Stop Sign Detection with Logistic Regression

Jiaming Lai[1]

I. INTRODUCTION

Stop sign detection problem contains two part: (1) use a classifier to segment red and non-red regions given an image; (2) based on the segmentation result, detect the stop sign position on the original image. In general, this is a computer vision task. This problem is quite important for the following reasons: (1) an autonomous vehicle or robot should be able to detect traffic signs when running on real road; (2) LiDARs and RADARs couldn't provide color space information, it is not a good idea to detect stop sign with LiDARs or RADARs, hence one possible way is to use camera sensors, which leads to the visual task we talk above. The main challenges of this problem comes from the fact that: (1) lighting and resolution of images vary; (2) some part of a stop sign may be obscured by other objects in images.

In our case, we propose a linear classifier by Logistic Regression to detect red and non-red pixels given an image. The linear classifier generates a mask as a result of color segmentation. We then apply dilation and erosion to the segmentation mask, following by filling holes and removing small clusters. Finally, we use processed result decide the position of stop signs and draw bounding boxes.

Data set was divided into two part: one is train-set for training a linear classifier using Logistic Regression, the other one is validation-set for evaluating detection performance.

The rest of paper is arranged as follows. First we give the detailed formulations of stop sign detection problem in Section II. Techinical approaches are introduced in Section III. And in the end we setup the experiment, results and discussion are presented in Section IV.

II. PROBLEM FORMULATION

A. Color Segmentation

The object of this part is to classify red and non-red pixels in an image, using a color classifier. Given a new image $\{I \in \mathbb{R}^{r\times c\times 3}, 0 \le I_{ijk} \le 255\}$(where r and c is the number of row and column of the image, number 3 corresponds the three color channels), try to generate a mask $\{M \in \mathbb{R}^{r\times c}\}$ where $M_{ij} = 1$ if the pixel in row i and column j is red, otherwise $M_{ij} = -1$ if the pixel in row i and column j is non-red.

*This work was an project of ECE276A, University of California San Diego

[1]Jiaming Lai is with Master of Department of Electrical and Computer Engineering, University of California San Diego, 9500 Gilman Dr, La Jolla
jil136@ucsd.edu

B. Stop Sign Position Detection

In this part, given a segmentation mask from Section A, the object is going to determine the position of stop signs in the original image. The position of stop sign is represented by a bounding box $\{B = (x_{BL}, y_{BL}, x_{TR}, y_{TR})^T \in \mathbb{R}^4\}$, where x_{BL}, y_{BL}, x_{TR} and y_{TR} is corresponding to x coordinate of bottom left point, y coordinate of bottom left point, x coordinate of top right point and y coordinate of top right point of the bounding box. The coordinate and origin we use here is shown as figure below.

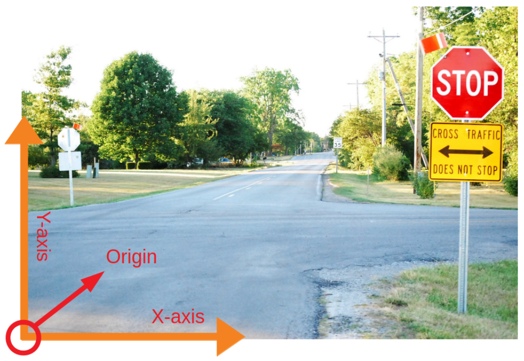


Fig. 1: Coordinate and origin of bounding boxes

In some cases, there would be more than one stop signs in an image. So the final result should be a matrix $\{F \in \mathbb{R}^{4\times n}\}$, where n is the number of bounding boxes.

III. TECHNICAL APPROACH

In this section, we will discuss about the algorithms and methods used in our project.

A. Image pre-processing and color space

In this project, we use RGB color space. In every iteration during training phase, every pixel of training image batch is exacted and finally form a matrix $\{T \in \mathbb{R}^{n\times 3}\}$, where every row is one pixel with three values of RGB color space and n is the number of pixel. An extra dimension of 1s is stacked as 4th column of matrix T, and finally we get our training input

$$X = [T, \mathbf{1}] \in \mathbb{R}^{n\times 4}$$

The masks corresponding to training image batch are reshaped and stacked to form a vector $\{\mathbf{y} \in \mathbb{R}^{n\times 1}\}$, where $\mathbf{y}_i = 1$ if the pixel in row i of X is red, otherwise $\mathbf{y}_i = -1$.

