Use Gaussian Mixture Model to Detect Stop Sign

Bolin He
Department of Electrical Computer Engineering
University of California, San Diego
b2he@ucsd.edu

I. INTRODUCTION

Images are one of the most significant sources we store information and essential components in computer vision. In the recent years, with the development of convolution networks(CNN) theory and improvement of computation capacity, researchers have made great progresses in the field of computer vision. However, the old-school statistics models still flourish. In this report, I will propose a Gaussian Mixture Model(GMM) to detect the stop signs in images. I first classified the pixels into four different classes, and then labelled them into different regions. I analyzed the region eccentricity, region area and region area ratio to determine whether the region was a stop signed or not. Besides, I also explored the results among different classes classification and region classified criteria.

II. PROBLEM FORMULATION

A Gaussian Mixture Model(GMM) is a distribution assembled from weighted multivariate Gaussian distributions. Weighting factors assign each distribution different levels of importance. The GMM has the formula as

$$P_{X|Y}(x \mid i) = \frac{1}{\sqrt{(2\pi)^d \mid \Sigma_i \mid}} e^{-\frac{1}{2}(x-u_i)^T \Sigma_i^{-1}(x-u_i)}$$

where i is the class, d is the dimension, μ is the mean of the i class, Σ is the covariant of the i class.

To apply Maximum Likelihood Estimation(MLE), we can derive

$$u = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - u)(x^{(i)} - u)^T$$

In this case, I can utilize the MLE of GMM to calculate the weights of our classes. With the weight $w_{MLE} = (u_i, \Sigma_i)$, I can define our model$P_{X|Y}(x \mid i)$ for each class individually.

For each pixel in image, I can calculate the different class probabilities of the this single pixel, and later find the the maximum among them. I classify this pixel as stop sign if and only if

$$P_{X|red} = Max(P_{X|red}, P_{X|fake-red}, P_{X|yellow}, P_{X|black})$$

otherwise, this pixel will be classified as non stop sign.

In this experiment, I create four classes, which means we should have a four-dimension GMM. However, to simplify the computation and increase the accuracy, I define the three-dimension GMM for each class. In this case, the mean μ is a three by one vector, and the covariant Σ is a three by three matrix for each class.

III. TECHNICAL APPROACH

III-A. Step one: Extract information from training data.

At the very first beginning, I read the image into python IDE and convert its color space from BGR to YCRCB. I utilize the python tool roipoly to draw the contour on image, which can generate a binary mask image, with 1 inside the contour and 0 outside the contour. With these information from this mask image, I can locate and extract information from the pixel with the same position in original image. Through couple times manipulation, finally I can obtain information of four classes, which are red, fake_red, yellow and black. These data are N by three matrix, where N is the number of total pixel and three is the channel of YCRCB color space. At the end, I save the data into pkl files.

III-B. Step two: Calculate the mean μ and covariant Σ for each class

As the definition of mean μ and covariant Σ above, I load the data from pkl file and calculate them for each class individually. Different classes have different μ and Σ, which indicates that they have different distribution property.

Furthermore, I define probability model $P_{X|Y}(x \mid i)$ for future implement.

III-C. Step three: Generate mask image on validation set

I reshape the validation set image from three dimensional array into N by three matrix, where N is the total number of pixels. with feeding the data into $P_{X|Y}(x \mid i)$, I can derive a probability matrix

$$P = [P_{X|red}, P_{X|fake-red}, P_{X|yellow}, P_{X|black}]$$

for each pixel. As I mentioned above, each pixel will be classified as stop sign if and only if

$$P_{X|red} = Max(P).$$

If this pixel belongs to class red, I set it as 1, and 0 otherwise. After all the pixels are classified, I try to reshape this new N by one matrix into a binary image. This binary image is the segmented mask image, with 1 as stop sign area and 0 as background.

