



Static Sign-Language Recognition

NOVEMBER, 2022

Introduction

Sign Language Recognition is an approach that can be used to remove linguistic barriers between those who rely on sign language and those who do not. In order to do this, systems recognize gestures and then convey that information to a device or human. For instance, a program that can identify a gesture and textual convey that information would greatly facilitate communication. The application can also be extended to Virtual Reality, for the purposes of entertainment or simulation, and recognition of hand signs can be used to control objects (Wen et al., 2021). There is also an application in Robotics to mimic human gestures to perform anything from mundane household chores to space exploration (Schioppo et al., 1970).

The assignment will demonstrate the process of predicting which static ASL gesture corresponds with a number ranging from 0-9. The entire pipeline will be explained including concepts such as data collection and preprocessing.

Pipeline

In order to successfully complete the bonus portion of the assignment, the following steps will be deployed:

- 1) DATA COLLECTION - Find/create a dataset for classes 0-9
- 2) DATA CLEANING - Ensure the image is a connected component
- 3) SPLIT DATA - Perform 70-30 split into training images and testing images
- 4) EXTRACT FV - Through Image Processing, extract feature vector for each image
- 5) TRAIN MODEL - Store features of each class of the training set in separate clusters
- 6) PREDICT CLASS OF TEST SET - Based on minimum Euclidean distance measure, of each test image and each training class cluster, determine the class that the image is closest to

Steps 1 & 2 - Data Collection & Cleaning

Collecting the data was done in a systematic way, images were selected using Google and captured. The criteria to select the images was rigid, the images found had to be of digital art format and contain colours on the grayscale spectrum. It is important to note that this is not the typical procedure for image processing tasks - it is not feasible to carefully curate an extensive database of images.



Figure 1.0 - Examples of images gathered for Class 0

Once the images were gathered, each image needed to be modified to ensure the hand was not disconnected.

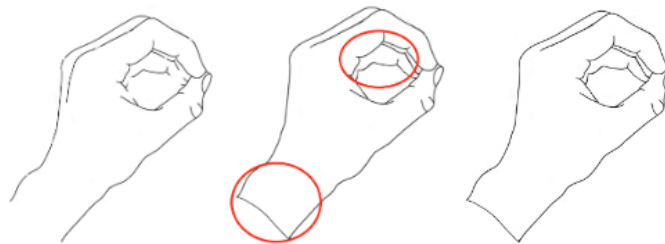


Figure 1.1 - Example of modification

The last thing to consider is the orientation - images were flipped horizontally to ensure that they all looked similar. This is a requirement as the second and third points in the feature vector (y, z) is highly dependent on the number of black pixels in the right or left section of the image. If the left-hand side typically holds more black pixels but the image is

flipped the opposite will be true. As evidenced by the image below, the number of black pixels on the RHS of the first image is not the same as on the RHS of the second image

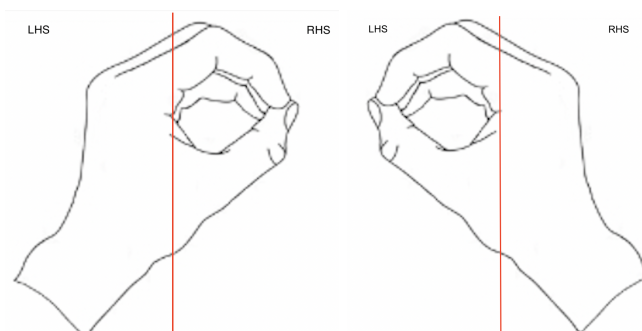







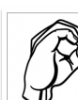

















Figure 1.2 - Importance of Orientation

Step 3 - Split Data

A 70-30 split was applied to all the images and their respective classes.

