Color Classification with Logistic Regression For Stop Sign Detection

James Salem
Department of Electrical and Computer Engineering
University of California, San Diego
jgsalem@eng.ucsd.edu

***Abstract*—This paper presents a method for detecting stop signs by using logistic regression for color classification. Images are segmented into regions of "red" and "not red" and computer vision techniques are used to determine if a red region is a stop sign. The result of using this technique was good image segmentation however the method for determining stop sign regions was not very robust.**

Keywords—Logistic Regression, Image Segmentation, Color Classification, styling

I. Introduction

Since driverless vehicles will be expected to follow the same rules of the road as human drivers, it is critical that they are able to identify street signs. Signs on the road inform drivers of speed limits, pedestrian crossings, when to stop, etc. and are often there to protect drivers and pedestrians. It is important that an autonomous vehicle is able to identify a sign and react appropriately.

In the case of this project, we are focusing on the detection of stops signs, which are red octagonal signs with the word "STOP" written on them. They are meant to signal to drivers that they must come to a full stop.

The approach in this paper will be to first segment the image into regions of "red" and "not red" using a logistic regression color classifier. The next step is to identify which of the "red" regions are similar to the shape of a stop sign.

II. Problem Formulation

A. Color Classification

The first step in this approach is to segment the image into "red" and "not red" regions. In order to perform segmentation, each pixel of the image must be classified as either "red" or "not red." A pixel is the smallest element of an image and consists of a single color. The pixel color can be described by its position in color space, however there are many different color spaces. For example, a pixel in the RGB colorspace can be described by three values: the Red, Green, and Blue components.

Given an image, ie a set of pixels values, classify each pixel as either belonging to the "red" class or the "not red" class.

B. Shape Identification

Once segmentation is performed, the result is a binary mask where "red" regions are represented by 1's and "not red" regions are represented by 0's. The goal is given a binary mask, to determine if any of the "red" regions are similar in shape to a stop sign.

III. Technical Approach

Below I will describe technical approaches taken to solve the problem of color classification and shape identification with regards to stop sign detection.

A. Color Classification with Logistic Regression

.In this case, we are trying to classify colors into two classes: red and not red. For this binary classification problem, I chose to use a logistic regression model to represent the class conditional probabilities:

$$p(y \mid x, \omega) = \sigma(yx^T\omega) = \frac{1}{1 + e^{-yx^T\omega}} \quad (1)$$

where the class $y \in \{-1,1\}$ and $x = (x_R, x_G, x_B) \in RGB$

In order to estimate the model parameters, $\omega \in \mathbb{R}^3$, I used Maximum Likelihood Estimation (MLE) which results in the following expression:

$$\omega_{MLE} = \operatorname{argmin} \sum_{i=1}^{n} (1 - e^{-y_i x_i^T \omega}) \quad (2)$$

Where y_i is the i'th label and x_i is the i'th pixel value in the training dataset.

Since equation 2 has no analytical solution, we must rely on solving it numerically. In order to find the solution for ω_{MLE}, I have chosen to use the optimization method of gradient descent. Gradient descent is an iterative optimization method so it requires an initial guess for the parameters ω. I have chosen $\omega = (0,0,0)$ as my initialization of parameters though this decision should not have a great impact on the results. The update equation for finding the MLE solution for the parameters is the following:

$$\omega_{MLE}^{(t+1)} = \omega_{MLE}^{(t)} + \alpha \sum_{i=1}^{n} y_i x_i (1 - \sigma(y_i x_i^T \omega_{MLE}^{(t)})) \quad (3)$$

Where α is the step size, which is a scaling factor that tells the update equation how much it should go in the direction of the gradient, and t is the iteration. $\omega_{MLE}^{(t+1)}$ is the parameter being computed and relies on the previous iteration's parameters, $\omega_{MLE}^{(t)}$. I chose a step size of $\alpha = 0.1$ which is small enough so that the solution does not diverge.

