CS231A Project Milestone
Sign Language Gesture Recognition with Unsupervised Feature Learning

Justin Chen
Stanford University
justinkchen@stanford.edu

Abstract

This paper focuses on applying different segmentation approaches and unsupervised learning algorithms to create an accurate sign language recognition model.

Future Distribution Permission

The author of this report gives permission for this document to be distributed to Stanford-affiliated students taking future courses.

1. Introduction

The problem I am investigating is sign language recognition through unsupervised feature learning. Being able to recognize sign language is an interesting computer vision problem while simultaneously being extremely useful for deaf people to interact with people who don't know how to understand American Sign Language (ASL). I am planning on trying out different combinations of segmentation based on colors, shapes, edges, and depth data in order to obtain a clean, centered bounding box around hands in image frames which will then be passed through the unsupervised feature learning algorithm to classify the hand configuration.

2. Methodology

2.1. Problem Statement

2.1.1 Dataset

The data that I plan to use will be collected off of a Microsoft Kinect 3D depth camera. Videos will be taken of the test subject's hands while forming sign language letters. The data will be categorized as easy or difficult based on the amount of background clutter. This will be done with as many letters as possible to form a comprehensive model of sign language recognition. the initial dataset consists of the following 5 letters: (a, b, d, f, l).

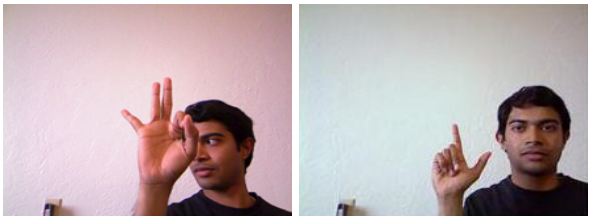

Figure 1: Example Image in Dataset of (Left) letter 'F', and (Right) letter 'L'

2.1.2 Expected Results and Evaluation

I expect to be able to segment out the hand and perhaps even be able to segment out specific visible fingers in any given frame. Other objects, such as faces, will be inappropriately detected by some of the segmentation methods I proposed, however, the depth data that we are collecting from Kinect will help differentiate between actual hands and other similar objects. These segmented results will be run through the unsupervised learning and tested on a separate held-out dataset. I believe that this approach should achieve a relatively high-performance classification since it is, at its core, similar to MNIST digit classification.

The final measure of my model performance will be based on the ratio of correct classifications out of a prepared validation set (recognition rate). False positives during hand-transition phases will also need to be penalized.

2.2. Technical Approach

2.2.1 Segmentation Methods

I am trying out many different methods of image segmentation and then combining the more successful ones together into a final polished segmentation tool.

One of the methods I am trying is using a Canny edge detector to find relevant "objects" in the field of view of the camera. The edges would then be dilated, and then all remaining holes in the mask will be filled to create a solid, continuous mask. Once this is done, the only the largest areas are taken in order to remove all the background clutter objects. This approach makes the simplifying assumption that the biggest objects seen in segmentation are typically of the most interest as well.

Figure 2: (Left) Edge detection results, (Right) Edge detection mask applied to image shown by black outline

I also tried out the following two approaches for skin segmentation using only color information. The first approach involved modeling the skin color by a 2D Gaussian curve and then using this fitted Gaussian to estimate the likelihood of a given color pixel being skin. First, I collected skin patches from 40 random images from the internet. Each skin patch was a contiguous rectangular skin area. I collected skin patches from people belonging to different ethnicities so that our model is able to correctly predict skin areas for a wide variation of skin color. The colors were then normalized as follows : $r = \frac{R}{R+G+B}, b = \frac{B}{R+G+B}$. The g component is ignored as it is linearly dependent on the other two. The mean and covariance matrix of the 2D Gaussian (with r, b as the axes) is estimated as follows : Mean $m = E[x]$, where $x = [r, b]^T$, Covariance $C = E[(x-m)(x-m)^T]$.

