SIGN LANGUAGE RECOGNITION

Machine Learning Final Project

Sabeen Admani
Nili Krausz
Athulya Simon
Austin Laurence

Abstract

Sign language is poorly understood among the general public and, as a result, imposes a communication barrier between those who need to communicate with it and those who do not. It is in our interest to develop a visual classification method in which static images of signed alphabet letters are translated into readable text. Doing so could serve as first step in creating a meaningful mode of communication between the audibly impaired and remaining community.

Goals

Our goal was to distinguish between a few letters of the signed alphabet, with the overall goal of being able to convert from sign language to written letters. More specifically we wanted to distinguish the letters c, e, h, l, o, w, y, as these represent a good subset of the ASL alphabet. As a first step, we would distinguish between the letters that are very different from each other (e, h, l, o, and y) and then add in two letters that are similar to others (letters c and w would be the added letters, similar to o and y) in the first group if we make it past the first stage with time to spare. An additional goal was to recognize sign language regardless of the skin color of the user.

Methods

Feature Extraction

The first step in performing the classification was to extract features from the images. To ensure that we could produce a good classifier of the different hand poses, we wanted to produce several features that could allow for cross validation and selection of the best features. Since our goal was to be able to extract hands of varying skin colors, we chose to use a single blue background which would be easily detectable. We collected a large sample of images of the different sign language gestures using this background and had varied lighting between images. This allowed us to normalize for differences in skin color, or for variations in hand color (such as for one subject who had henna in several of the images).

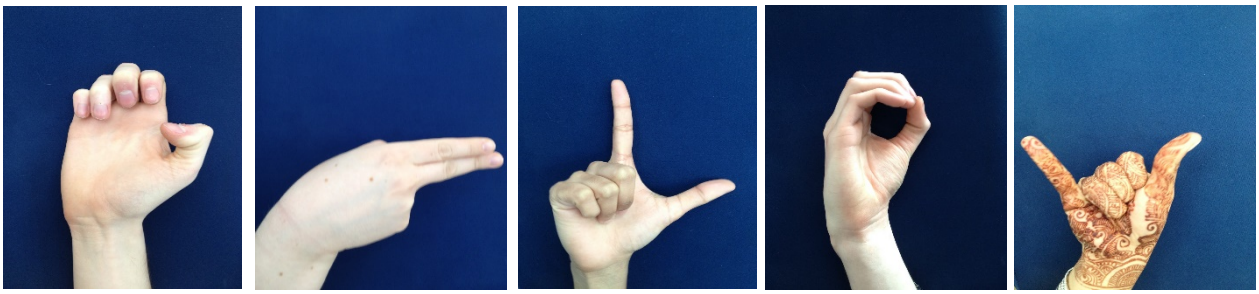

Figure 1: Subset of images that show the letters that were selected for this project, with the same background for each image, but with slight differences in lighting.

To extract the features we first find the color of the background using an imcrop() function in Matlab. This allows us to select part of the background and by converting to an HSV (hue, saturation and value) colorspace we could produce a histogram of the hue of the selected region. See the histogram shown in Fig. 2.

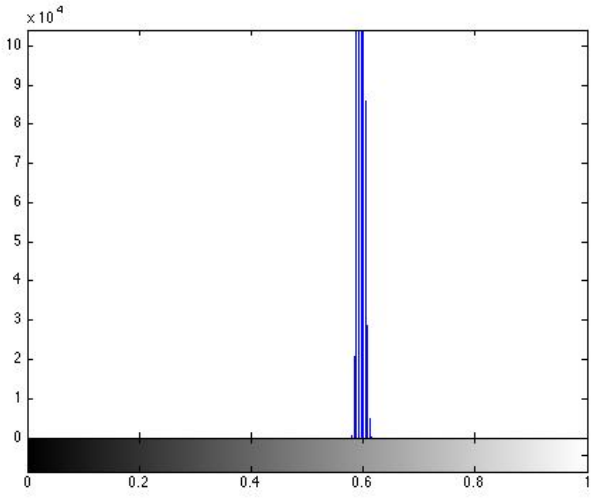

Figure 2: Histogram of Background Hue for Extraction of Testing Background

To demonstrate the feature extraction we will show a single example of the letter H. By defining a threshold of just above and just below the range of hues given by the histogram for the background we can segment out only those pixels that are in this range. This allows us to extract out the background and only find the hand. Figure 3 shows the sample image of the letter H, with the background extracted and shown as black on the right. This also gives us the first feature that we extracted from our data. From the foreground we could obtain the normalized surface area (normalized by the area of a rectangle the size of the hand).