After each iteration of the gradient descent algorithm, I perform segmentation on a validation set, which consists of images not in the training set, and compute an accuracy score:

$$\text{accuracy} = \frac{\text{number of pixels correctly segmented}}{\text{total number of pixels}} \quad (4)$$

Once the parameters have been found, the decision boundary can be determined by setting the class conditional probabilities, $p(y = -1 \mid x, \omega)$ and $p(y = 1 \mid x, \omega)$, equal to each other. The resulting decision boundary is:

$$y = \begin{cases} 1 & x^T\omega \geq 0 \\ -1 & x^T\omega < 0 \end{cases} \quad (5)$$

In order to make my model more expressive, I added a bias term to the parameters making $\omega \in \mathbb{R}^4$ and in turn added a dimension to the data making each pixel $x = (1, x_R, x_G, x_B)$. Therefore the resulting decision boundary is:

$$x^T\omega_{MLE} = \omega_0 + \omega_1 x_R + \omega_2 x_G + \omega_3 x_B \quad (6)$$

Which represents a plane in RGB space. The added bias term, ω_0, acts as the intercept allowing the decision boundary to be shifted thus giving the model more flexibility.

B. Shape Identification

Once each pixel in an image has been classified as either red or not red, the result is a binary mask with 0's representing not red and 1's representing red. The goal of this section is to identify red regions that are similar in shape to a stop sign.

The first problem that I ran into is that if a stop sign is small, sometimes the top and bottom of the sign will be separated into two regions. In order to try and connect close regions, I used a dilation operation with a (3,3) kernel to expand the edges of regions.

The dilation operation was able to connect close regions but created a new problem: small noise or small inconsequential regions of red were becoming larger and more easily confused with stop signs after dilation. To combat this issue, prior to performing the dilation, I removed regions deemed too small. The following is the criteria used to determine if a region should be removed:

$$\text{Region Area} < \frac{(\text{image width} * \text{image height})}{1000} \quad (7)$$

After removing noise and dilating, the next step was to approximate red regions as polygons. My first instinct was to simply check if the number of sides of the polygon were equal to eight, but, due to the rounding nature of the dilation operation, this would not work for small stop signs. After dilation, small stop signs became more circular than octagonal. So instead of requiring the number of sides to be precisely equal to eight, I required that the number of sides be greater than or equal to seven.

The final criteria was to check the squareness of the region's bounding box. Since stop signs area regular polygons, the ideal bounding box has equal width and height. Of course in reality pictures can be taken at various angles so this equality will never really be true. In general, however, the bounding box of a stop sign region should be neither "tall" or "fat." The following is the criteria I used to determine if a region bounding box is sufficiently square:

$$\text{width} - \text{height} < 0.50 * \text{width} \quad (8)$$

IV. Results

A. Training the Logistic Regression Model

After each iteration of the gradient descent algorithm, I tested the parameters by segmenting each of the ten images in my validation set, which consist of images not found in the training set. The accuracy is calculated according to (4). A plot of the accuracy after each iteration of the gradient descent algorithm is shown Figure 1. Over the first 100 iteration the variation accuracy decreases but from 100 to 300 the variation stays rather consistent.

I selected the parameters after iteration 292 because they result in high accuracy when tested on the validation set. The resulting parameters used in the logistic regression model are $\omega = (-37532.89, -171158.85, -185771.03, 192951.91)$. These parameters result in an accuracy of 99.0% when tested on the validation set.

B. Image Segmentation

To segment an image, I plug each pixel value along with the parameters listed in the section above into (5) in order to get the label for that pixel. I then map the -1's to 0's in order to get a binary mask. Good examples of the segmented images are shown in Fig. 2, 3, and 4 along with the original images which are from the validation set. Fig. 5, on there other hand, shows the worst segmentation example from the validation set. The bottom stop sign is well segmented but the top one is only partly captured in the segmentation. This type of issue may be reduced by training the model with pixel data in a different colorspace.