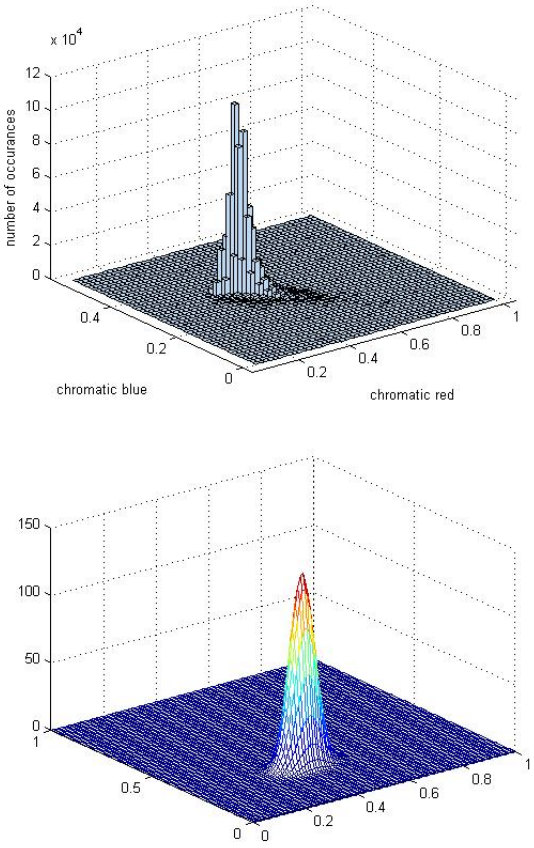

Figure 3: (Top) Histogram of color distribution for skin patches, (Bottom) Gaussian model fit

With this Gaussian fitted skin color model, the likelihood of skin for any pixel of a given test image can be obtained. If the pixel, has a chromatic pair value of (r, b), then the likelihood of skin for this pixel is given by:

$Likelihood = e^{[-0.5(x-m)^T C^{-1}(x-m)]}$, where $x = [r, b]^T$.

Finally, I thresholded the likelihood to classify it as skin or non-skin. However, this approach did not give significantly good results and failed to detect dimly illuminated parts of skin.

The second approach which I used is motivated by the paper [1], in which the authors first transform the image from the RGB space to the YIQ and YUQ color spaces. Then they compute the parameter $\Theta = tan^{-1}(V/U)$ and combine it with the parameter I to define the region to which skin pixels belong. Specifically, the authors called all pixels with $30 < I < 100$ and $105^o < \Theta < 150^o$

as skin. For my experiments, I tweaked these thresholds a bit, and found that the results were significantly better than our Gaussian model in the previous approach. This might have been because of two reasons:

1. The Gaussian model was trained using data samples of insufficient variety and hence was inadequate to correctly detect skin pixels of darker shades

2. Fitting the model in the RGB space performs poorly as RGB doesnt capture the hue and saturation information of each pixel separately.

Figure 4: (Left) Skin detected using Gaussian model, (Right) Skin detected using YIQ and YUV color spaces

In order to segment out the fingers, I am using convex hull detections to find the fingers after the hand has already been segmented out. The fingers will ideally be oriented along the direction from the convex hull point to the centroid of the hand as seen in Figure 5.

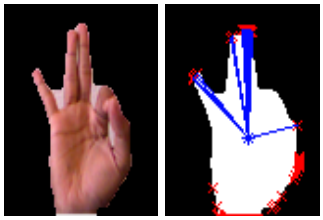

Figure 5: (Left) Skin model segmentation, (Right) Using convex hull detection to find potential "fingers"

In order to further improve the hand segmentation results as well as to get rid of a lot of extraneous clutter in the background (including the detected faces), I will be using the corresponding depth data from the Kinect to further segment the image using a simple thresholding function.

2.2.2 Feature Learning and Classification

The extracted data of hand images is fed into an autoencoder in order to perform the actual recognition part of the project. This stage implements an unsupervised learning algorithm. We feed all the data samples into the sparse autoencoder. The input data from the segmentation block are images of size 48x32 pixels. A sparse autoencoder is chosen initially with an input layer with 48x32 nodes and one hidden layer of 50 nodes. We used L-BFGS to optimize the cost function. This was run for about 400 iterations to obtain estimates of the weights. Now the autoencoder has learnt a set of features similar to edges. The next step is to classify the 5 different letters based on the features learnt. The output of the hidden layer of the autoencoder is fed into a softmax classifier to now classify the data into 5 categories. The softmax classifier again learns using the L-BFGS optimization function. This algorithm converges after about 20 iterations. Further improvements to these results include gathering more training samples and trying cross validation to improve prediction rate.

3. Preliminary Results

Segmentation examples/results can be seen throughout the "Technical Approach" section. The autoencoder and softmax classifier learning approach achieved 95-97% accuracy on the training set. I haven't done any actual cross-validation yet, but we aim to get that done before the final project deadline. I aim to improve the segmentation through use of depth data, and then adding more letters to our dataset to make the project more interesting and useful.

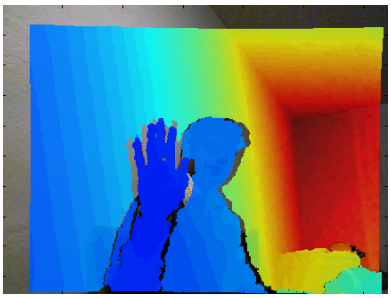

Figure 6: Depth data overlayed on top of RGB data

References

[1] X. Teng. A hand gesture recognition system based on local linear embedding, April 2005. Journal of Visual Languages and Computing.

4. Appendix

This project is done in combination with the CS229 Machine Learning final project. The CS231A Computer Vision primary component is the hand and finger segmentation using 3D camera.