Class	Training Set	Test Set
0	 0_5  0_6  0_7  0_8  0_9  0_10  0_11	 0_1  0_2  0_3  0_4
1	 1_6  1_7  1_8  1_9  1_10  1_11  1_12  1_13	 1_1  1_2  1_3  1_4

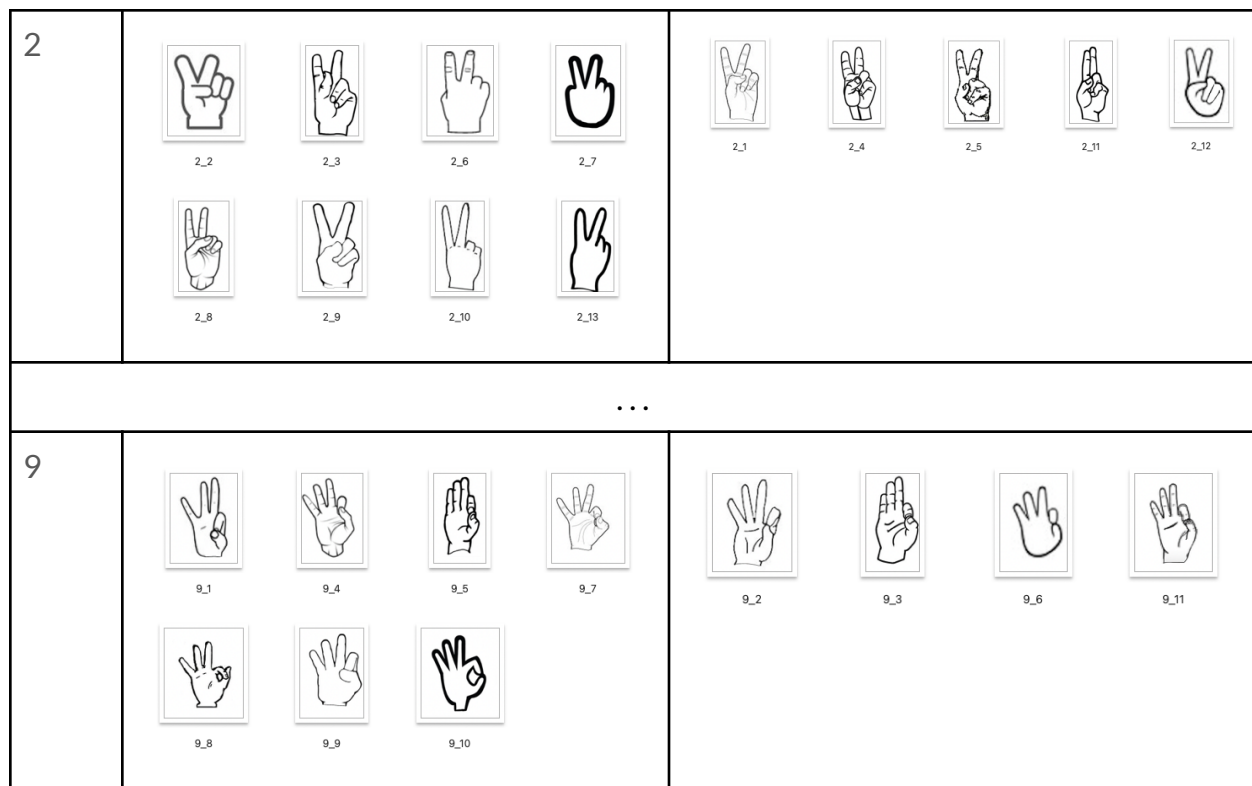


Figure 1.3 - Visual Representation of Train/Test Split



Step 4 - Extract FVs

After thresholding, and using the OpenCV function, `floodfill` (theory explained in A03) on the images: $[x, y, z] = [\text{the aspect ratio, the ratio of the left half black pixels area to the rectangle area, the ratio of the right half black pixels area to the rectangle area}]$




Figure 1.4 - Thresholded and Filled Training set of Class 0




Steps 5 & 6 - Train Model and Predict

In these steps, the feature vectors of the training set are stored in their respective clusters. In order to predict which class an image of the test set belongs to, the euclidean distance is calculated using d_{min} which is the distance between the nearest points of 2 clusters. The d_{min} of all 9 clusters and the image is compared, the smallest distance will denote the predicted class. Predictions will be made for each image in the test set and the accuracy will be based on how well the model can detect the correct class.



