# Real-Time First-View Hand Gestures Recognition

Yipeng Lin and Xinyang Chen
Carnegie Mellon University
{yipengli, xinyangc}@andrew.cmu.edu

May 31, 2022

## Contents



## 1 Introduction

Hand gestures recognition has been a trending topic for sometime in the machine learning and computer vision fields. This hand gestures recognition system is an application-focused product, aiming of providing a fast, light-weight, scalable, and user-friendly system. The system is compatible with the current highest-precision models and functions including dynamic hand gesture recognition model, static hand poses recognition function, and gesture recognition with gloves function. In this document, we would like to briefly introduce the general design of the *Hand Gesture Recognition* project and the trail of tests we have done.

This project deals with two recognition tasks, recognizing *static poses* and *dynamic gestures*. Static poses refer to poses of hands that can be recognized using a single image, e.g., hand poses for numbers and “ok”. Dynamic gestures refer to gestures that contain meaningful information in multiple consecutive frames, e.g., waving hands. The task to recognize static poses is achieved by using MediaPipe [1]. The task to recognize dynamic gestures is achieved by a customized deep Recurrent Neural Network (RNN).

The general idea of recognizing static poses is to define the range of angles for each key points of a human hand. Static images will first be fed to MediaPipe to perform a key-point recognition. Each hand will be represented by 20 hand landmarks. A list of landmarks and their corresponding positions in a hand are depicted in Fig. 1. Static poses are recognized by defining the range of angles or distances between several landmarks. For instance, the pose "one" or "point" can be classified when only landmark 5, 6, 7, 8 are almost in a line. The task to include a new pose is equivalent to mathematically defining the relative positions of each landmark. In this project, the recognition of fifteen poses are supported, the supported poses are number counting (e.g. 1, 2, and 3), thumb up, thumb down, yeah, gun, and point.

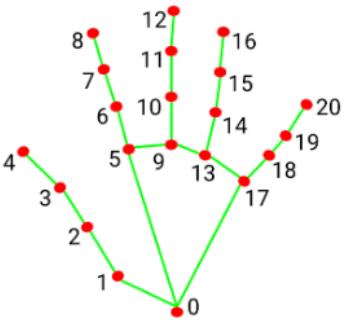



Figure 1: Hand landmarks [2].

The module to recognize dynamic gestures is built on top of static pose recognition. Particularly, we only perform the recognition of dynamic gesture when the user is posing “point”, and only the positions of the index finger tip are collected. The recognition task is achieved by building a Recurrent Neural Network (RNN). We fed the trace of the finger tip positions for consecutive 45 frames into a pre-trained RNN model, and the model will perform the recognition task and give a prediction result. Limited by the amount of data we can collect, we adopted Long-Short-Term-Memory (LSTM) model. Due to the adoption of a learning-based model, we built a gesture dataset. In this project, we focused on the recognition task of six gestures, which are listed in Table 1.

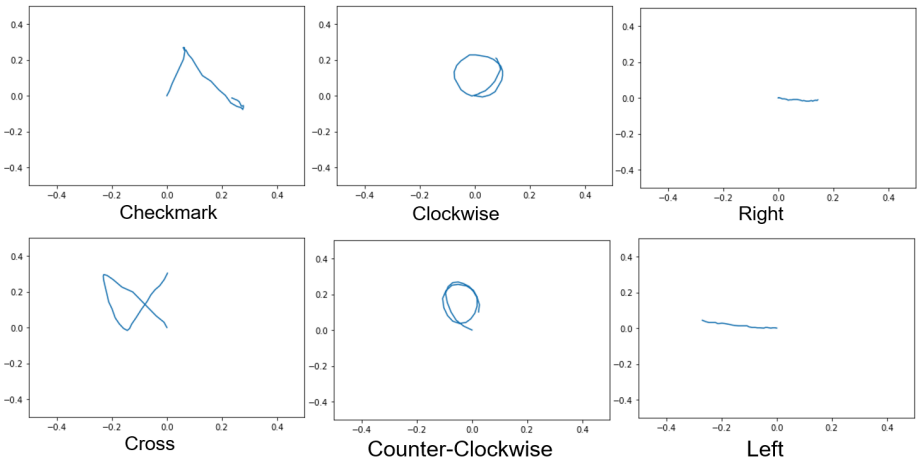



Figure 2: Trajectories of dynamic hand gestures.

