Pattern Recognition System Report

--A gesture recognition system based on MediaPipe and KNN

Li Jinying 2219040107 (Design and Implementation) Wang Yifei 2206060105

1 Project Overview

This project implements a gesture recognition system based on MediaPipe and KNN, recognizing sign language numbers 0 to 9. The implementation process includes modules for data acquisition, feature extraction and processing, model training, real-time recognition, and visualization, and can ultimately run locally.

The implementation environment and tools are as follows:

Operating System: Windows 11

Development language: Python 3.10

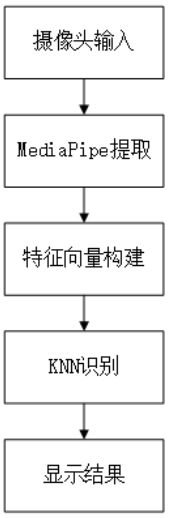
Development tools: VS Code + Jupyter Notebook

Project address:

Training and related code are available at https://github.com/zing53/slr

Real-time gesture recognition : https://github.com/zing53/slr-realtime

2 System Design Framework



The system is divided into five main modules:

Data acquisition module: Uses a camera to capture real-time image frames.

Feature extraction module: Call MediaPipe to obtain the 3D coordinates (x, y, z) of 21 key points on the hand.

Feature construction module: Calculates the distance between 20 key points, forms a feature vector, and performs normalization processing.

Classification and recognition module: Based on a trained KNN classifier, it identifies the numerical labels corresponding to the features.

Output module: Displays the recognition results in the image in real time.

3 Performance indicators

To measure the performance of this system, the following evaluation metrics are designed:

Accuracy: The ratio of the recognition result to the actual label.

Response time: The time required from image acquisition to output.

4 Specific Algorithm

4.1 Feature extraction

MediaPipe is an open-source, cross-platform framework provided by Google. Its MediaPipe Hand Landmarker Task supports efficient hand detection and tracking, detecting 21 hand keypoints in a single frame image, including the (x , y , z) coordinates of each point. This system will call the MediaPipe API to obtain these keypoint coordinates in real time, thereby achieving feature extraction of the gesture.

4.2 Feature construction

In this stage, the system will adopt a spatial normalization strategy based on the palm reference point. Specifically, using the palm point as the reference point, the Euclidean distance between the remaining 20 keypoints and the reference point is calculated using two-dimensional coordinates (x, y), forming a distance vector. This distance vector is then mapped to the [0, 1] interval to achieve normalization.

In this way, instead of using absolute coordinates directly, relative distance is used as the feature input, which reduces the dimensionality of the features, suppresses feature shifts caused by different hand sizes and camera positions, improves the scale invariance of the classifier, and makes subsequent classification tasks more stable and efficient.

4.3 Classification and Recognition

The classification part uses the K-Nearest Neighbors algorithm. Its basic principle is: for a sample to be classified, calculate its distance to all samples in the training set, select the K nearest samples , and use voting to determine the category.

KNeighborsClassifier from scikit-learn for training and prediction. The feature input is the feature vector processed as described above. During training, the dataset size is approximately 100 images per digit, the K value is selected using 5-fold cross-validation, and Euclidean distance is used as the distance metric.

The KNN method has a training time complexity of O(1) and a prediction time complexity of O (n) . It is simple to implement, highly accurate, and suitable for situations with a small sample size. Its drawbacks are that it is sensitive to noise and the computational cost increases with the sample size.

5 Implementation process

5.1 Dataset

Sign Language Digits Dataset [1]was used , which covers numbers 0 to 9, with about 200 images for each gesture, for a total of 2062 images. The data is stored in folders, and the image quality is generally average with a resolution of 100×100.

5. 2 Feature extraction and construction

The corresponding file is feature_extraction.py.

The MediaPipe Hand Landmarker Task is used to detect hand landmarks in a single frame image. The output, HandLandmarkerResult , consists of three parts. This project only uses WorldLandmarks , where each landmark is composed of x , y, and z , representing real 3D coordinates in meters, with its origin located at the geometric center of the hand.

During the feature extraction stage, the detection results for each image are obtained , and the image name, the two-dimensional world coordinates (x, y) of 21 hand key points , and the corresponding gesture labels are saved in the file ./data/hand_landmarks.csv .

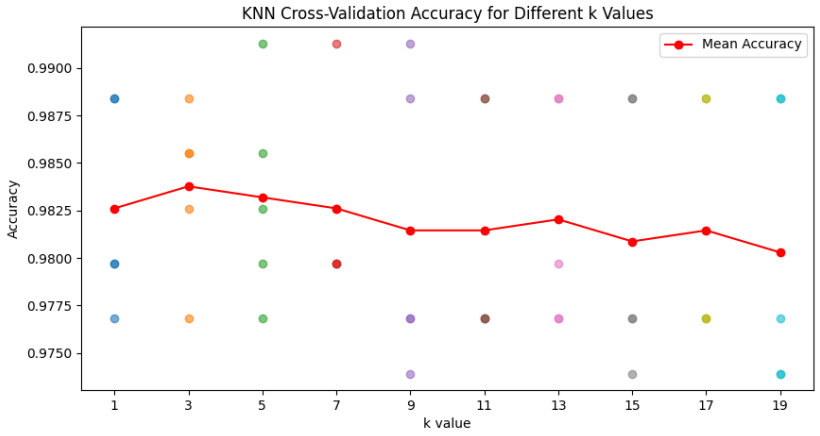
Because some images were not correctly recognized by MediaPipe , the number of images in different categories varies, but all gestures have more than 130 images: gesture 0 has 150 images, gesture 1 has 158 images, gesture 2 has 132 images, gesture 3 has 198 images, gesture 4 has 181 images, gesture 5 has 207 images, gesture 6 has 161 images, gesture 7 has 161 images, gesture 8 has 174 images, and gesture 9 has 203 images.