The next step to obtaining our features is to extract edges from our image. We do this using a Sobel edge detector, and obtain the resulting edges shown in Fig. 3. These edges are faint and somewhat disconnected, so we then improved the edges extraction using a standard image processing method known as dilation. Our dilation procedure utilized a disk structuring element of size 2. The edges that we obtained after dilation are shown in the middle of Fig. 3. Using these edges we then performed a Hough Transform, which converts the edge points into the Hough domain, which is in polar coordinates rather than cartesian coordinates. This will convert lines into points, and by finding the points with the highest magnitude we can select the most significant lines in the image. The lines obtained using the Hough Transform are shown at the right of Figure 3.

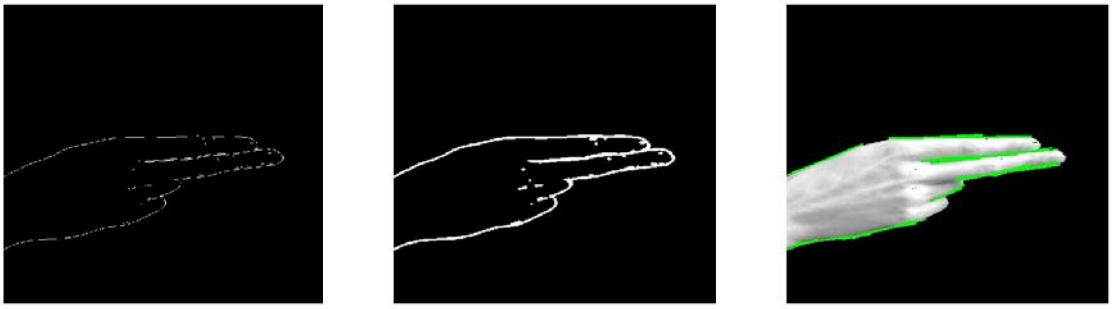

Figure 3: (a) Edges obtained by Sobel edge detector, (b) Edges after dilation with a size 2 disk structuring element, (c) important Hough lines extracted from Hough Transform

The next feature that we can extract is the histogram of the Hough lines. We separated the Hough lines into bins based on direction with each bin representing a range of 45°, and then the magnitude of each bin was a feature that we used in our final classifier.

The next thing we did for our feature extraction was to consider a method to compensate for color variations in the middle of the hand (such as the henna color difference). To eliminate errors due to the color variation we converted all of the foreground/hand pixels to an intensity of 255 (or white), and found edges and Hough lines. The Hough lines with and without converting the pixel intensity are shown for an example image in Figure 6. The Hough line histogram was then used as an additional feature for our classifier.

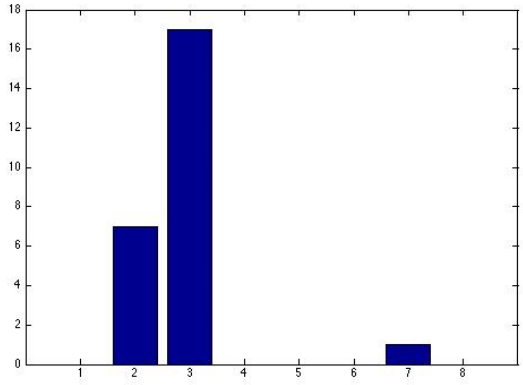

Figure 4: Histogram of the Hough Lines for letter H. Each in represents a range of 45 degrees, with bin 1 equal to the range of -22.5 to 22.5 degrees, etc. The larger the bar that is shown, the more lines there are in a given direction. In this case, there are the most lines in bin 3, which represents 67.5 to 112.5 degrees.