Table 1: Supported hand gestures.

| Gesture Id | Gesture Description |
|---|---|
| 0 | Rotate clockwise |
| 1 | Rotate counter-clockwise |
| 2 | Checkmark |
| 3 | Cross |
| 4 | Move left |
| 5 | Move right |
| 6 | None |

# 2 Dynamic Gesture Recognition

In this section, we introduce the designed dynamic gesture recognition system. We first describe how a gesture dataset is built and cleaned. Then, we discuss the methods to augment data points. Lastly, we sequentially describe different LSTM models and justify their performance.

## 2.1 Dataset Creation and Cleaning

We collect 400 data points for each of the six gestures. For each gesture, we use a web-camera to record a 30-frame (1 second) video slice and use the MediaPipe module to get the x-y position of index finger tip in each frame. Each data point is a vector of 60 real numbers. For most of the gestures in Table 1, 30 frames are adequate.

Since MediaPipe may fail to get the finger tip positions in some frames, after collecting the data, we proposed to clean the dataset as follows

- Define a tolerant moving distance of the finger tip in consecutive two frames. If the finger tip movement exceeds the tolerance in the current frame, we use the position in the previous frame to estimate the position in the current frame.
- We wrote a small program to visualize the trace of the finger tip in 30 frames, and manually decide whether the trace matches the corresponding gesture type.

The last gesture is "None", which represents gestures that are not categorized as one of the six gestures. We consider three cases for the “None” gesture: random walking, sudden points, and stop moving. Examples of the trajectories are depicted in Figure 3.

- **Random walking** mimics the situation that a user moves the hand subconsciously and does not mean to express any meaning.
- **Sudden points** mimics the situation that a user makes some sudden movements which may exceed the detection range.
- **Stop moving** mimics the situation that a user stops doing gestures and hovers the hand at a random position.

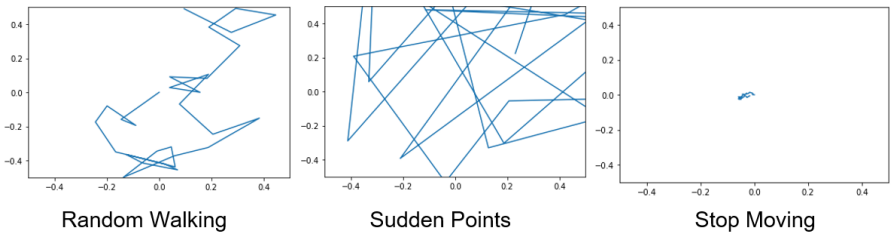



Figure 3: Trajectories of "None" gestures.

## 2.2 Data Augmentation

The cost to collect data for a large dataset is high. We proposed to conduct data augmentation in the following ways.

- **Rotation** The input data point is rotated by a randomly selected degree while its associated gesture type is unchanged.
- **Translation** The input data point is shifted in x and y axes at random distance.
- **Scaling** The input data point is scaled up/down at a random ratio.

Each data point is applied sequentially with multiple augmentation operations, and which operation can be applied is shown in Table 2.

Table 2: Applied augmentation operations.

| Gesture Description | Rotation | Translation | Scaling |
|---|---|---|---|
| Rotate clockwise | ✓ | ✓ | ✓ |
| Rotate counter-clockwise | ✓ | ✓ | ✓ |
| Checkmark | | ✓ | ✓ |
| Cross | ✓ | ✓ | ✓ |
| Move left | | ✓ | ✓ |
| Move right | | ✓ | ✓ |

Table 3: Fully-connected model results before augmenting data.

| Gesture Description | Precision | Recall | f1-score | Support |
|---|---|---|---|---|
| Clockwise | 0.44 | 0.38 | 0.41 | 92 |
| Counter Clockwise | 0.67 | 0.55 | 0.61 | 112 |
| Checkmark | 0.70 | 0.76 | 0.73 | 97 |
| Cross | 0.58 | 0.64 | 0.61 | 110 |
| Right | 0.52 | 0.47 | 0.49 | 111 |
| Left | 0.72 | 0.78 | 0.75 | 85 |
| None | 0.75 | 0.89 | 0.81 | 96 |
| Accuracy | | | 0.63 | 703 |

# 3 Recognition Models

## 3.1 Fully-Connected Model

We proposed to use a Fully-connected network to perform the prediction. The testing results on the original dataset that does not perform data augmentation are listed in Table 3.

