Color Segmentation

Saurabh Himmatlal Mirani (A53319557)
Sensing and Estimation in Robotics
University of California, San Deigo
smirani@ucsd.edu

***Abstract*—Automatic traffic sign detection and recognition is a computer vision problem and is very important for autonomous driving. This paper proposes a framework that will segment the image into two two parts, red and not red using Gaussian Discriminative Model and then detect if the red part has any stops signs based on contours. If present, a bounding box will be created around the sign and the co-ordinates will be returned. The experimental results show the detection rate is more than 95% and the accuracy of recognition is more than 85%**

***Index Terms*—STOP Sign; Colour Feature; Autonomous Driving; Detection and Recognition; Segmentation; Gaussian Discriminative Model**

I. Introduction

Automatic traffic sign detection and recognition is a computer vision problem and is very important for autonomous driving. The traffic signs provide a lot of information about the rules that need to obeyed, for example, speed limit, no entry, prohibit or permit certain actions like STOP sign etc. This helps in autonomous navigation by correctly identifying the road-signs [1]

The traffic signs have many unique features that can be used for their identification. Colours and shapes are the most important features that help in identification. Moreover, the colour used are almost standard in most countries which mostly include primary colors i.e. red, green, blue. Therefore, this can be used to identify traffic signs.

In this paper, we restrict our goal to identification of only STOP signs. STOP signs have a unique characteristic which is its octagonal shape. We will exploit this characteristic to distinguish STOP signs from other red objects in our image. The detection of STOP signs is achieved by three main modules: pre-processing, segmentation, and recognition. In the pre-processing module, the input images are pre-processed to remove the noise and enhance the image which would aid in better recognition. In the segmentation module, a binary mask is created using Gaussian Discriminative Model (GDM) with 1s representing red colour in the image and 0 representing not-red in the image. In the recognition phase, an octagon is tried to fit in the segmented mask. If it fits, then there is a high probability that it is a STOP sign.

The rest of the paper is organized as follows, Section II describes basic properties of STOP sign, Section III describes the Problem Formulation, Section IV describes the Technical Approach, experimental results are reported in section V, Section VI concludes the paper.

II. Properties of STOP sign

STOP signs are characterized by unique features that make a difference from the other objects.

- They have 2D octagonal shape.
- STOP signs are always red in colour
- The colors of the characters written on the sign has other color, i.e. white

(a) Compliant Sign (b) Non-Compliant Sign

Fig. 1. Different Stop signs

III. Problem Formulation

A. Color Segmentation Problem

Color segmentation problem [2] requires segmenting a 3D color space into a set of discrete volumes associated with semantic colors.

Each pixel is a 3D vector:

$$\mathbf{x} = (B, G, R)$$

where B, G & R represent Blue, Green and Red color channels respectively. Discrete color labels $y \in \{0, 1\}$, where

$$y = \begin{cases} 1, & \text{if red} \\ 0, & \text{if not red} \end{cases}$$

A probabilistic model $p(y, \mathbf{x})$ is used for the color class y of a pixel x

B. Training

Given a dataset $D := \{\mathbf{x}_i, y_i\}_{i=1}^{n}$ of **iid** examples $\mathbf{x}_i \in \mathbb{R}^3$ with associated scalar labels y_i generated from an *unknown* joint pdf $p_*(y, \mathbf{x})$

The training dataset can also be written in matrix notation, $D = (X, \mathbf{y})$, with $X \in \mathbb{R}^{nxd}$ and $\mathbf{y} \in \mathbb{R}^n$

Goal: Define a function: $h : \mathbb{R}^3 \to \mathbb{R}$ that can assign a label y to a given data point $\mathbf{x}$, either from the training dataset D or from an unseen test set generated from the *same* unknown pdf $p_*(y, \mathbf{x})$

For generative model,

$$h(x) := \arg\max_y p(y, \mathbf{x})$$

where $p(y, \mathbf{x})$ is chosen such that it approximates the unknown data-generating pdf and can generate new examples of $\mathbf{x}$ with associated labels y by sampling from $p(y, \mathbf{x})$

