Stop Sign Detection and Localization Using Logistic Regression

Baoqian Wang
Department of Electrical and Computer Engineering
University of California, San Diego
San Diego, CA, 92093
Email:bawang@ucsd.edu

***Abstract*—This project investigates detecting and localizing stop sign in an image. In particular, the logistic regression model is used to classify the pixel values of the image and produce the segmented image. The regions of interests are then identified. By analyzing the geometry properties of the region, stop sign can be detected and localized The test results show the promising performance of the method.**

I. Introduction

In recent years, autonomous driving becomes a hot topic in both industry and academia. To achieve that, one of the most important issue is the objection detection and localization through varies sensors such as cameras, GPS, Lidar and so on. Examples include traffic light detection [1], road detection [2] and so on. This project focuses on stop sign detection and localization.

Many studies have been done on stop sign detection and localization. Existing works include using deep neural networks such as deep convolutional neural network [3], [4] and traditional machine learning approaches such as classifier algorithms [5], also image processing algorithms such as clustering of images pixels to extract features of stop sign [6] . Among these methods, deep learning gains large success and outperforms other methods due to the large amount of available images for training and powerful computation capability to train the model. The typical one is YOLO (You only look once) [7], which can be used to detect and localize multiple object types including car, person, traffic light and so on. More importantly, it can also be used for stop sign detection and localization by transfer learning.

Since the deep learning algorithms in objection detection and localization has been mature, this project focuses on traditional machine learning approaches. In particular, in this work, a logistic regression model is used to classify the the pixel value of the image based on the labels of training images. The trained model can then be used to generate a segmented image, on which regions of interest can be identified. To determine if a stop sign exists or not, the geometry properties of the region are analyzed. After the detection of the stop sign, a bounding box is further generated.

The model is evaluated by testing on images that contains different positions, sizes, patterns of stop signs. The results show that the model can achieve promising performance on most images, while it fails in some cases, where there are similar regions with the stop sign in the image.

The rest of this paper is organized as follows. Section II formulates the stop sign detection and localization problem. Section III presents the technical approaches to solve the problem. The performance of the approaches are evaluated comprehensively in Section IV. Section V concludes the paper with a brief summary.

II. Problem Formulation

The problem of stop sign detection and localization is formulated as follows. Given the set of n images $\mathcal{I} = \{I_1, I_2, ..., I_n\}$, this project aims to detect the stop sign in each image. In addition, if the stop sign is detected on the image, the bounding box is generated to localize the stop sign. The bounding boxes for each image are represented by the set $\mathcal{B} = \{B_1, B_2, ..., B_n\}$, where B_i is the vector in the form $[x_1, y_1, x_2, y_2]^T$, (x_1, y_1) and (x_2, y_2) are the coordinate values of top left point and bottom right point of the rectangle, respectively, in x-y coordinate system.

III. Technical Approaches

This section presents technical approach to solve the formulated problem. In particular, the approach consists of two steps. The first step is to segment the image. And the second step is to detect and localize the stop sign in the segmented image.

A. Image Segmentaion

The first step is the image segmentation, which is achieved through logistic regression model. The model is represented by

$$p(y|\mathbf{x}, \omega) = \sigma(y\mathbf{x}^T\omega) \tag{1}$$

where $y \in \{-1, 1\}$ is the binary label, $x = \{v_B, v_G, v_R, 1\}$ is the input vector with v_B, v_G, v_R denoting the pixel values in BGR (Blue, Green, Red) order, respectively. In this case, the pixel values of stop sign region are labled as 1, while other regions are labled as -1, ω is the weight to be trained. $p(y|\mathbf{x}, \omega)$ represents the probability of y in the condition of $\mathbf{x}$ and w, σ is the sigmoid function represented by

$$\sigma(z) = \frac{1}{1 + exp(-z)} \tag{2}$$

The model is trained using maximum likelihood estimation (MLE). The procedures are described as follows. Given the data set $(X, \mathbf{y})$, where $X = [\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, ..., \mathbf{x_N}]^T$, $\mathbf{y} = [y_1, y_2, y_3, ..., y_N]$, N is the total number of data points. The data (X, y) is i.i.d., therefore the joing data likelihood is captured by

$$p(\mathbf{y}|X, \boldsymbol{\omega}) = \prod_{i=1}^{N} \sigma\left(y_i \mathbf{x}_i^\top \boldsymbol{\omega}\right) = \prod_{i=1}^{N} \frac{1}{1 + \exp\left(-y_i \mathbf{x}_i^\top \boldsymbol{\omega}\right)} \tag{3}$$