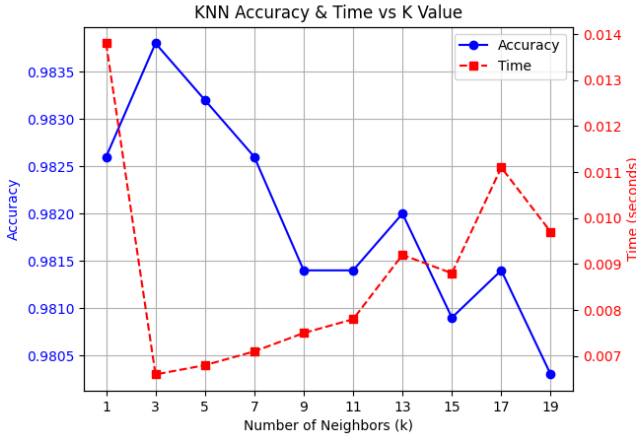
In the feature construction stage, the key point coordinates of each image are read from the hand_landmarks.csv file. Taking the palm point (index=0) as the reference point, the Euclidean distances from the other 20 points to it are calculated to form a 20-dimensional feature vector. The image name and corresponding label are saved as a row in the file /data/processed_hand_landmarks.csv .

5. 3 KNN model training

The corresponding `train.py` file uses the scikit-learn library. The training process is as follows:

First, features and labels were read from `processed_hand_landmarks.csv` , and the features were normalized using MinMaxScaler . Then, StratifiedKFold cross-validation was performed, with a 5-fold scrambled dataset, and the optimal K value was selected from odd numbers between 1 and 19. The evaluation metrics were accuracy and average prediction time. The experimental results are shown in the figure, indicating that K=3 yielded the best results in both accuracy and prediction time.





Finally, KNeighborsClassifier is called to train the model using the complete dataset and the optimal K value (3) , and the model and normalizer are saved locally.

6 Module implementation

The system design framework is detailed in the design report. Real-time gesture recognition for local live streams is implemented in the ` main.py` file of the `slr - realtime` project . This project consists of 5 modules, and the specific algorithms are as follows.

6. 1 Data Acquisition and Feature Extraction

The system uses OpenCV (cv2.VideoCapture) to capture images in real time from a laptop or external camera, obtaining continuous frame data as the input source. After acquisition, the images are mirrored (cv2.flip) to improve the user experience and ensure they are aligned with the screen orientation.

Feature extraction is achieved through the landmarker_and_result class. This class contains the following methods: createLandmarker ( ) creates a hand keypoint detector for MediaPipe, sets the mode to LIVE_STREAM live stream, and can detect up to two hands at the same time; detect_async ( ) receives image frames in real time and performs detection asynchronously, stores the detection results for subsequent drawing and classification. The design of this class is referenced in the code [2].

6. 2 Feature construction, classification and output

Feature construction and classification are implemented in the `classifier()` function. Based on the extracted features, the distance is calculated, a 20-dimensional feature vector is constructed and normalized as the input to the model; the trained KNN model is loaded for classification; and the prediction result and the confidence score of the predicted class are returned.

The output is implemented in the draw_landmarks_on_image ( ) function. The hand keypoints and skeleton structure detected by MediaPipe are drawn on the image frame in real time, and the KNN classification results and confidence scores are also drawn. The design of this function is referenced in the code [3].

7 Testing and Evaluation

In the KNN classifier, the choice of the K value has a significant impact on the results. In actual testing, it was found that when K=3, there was some confusion between the gesture 5 and

gesture 1 categories. Ultimately, K=5 was chosen as the parameter, which improved stability for easily confused categories while maintaining overall accuracy.

K=5 , the average accuracy of 5-fold cross-validation is 98.32 % , and the average prediction time is 0.0068 seconds.

The shortcoming of this project is that the key points of some gestures are similar in distribution and the feature vectors are very similar, making them difficult to distinguish. Further improvements can be made in feature construction, such as adding key point angles.

The complete training process and image gesture recognition interface are implemented in the slr_final.ipynb file , including image selection and image upload for recognition. For these 9 test images, the average prediction time was 0.0012 seconds , and the average prediction

confidence was 1, indicating that the system is relatively stable in recognizing static images . The following image shows the interface for selecting images for gesture recognition.

the slr - realtime project was built to achieve local real-time gesture recognition, and the results are shown in the upper right figure. Statistical analysis of 700 frames of gestures showed that the average response time from frame acquisition to output processing result was 0.038 seconds , indicating good real-time performance.

refer to:

[1] https://github.com/ardamavi/Sign-Language-Digits-Dataset

[2] https://github.com/OwenTalmo/finger-counter

[3] https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker/python?hl=zh-cn