Parametric Learning: Represent the pdfs $p(y, \mathbf{x}; \omega)$ using parameters ω

Estimate/optimize/learn ω based on the training set $D = (X, \mathbf{y})$ in a way that ω^* produces good results on a test set $D_* = (X_*, \mathbf{y}_*)$

Maximum Likelihood Estimation (MLE): is the most commonly used strategy. It maximizes the likelihood of the data D given the parameters ω

$$\omega_{MLE} = \arg\max_\omega p(\mathbf{y}, X|\omega)$$

C. Testing

The optimized model $p(y, x)$ is used to define a function that transforms a new color-space input $\mathbf{x}_*$ into a discrete color label y_*:

$$y_* \in \arg\max_y p(y, \mathbf{x}_*|\omega_{MLE})$$

IV. Technical Approach

The main steps of the proposed approach are illustrated in Fig. 2. The algorithm has three main stages: 1) Pre-processing 2) Color Segmentation 3) Recognition. The detailed design flow in Fig. 5. The system takes static color images as input. The output is the coordinates of the bounding box of the STOP sign.

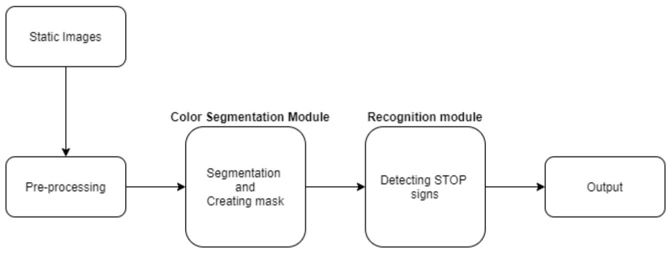
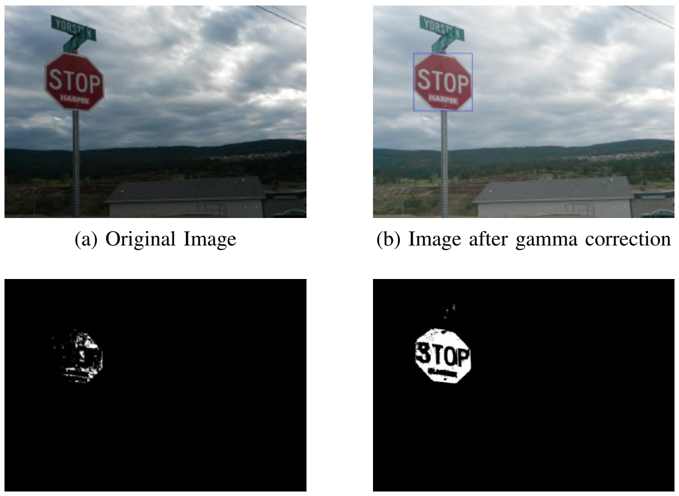


Fig. 2. Stages of the Proposed Method

In the Color Segmentation module, a mask is created using Gaussian Discriminative Model (GDM) and made ready for the recognition stage. The recognition module tries to fit contours in the mask and hence carries out the recognition job.

A. Pre-processing

This step is used to increase the accuracy of the classification step. Gamma correction is used to improve the results. When twice the number of photons hit the sensor of a digital camera, it receives twice the signal (a linear relationship). However, that's not how human eyes work. Instead, double the amount of light is perceived as only a fraction brighter (a non-linear relationship). Furthermore, eyes are also much more sensitive to changes in dark tones than brighter tones (another non-linear relationship). In order to account for this gamma correction is applied, a translation between the sensitivity of eyes and sensors of a camera. Benefits of gamma correction can be clearly seen in Fig. 3 and Fig. 4