Results

In the implementation of the non-bonus assignment, the horns image is **predicted** to belong to **class 9**.

	Euclidean distance
	Zero
0	0.216
1	0.156
2	0.216
3	0.228
4	0.062
5	0.288
6	0.235
7	0.093
8	0.065

9	0.023
---	-------

This result is definitely understandable as objects from class 9 and 7 are structurally similar. Since the model does not care about the location of the digits, if we reconstruct an image from class 9 by removing the middle finger, or a third digit from the left, as shown in Figure 1.3, we end up with an image that is not far off from the horns image.

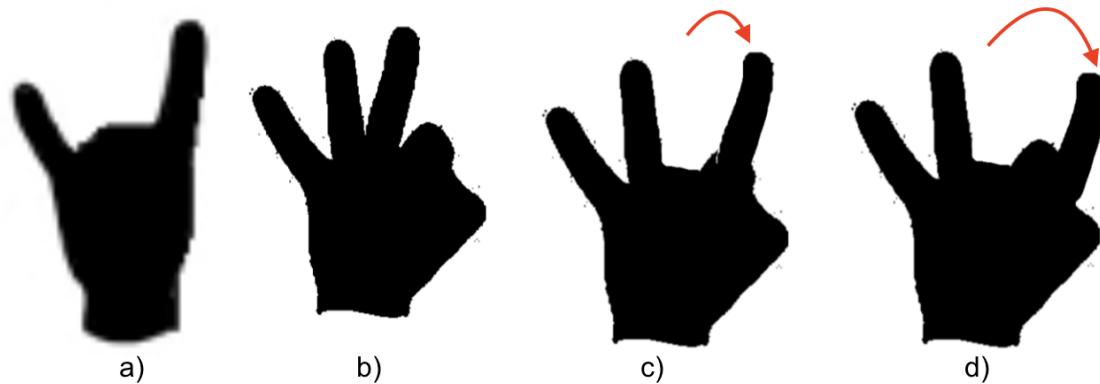


Figure 1.3 - a-d

a) Horns Image b) Image from class-9 c) Repositioned finger of image b)
d) Another repositioned finger of image b)

In the implementation of the bonus, the model was defined to have a **46.875%** recognition rate. The model often confused the sign number 4 with 5. It was able to predict the sign number 2 with high accuracy. These findings can lead way to improvements within the model.

Limitations

The accuracy of the model is low, this can be attributed to a few things. For instance, the scale, the dataset does not consist of enough images. More training images will increase the predictive power of the model. A method of augmenting the data without having to find new data is to apply techniques like Gaussian Blur or a slight rotation to the image.

Suggested Improvements

A frequent issue is the misclassification of the sign numbers 4 and 5, the differences between these classes are subtle. One way to correct this problem and improve accuracy is to add a new feature. I believe this feature should have something to do with the euclidean distance from the tip of the leftmost finger to the tip of the rightmost finger. For example, this distance would be 0 in class 0, and large for class 5.



References

- Schioppo, J., Meyer, Z., Fabiano, D., & Canavan, S. (1970, January 1). *[PDF] sign language recognition in virtual reality: Semantic scholar*. undefined. Retrieved December 1, 2022, from <https://www.semanticscholar.org/paper/Sign-Language-Recognition-in-Virtual-Reality-Schioppo-Meyer/6dc21e34adec60713c52e9ecc7ae5dc28fcfecb9>
- Wen, F., Zhang, Z., He, T., & Lee, C. (2021, September 10). *AI enabled Sign language recognition and VR space bidirectional communication using Triboelectric Smart Glove*. Nature News. Retrieved December 1, 2022, from <https://www.nature.com/articles/s41467-021-25637-w>