Project: Computer Interaction For The People With Disabilities

Project Overview

This project implements a multi-modal system to interact with a computer's mouse using three major input mechanisms: **hand gestures**, **head/eye movements**, and **voice commands**. Each feature offers users alternative methods to control the mouse, catering to accessibility and enhancing user interactivity.

Features

1. Hand Gesture Tracking for Mouse Interaction

This feature uses **MediaPipe** and **OpenCV** to detect hand gestures via a webcam and translates them into mouse actions:

- **Cursor Movement**: Moves the mouse pointer based on the index finger's tip position.
- **Left Click**: Identified by specific hand angles and distances between finger landmarks.
- **Right Click**: Detected through distinct finger configurations.
- **Double Click**: Recognized when specific landmark conditions are met.
- **Screenshot Capture**: Triggered by a gesture involving thumb and index finger proximity.

Key Libraries Used:

- **MediaPipe**: For detecting hand landmarks and tracking gestures.
- **PyAutoGUI**: To control the system mouse and take screenshots.

---

2. Head and Eye Tracking for Mouse Movement

This feature leverages **FaceMesh** to detect facial landmarks and track eye movement via a webcam:

- **Cursor Control**: Tracks the user's head position and moves the cursor accordingly.
- **Click Action**: Detects eyelid movement (e.g., a blink) to perform mouse clicks.

Key Features:

- Real-time responsiveness with minimal latency.
- Cursor movement scaled to the screen size.

Key Libraries Used:

- **MediaPipe**: For detecting facial landmarks.
- **PyAutoGUI**: For controlling mouse pointer and clicks.

---

3. Voice Recognition for Mouse Commands

This feature uses **SpeechRecognition** and **Pyttsx3** to translate spoken commands into mouse actions:

- **Movement Commands**: Move the mouse in specific directions (e.g., "move left", "move down").
- **Click Commands**: Perform actions like "click", "double click".
- **Scrolling**: Scroll the screen up or down using voice instructions.
- **Stop Command**: Halts the voice recognition process.

Key Features:

- Noise handling and ambient sound adjustments.
- Feedback provided to users via text-to-speech responses.

Key Libraries Used:

- **SpeechRecognition**: For speech-to-text conversion.
- **Pyttsx3**: For text-to-speech feedback.
- **PyAutoGUI**: For executing mouse actions.

Technical Implementation

- **Multi-threading**: Each input mechanism runs in its thread to allow simultaneous tracking and command execution without blocking other features.
- **Real-time Processing**: The project leverages the processing power of libraries like OpenCV and MediaPipe for real-time interactions.
- **Fallback Mechanism**: Users can stop any feature using predefined commands (e.g., "stop" for voice recognition).

Usage Scenarios

1. **Accessibility Support**: Enables physically challenged users to control their computers using alternative methods.
2. **Hands-Free Operation**: Useful for scenarios requiring non-contact interactions, such as sterile environments.
3. **Enhanced Interaction**: Adds intuitive and futuristic methods for controlling digital interfaces.

Future Enhancements

- **Improved Gesture Detection**: Adding more gestures for finer control.
- **Eye Tracking Precision**: Incorporating advanced algorithms to handle small head movements.
- **Customizable Voice Commands**: Allowing users to define their own commands for greater flexibility.
- **Keyboard Integration**: Allowing for more sophisticated usage making it more accessible

---

Current Limitations:

Issue: The tracking of each function is quite unreliable. The project may give hope for scalability but currently it is quite unreliable for its motion of tracking.
Plan: To solve this issue further testing is needed. By testing extensively we can pinpoint the more accurate speed of cursor movement to be set for the project.

Issue: The accuracy or pinpointing the object of interest is sometimes unreliable. We have pinpointed some of the issues through our observation.

1) The clashing of Sensors or multiple object of interest on the screen trying to do same things
2) The Distance of the object of interest. The further the object is, the harder it is to pinpoint the object.
3) The devices themselves which we are using to identify the objects are subpar or not up to the quality we are expecting.
4) Some objects of interest are quite small making them harder to pinpoint for the program, resulting in an unreliable machine.

Plan:

1) Will try to implement the project so that the user may choose which object of interest to prioritise if multiple objects are trying to do the same thing.
2) For this further testing is required to know how far the program can identify objects.
3) We first need to try and optimize the logic in our program first before trying to get better devices to identify the objects for better accuracy.
4) Will try to make the object of interest larger if possible (like: hand and finger objects).

Conclusion

This project integrates cutting-edge computer vision and speech technologies to create a versatile, multi-modal input system. It demonstrates the potential of combining hardware and software to deliver innovative and accessible user experiences.