Obviously, this mask image will have a stop sign with fuzzy boundary, also contain some noise, the separated distributed pixel classified as stop sign but actually not. The accuracy of the weight of GMM can account for these noise. Moreover, some pixels do have similar property as the pixel within stop sign, that is why they are also classified into class red.

III-D. Step four: Find bounding box on validation set

Generally, the mask image will have couple regions. Some of them are real stop signs, while some of them are random noise or even irrelevant regions, such as other sign, red car, ornament and so forth.

In this step, I set the criteria to eliminate the interferential elements.

Inspired by the train set, all the stop signs are hexagons, which indicates that the region eccentricity, region area ratio in the bounding box and the whole image are in specific range. These criteria on region property will help me to find the real stop sign. Moreover, since the region area should have a number of pixels, I also set a criterion on it, to eliminate the noise.

I classify the region belongs to stop sign, if it meets all the following criteria:

- the region area to the bounding box ratio is between 0.3 to 0.8
- the region eccentricity is between 0.3 to 0.7
- the region area to the whole image ratio is larger than 0.003
- the region area has at least 45 pixels

Finally, I can find the real stop sign and obtain the bounding box information by using python tool regionprops on the classified region.

IV. RESULT

IV-A. Result one: Training results

The mean μ and covariant Σ for each class:

μ_red = [[96.06214209877389], [195.989889718473 88], [110.16503442016577]]
μ_yellow = [[153.4965087529189], [164.0425092718365], [76.24108873490943]]
μ_black = [[39.909135739624 26], [125.14013012423842], [131.4960628668692]]
μ_fake_red = [[140.30234245338], [122.3181868915784], [121.58589481040]]

Σ_red = [[3232.40911135, -11132.20292696, -186.67559589],
[-11132.20292696, 1360.52123188, -7647.95013781],
[-186.67559589, -7647.95013781, 150.38283229]]

$\Sigma\_yellow = [[\ 979.68959055,\ -112.29872421, -6447.34687934],$
$[\ -112.29872421,\ \ 72.47298057, -7755.79175697],$
$[-6447.34687934, -7755.79175697,\ \ 510.43182111]]$

$\Sigma\_black = [[\ 531.69608004, -7263.92179451, -8398.71807693],$
$[-7263.92179451,\ \ 18.09386583,\ \ -49.49034095],$
$[-8398.71807693,\ \ -49.49034095,\ \ 17.77832038]]$

$\Sigma\_fake\_red = [[\ 3074.48732956,\ \ 391.05814379,\ -734.2280698\ ],$
$[\ 391.05814379,\ 1328.30199029, -1002.63908271],$
$[\ -734.2280698\ , -1002.63908271,\ 1036.21677167]]$

IV-B. Result Two: Successful cases

In my validation set, most of them have great performance. Generally speaking, these cases have obvious stop sign and contrast background. It makes sense, because even for the human beings, these stop sign are more detectable.

