Color Segmentation

Yahsiu Hsieh
Department of Electrical & Computer Engineering
University of California, San Diego
y4hsieh@eng.ucsd.edu

***Abstract*—This paper presented approaches using three methods for color segmentation problem. In this work, I basically followed the idea of Bayes' theorem to segment the image. My goal is to use probability theory and supervised learning to classify the red color and compare the results.**

I. Introduction

Computer vision is a discipline science, and it includes picture recognition, identification, verification, and more. It is the fundamental component in AI problem. Various applications can be found in medical research, human robot interaction and self driving vehicle.

The most essential part of color segmentation problem is how to classify pixels. The challenge comes from the fact that same color in different pictures may have different rgb value due to lightning conditions. Moreover, colors in a picture may not be i.i.d, which adds difficulty in classifying pixels.

Here I presents three models for classifying colors, (1) Simple Gaussian, (2) Naive Bayes, and (3) Logistic regression. Simple Gaussian solution for this situation is try to model color distribution to Gaussian and find simple way to calculate the mean and covariance. Gaussian naive Bayes utilize the notion of prior and posterior, making the data distribution more like real world. Logistic regression try to recognize if a pixel is red or not. For stop sign region detection, I use OpenCV findContour and minAreaRect for the implementation.

The rest of paper is as follows. First we give the detailed formulations of color segmentation problem in Section II. Technical approaches are introduced in Section III. And at last we setup the experiment, results are presented in Section IV.

II. Problem Formulations

A. Color Segmentation Problem

The color segmentation problem is that: given an image $I \in \mathbb{R}^{h\times w\times 3}$ (h and w is the number of rows and columns for the image), try to classify the stop sign red pixel $\mathbb{P} = \{R, G, B\}$, and at last output a masked image $I_m \in \mathbb{R}^{h\times w\times 3}$, representing the stop sign red region.

B. Stop Sign Detection Problem

The stop sign detection problem is that: given an image $I \in \mathbb{R}^{h\times w\times 3}$ (h and w is the number of rows and columns for the image), try to find the region of stop sign (may or may not be in the image), and return a list of bounding boxes $\mathbb{B} = \{(x, y, x + a, y + b)\}$($x$, y, a, and b is the upper left coordinate, width and height of the bounding box, respectively), representing the stop sign region.

III. Technical Approaches

In this section we discuss about algorithms and models used in this project.

A. Simple Gaussian

Simple Gaussian uses two steps to classify the stop sign red pixel. In the model building step, we calculate the mean and covariance for six different color data sets (stop sign red, other red, orange, brown, blue, and others). We then build a 3D Gaussian model. In the prediction step, we compare the likelihood given by the constructed 3D Gaussian model to decide which of the six classes does the pixel belong to.

a) Model Building Phase

First of all we use labeled training data set to calculate mean μ (3-by-1 vector) and covariance Λ (3-by-3 matrix) for constructing 3D Gaussian model.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} X_i \tag{1}$$

$$\Lambda = \frac{1}{N-1}\sum_{i=1}^{N}(X_i - \mu)(X_i - \mu)^T \tag{2}$$

where X_i is a 3-by-1 vector to represent the pixel with RGB value of each color segment, N is the number of samples in that segment.

b) Prediction Phase

Here we identify the probability of the pixel x in each color segment using multivariate Gaussian equation.

$$p(x|class) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Lambda|^{\frac{1}{2}}} exp(-\frac{1}{2}(x-\mu)^T\Lambda^{-1}(x-\mu)) \tag{3}$$

where n is the input dimension.

