gesture Recognition:**
    - Input: Hand gestures detected using a camera.
    - Output: Interactions with the computer, including:
        - Scrolling (up/down).
        - Clicking (left/right).
        - Zooming in/out.
        - Volume and brightness control.
    - Gesture Actions:
        - Scroll Up/Down: Using a hand motion or pinch gesture.
        - Click: Using a "pinch" gesture.
        - Zoom: Open and close fingers.
        - Volume/Brightness Control: Specific gestures will increase or decrease these settings.
2. **Math Recognition and Solving:**
    - Input: Handwritten math problems (captured by camera).
    - Output: Solved mathematical expression using a symbolic computation library.
    - Features:
        - Recognizing common arithmetic problems.
        - Solving equations.
        - Displaying results after parsing and solving.

Algorithms Used:

1. **Hand Gesture Recognition:**
    - **Mediapipe:** A library for real-time hand tracking and gesture recognition. It uses machine learning models to detect 21 key landmarks on a hand and can distinguish different hand gestures.
        - Hand Landmarks Detection: The system identifies the positions of key hand landmarks and calculates features like angles, distances, and relative positions between them.
        - Gesture Classification: Custom algorithms based on finger states (open/closed) and distances between landmarks classify gestures like a fist, pinch, or spread fingers.
2. **Optical Character Recognition (OCR):**
    - **Tesseract OCR:** Used to recognize handwritten math problems from images captured by the camera. Tesseract is trained to detect text in various fonts and handwriting styles.
    - Preprocessing:
        - Grayscale Conversion: Simplifies the image.
        - Thresholding and Noise Removal: Improves OCR accuracy.
3. **Mathematical Computation:**
    - **SymPy:** A Python library for symbolic mathematics. It allows us to parse mathematical expressions, solve equations, and evaluate results symbolically.
    - **Math Problem Solving:** After OCR recognition, SymPy is used to handle basic arithmetic and solve more complex algebraic equations.

Datasets Used:

Hand Gesture Data:

- Mediapipe's Hand Tracking Model.
- Custom Gesture Dataset.

Handwritten Math Dataset:

- CROHME (Competition on Recognition of Handwritten Mathematical Expressions).
- Custom Math Problem Dataset.

Libraries:

- **OpenCV**: For image processing and camera interaction.

- **Mediapipe**: For hand gesture recognition and tracking.
- **Pytesseract**: For OCR.
- **SymPy**: For solving mathematical expressions.
- **PyAutoGUI**: For controlling the mouse and keyboard.
- **Pycaw**: For controlling system volume.

Week 3: Understanding the Scope and Setting Up the Environment

Goals:

- Define project requirements and functionality:
    - Gesture-based PC control (scrolling, clicking, zooming, etc.).
    - Handwritten math problem recognition and solution.
- Set up the development environment.

Week 4: Implementing Hand Tracking

Goals:

- Detect hands and landmarks using Mediapipe.
- Understand the output data structure of Mediapipe's hand model.

Week 5: Gesture Recognition Implementation

Goals:

- Convert hand landmarks into meaningful gestures.

Week 6: Gesture-Based PC Controls

Goals:

- Use gestures to interact with system functionalities.

Week 7: Math Recognition Prototype

Goals:

- Implement basic OCR for handwritten math recognition.

Week 8: Math Solving Integration

Goals:

- Solve recognized math problems using SymPy.

Week 9: Combining Gesture and Math Functionality

Goals:

- Create a unified pipeline for gesture control and math solving.

Week 10: Testing and Fine-Tuning

Goals:

- Test the system thoroughly for accuracy and usability.

Week 11: User Interface and Experience Design

Goals:

- Create a simple and intuitive user interface.

Week 12: Documentation and Final Presentation

Goals:

- Prepare final documentation and showcase the project.