Implementation of Contactless UI using Hand Pose AI and Cosine Similarity

Abstract

This paper details the development of a contactless user interface (UI) system leveraging Google Mediapipe and Hand Pose AI. Motivated by the need for hygienic and accessible computer interaction methods—particularly highlighted by the COVID-19 pandemic—this system interprets hand gestures captured via webcam to execute commands such as window creation, resizing, and cursor movement. By utilizing cosine similarity to match user gestures against a pre-defined dataset, the system achieves intuitive control without requiring specialized hardware. This study discusses the system architecture, algorithmic implementation, testing results, and potential future applications.

1 Introduction

1.1 Background and Necessity

The COVID-19 pandemic significantly increased the time individuals spent at home, leading to a rise in media consumption. A common inconvenience identified during this period was the displacement of remote controls or the need to interact with peripherals while eating, which often resulted in the contamination of keyboards and mice.

These scenarios highlighted a need for computer control interfaces that are accessible even when physical input devices are difficult to locate or use. Inspired by user interfaces depicted in films such as *Minority Report* and *Iron Man*, this project aims to realize a functional contactless control system.

1.2 Project Objective

The primary objective is to implement a contactless UI utilizing "Google Mediapipe" and "Hand Pose AI." The system is designed to recognize specific hand shapes presented before a camera and execute corresponding tasks on a computer window.

Key functionalities implemented include window creation, expansion, contraction, cursor movement, and dragging. Due to technical constraints in manipulating third-party software directly, the system operates within a virtual program environment developed for this purpose.

1.3 Differentiation from Existing Solutions

Existing contactless solutions often have significant limitations. For instance, gesture-based extensions for Chrome browsers typically support only vertical scrolling. Hardware-

based solutions, such as PT rings, require the user to wear a device, which can be cumbersome. Trackpads, while common, remain contact-based and susceptible to contamination.

In contrast, the proposed system supports a wider range of actions (creation, resizing, movement) and operates solely via webcam recognition, eliminating the need for wearable devices or physical contact.

2 System Design and Implementation

2.1 Design Direction

The system utilizes Hand Pose AI to extract 21 landmark points from the webcam video feed in real-time. The core challenge lay in mathematically interpreting this data to determine the user's pose and executing one of five commands fluidly.

Poses were selected to be intuitive and distinct to ensure ease of use for first-time users and to prevent command overlap.

2.2 Language and Platform

Javascript was selected as the development language to facilitate a web browser-based implementation. The program runs directly in a web browser and accesses the webcam.

2.3 System Architecture

The program flow is as follows:

1. Start
2. Acquire webcam feed
3. Load Hand Pose Recognition AI Module
4. Begin Hand Recognition / Video Capture
5. Hand Detected $\rightarrow$ Find closest match among pre-defined hand shapes
6. Trigger Event (e.g., Expand Window, Move Cursor, Shrink Window, Drag Window)

2.4 Usage Method

Users present one of five poses approximately 50cm from the screen:

- **Create:** Generate a new window.
- **Expand:** Increase window size.
- **Shrink:** Decrease window size.
- **Point:** Move the cursor.
- **Drag:** Move the window.

A red border indicates the currently selected window.

3 Technical Algorithms

3.1 Libraries Used

3.1.1 Mediapipe Hands

Google Mediapipe offers a suite of machine learning solutions. This project utilizes the **Mediapipe Hands** model, which tracks hand and finger movements with high accuracy. It combines a palm detection model with a hand landmark model to identify 21 specific 3D points on the hand.

3.1.2 Tensorflow.js

Tensorflow.js is utilized to run the machine learning models directly within the browser using Javascript.

3.2 Data Pre-processing: Normalization

To identify hand gestures, the system compares the current hand data against stored templates. However, raw coordinate data varies based on the hand's position in the frame and its size (distance from camera). To address this, a normalization function was implemented to scale all points to a range between 0 and 1 relative to the hand's bounding box.

The normalization formula for a point (x, y, z) is:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

$$y' = \frac{y - y_{min}}{y_{max} - y_{min}} \tag{2}$$

$$z' = \frac{z - z_{min}}{z_{max} - z_{min}} \tag{3}$$

```
function normalize(handLandmarks) {
    if (!Array.isArray(handLandmarks)) {
        return null;
    }
    const minX = handLandmarks.map((p) => p[0]).minN();
    const maxX = handLandmarks.map((p) => p[0]).maxN() - minX;
    const minY = handLandmarks.map((p) => p[1]).minN();
    const maxY = handLandmarks.map((p) => p[1]).maxN() - minY;
    const minZ = handLandmarks.map((p) => p[2]).minN();
    const maxZ = handLandmarks.map((p) => p[2]).maxN() - minZ;

    return handLandmarks.map((p) => [
        (p[0] - minX) / maxX,
        (p[1] - minY) / maxY,
        (p[2] - minZ) / maxZ,
    ]);
}
```

Listing 1: Normalization Function

3.3 Cosine Similarity

Cosine similarity is employed to quantify the resemblance between the current hand pose and the stored templates. It measures the cosine of the angle between two vectors; a value of 1 indicates identical direction.

Since the hand data consists of 3D points (an array of arrays), it is flattened into a 1D vector of length 63 (3×21) using `Array.flat()` before calculation.

The pose with the highest cosine similarity score is selected as the current action. A threshold of 0.95 is enforced; similarity scores below this value are ignored as undefined or noise.

4 Testing and Refinement

4.1 Test Environment

- **Hardware:** Macbook Air M1, Logitech Webcam 720P
- **Software:** macOS Monterey 12.4, Chrome 102.0
- **Methodology:** Real-time monitoring of cosine similarity values for various hand poses.

4.2 Results and Analysis

Initial testing revealed detection errors:

- The "Point" pose was occasionally misidentified due to camera angle variations, dropping similarity to 0.88.
- A slightly widened hand in the "Shrink" pose was misidentified as "Point".
- While "Create" and "Drag" were distinct, "Expand," "Shrink," and "Point" shared structural similarities leading to confusion.

4.3 Improvements

To mitigate these errors, the dataset was augmented. Specifically for "Point" and "Drag" actions which involve movement, the screen was divided into 9 zones, and pose data was captured for each zone to account for angular variations.

Additionally, a **deadZone** variable was introduced for the "Point" cursor movement. This allows for cursor control with smaller hand movements near the center of the screen, addressing the issue where reaching screen edges required excessive physical motion.

5 Conclusion

5.1 Summary

This project successfully demonstrated that contactless UI technologies, once the domain of science fiction, can be implemented using accessible AI libraries. The democratization of AI tools allows for the development of sophisticated interfaces without specialized hardware.

5.2 Future Work

Future iterations could include:

- Collecting a more extensive dataset to further reduce detection errors.
- Implementing a neural network to learn and classify poses instead of relying solely on raw cosine similarity.
- Packaging the functionality into a Chrome Extension for broader usability.

6 References

1. Google Developers. *Mediapipe*. `https://google.github.io/mediapipe/`
2. Google Developers. *Mediapipe Hands*. `https://google.github.io/mediapipe/solutions/hands`
3. Tensorflow. *Tensorflow.js*. `https://www.tensorflow.org/js`
4. Tensorflow Blog. *Move Mirror: An AI Experiment with Pose Estimation*. `https://blog.tensorflow.org/2018/07/move-mirror-ai-experiment-with-pose-estimation-tenso` `html`