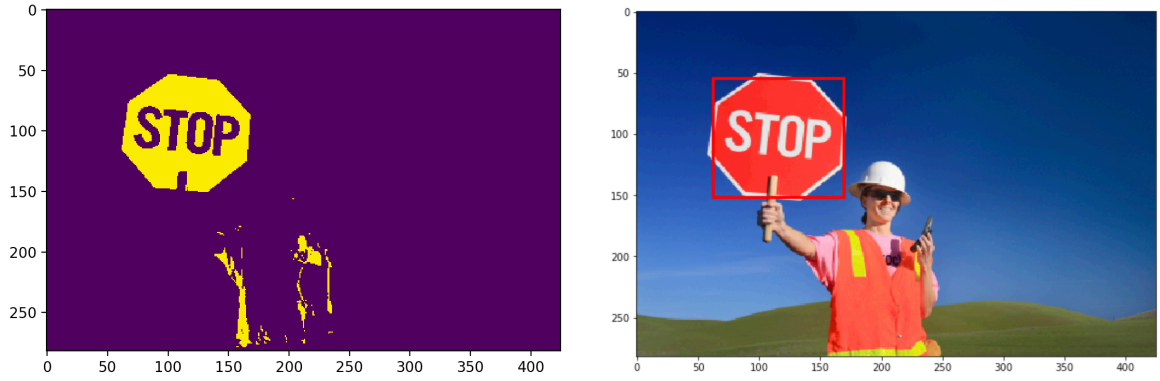

Fig. 1 Successful cases with fake red interference

Fig. 2 Successful cases with contrast background

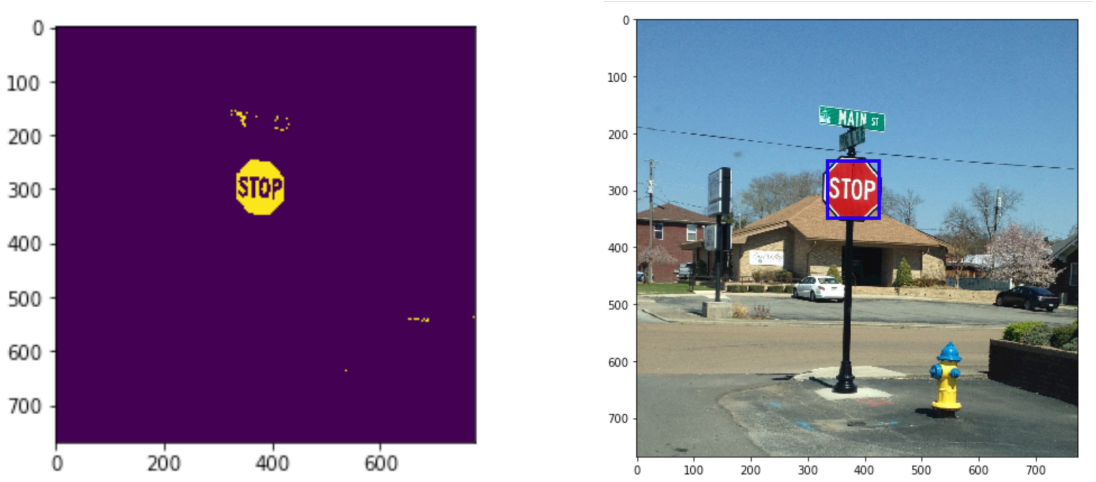

Fig. 3 Successful cases with noisy background

IV-C. Result Three: Failure cases

Some cases in validation set fail in test. These images usually have small stop signs or in the environment with abnormal light, such as too dark or too bright.

