ECE 276A Project 1: Color Segmentation

Albert Tan
Department of Electrical and Computer Engineering
University of California, San Diego
aktan@eng.ucsd.edu

***Abstract*— This report presents an approach to detecting stop signs in an image using color segmentation. Gaussian Mixture Models were used to segment the image into regions of red pixels and non-red pixels. Out of all the red pixels, the red regions of highest likelihood were then bounded as stop-signs**

I. Introduction (*Heading 1*)

Color segmentation is used to partition a digital image into multiple segments, and has become an increasingly important task in the field of Computer Vision and Autonomous Vehicles. Segmentation allows regions to be more easily analyzed for patterns and defining features. In this project, the red pixels of images were segmented from all other colors using a Gaussian Mixture Model. The result was a partitioned image into regions of red pixels and non-red pixels based on those pixels that met a likelihood threshold. Bounding boxes of particular sizes were then used to locate stop signs out of the red segments.

II. Problem Formulation

The task of color segmentation can be reframed in this particular project as creating a binary mask from any arbitrary image. The conditions for whether a pixel is a '1' in the mask versus a '0' was determined through the use of Gaussian Mixture Models, which attempts to fit multiple gaussians and find the parameters that best maximize the probability of the data.

We know that given a test image, the probability of a particular pixel is given by:

$$\log \mathrm{P_X(x)} = \log \sum_{\mathrm{k=1}}^{7} \alpha_{\mathrm{k}}\, \mathcal{G}(\mathrm{x}, \mu_{\mathrm{c}}, \Sigma_{\mathrm{c}})$$

The problem is then to find the parameters α, μ, and Σ that satisfy the following optimization problem:

$$\mathrm{Q}(\theta, \theta^0) = \mathrm{E_{Z|X}}[\log \mathrm{P}(\mathrm{X}, \mathrm{Z}|\theta)]$$

$$\hat{\theta} = \max_{\theta} \operatorname{argmax}\, \mathrm{Q}(\theta, \theta^0$$

III. Technical Approach

a. *Color Segmentation*

To segment the image into red and non-red regions, a Gaussian Mixture model of 7 components was used. Once I gathered the training set of red pixels extracted through roiPoly, I trained the GMM off of it and obtained the optimal parameters through the following update equations:

Start with initial guess $\boldsymbol{\omega}^{(t)} := \left\{\alpha_{kj}^{(t)}, \boldsymbol{\mu}_{kj}^{(t)}, \Sigma_{kj}^{(t)}\right\}$ for $t = 0$, $k = 1, \ldots, K$, $j = 1, \ldots, J$ and iterate:

(E step)
$$r_k^{(t)}(j \mid \mathbf{x}_i) = \frac{\alpha_{kj}^{(t)} \phi\left(\mathbf{x}_i; \boldsymbol{\mu}_{kj}^{(t)}, \Sigma_{kj}^{(t)}\right)}{\sum_{l=1}^{J} \alpha_{kl}^{(t)} \phi\left(\mathbf{x}_i; \boldsymbol{\mu}_{kl}^{(t)}, \Sigma_{kl}^{(t)}\right)}$$

(M step)
$$\alpha_{kj}^{(t+1)} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i)}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\}}$$

$$\boldsymbol{\mu}_{kj}^{(t+1)} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i) \mathbf{x}_i}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i)}$$

$$\Sigma_{kj}^{(t+1)} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i) \left(\mathbf{x}_i - \mu_{kj}^{(t+1)}\right) \left(\mathbf{x}_i - \mu_{kj}^{(t+1)}\right)^T}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i)}$$

The resultant weights (α) were:

```
[0.15644498 0.01170396 0.25656512 0.04695649 0.03157741 0.32426151
 0.17249054]
```

The resultant means (μ) were:

```
[[ 26  15 159]
 [ 26  15 159]
 [ 26  15 159]
 [ 26  15 159]
 [ 26  15 159]
 [ 26  15 159]
 [ 26  15 159]]
```

The resultant covariances (Σ) were:

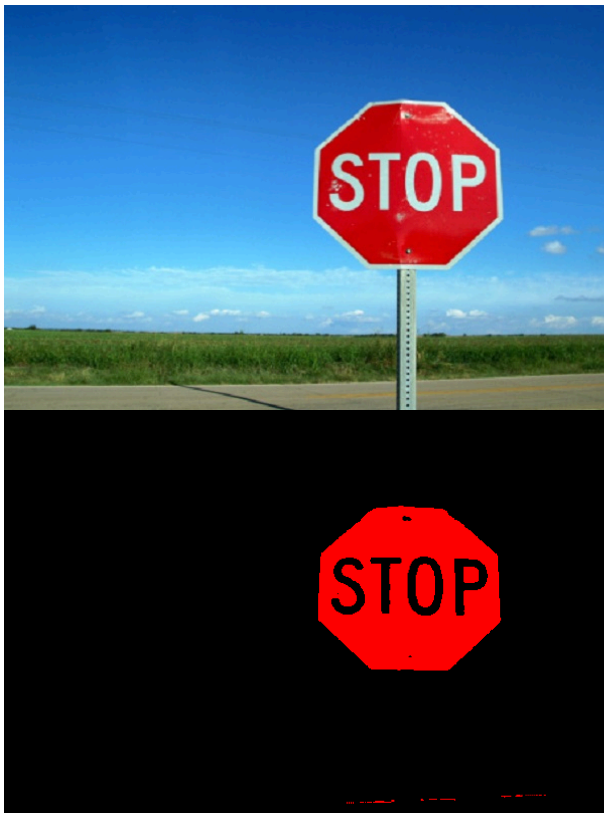
```
[[[28559.37891721 26639.56174963 17575.21677028]
  [26639.56174963 37249.67079203 20975.75854094]
  [17575.21677028 20975.75854094 21364.60047526]]

 [[28559.37891747 26639.56175006 17575.21677038]
  [26639.56175006 37249.67079162 20975.75854103]
  [17575.21677038 20975.75854103 21364.60047542]]

 [[28559.37891675 26639.56174938 17575.21676993]
  [26639.56174938 37249.67079068 20975.75854049]
  [17575.21676993 20975.75854049 21364.60047488]]

 [[28559.37894535 26639.56176616 17575.21678001]
  [26639.56176616 37249.67080091 20975.75854078]
  [17575.21678001 20975.75854078 21364.6004806 ]]

 [[28559.37891697 26639.56174923 17575.21677022]
  [26639.56174923 37249.67079251 20975.75854088]
  [17575.21677022 20975.75854088 21364.60047513]]

 [[28559.37891731 26639.56174972 17575.21677034]
  [26639.56174972 37249.67079215 20975.758541  ]
  [17575.21677034 20975.758541   21364.60047533]]

 [[28559.37891649 26639.56174878 17575.21676992]
  [26639.56174878 37249.67079185 20975.75854052]
  [17575.21676992 20975.75854052 21364.60047476]]]
```