## 3.2 Single-Network Model

We proposed to use an LSTM network to capture the temporal information. The testing results on the original dataset are listed in Table 4. Compared with results in Table 3, LSTM model has a better ability to capture the temporal information and results in a higher prediction result. We also conducted another experiment on augmented dataset whose results are listed in Table 5. Augmentation largely helps to improve the prediction accuracy.

## 3.3 Double-Network Model

After using a single-network model, we also tried a double-network model to increase the prediction accuracy of complex gestures (i.e., $checkmark$ and $cross$).

Table 4: Single-network model results before augmenting data.

| Gesture Description | Precision | Recall | f1-score | Support |
|---|---|---|---|---|
| Clockwise | 0.49 | 0.58 | 0.53 | 92 |
| Counter Clockwise | 0.80 | 0.65 | 0.72 | 112 |
| Checkmark | 0.80 | 0.82 | 0.81 | 97 |
| Cross | 0.69 | 0.58 | 0.63 | 110 |
| Right | 0.77 | 0.59 | 0.67 | 111 |
| Left | 0.65 | 0.86 | 0.74 | 85 |
| None | 0.81 | 0.95 | 0.88 | 96 |
| Accuracy | | | 0.71 | 703 |

Table 5: Single-network model results after augmenting data.

| Gesture Description | Precision | Recall | f1-score | Support |
|---|---|---|---|---|
| Clockwise | 0.98 | 0.95 | 0.97 | 979 |
| Counter Clockwise | 0.97 | 0.95 | 0.96 | 966 |
| Checkmark | 0.90 | 0.88 | 0.89 | 845 |
| Cross | 0.87 | 0.88 | 0.88 | 971 |
| Right | 0.96 | 0.97 | 0.97 | 778 |
| Left | 0.93 | 0.96 | 0.94 | 853 |
| None | 0.95 | 0.98 | 0.97 | 997 |
| Accuracy | | | 0.94 | 6389 |

The proposed double-network model contains two neural networks, model 1 and model 2. Model 1 represents the single-network model described in Sec. 3.2 that detect all seven gestures. Model 2 represents a model designated for *checkmark*, *cross*, and *none*, and the *none* means all the other gestures except *checkmark* and *cross*. The structure of the double-network model is depicted in Fig. 4.

During the test, model 1 and model 2 process the same frame simultaneously. The prediction result of model 1 will enter a counter that counts the frequency of each prediction result in the last 30 frames. Once the prediction result of model happens to be in the counter, i.e., the model 2's result shows up in the previous result of model 1 in the most recent 30 frames, we ignore model 1's prediction result and directly use model 2's prediction result.

The proposed double-network model achieves good prediction results in the training process while its prediction accuracy decreases dramatically on a real-application. We think that this is a problem of missing alignment of input data. For data points in the dataset, the first frame indicates the starting of a gesture and there is no data point that happened to be a combination of two gestures. However, it is common to ask a model to predict the gesture in real application while the user is in the transition stage between two gestures.

