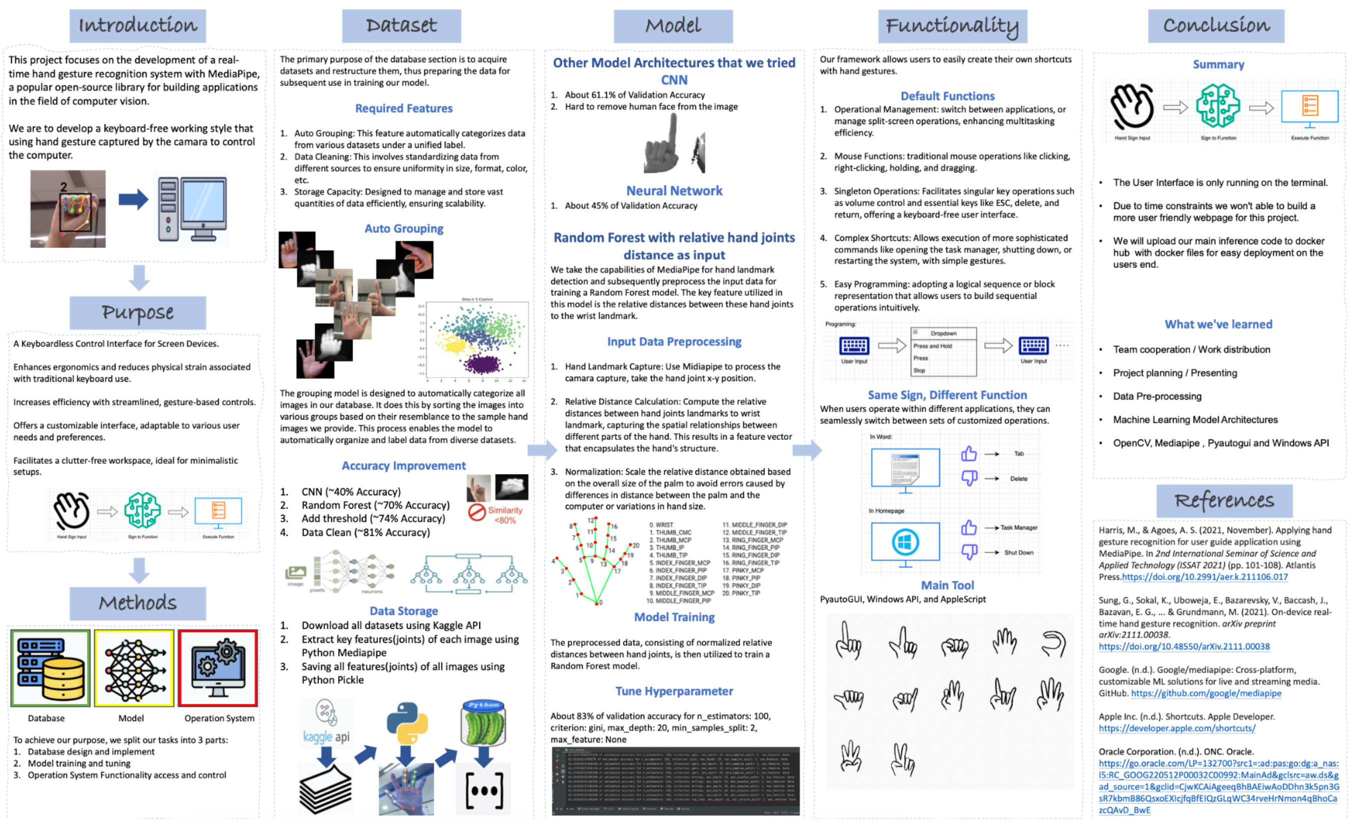
Our poster: (Summarizes our project every well)



Introduction

This project focuses on the development of a real-time hand gesture recognition system with MediaPipe, a popular open-source library for building applications in the field of computer vision.

We are to develop a keyboard-free working style that using hand gesture captured by the camara to control the computer.

Purpose

A Keyboardless Control Interface for Screen Devices.

Enhances ergonomics and reduces physical strain associated with traditional keyboard use.

Increases efficiency with streamlined, gesture-based controls.

Offers a customizable interface, adaptable to various user needs and preferences.

Facilitates a clutter-free workspace, ideal for minimalistic setups.