As you can see, the GMM produced nearly identical means and covariances across all 7 of its components, though the weights were different.

Once all the parameters were obtained, the log-likelihood for each equation was determined through the GMM equation:

$$\log P_X(x) = \log \sum_{k=1}^{7} \alpha_k \mathcal{G}(x, \mu_c, \Sigma_c).$$

From there, I set the segmentation mask so that that all the pixels whose likelihood exceeded a fraction of the highest likelihood in the image were set as 1, and 0 otherwise.



The above pair of images show the original image and the resultant mask segmentation

b. Bounding Box

Once the images were segmented, the next task was to bound each of the red pixels. I used the skimage.draw and skimage.measure to draw and compute the optimal bounding boxes. Those boxes were determined through through 2 main criteria.

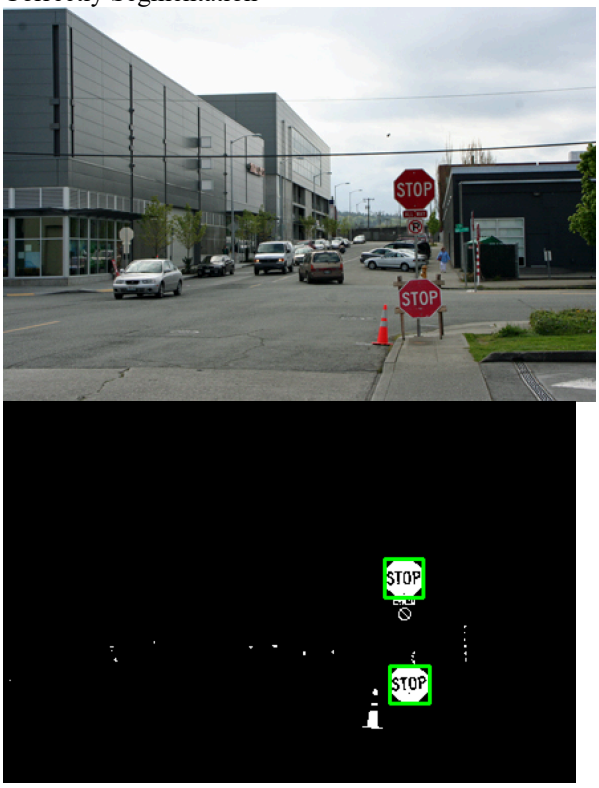
The first criteria was to ensure that the ratio of the bounding box area to the entire image area was greater than a certain threshold. This ensured that whatever was bounded was large enough and increased the probability that the region was indeed a stop sign, and not smaller patches of noise.

The second criteria was that the bounding box height to width ratio also needed to exceed a certain threshold. There were many instances where long streaks of red were being bounded. Thus, to eliminate these false positives and increase the chances of the region being a stop sign, I set the threshold to be most like a square, which resembles an octagon.

IV. Results

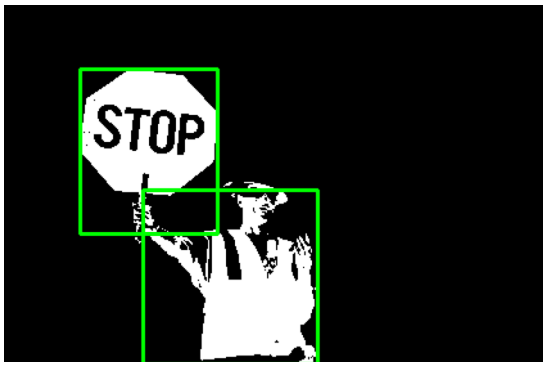
The model ended up segmenting 9/15 test images correctly. Below were some results

a. Correctly Segmentation



In the above pairs, the two stop lights were correctly identified.

b. Incorrect Segmentations



In the above pairs, the output produced a false positive. That is, it bounded the red vest although it wasn't a stop sign. This was likely because my conditions weren't strict enough, or that I was missing a condition that would eliminate these false positives.

This last pair was an example of a false negative. Here, there were 3 stop signs in the image but the model failed to detect a single sign. This was because the stop sign text filled up too much of the red sign, causing the area of the sign to decrease and the overall bounding box to overall area ratio to not be high enough.

ACKNOWLEDGEMENT

The author of this report acknowledges the following collaborators:

Austin Choe (A92091213)