(a) Original Image (b) Image after gamma correction

(c) Mask of original image (d) Mask of image after gamma correction

Fig. 3. Advantages of gamma correction

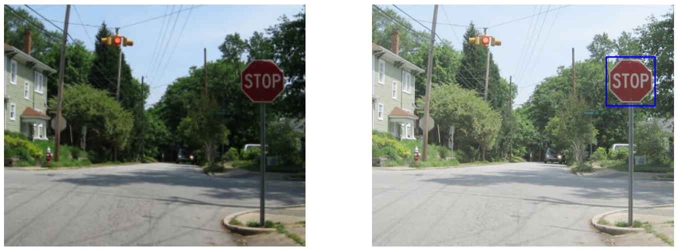
B. Color Segmentation

The main task in this module is to generate a binary mask from the images and to make them ready for the recognition stage. Consider a generative mode $p(y, \mathbf{x}|\omega)$ for discrete labels $y \in \{0, 1\}$ where,

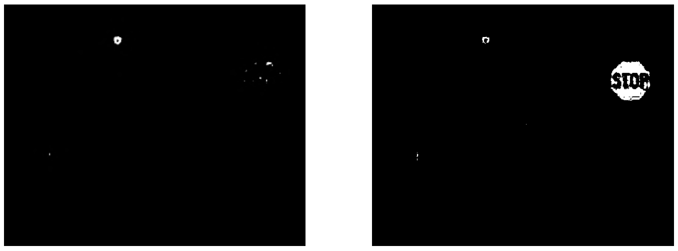
$$y = \begin{cases} 1, & \text{if red} \\ 0, & \text{if not red} \end{cases}$$

Different steps of this module are shown in Fig. 5 and discussed in brief.

1) *Conversion from BGR to 2D array*: BGR (Blue, Green, Red) is the primary color space commonly used in OpenCV [3]. However, BGR is a 3D array and results in slower computations. In order to make the computations faster we convert the input image into 2D array where each row represents a pixel and the columns represent the Blue, Green and Red values respectively. So, if we have an image of dimensions n x m, our converted matrix will be of dimensions n*m x 3



(a) Original Image (b) Image after gamma correction



(c) Mask of original image (d) Mask of image after gamma correction

Fig. 4. Advantages of gamma correction

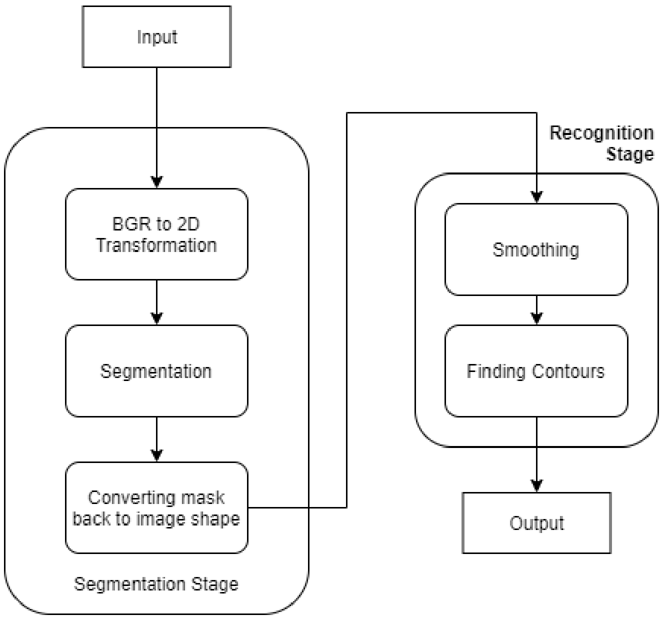


Fig. 5. Design Flow of the Proposed Method

2) *Segmentation*: In this step we classify each pixel into two classes viz. red and not red. For this classification Gaussian Discriminative Model (GDM) is used.
Gaussian Discriminative Model (GDM):

$$p(\mathbf{y}, X|\omega, \theta) = p(\mathbf{y}|\theta)p(X|\mathbf{y}, \omega) = \prod_{i=1}^{n} p(y_i|\theta)p(x_i|y_i, \omega)$$

$$p(y_i|\theta) := \prod_{k=0}^{1} \theta_k^{\mathbb{1}\{y_i=k\}} \;\&\; p(\mathbf{x}_i|y_i = k, \omega) = \phi(\mathbf{x}_i; \mu_k, \Sigma_k)$$

where, $\omega := \{\mu_k, \Sigma_k\}$ and MLE estimates of θ and ω are obtained from:

$$\theta_k^{MLE} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} \quad (1)$$

$$\mu_k^{MLE} = \frac{\sum_{i=1}^{n} \mathbf{x}_i \mathbb{1}\{y_i = k\}}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\}} \quad (2)$$

$$\Sigma_k^{MLE} = \frac{\sum_{i=1}^{n}(\mathbf{x}_i - \mu_k^{MLE})(\mathbf{x}_i - \mu_k^{MLE})^T \mathbb{1}\{y_i = k\}}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\}} \quad (3)$$