B. Logistic Regression

1) *Parameter Initialization:* Parameter initialization is very important. In this project, parameter W is initialized as

$$\omega = std * [\omega_1, \omega_2, \omega_3, \omega_4]^T$$

where ω_1, ω_2, ω_3 and ω_4 is sampled from the "standard normal" distribution. Here std is a hyper-parameter, by default, we set it as $std = 1e3$.

2) *Training Phase:* There are several hyper-parameters in this project during training phase. The following is the hyper-parameters name and corresponding meaning.

   – *Number of iteration*: number of steps to take when optimizing. By default we set as 150.
   – *Batch size*: number of training examples to use per step. In this project, we select 40 images as training data and randomly choose 8 images as training examples to use per step.
   – *Learning rate*: scalar giving learning rate for optimization. By default, we set it as 0.001.
   – *Learning rate decay*: scalar giving factor used to decay the learning rate after each epoch. By default, epoch is set as 10 steps during training and learning rate decay as 0.95.

   The objective function during training is

$$\begin{aligned}\omega^* &= \min_{\omega}(-logP(\mathbf{y}|X,\omega)) \\ &= \min_{\omega}\left(-log\prod_{i=1}^{n}\frac{1}{1+exp(-\mathbf{y}_i X_i^T\omega)}\right)\end{aligned} \tag{1}$$

   Negative log-likelihood can be minimized to obtain a minimum by using SGD(stochastic gradient descent). The parameter update rule is as following

$$\begin{aligned}\omega^{(t+1)} &= \omega^{(t)} - \alpha\nabla_{\omega}(-logP(\mathbf{y}|X,\omega))|_{\omega=\omega^{(t)}} \\ &= \omega^{(t)} + \alpha\sum_{i=1}^{n}\mathbf{y}_i X_i(1-\sigma(-\mathbf{y}_i X_i^T\omega^{(t)}))\end{aligned} \tag{2}$$

   The convergence condition is depending on gradient. If the gradient in current step is smaller than a threshold we set, then we regard is as convergence.

3) *Predicting Phase:* Given a new test image, pre-process it as we discuss in Section A and finally get our test input

$$X_* = [T_*, \mathbf{1}] \in \mathbb{R}^{n\times 4}$$

   Given a test pixel x_*, use the optimized parameters to predict the label and the decision rule is:

$$y_* = \begin{cases} 1 & , x_*^T\omega^* \geq 0 \\ -1 & , x_*^T\omega^* < 0 \end{cases}$$

   The predicted result is reshaped as a segmentation mask $\{M_* \in \mathbb{R}^{r\times c}\}$, where r and c is the number of row and column of the test image.

C. Stop Sign Position Detection

Once we get the segmentation mask $\{M_* \in \mathbb{R}^{r\times c}\}$ from classifier given a new test image, we can perform stop sign detection and draw a bounding box of the stop signs.

However, due to lighting variance and other negative influence, we could not always get all red region on a stop sign. So the segmentation mask of stop sign is usually broken and the contour is not a ideal Octagon. Moreover, there are other red region in some images, so we need to filter out these region on the segmentation mask. One example segmentation mask is shown below:

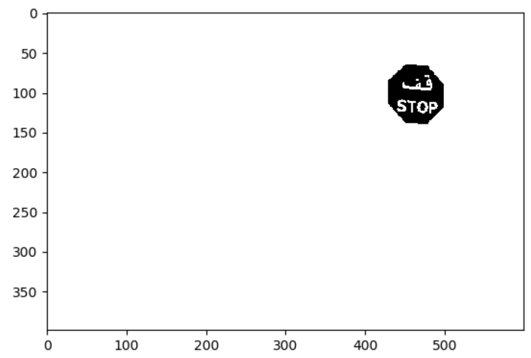


Fig. 2: Example of segmentation mask

In this project, we perform several methods to process the predict mask before detecting stop sign and drawing bounding box. These methods are shown as following step by step:

- *Dilation and erosion*: since the red region in our predict mask is black, we perform **CLOSE** operation onto predict mask, in order to close the black region in segmentation mask and remove "STOP" string white region within stop sign region.
- *Fill small holes*: in this step, we fill some small holes in black region of mask.
- *Remove small cluster*: this step is for removing small cluster in segmentation mask.

