Gesture-Based Interactive UI and Mini-Game

Can İnanır - 31159 Mert Polat - 31326

December 10, 2024

1 Introduction

In this project we implemented a system that can detect gestures and allows users to interact with UI with only their camera. We used MediaPipe's hand tracking technology to make the system detect specific hand gestures (e.g., an open palm or a closed fist) and use them to emulate cursor actions, clicks, and dragging actions. The app includes a couple movable buttons and a small mini-game where the user can catch falling geese in a basket by using hand gestures.

2 Contributors

Mert: Designed and implemented the core logic that maps hand gestures to interaction states (click, grab, release). Also ensured stable cursor behavior based on landmarks.
Can: Implemented the interactive application and the mini-game portion ("Goose Grab"). Integrated the UI logic, gesture recognition results, and spawning/collision detection for falling objects.

3 Overview of the System

The system tracks the user's hand in real-time through the webcam feed. MediaPipe's Gesture Recognizer identifies hand landmarks and classifies gestures. Two primary gestures control the cursor:

Closed Fist: Used as a click-and-hold action, allowing the user to grab and move UI elements or the basket.

Open Palm: Used to release a currently held UI element after maintaining the gesture for a short interval (0.1s), ensuring deliberate releases.

3.1 Landmarks and Gesture Choice

We use the wrist (landmark index 0) and index finger MCP (landmark index 5) to determine cursor position. Averaging their coordinates yields:

$$palm\_x = \frac{wrist_x + index\_finger\_mcp_x}{2}, \quad palm\_y = \frac{wrist_y + index\_finger\_mcp_y}{2}$$

These coordinates, mapped to screen space, provide stable cursor control.

3.2 Interactions

- **UI Buttons:** Grabbing buttons with a closed fist allows the user to reposition them. Than the user can release them with an open palm. A reset button resets the possition of all buttons. There is also a text that displays which gesture the program recognizes at the moment.

- **Mini-Game (Goose Grab):** Selecting the "Goose Grab" button starts the mini-game. Geese fall from the top of the screen, and the user must catch them with the basket by grabbing and moving it with the closed fist gesture. Points are earned for each goose caught. A back button is also added to go back to the main menu.

4 Implementation Details

4.1 Landmark Calculation and Cursor Mapping

MediaPipe's Gesture Recognizer outputs normalized hand landmarks. We select the wrist and index finger MCP to define a stable reference point. The midpoint of these landmarks is converted to pixel coordinates to control the cursor. This way we can have a stable hand-to-screen mapping.

4.2 Gesture Recognition and State Management

Closed fists initiate or maintain a click state. When the user makes an open palm and holds it for at least 0.1 seconds, the system interprets this as a release action. This delay prevents accidental releases caused by mis-inputs or camera errors.

4.3 Audio and Performance Considerations

Audio playback is handled asynchronously with separate threads. Without this the program would freeze every time we played a sound until the sound had finished playing. This way we prevent blocking the main loop and avoid stuttering. Visual updates, gesture recognition, and collision detection proceed smoothly, even while sound effects play.

4.4 Code Structure

Cursor Class: Simulates mouse-like behaviors (click, drag), triggered by gestures.

Button Class: Manages UI elements' hover, click, and drag states.

Goose Class: Represents falling objects, updated each frame, and rendered over the main feed.

GestureRecognizerApp Class: The main application that orchestrates video capture, landmark/gesture processing, UI drawing, mini-game logic, and event handling.

5 GitHub Link

The source code is hosted at:

`https://github.com/mert577/gesture_project`

6 Images

Figures can be inserted below to illustrate the interface and mini-game scene.

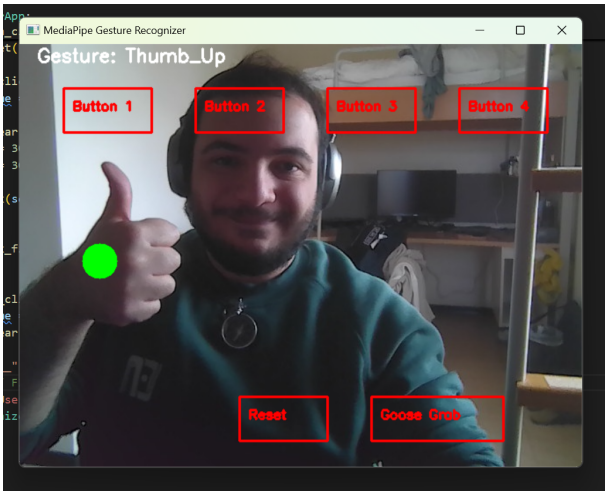

Figure 1: Main Screen with draggable buttons and Goose Grab button

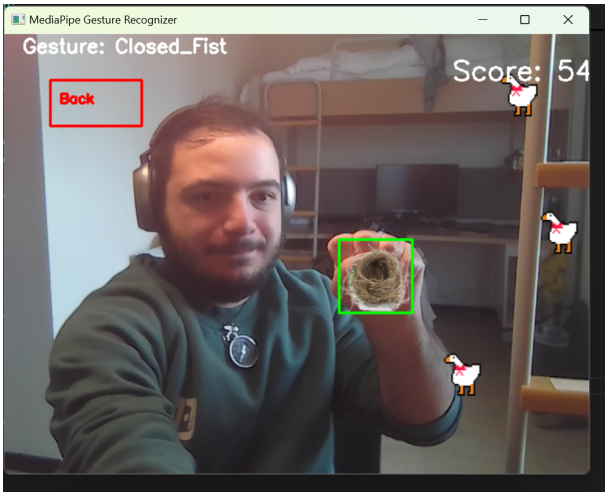

Figure 2: Mini-Game Screen: The basket can be dragged to catch falling geese.

7 Conclusion

This project demonstrates an effective combination of computer vision and gesture recognition with a simple interactive UI and mini-game. By selecting gestures that naturally map to cursor actions and by implementing asynchronous audio playback, the system remains responsive and user-friendly. Contributors Mert and Can have successfully integrated these components into a cohesive application that highlights the potential of hand-tracking interfaces for intuitive and accessible interactions.