Methods

To achieve our purpose, we split our tasks into 3 parts:
1. Database design and implement
2. Model training and tuning
3. Operation System Functionality access and control

Dataset

The primary purpose of the database section is to acquire datasets and restructure them, thus preparing the data for subsequent use in training our model.

Required Features

1. Auto Grouping: This feature automatically categorizes data from various datasets under a unified label.
2. Data Cleaning: This involves standardizing data from different sources to ensure uniformity in size, format, color, etc.
3. Storage Capacity: Designed to manage and store vast quantities of data efficiently, ensuring scalability.

Auto Grouping

The grouping model is designed to automatically categorize all images in our database. It does this by sorting the images into various groups based on their resemblance to the sample hand images we provide. This process enables the model to automatically organize and label data from diverse datasets.

Accuracy Improvement

1. CNN (~40% Accuracy)
2. Random Forest (~70% Accuracy)
3. Add threshold (~74% Accuracy)
4. Data Clean (~81% Accuracy)

Data Storage

1. Download all datasets using Kaggle API
2. Extract key features(joints) of each image using Python Mediapipe
3. Saving all features(joints) of all images using Python Pickle

Model

Other Model Architectures that we tried

CNN

1. About 61.1% of Validation Accuracy
2. Hard to remove human face from the image

Neural Network

1. About 45% of Validation Accuracy

Random Forest with relative hand joints distance as input

We take the capabilities of MediaPipe for hand landmark detection and subsequently preprocess the input data for training a Random Forest model. The key feature utilized in this model is the relative distances between these hand joints to the wrist landmark.

Input Data Preprocessing

1. Hand Landmark Capture: Use Midiapipe to process the camara capture, take the hand joint x-y position.
2. Relative Distance Calculation: Compute the relative distances between hand joints landmarks to wrist landmark, capturing the spatial relationships between different parts of the hand. This results in a feature vector that encapsulates the hand's structure.
3. Normalization: Scale the relative distance obtained based on the overall size of the palm to avoid errors caused by differences in distance between the palm and the computer or variations in hand size.

Model Training

The preprocessed data, consisting of normalized relative distances between hand joints, is then utilized to train a Random Forest model.

Tune Hyperparameter

About 83% of validation accuracy for n_estimators: 100, criterion: gini, max_depth: 20, min_samples_split: 2, max_feature: None

Functionality

Our framework allows users to easily create their own shortcuts with hand gestures.

Default Functions

1. Operational Management: switch between applications, or manage split-screen operations, enhancing multitasking efficiency.
2. Mouse Functions: traditional mouse operations like clicking, right-clicking, holding, and dragging.
3. Singleton Operations: Facilitates singular key operations such as volume control and essential keys like ESC, delete, and return, offering a keyboard-free user interface.
4. Complex Shortcuts: Allows execution of more sophisticated commands like opening the task manager, shutting down, or restarting the system, with simple gestures.
5. Easy Programming: adopting a logical sequence or block representation that allows users to build sequential operations intuitively.

Same Sign, Different Function

When users operate within different applications, they can seamlessly switch between sets of customized operations.

Main Tool

PyautoGUI, Windows API, and AppleScript

Conclusion

Summary

- The User Interface is only running on the terminal.
- Due to time constraints we won't able to build a more user friendly webpage for this project.
- We will upload our main inference code to docker hub with docker files for easy deployment on the users end.

What we've learned

- Team cooperation / Work distribution
- Project planning / Presenting
- Data Pre-processing
- Machine Learning Model Architectures
- OpenCV, Mediapipe , Pyautogui and Windows API

References

Harris, M., & Agoes, A. S. (2021, November). Applying hand gesture recognition for user guide application using MediaPipe. In *2nd International Seminar of Science and Applied Technology (ISSAT 2021)* (pp. 101-108). Atlantis Press.https://doi.org/10.2991/aer.k.211106.017

Sung, G., Sokal, K., Uboweja, E., Bazarevsky, V., Baccash, J., Bazavan, E. G., ... & Grundmann, M. (2021). On-device real-time hand gesture recognition. *arXiv preprint arXiv:2111.00038*. https://doi.org/10.48550/arXiv.2111.00038

Google. (n.d.). Google/mediapipe: Cross-platform, customizable ML solutions for live and streaming media. GitHub. https://github.com/google/mediapipe