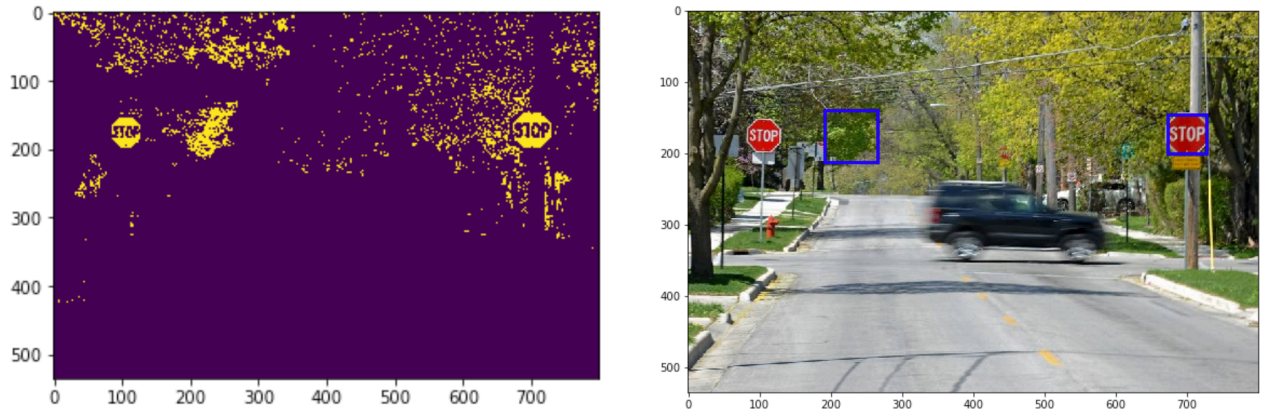

Fig. 3 Failure cases with small stop sign

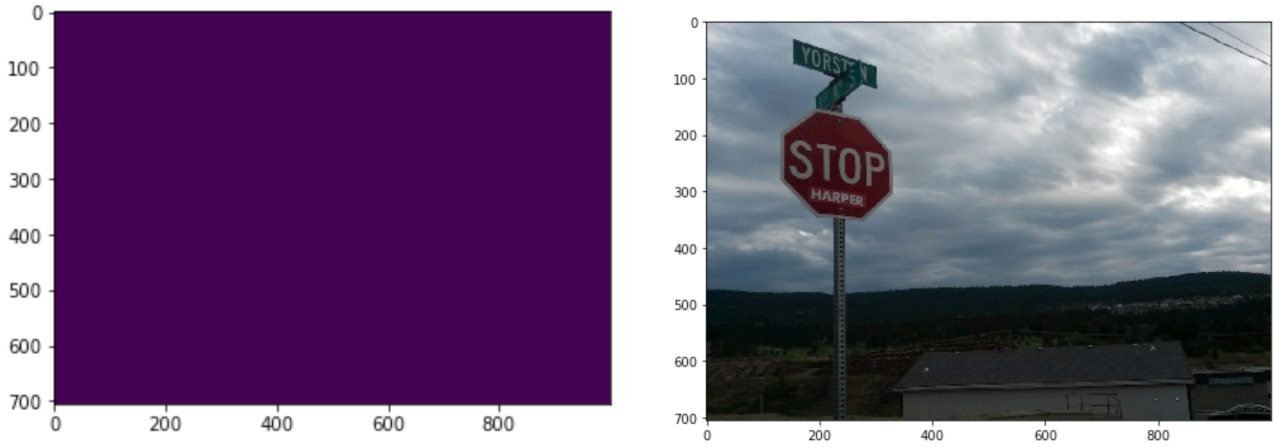

Fig. 4 Failure cases with dark light environment

IV-D. Discussion

In my experiment, I have explored different combination methods and elements to achieve a better performance. I discover several aspects below:

a. At the very first beginning, actually I tried to divide the pixels into eight classes. Besides red, fake red, black and yellow which I finally used in the experiment, I also created classes white, grey, blue and green. One interesting thing was, once I used either class green or class grey, I could not get any result. In this case, this GMM would always achieve mask image with all 0. Another interesting thing was, class blue and class white could not affect the classification result, which meant that these two classes are redundant.

b. Except these classes, I also tried to create a class named random. It is obvious that this class would contain random pixels. From my point of view, because I only want the red pixel in stop sign, I can shuffle all other pixels into one class. However, I achieved bad result through this way. To account for this, I suspected that due to the different mean and variance of different color, mixing them up would generate a class represents nothing. In this way, the class random had unrepresentative mean and variance, which increased the uncertainty of experiment. To a great extent, the result depended on how this class was created and the result was largely unpredictable.

c. Different resize methods will influence the result. At the beginning, I used two *for loop* to classify the pixels, which resulted in low processing speed. I tried to resize the image, and achieved different result with different resize ratio. Some of them was trivial, while some of them generated a huge gap among results. Later, I reshaped the image into matrix to classified the pixels and then reshaped them back as image. Without using two *for loop*, this approach increased the speed significantly.

d. The region criteria can be improved. Date back to the fig.3 above, it did detect both the left and right stop signs. However, it recognized the irrelevant area as stop sign but ignored the real stop sign on the left. I think modifying the criteria or using other discriminative approach would be helpful.