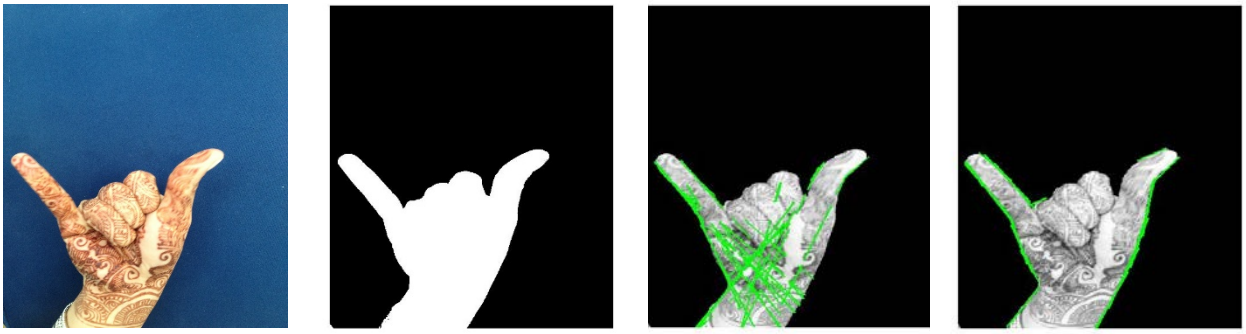

Figure 5: Improvements of changing pixel intensities for color variation. L-R: Original Image, Modified Foreground Intensity, Original Hough Lines, Modified Hough Lines

Finally, we used several other feature extraction methods, such as Histogram of the Gradients, or the maximum hough line. However none of the other methods ended up being used in the final classification, so we will not go into greater depth in this paper.

Cross Validation for Feature Selection:

Once we developed a method for extracting several different types of features from our image-sets, we needed to see which features could actually separate the one class from the rest. Another element that we wanted to determine was, if the data was separable, whether it was linearly so or not. If not, could we see a trend in the graph that would give us a hint as to how we could linearize that data?

Because we had so much data, we edited the log_poly_discrete_cross_validation_feat_select.m program that was uploaded in class such that it could go through iteratively and create a plot of each combination of the features with all of the classes on it in different colors such that they would be easy to distinguish from one another. This made it easy to go plot by plot (feature combination by feature combination) and then color by color to see how well one letter could be distinguished from the rest. An example of one

such figure is shown below. From this, we could see that the letter 'H' (yellow) is clearly linearly separable from the rest of the letters using the feature combination 8 and 4. In other plots, the data was very clearly separable using a quadratic or an ellipsoid.

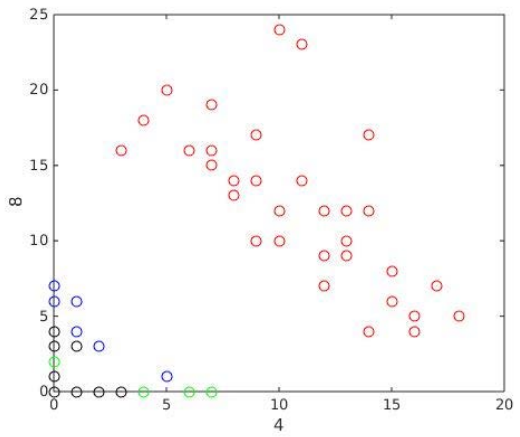

Figure 6: Figure showing all of the data for all of the letters plotted using features 8 and 4. Red is the letter 'H', which is clearly linearly separable from the rest using these features.

Support Vector Machine

Once we decided which features best separated all of the data, we needed to run this through the support vector machine (SVM) to identify a classifier. We used a soft-margin SVM with an approach that was similar to one vs. all in order to classify our five classes. The reason why we chose a soft-margin SVM is because we assumed our data does not exhibit *perfect* linear separability and wanted to relax the constraints of the original hard-margin SVM model. In order to accomplish this, we used the same method we were taught in class of adding in the regularization parameter (lambda).