C. Shape Identification

Regions that meet the criteria described in section III part B are considered stop signs and the bounding boxes that surround these regions are found. Fig. 2, 3, and 4 show good examples with bounding boxes drawn on the images. Fig. 3 is particularly good as it shows the stop sign correctly identified despite there being other red regions in the segmented image. Fig. 5 shows a bad example of identification as a red car is erroneously identified as a stop sign. This method is far from perfect. There are occasional false positives particularly in cases where there are other red regions in an image and false negatives particularly in cases where stop signs are far away.

D. Conclusion

Overall, this method of image segmentation with a logistic regression color classifier and shape identification works decently but is far from perfect. A generative model for color classification may prove more robust for image segmentation than the discriminative logistic regression that was demonstrated in this paper.

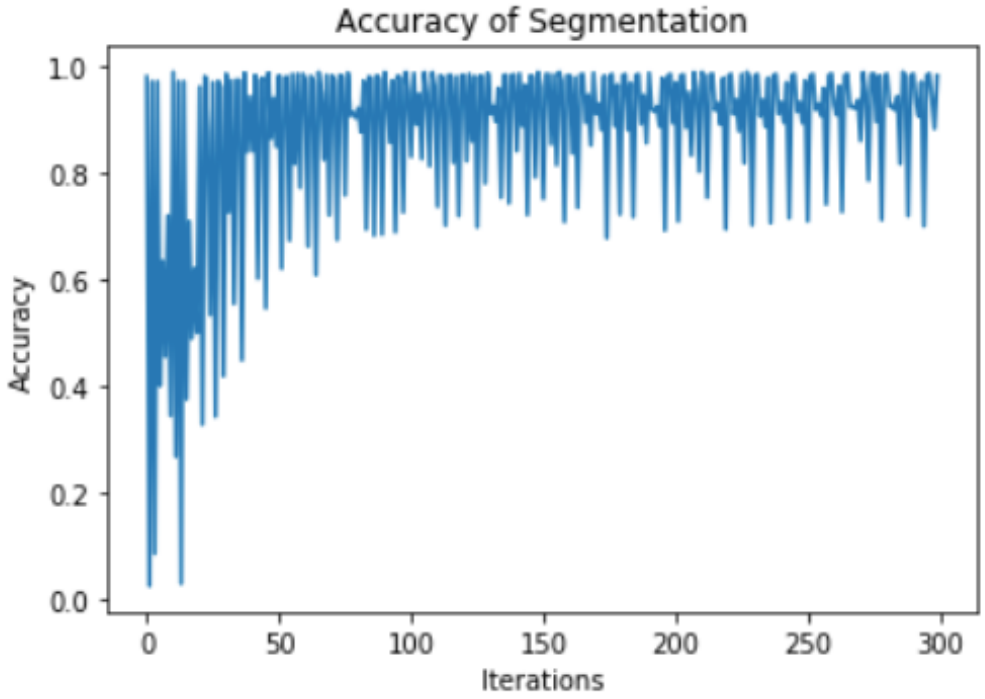

Figure 1: Accuracy of segmentation on validation set during training

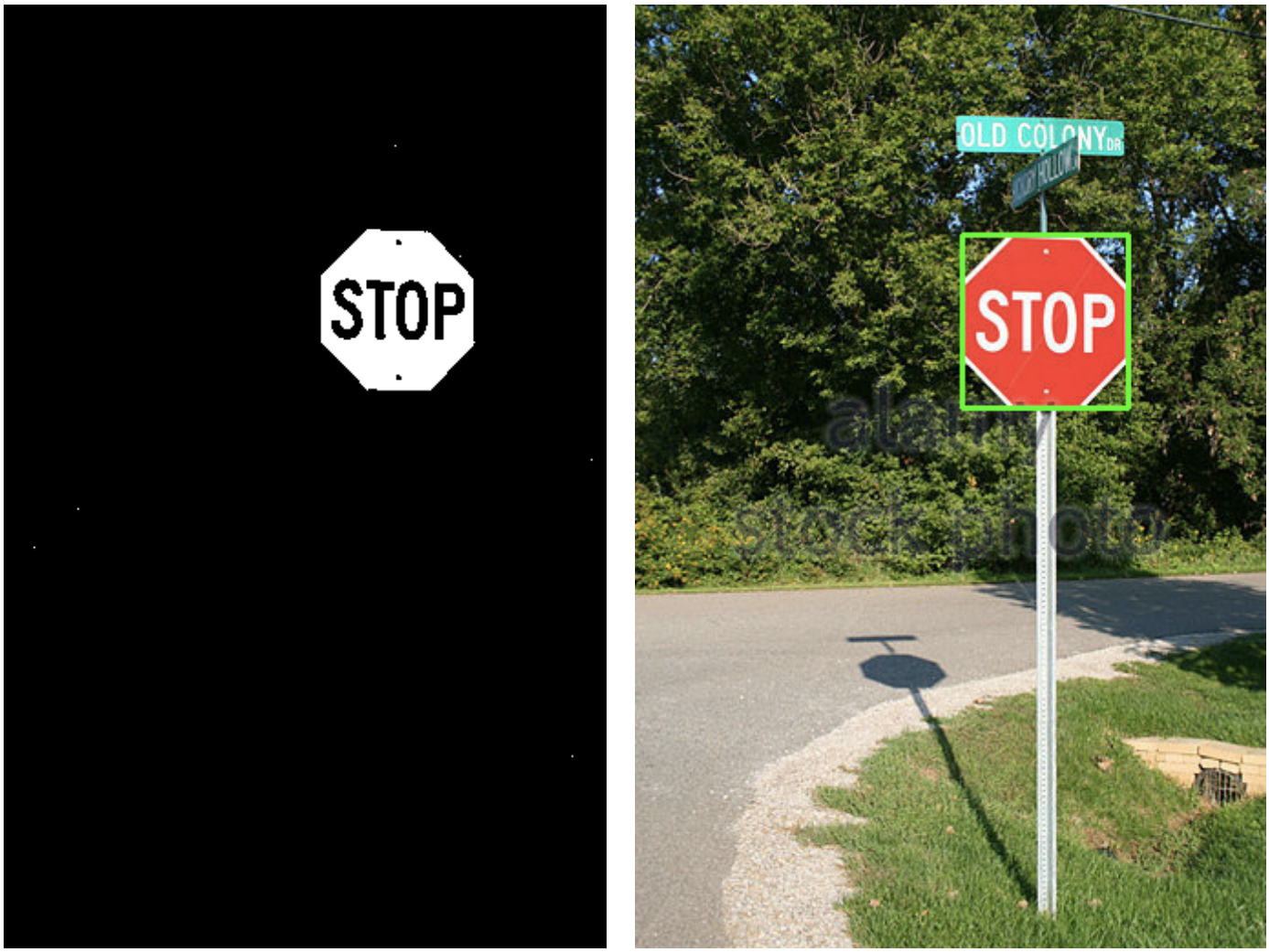

Figure 2: Good example from validation set with segmented image (left) and the original image with bounding box (right). The coordinates of the bounding box are (180, 318, 271, 413)

Figure 3: Good example from validation set with segmented image (left) and the original image with bounding box (right). The coordinates of the bounding box are (95, 148, 125, 179)

Figure 4: Good example from validation set with segmented image (left) and the original image with bounding box (right). The coordinates of the bounding box are (779, 482, 957, 659)