The estimation of ω using MLE is then represented by

$$\begin{aligned} \omega^* &= \arg\max_{\omega} \log p(\mathbf{y}|X, \omega) \\ &= \arg\min_{\omega} \sum_{i=1}^{N} \log\left(1 + \exp\left(-y_i \mathbf{x}_i^\top \boldsymbol{\omega}\right)\right) \end{aligned} \tag{4}$$

As the $\nabla_\omega(-\log p(\mathbf{y}|X, \omega)) = 0$ does not have a closed form solution. Therefore, the gradient descent algorithm is used here to update the weights ω. In particular, the ω is updated by

$$\begin{aligned} \boldsymbol{\omega}^{(t+1)} &= \boldsymbol{\omega}^{(t)} - \alpha \nabla_{\boldsymbol{\omega}}(-\log p(\mathbf{y}|X, \boldsymbol{\omega}))|_{\omega=\omega^{(t)}} \\ &= \boldsymbol{\omega}^{(t)} - \alpha \sum_{i=1}^{n} \frac{1}{1 + \exp\left(-y_i \mathbf{x}_i^\top \boldsymbol{\omega}^{(t)}\right)} \\ &\exp\left(-y_i \mathbf{x}_i^\top \boldsymbol{\omega}^{(t)}\right)(-y_i \mathbf{x}_i) \\ &= \boldsymbol{\omega}^{(t)} + \alpha \sum_{i=1}^{n} y_i \mathbf{x}_i \left(1 - \sigma\left(y_i \mathbf{x}_i^\top \boldsymbol{\omega}^{(t)}\right)\right) \end{aligned} \tag{5}$$

The above updating process stops until the $\omega^{(t+1)} \approx \omega^{(t)}$. After the logistic regression model is trained, It is used to predict the pixel values of an image. In particular, for each pixel with value $[v_B^*, v_G^*, v_R^*]$ in an image, the predict value is determined by

$$y_* = \begin{cases} 1 & \mathbf{x}_*^\top \boldsymbol{\omega}^* \geq 0 \\ -1 & \mathbf{x}_*^\top \boldsymbol{\omega}^* < 0 \end{cases} \tag{6}$$

where $x^* = [v_B^*, v_G^*, v_R^*, 1]$, ω^* is the weight of the logistic regression model after training. The two-dimensional array is then obtained with the element value either be 1 and -1. Then the binary segmented image can be generated by converting the -1 value to 0 value.

B. Localization

In this section, how to detect and localize the stop sign on the segmented image is described. The segmented image generated by the logistic regression model is supposed to identify the stop sign region. However, due to the inaccuracy of prediction of logistic regression model, other regions that are similar to stop sign may also be identified. Therefore, a criteria is needed to distinguish the stop sign region with other regions. In particular, some built-in $python - opencv$ functions are adopted here to analyze the properties of the identified region and compare with the geometry properties of the stop sign shape, i.e., octagon. Next, let me first introduce some important $opencv$ functions that are used for the localization.

(a)

Figure 1. Illustration of the shape of stop sign

1) Python-OpenCV functions:

a) *cv2.findContours* This function can find all the contours in a binary image.
b) *cv2.approxPolyDP* This function can approximate the contour using a polygonal curve and return an array of points that specify the vertex of the polygon. For instance, if the polygon is the rectangle, it will return 4 points. And if the polygon is triangle, it will return 3 points and so on.

2) Shape properties of stop sign: The Python-OpenCV functions, cv2.findContours and cv2.approxPolyDP can approximate the identified region with a polygon. As shown in the Figure 1, we can see that the stop sign is in octagon shape. The $cv2.approxPolyDP$ will get 8 points for the stop sign, which can be used as an criteria to identify the stop sign.