Once we finish the step above, we can get a better processed segmentation mask for stop sign position detection. We then use **findcontours** function to find contours on processed segmentation mask. Sort these contours by the key of their contour area and only keep three largest contour area. We then perform conditional detection based on different information such as: contour area, number of edges. Finally, we use **regionprops** function to draw the bounding boxes and print coordinate of bounding boxes.

IV. RESULTS

In this section, we will discuss test result on validation set and Autograder. The validation images we use in following discussion are: **19** and **33**.

A. *Validation set and autograder results*

The following test result of figure **19** and **33** choose from validation set. We will first post the mask result corresponding to red region in original image. After that, we will post bounding box result.

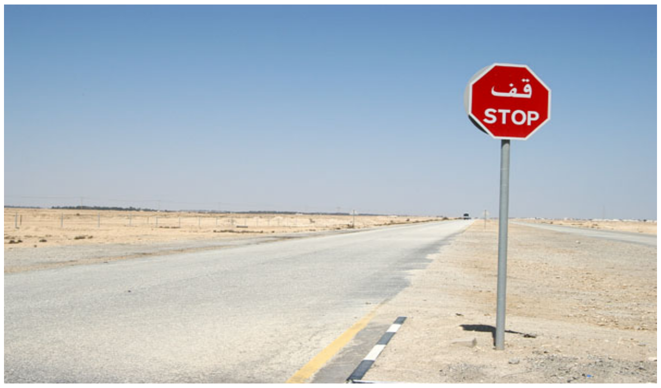


Fig. 3: Original image of image 19

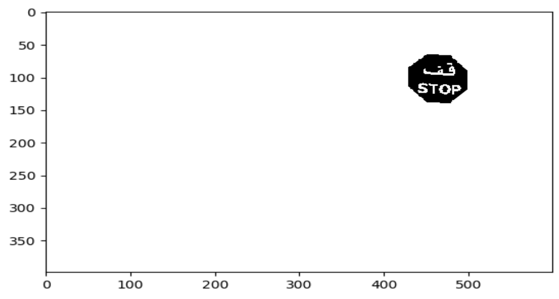


Fig. 4: Segmentation mask of image 19

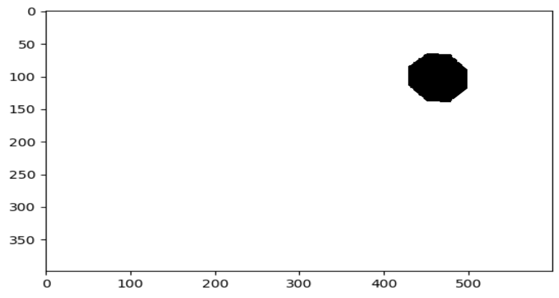


Fig. 5: Processed segmentation mask of image 19

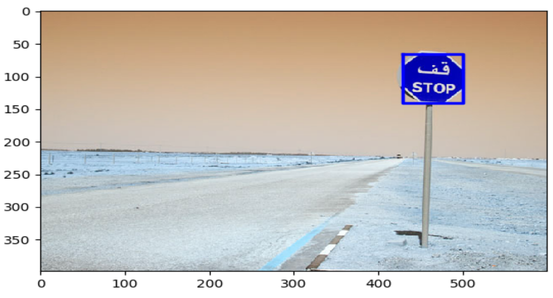


Fig. 6: Bounding box result of image 19. Box coordinate is [429, 259, 500, 334]