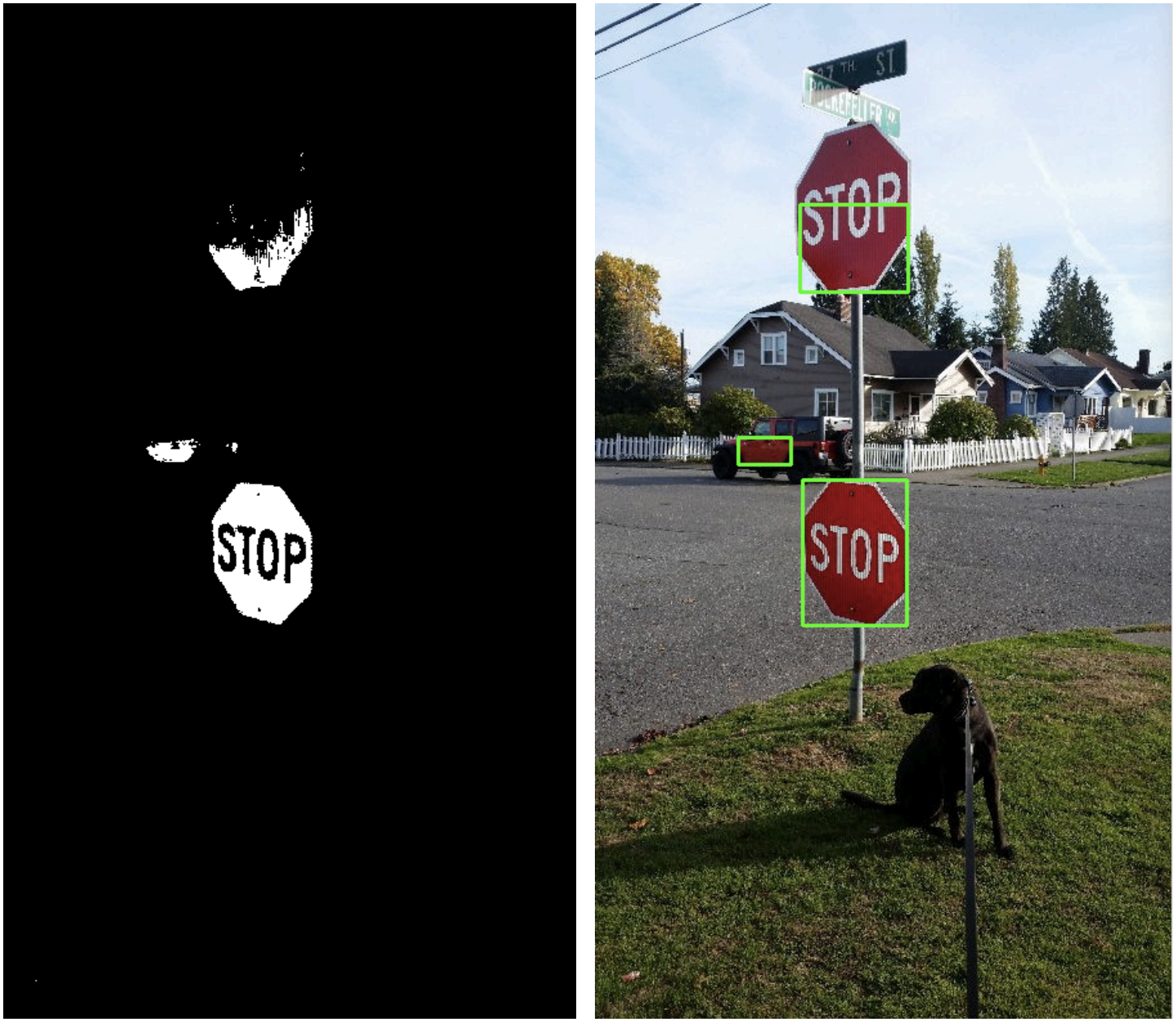

Figure 5: The worst example from the validation set with segmented image (left) and the original image with bounding boxes (right). The coordinates of the bounding boxes are (113, 441, 154, 462), (161, 575, 245, 643), (163, 316, 244, 429)