The whole algorithm of stop sign detection and localization is summarized in Algorithm 1.

IV. Results

In this section, the proposed stop sign detection and localization approaches are implemented on collected images. The performance is evaluated by using it to detect and localize stop sign in an image.

A. Data Preparation

The data set used in this project consists of nearly 200 images. Among these images, 100 images contain stop sign, while the rest of images do not. Two sample images are shown in Figure 2.

To create the training data set, the images are further labeled manually. In particular, for each image, the stop sign region is manually identified using Matlab $impoly$ function, which can draw a polygon on a specific region on an image. Then the polygon area is then converted into a binary image mask for further usage. In this work, only the stop sign region is labeled as 1 while all other regions are labeled as -1. The generated binary image mask of image shown in Figure 5(a) is shown in Figure 3.

Algorithm 1: Stop Sign Detection and Localization

Input: $\mathcal{I} = \{I_1, I_2, ..., I_n\}$
Output: $\mathcal{B} = \{B_1, B_2, ..., B_n\}$
// **Step 1:** Image Segmentation
for $i = 1 : n$ **do**
  for *each pixel* $x = \{v_B, v_G, v_R, 1\}$ *in* I_i **do**
    Using Equation 1 and 6 to get the predicted value
    Generating binary image $\hat{I}_i$ by converting -1 to 0
// **Step 2:** Detection and Localization
Create an empty list $\mathcal{B}$
for $i = 1 : n$ **do**
  Create an empty list B_i
  contours=cv2.findContours($\hat{I}_i$)
  for *each contour in conours* **do**
    polygon=cv2.approxPolyDP(contour)
    if(len(polygon)==8)
      B_i.append(polygon.bbox)
  $\mathcal{B}$.append(B_i)
Return $\mathcal{B} = \{B_1, B_2, ..., B_n\}$

(a)

(b)

Figure 2. Samples images a) with a stop sign b) without stop sign.

B. Model Training

The generated images and their associated binary image masks can be used for logistic model training. In particular, the model is trained as described in Section III-A. The hyperparameters are listed in the Table I.

Figure 3. Illustration of the binary image mask.

Table I. HYPERPARAMETERS FOR LOGISTIC REGRESSION MODEL TRAINING

Learning rate α	0.002
Number of images n	50
Number of epoch	100

C. Image Segmentation Testing

After training the model, the performance of logistic regression model is evaluated on a test image set consisting of 10 images.

The accuracy of the logistic regression model is calculated by

$$\frac{\sum_i^k \mathbb{1}_{\hat{y}_i = y_i}}{k} \times 100\% \tag{7}$$

where y_i is the labeled value of the pixel, $\hat{y}_i$ is the predicted value of the pixel, k is the total number of pixels. $\mathbb{1}$ is the indicator function.

The accuracy of the trained model is 76%. The prediction results of two typical sample test images are shown in Figure 4 and Figure 5. First look at the Figure 4. The test image contains one stop sign. Moreover, the stop sign is the only red item in the image, while other items such as trees, sky, house have color that are quite different from the red color. Therefore, the logistic regression model can accurately obtain the segmented image that only identifies the stop sign region (see Figure 4(b) and 4(c) as an illustration).

The Figure 5, as we can see, the sample test image also contains one stop sign, which is located in the right middle on the image. However, in the predicted segmented image, multiple regions are identified in addition to the stop sign region. This is because that in the image, there are multiple regions that are similar with the stop sign. For instance, the rear light of the car has same color with the stop sign. The model also identifies it as a stop sign region. Therefore only using the logistic regression model can not detect and localize the stop sign region in the image.