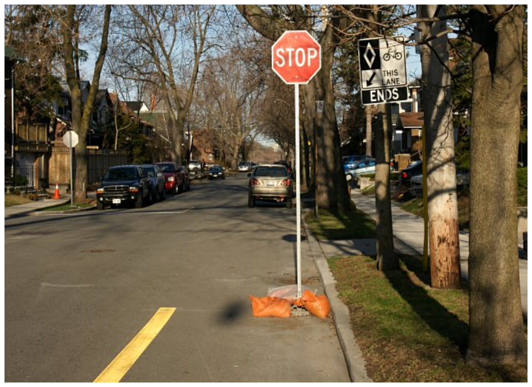


Fig. 7: Original image of image 33

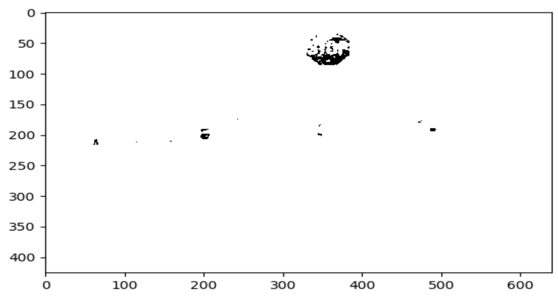


Fig. 8: Segmentation mask of image 33

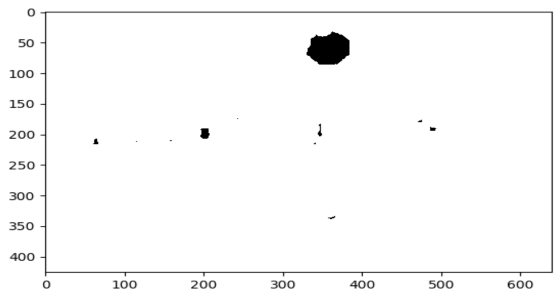


Fig. 9: Processed segmentation mask of image 33

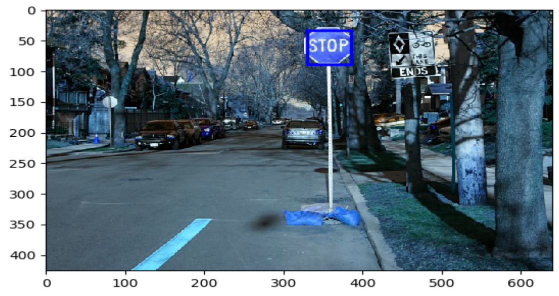
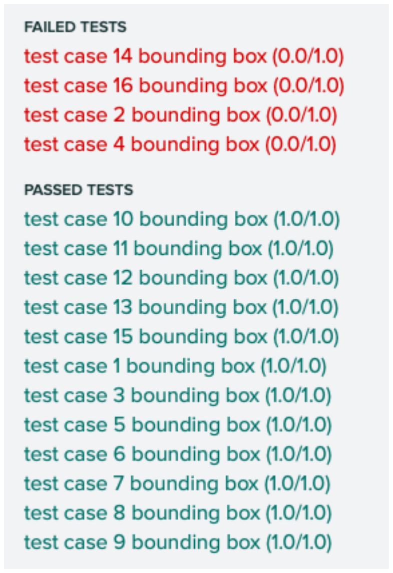


Fig. 10: Bounding box result of image 33. Box coordinate is [330, 338, 385, 393]



```
FAILED TESTS
test case 14 bounding box (0.0/1.0)
test case 16 bounding box (0.0/1.0)
test case 2 bounding box (0.0/1.0)
test case 4 bounding box (0.0/1.0)

PASSED TESTS
test case 10 bounding box (1.0/1.0)
test case 11 bounding box (1.0/1.0)
test case 12 bounding box (1.0/1.0)
test case 13 bounding box (1.0/1.0)
test case 15 bounding box (1.0/1.0)
test case 1 bounding box (1.0/1.0)
test case 3 bounding box (1.0/1.0)
test case 5 bounding box (1.0/1.0)
test case 6 bounding box (1.0/1.0)
test case 7 bounding box (1.0/1.0)
test case 8 bounding box (1.0/1.0)
test case 9 bounding box (1.0/1.0)
```

Fig. 11: Autograder result

B. *Discussion*

As is shown above, our method work well on 19 and 33 image from validation set. The resolution of these images are relatively high and there is few red region in original image. So our method work perfectly in these image.

But as we can see in the result of autograder, our method fails to detect stop sign in 4 test cases. Possible reason would be that: (1) there is large red region in some test cases, leading to huge interference onto our segmentation mask; (2) dilation and erosion method would destroy some contour information of stop sign region, resulting in bad influence on our stop sign detection; (3) some stop sign is small on original image, so it is easy to be filtered out during sorting out three largest contour area.

V. COOPERATION

The morphology part for processing segmentation mask is under cooperation with Zhiqiang Sun and Yang Yue.