Apple Inc. (n.d.). Shortcuts. Apple Developer. https://developer.apple.com/shortcuts/

Oracle Corporation. (n.d.). ONC. Oracle. https://go.oracle.com/LP=132700?src1=:ad:pas:go:dg:a_nas:l5:RC_GOOG220512P00032C00992:MainAd&gclsrc=aw.ds&gad_source=1&gclid=CjwKCAiAgeeqBhBAEiwAoDDhn3k5pn3GsR7kbmB86QsxoEXIcjfqBfEIQzGLqWC34rveHrNmon4qBhoCazcQAvD_BwE

Overview of Project:

We found it to be a revolutionary change if we could control our systems with no other devices than our hands. As Computer Science students ourselves, we sit in a fixed position in front of our laptops/PCs for the majority of the time of the day, which is detrimental for us. We would like to create this new way of interacting with the screen so that we have the freedom to move around more while doing our work. We also hope users can customize our application to fit their needs.

Distribution of Responsibilities in the Group

- Yo-Hsuan Chen:
  - Model Training and Training Algorithm Selection
  - “Model section”
- Haike Yu:

    - Data Gathering and Filtering, Data Model Training
    - “Dataset section”
- Suqing Liu:
    - Connection of Operation System. Building Software Development Kit
    - “Functionality section”

Overview of Implementation:

We designed our architecture separated to 3 main categories: dataset, model, and functionality.

Dataset:

- The primary purpose of the database section is to acquire datasets and restructure them, thus preparing the data for subsequent use in training our model.
- Required Features
    - Auto Grouping: This feature automatically categorizes data from various datasets under a unified label.
    - Data Cleaning: This involves standardizing data from different sources to ensure uniformity in size, format, color, etc.
    - Storage Capacity: Designed to manage and store vast quantities of data efficiently, ensuring scalability.
- Auto Grouping
    - The grouping model is designed to automatically categorize all images in our database. It does this by sorting the images into various groups based on their resemblance to the sample hand images we provide. This process enables the model to automatically organize and label data from diverse datasets.
- Accuracy Improvement
    - CNN (~40% Accuracy)
    - Random Forest (~70% Accuracy)
    - Add threshold (~74% Accuracy)
    - Data Clean (~81% Accuracy)
- Data Storage
    - Download all datasets using Kaggle API
    - Extract key features(joints) of each image using Python Mediapipe
    - Saving all features(joints) of all images using Python Pickle

Model:

- Train on the previously filtered data and use it for inference of the label of the input hand sign.
- Random Forest with relative hand joint distance as input.
    - We take the capabilities of MediaPipe for hand landmark detection and subsequently preprocess the input data for training a Random Forest model. The key feature utilized in this model is the relative distances between these hand joints to the wrist landmark.
- Input Data Preprocessing
    - Hand Landmark Capture: Use Midiapipe to process the camera capture. Take the hand joint x-y position.
    - Relative Distance Calculation: Compute the relative distances between hand joint landmarks to wrist landmarks, capturing the spatial relationships between different parts of the hand. This results in a feature vector that encapsulates the hand's structure.
    - Normalization: Scale the relative distance obtained based on the overall size of the palm to avoid errors caused by differences in distance between the palm and the computer or variations in hand size.
- Model Training
    - The preprocessed data, consisting of normalized relative distances between hand joints, is then utilized to train a Random Forest model.
- Tune Hyperparameter
    - About 83% of validation accuracy for n_estimators: 100, criterion: gini, max_depth: 20, min_samples_split: 2, max_feature: None

Functionality:

- Pair individual functions with the hand sign. Allow customization to each user’s need.
- Default Functions:
    - Operational Management: switch between applications or manage split-screen operations, enhancing multitasking efficiency.
    - Mouse Functions: traditional mouse operations like clicking, right-clicking, holding, and dragging.
    - Singleton Operations: Facilitates singular key operations such as volume control and essential keys like ESC, delete, and return, offering a keyboard-free user interface.

    - Complex Shortcuts: Allows execution of more sophisticated commands like opening the task manager, shutting down, or restarting the system with simple gestures.
- Same Sign, Different Function
    - When users operate within different applications, they can seamlessly switch between sets of customized operations.