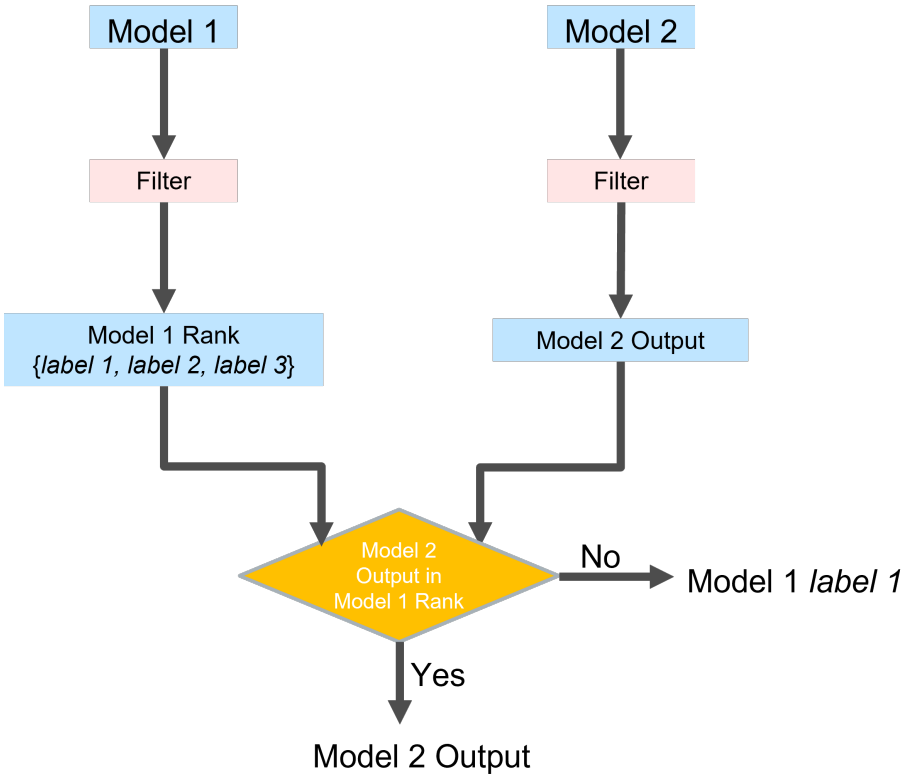



Figure 4: Double network model.

Table 6: Model 2 Testing Validation Results.

| Gesture Description | Precision | Recall | f1-score | Support |
|---|---|---|---|---|
| Charkmark | 0.92 | 0.98 | 0.95 | 894 |
| Cross | 0.97 | 0.92 | 0.94 | 956 |
| None | 0.98 | 0.97 | 0.98 | 997 |
| Accuracy | | | 0.96 | 2847 |

Table 7: Model 2 Video (1408 frames) Testing Results.

| Gesture Description | Precision | Recall | f1-score | Support |
|---|---|---|---|---|
| Charkmark | 0.54 | 0.80 | 0.64 | 374 |
| Cross counter-clockwise | 0.57 | 0.29 | 0.39 | 388 |
| None | 0.94 | 0.10 | 0.18 | 646 |
| Accrucy | | | 0.34 | 1408 |

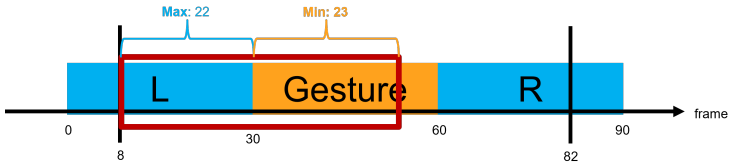
### 3.4 Single-Network Model with Random Prefix and Suffix

To deal with the alignment issue found in Sec. 3.3, we randomly select two other data points and add one as a prefix data point and another as a suffix data point. We first directly concatenate the three data points one after another and then truncate it to the designated length. The padding visualization can be seen in Figure 5, where the orange $Gesture$ is the target gesture for model to predict and two other gestures are randomly selected as its prefix ($L$) and suffix ($R$). In the truncate step, we keep a constant length (45 frames) and make sure that the target gesture always occupies the most number of frames, i.e., >= 23 frames. Figure 5 shows the maximum length of prefix (i.e. $L = 22$) in 5a, balanced prefix and suffix in 5b, and the maximum length of suffix (i.e. $R = 22$) in 5c. By controlling the maximum number of allowed frames, we control the difficulty of the training task. For instance, the difficulty of case 5b is the smallest because the target gesture takes up the longest time.

We also make experiments of the three cases in Figure 5 and the results can be seen in Table 8, 9, and 10 separately. From these three tables, we can find that the Table 8 of case 5b has the highest accuracy. This may because the target gesture is complete and with lowest distraction from its prefix and suffix gestures.

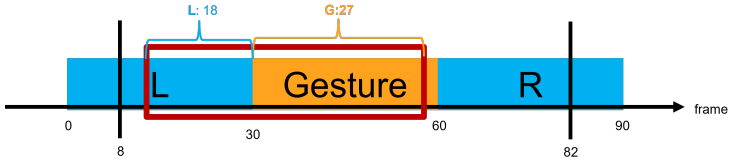
## 4 Glove Preprocessing Method

Mediapipe's hand tracking solution is based on deep learning models trained with a hand dataset. Therefore, it's sensitive to not only the hand's shape but also the skin color. Gloves have little influence on the hand's shape but change the color which significantly decreases the tracking success rate and accuracy. We develop a preprocessing method for gloves that can fit them into Mediapipe's suitable color




(a) Case 1


(b) Case 2




(c) Case 3

Figure 5: Prefix and suffix visualization.