And we will pick the class that has the maximum likelihood.

$$class = argmax_{class} p(x|class) \tag{4}$$

B. Naive Bayes

The Naive Bayes classifier takes class prior into consideration. We use likelihood obtained from simple Gaussian model and class prior to calculate posterior.

$$p(class|x) = \frac{p(x|class) \times p(class)}{p(x)} \tag{5}$$

Since $p(x)$ is a common denominator, we can compare the numerator. Therefore, we pick the class that has the maximum posterior.

$$class = argmax_{class} p(class|x) \tag{6}$$

C. Logistic Regression

Logistic regression uses two steps to classify the stop sign red pixel. In the training step, we use all images in training set to train the network. The model architecture is shown in Fig 2. In the prediction step, we input the pixel into the network and the network will decide if the pixel is red or not.

a) Training Phase

First of all we have our input data $x = \{R, G, B\}$ (3-by-1 vector), a weight matrix w (1-by-3 vector), and bias value b (1-by-1 vector). In Fig 2, the hidden node H_1 represents sum followed by a Sigmoid function. The output node O_1 is either 0 or 1, where 0 means the input pixel is not red. I set the learning rate to 0.001, and trained the model for 25 epochs. Here the loss function $J(\theta)$ I used is Cross-Entropy. Fig 1 shows the training loss over 25 epochs.

$$J(\theta) = \frac{1}{m}(-y^T log(h) - (1-y)^T log(1-h)) \tag{7}$$

where m is the total sample size, h is the value after the hidden node H_1, and y is the corresponding ground truth.

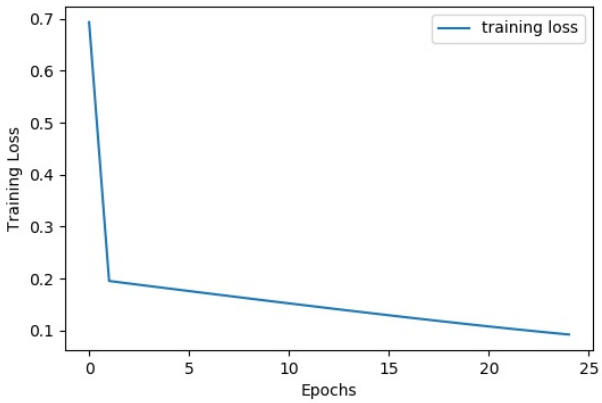


Fig. 1: Loss evaluation of Logistic Regression

b) Prediction Phase

In the prediction step, we input the pixel data x into our logistic model and obtain a value h after a Sigmoid function. We set a threshold $h^* = 0.5$, if $h > h^*$, then the current input pixel is red.

D. Stop Sign Detection

Stop sign detection is given a masked image, find one or multiple bounding box of a potential stop sign region. My main method is to utilize OpenCV and sci-kit image library to achieve the task.

I mainly apply *approxPolyDP* from OpenCV to detect if a region is octagon. Also I utilize sci-kit image functions, *regionprops* to get the graphical features of each detected region. By checking features such as *euler number*, *ratio of pixels in the region to pixels in the total bounding box*, and *width height ratio of the bounding box*, we can further remove irrelevant bounding boxes.

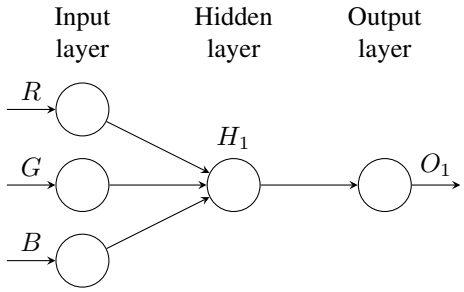


Fig. 2: Logistic Regression Model Architecture

Moreover, by utilizing a self-built function *mergeRect*, we can merge two bounding boxes into one in case the two bounding boxes belong to the same stop sign. See Fig 3 for the result.

(a) Bounding box before *mergeRect*.

(b) Bounding box after *mergeRect*.

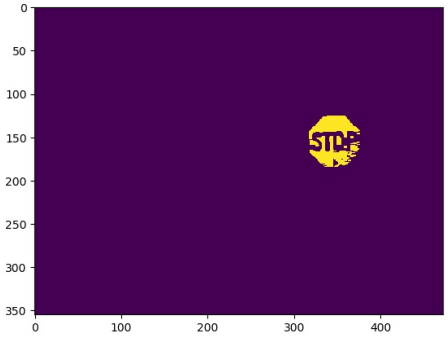
Fig. 3: Merging bounding boxes, 27.jpg

IV. Results

In this section I will present the segmentation and detection results of different models, and further compare their classification accuracy.