After we combine all of our sections, we have the final product that takes a hand sign from the user and asks the model to pair the hand sign with the required program. The User Interface is only running on the terminal. Due to time constraints, we won't be able to build a more user-friendly webpage for this project. We will upload our main inference code to docker hub with docker files for easy deployment on the user's end.

Testing Results:

Robustness: We run the model with real-time camera captures and the model still maintains a good performance.

Speed: It takes 890.6509351730347 seconds to train our model including hyperparameter tuning.

Quality: With improvement in the database, we reduce the overfitting in training data.

Accuracy: 100.0% of training accuracy for n_estimators: 100, criterion: gini, max_depth: 20, min_samples_split: 2, max_feature: None; 83% of validation accuracy for n_estimators: 100, criterion: gini, max_depth: 20, min_samples_split: 2, max_feature: None

Datasets Used:

Leap Gesture Recognition Dataset :

https://www.kaggle.com/datasets/gti-upm/leapgestrecog

ASL Dataset :

https://www.kaggle.com/datasets/ayuraj/asl-dataset

Hand Sign Images Dataset :

https://www.kaggle.com/datasets/ash2703/handsignimages

Describe the data (both input and output) and give examples

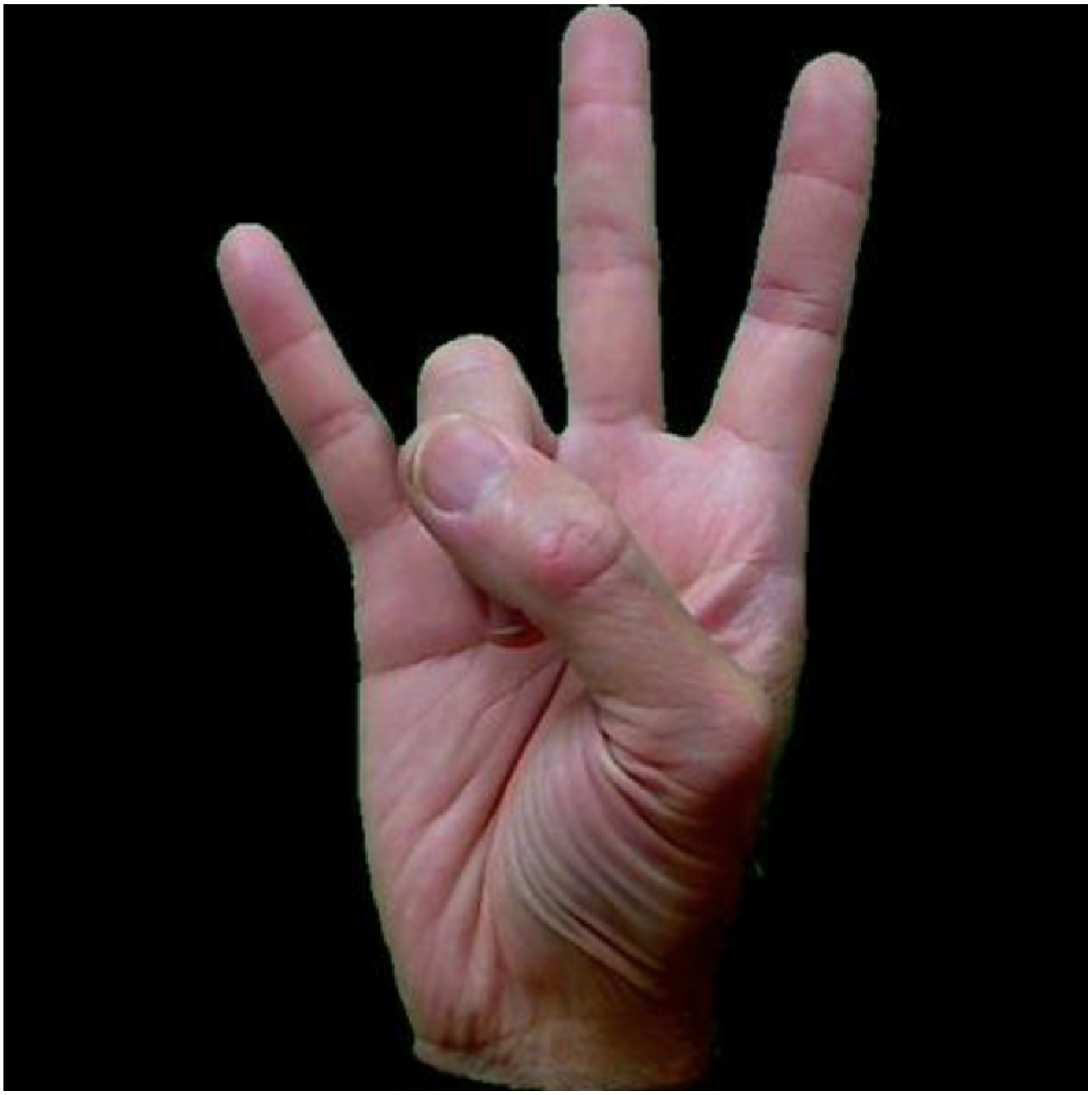
Input Structure:

A jpg file of hand pose (different datasets have different colours, sizes, and shapes)
Examples:

Hand Gesture Recognition Dataseats:

Hand-sign-images:

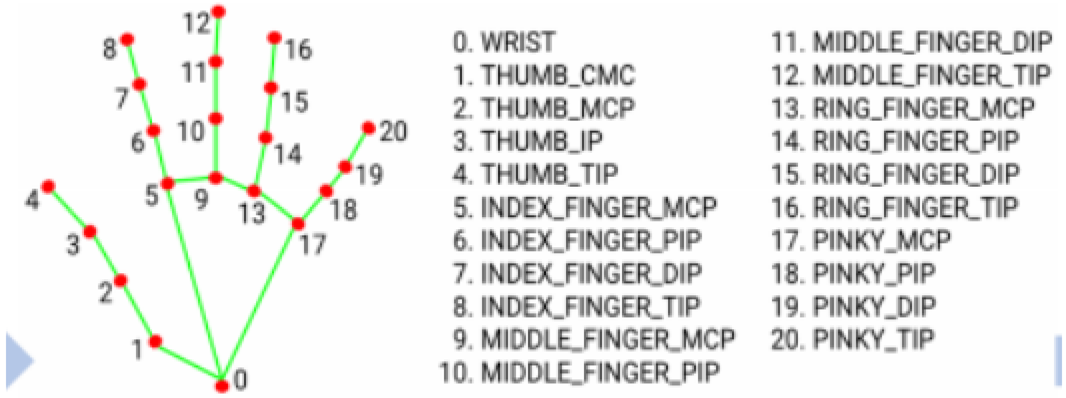
American Sign Language Dataset:



Output Structure:

the normalized relevant Euclidian distance from landmark i to the wrist landmark excluding the landmark [0, 1, 5, 9, 13, 17]

```
[0.5307661944555954, 0.7383509435265254,
0.7444541027088238, 1.2860871105701768,
1.5122054777178713, 1.7409645155535165,
0.9020862872808414, 0.5445285476667695,
0.5213874979488742, 0.7723513606865626,
0.4571625057792881, 0.46549396148361244,
0.7043143953381948, 0.49703536031354273,
0.48741721693968004]
len(data[0]): 15
```



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

References:

- Harris, M., & Agoes, A. S. (2021, November). Applying hand gesture recognition for user guide application using MediaPipe. In 2nd International Seminar of Science and Applied Technology (ISSAT 2021) (pp. 101-108). Atlantis Press.https://doi.org/10.2991/aer.k.211106.017
- Sung, G., Sokal, K., Uboweja, E., Bazarevsky, V., Baccash, J., Bazavan, E. G., ... & Grundmann, M. (2021). On-device real- time hand gesture recognition. arXiv preprint arXiv:2111.00038. https://doi.org/10.48550/arXiv.2111.00038
- Google. (n.d.). Google/mediapipe: Cross-platform, customizable ML solutions for live and streaming media. GitHub. https://github.com/google/mediapipe
- Apple Inc. (n.d.). Shortcuts. Apple Developer. https://developer.apple.com/shortcuts/

- Oracle Corporation. (n.d.). ONC. Oracle. https://go.oracle.com/LP=132700?src1=:ad:pas:go:dg:a_nas: l5:RC_GOOG220512P00032C00992:MainAd&gclsrc=aw.ds&g ad_source=1&gclid=CjwKCAiAgeeqBhBAEiwAoDDhn3k5pn3G sR7kbmB86QsxoEXIcjfqBfEIQzGLqWC34rveHrNmon4qBhoCa zcQAvD_BwE