In order to identify to which class a particular pixel belongs, Bayes' Decision Rule (BDR) is used.
Let y_i represent the class of the i^{th} data point or pixel, i.e. if $y_i = 1$ then it is red and if $y_i = 0$ it is not red. The Bayes' Decision rule is thus; i^{th} pixel is red if:

$$p(x_i|y_i = 1, \omega) * p(y_i = 1|\theta) >$$

$$p(x_i|y_i = 0, \omega) * p(y_i = 0|\theta)$$

where $\omega := \{\mu_k, \Sigma_k\}$

3) *Converting back to image shape*: Since the created mask is now of dimension n*m x 1, it is converted to n x m, i.e. the original shape of the image.

C. Recognition

The recognition stage takes the output mask from the previous color segmentation stage as its input. Different steps of this module are shown in Fig. 5 and discussed in brief.

1) *Smoothing*: Gaussian blur is used for image smoothing. It is the result of blurring an image using a Gaussian function. It helps in reducing the noise and reduce the detail. Reducing detail is necessary since too much details may hinder in finding contours. Gaussian blur is chosen over averaging blur as it provides better results experimentally.
2) *Finding Contours*: OpenCV's findContour function is used here. Certain criteria are used to determine how close the contour is to an octagon;
   - Number of edges: Ideally an octagon has 8 edges. However this may not be the case always during recognition. If the STOP sign is too far in the image, it can look like more a circle and hence can have more than 8 edges. Considering another case where a STOP sign is tilted it may have less than 8 edges. Hence, instead of considering exactly 8 edges, we consider a range of edges from 6 to 13, using which a similarity score can be given as to how close it is to an octagon.
   - Height/Width ratio: Ideally an octagon has $h/w = 1$. Again due to STOP sign being tilted or being far away can result in non-ideal h/w ratios. Hence, a range of 0.9 to 1.3 is selected (based on experiments on training set)

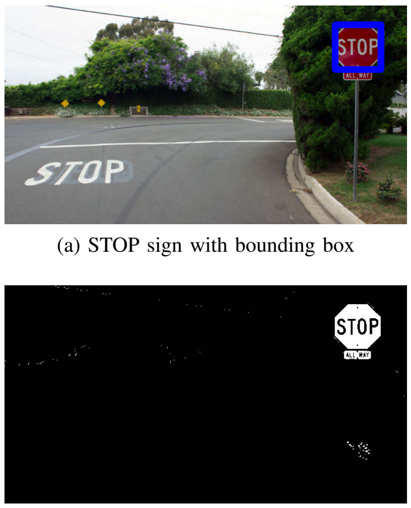
- Contour Area: This is an important factor to remove unwanted contours. Since any segmented image is bound to have noise even after smoothing, there is a high probability that very small contours may fulfil all the above criteria. Hence, we use ratio of contour area and original image size to determine if the contour is too small to be a STOP sign. There is a slight chance that an actual STOP size may be excluded due to this criteria. Keeping a threshold of 0.002 (found experimentally) ensures that almost always only noise is excluded. This is reasonable for all practical purposes since a STOP sign smaller than this is too far in the image and might not pose any problem for autonomous driving.

All the above criteria seem to work well with high resolution images. However problem arises with low resolution images. Hence, we apply all the above criteria only for high resolution images. For low resolution images the criteria is relaxed a bit. This is necessary because of image being of low resolution, fitting approximate polygons is difficult and may result in non ideal cases.

3) *Bounding Box*:At the end, a bounding box of the contour is created using OpenCV's boundingRectangle function.