A. Segmentation and Detection

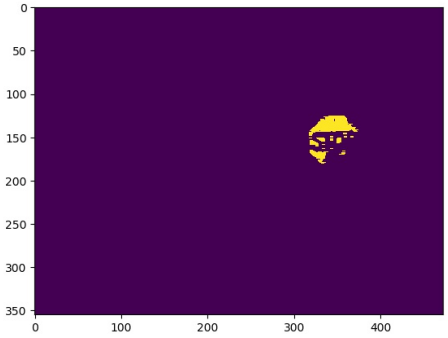
From the plots we can see that, Logistic Regression perform better than previous models on low light images. However, for images that are too bright, Logistic Regression seems to perform worse than Gaussian model.



(a) Masked image created by simple Gaussian model.

(b) Corresponding bounding box.

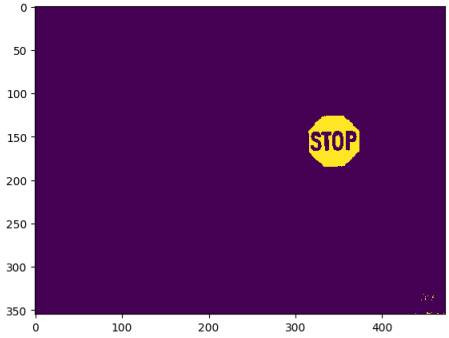
Fig. 4: Simple Gaussian Model, 27.jpg



(a) Masked image created by Naive Bayes Classifier.

(b) Corresponding bounding box.

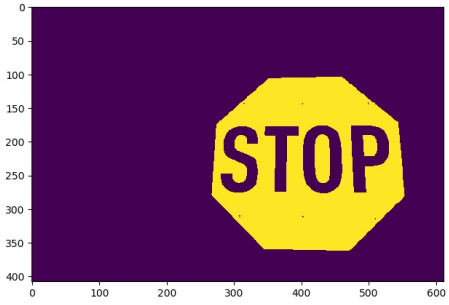
Fig. 5: Naive Bayes Classifier, 27.jpg



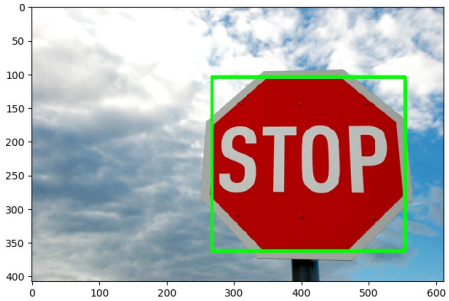
(a) Masked image created by Logistic Regression.

(b) Corresponding bounding box.

Fig. 6: Logistic Regression, 27.jpg

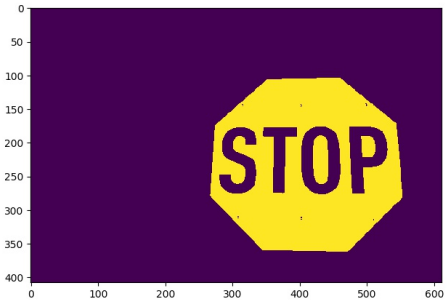


(a) Masked image created by Logistic Regression.

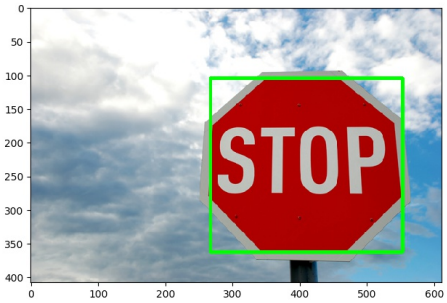


(b) Corresponding bounding box.

Fig. 7: Simple Gaussian Model, 29.jpg

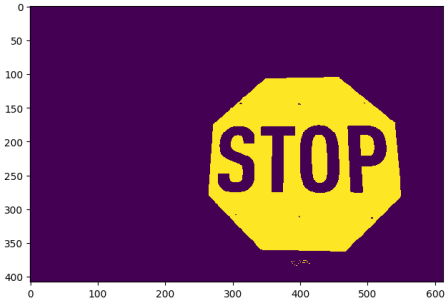


(a) Masked image created by Logistic Regression.