To find the value of lambda that led to the smallest amount of error we performed an L2 regularized cross-validation scheme. In other words, we first split our data into training and testing sets. Then, we used soft-margin SVM to get a minimum x value for each function and evaluated the resulting data. The evaluation was done by finding the value of Transpose(D)*X (if the value of Transpose(D)*X is greater than 0, that means that the data point we were looking at was classified into that group; if that value is less than or equal to 0, then that data point was not classified into that group). After we found the resulting error, we found the minimum testing error and the lambda associated with it. In order to distinguish more than one letter, we iterated through the process described above for each letter (class) that we have.

Results

Resulting percentages for correct classification of the letters are displayed below.

Letter	Percent Correctly Identified in Testing Set
E	75.000000
H	96.875000
L	68.229167
O	73.958333
Y	76.041667

The figure below shows the data after it is classified into classes through the algorithm described above. We are using seven features instead of three, however so you can imagine that the 3-dimensional plot shown below that already shows some degree of clustering for each class would be much better separated in a 9-dimensional space.

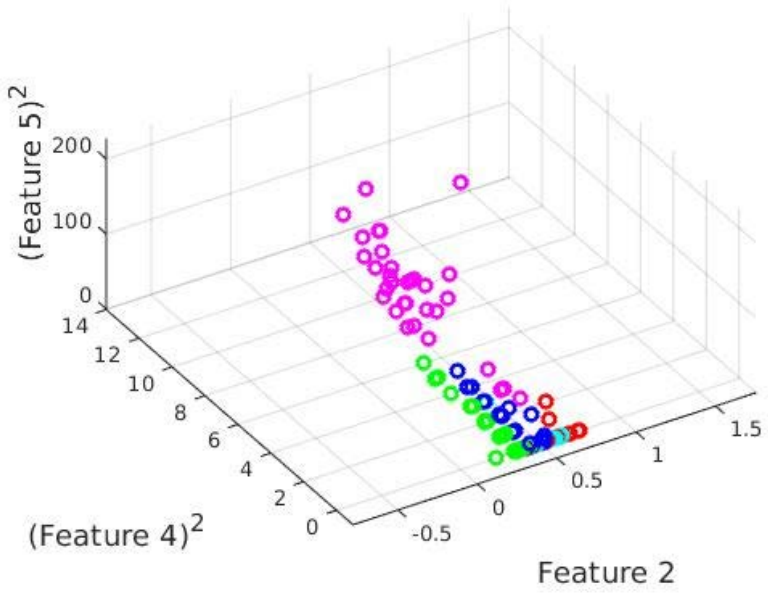

Figure 7: Figure showing all of the letters in 3 dimensional instead of 7 dimensional space.

Conclusions

As you can see, we achieved relatively good accuracy using the algorithm that we described, but there are always improvements that can be made. One improvement that we could make is to run the cross-validation algorithm to actually check which degree polynomial would best fit our nonlinear datasets. As stated before, we plotted them all and looked at what we thought would be appropriate divisions. This worked to an extent but the results could have benefitted from more appropriate feature linearization techniques. Additionally, we would like to make in the future is to make the feature selection algorithm more robust because, right now, we can only take in images that are taken against a solid background. As it stands, we're really proud of what we accomplished through this project; we learned a lot, so thank you so much for a great quarter!

Bibliography

J. Illingworth, J. Kittler, " A survey of the Hough Transform," Computer Vision, Graphics, and Image Processing. vol. 4, no. 1, 1988, p. 87-116.
http://www.sciencedirect.com/science/article/pii/S0734189X88800331

P. Kakumanu , S. Makrogiannis, N. Bourbakis, " A survey of skin-color modeling and detection methods," Pattern Recognition. vol. 40, no. 3, 2007, p. 1106-1122.
http://www.sciencedirect.com/science/article/pii/S0031320306002767

A. Katsaggelos, J. Watt, R. Borhani, "Machine Learning," unpublished book.