Table 8: Prefix and suffix case 1 Results.

| Gesture Description | Precision | Recall | f1-score | Support |
|---|---|---|---|---|
| Clockwise | 0.94 | 0.90 | 0.92 | 996 |
| Counter Clockwise | 0.93 | 0.86 | 0.89 | 963 |
| Checkmark | 0.87 | 0.77 | 0.82 | 864 |
| Cross | 0.75 | 0.79 | 0.77 | 1009 |
| Right | 0.89 | 0.90 | 0.89 | 749 |
| Left | 0.91 | 0.85 | 0.88 | 832 |
| None | 0.81 | 0.95 | 0.87 | 1101 |
| Accuracy | | | 0.86 | 6514 |

Table 9: Prefix and suffix case 2 Results.

| Gesture Description | Precision | Recall | f1-score | Support |
|---|---|---|---|---|
| Clockwise | 0.98 | 0.96 | 0.97 | 996 |
| Counter Clockwise | 0.98 | 0.94 | 0.96 | 963 |
| Checkmark | 0.92 | 0.88 | 0.90 | 864 |
| Cross | 0.89 | 0.85 | 0.87 | 1009 |
| Right | 0.93 | 0.95 | 0.94 | 749 |
| Left | 0.90 | 0.92 | 0.91 | 832 |
| None | 0.88 | 0.97 | 0.93 | 1101 |
| Accuracy | | | 0.93 | 6514 |

Table 10: Prefix and suffix case 3 Results.

| Gesture Description | Precision | Recall | f1-score | Support |
|---|---|---|---|---|
| Clockwise | 0.97 | 0.94 | 0.95 | 996 |
| Counter Clockwise | 0.95 | 0.91 | 0.93 | 963 |
| Checkmark | 0.85 | 0.82 | 0.83 | 864 |
| Cross | 0.83 | 0.77 | 0.80 | 1009 |
| Right | 0.90 | 0.94 | 0.92 | 749 |
| Left | 0.87 | 0.90 | 0.88 | 832 |
| None | 0.86 | 0.94 | 0.90 | 1101 |
| Accuracy | | | 0.89 | 6514 |

range.

We show the one-frame preprocessing pipeline in Figure 6. To begin with, we convert the raw input from RGB color space into CIELAB color space and get a target mask based on the value of $A$ and $B$. On the construction site, the lightning condition can be various. In CIELAB color space, $L$ is corresponding to light intensity while $A$ and $B$ are stable with the different lighting conditions. Thus, we only use the value of $A$ and $B$ to get the mask which can avoid the influence of lighting conditions. After getting the mask, we add or minus a specific value to both $L$, $A$, and $B$ of pixels inside the mask. For $A$ and $B$, the value is corresponding to the gloves' color and we can determine it by checking the distance between the suitable range and the gloves' color range. For $L$, we run an adjusting module to find the suitable value under a specific lighting condition. We introduce the module in Figure 7. Finally, we convert back to RGB color space and send the output to the Mediapipe hand detector to get the tracking result.

We show the complete pipeline of the glove preprocessing method in Figure 7. At the very beginning, we run the parameter initialization module to determine $L$'s shifting value until the module successfully finds a suitable shifting value. For each frame after determining $L$'s shifting value, the process is the same as in Figure 6. In the parameter initialization module, the first step is to initialize the shifting value as the difference between the maximum $L$ value inside the mask and the maximum possible $L$ value. Mediapipe is more likely to successfully track object with higher $L$ value, so we attempt to set the pixels' $L$ value as high as possible inside the region of interest. After initialization, we run a loop to gradually increase the shifting value and check whether the Mediapipe can track the hand successfully. When the module succeeds, we set $L$'s shifting value and stop running the parameter initialization module. Otherwise, we keep on trying until succeed.