D. Localization Testing

Based on the segmented images, the accuracy of localization of stop sign is further tested. Same set of test images are used here. The accuracy is captured by

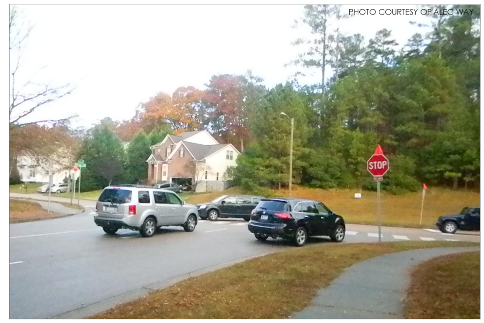
$$\frac{\sum_i^p \mathbb{1}_{|\hat{B}_i - B_i| <= \epsilon}}{p} \times 100\% \tag{8}$$

(a)

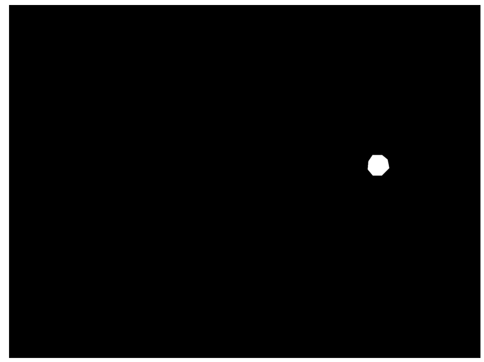
(b)

(c)

Figure 4. Samples images a) with a stop sign b) without stop sign.



(a)



(b)

(c)

Figure 5. Samples images a) with a stop sign b) without stop sign.

where p is the total number of bounding boxes, $\hat{B}_i$ is the bounding boxes obtained through localization, ϵ is the error tolerance, which means the predicted bounding boxes can be in a specific range of real bounding boxes.

The accuracy of the proposed localization method on the test image set is 62%. To further evaluate the approach, the localization results of the same test images used in Section IV-C are shown in Figure 6. As we can see, for the image where there are only one red item, i.e., stop sign, the proposed approach can effectively localize the stop sign. However, for the second image where there are multiple items with similar shapes and color with stop sign. The proposed localization method is not promising in distinguishing the stop sign region with from other regions. This is because that due to the inaccuracy of the logistic regression model, the predicted stop sign region in segmented image may not be a perfect octagon. Therefore, the approximated polygon may not contain 8 vertexes. Moreover, other regions may be approximated with polygons with 8 vertexes, which are then localized.

V. Conclusion

This project presents an approach to detect and localize stop sign in an image. In particular, the logistic regression model is first used to classify the pixel values of the image and generate the segmented image. Through analysis of the geometry properties and usage of built-in *opencv* functions, the stop sign regions in the segmented image can be identified. The testing results show that the approach achieves good performance when the image does not contain items that are similar to stop sign. These items generally have red color and polygon shapes such as rear lights of the car, red roof ans so on. For further improvements, two directions can be investigated, one is to improve the prediction accuracy of the logistic regression model or even try more advanced model. Another direction is to consider more geometry properties of the stop sign when distinguishing the stop sign

(a)

(b)

Figure 6. Samples images a) with a stop sign b) without stop sign.

region from other regions.

References

[1] S. Saini, S. Nikhil, K. R. Konda, H. S. Bharadwaj, and N. Ganeshan, "An efficient vision-based traffic light detection and state recognition for autonomous vehicles," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 606–611.

[2] L. Caltagirone, S. Scheidegger, L. Svensson, and M. Wahde, "Fast lidar-based road detection using fully convolutional neural networks," in *2017 ieee intelligent vehicles symposium (iv)*. IEEE, 2017, pp. 1019–1024.

[3] S. Jung, U. Lee, J. Jung, and D. H. Shim, "Real-time traffic sign recognition system with deep convolutional neural network," in *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE, 2016, pp. 31–34.

[4] J. Lu, H. Sibai, E. Fabry, and D. Forsyth, "No need to worry about adversarial examples in object detection in autonomous vehicles," *arXiv preprint arXiv:1707.03501*, 2017.

[5] S. Loussaief and A. Abdelkrim, "Machine learning framework for image classification," in *2016 7th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)*. IEEE, 2016, pp. 58–61.

[6] S. Dadras, H. Jamshidi, S. Dadras, and T. E. Pilutti, "Novel stop sign detection algorithm based on vehicle speed profile," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 3994–3999.

[7] N. Mittal and A. Kapoor, "Object detection and classification using yolo," 2019.