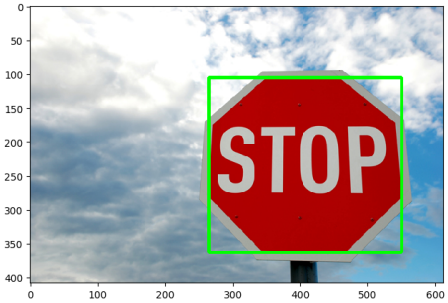


(b) Corresponding bounding box.

Fig. 8: Naive Bayes Classifier, 29.jpg

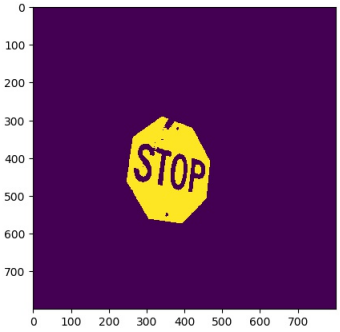


(a) Masked image created by Logistic Regression.



(b) Corresponding bounding box.

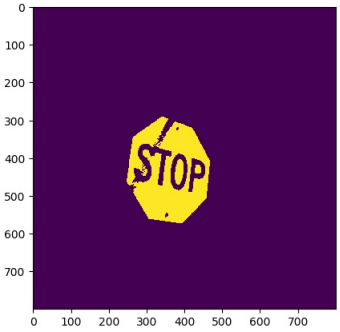
Fig. 9: Logistic Regression, 29.jpg



(a) Masked image created by Logistic Regression.

(b) Corresponding bounding box.

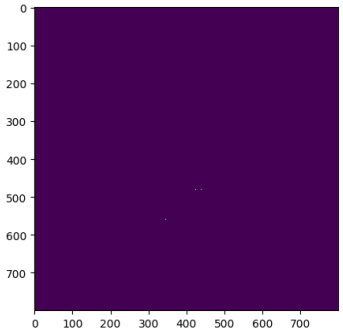
Fig. 10: Simple Gaussian Model, 94.jpg



(a) Masked image created by Logistic Regression.

(b) Corresponding bounding box.

Fig. 11: Naive Bayes Classifier, 94.jpg



(a) Masked image created by Logistic Regression.

(b) Corresponding bounding box.

Fig. 12: Logistic Regression, 94.jpg

B. Model Comparison

Here I compare the pixel classification accuracy and average detection time of different models. The average classification accuracy is calculated using equation 8, and the average detection time (per image) is calculated with total 100 images.

$$accuracy = \frac{\sum_{i=1}^{N} s_i}{\sum_{i=1}^{N} w_i \times h_i} \tag{8}$$

where w_i and h_i is the corresponding width and height of an image, is the number of total images, and s_i is the corresponding number of correctly classified as red or non-red pixel in an image.

TABLE I: Model Comparison

Model	Avg Detection time (sec)	Avg Accuracy(%)
Simple Gaussian	1.86	97.30
Naive Bayes	1.83	98.30
Logistic Regression	2.73	98.38

From Table I we can see that Logistic Regression model has the best classification accuracy out of three models. However, Gaussian models have faster average detection time. Overall, I would say all these three models perform quite well on this certain task.

C. Conclusion

To sum up, we implemented three models to classify pixels and detect the stop sign region. In the experiments we can see all three models' results. We presented fine results on images with normal light condition. However, we have a hard time detecting some low light images. Therefore, a future work is to try to solve low light images problem. A suggestive way is to perform histogram equalization or perform classification on color space other than RGB.

ACKNOWLEDGEMENTS

The hand-labeling work was in collaboration with Yunhsiu Wu, David Lu, Minhsueh Cheng, Yu-Tsun Yang, Sheng-Wei Chang, Chun-Yen Liou, Jui-Te Lin, Chun-Nien Chan, and Yu-Hao Liu.