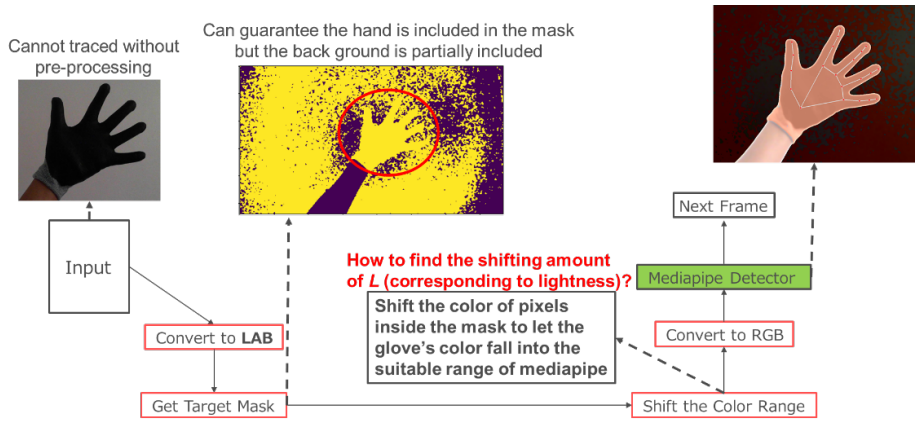



Figure 6: Preprocess in one-frame.

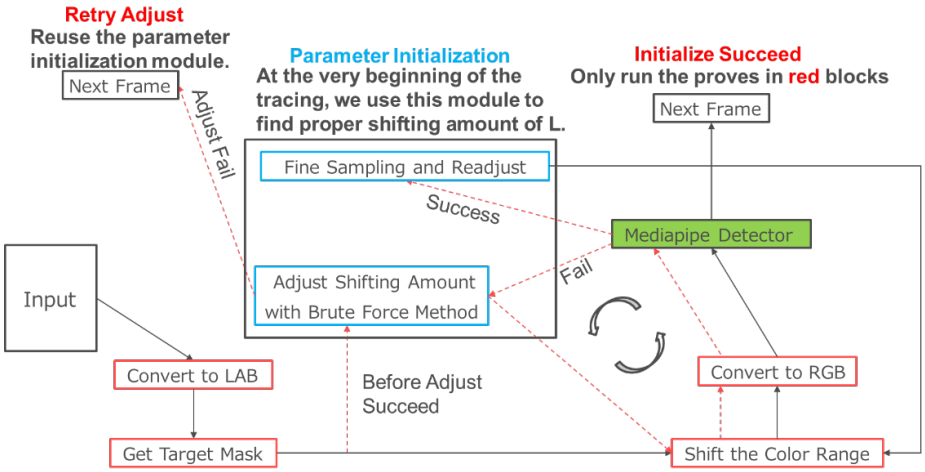



Figure 7: Glove preprocessing method pipeline.

# 5 Conclusion

Our project's goal is to recognize static and dynamic gestures with a vision-based method. Considering the demand on the construction site, we also develop a preprocessing method to adapt the recognition model to the situation of wearing gloves. We do a series of testing on the gesture recognition model, proving its reliability and applicability. For the performance of the glove preprocessing method, we haven't built a corresponding testing dataset, which is part of our future work. However, we have some sample demos on our project's GitHub page for fundamental reference [3]. We also include source code and instruction in the GitHub page. We will update the page as we step further in this project.

# 6 Future Plan

For the future plan we have two directions. The first one is refining the dataset. Even though using the padding prefix and suffix method in subsection 3.4 largely improves the performance in real-time video testing, the transition of different gestures in real world cannot be fully mimicked. Thus, we will using recording videos to create the dataset in the next step.

To test the glove preprocessing method scientifically, we are working on building a glove dataset including various validation samples. Given the complicated background in construction sites, we are scheduled to improve our method's robustness. Meanwhile, we are also interested in advancing the method's generalization ability for different kinds of gloves. In general, we will spend effort on enhancing the system's applicability and stability.

# References


[1] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "Mediapipe hands: On-device real-time hand tracking," 2019, arXiv:2006.10214.

[2] G. LLC, "Mediapipe hands," google.github.io. https://google.github.io/mediapipe/solutions/hands (accessed May. 22, 2022).

[3] Y. Lin and X. Chen, "Cerlab realtime handgesture recognition," https://github.com/Flora9978/CerLab_RealTime_HandGesture_Recognition/tree/glove (accessed May. 25, 2022).