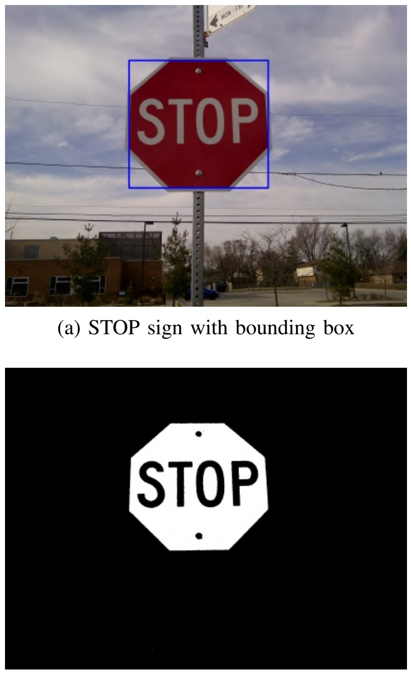
V. Results and Discussion

A. *Successful Tests*


(a) STOP sign with bounding box

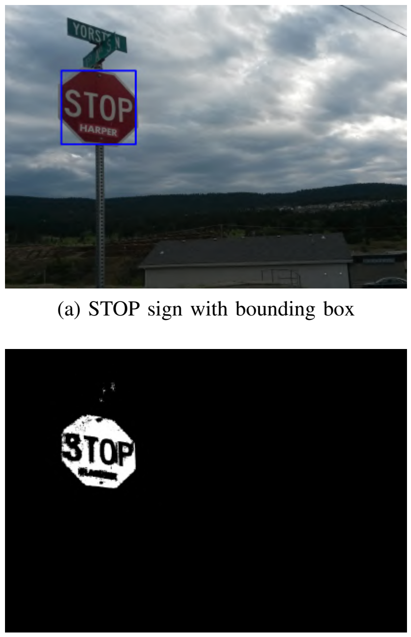
(b) Image mask

Fig. 6. Successful recognition of slightly dull image


(a) STOP sign with bounding box

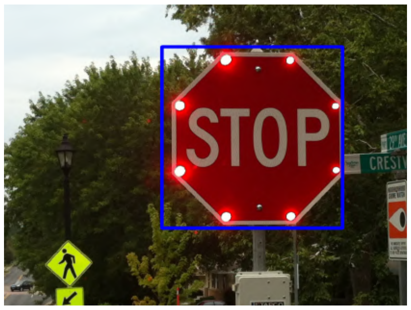
(b) Image mask

Fig. 7. Successful recognition of slightly dull image


(a) STOP sign with bounding box

(b) Image mask

Fig. 8. Successful recognition of highly dull image. It can be seen that even though the image has poor lighting the segmentation is good enough, the STOP sign is detected accurately. This demonstrates the robustness of the recognition model


(a) STOP sign with bounding box

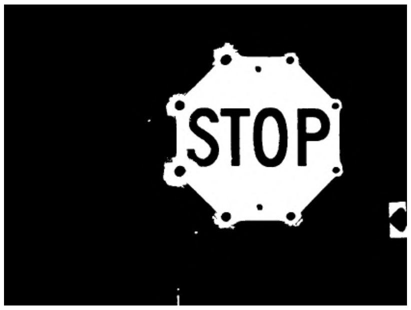
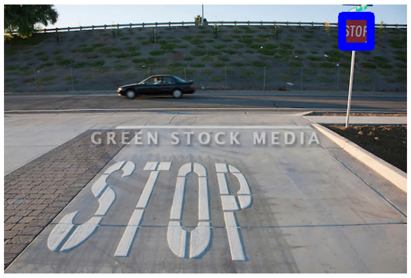
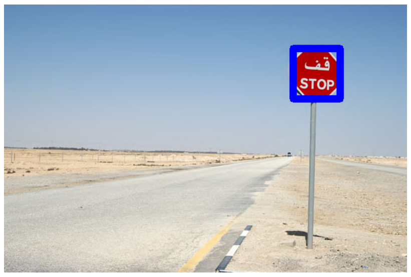

(b) Image mask

Fig. 9. Successful recognition of approximate octagonal shape. This demonstrates that the model is not over-fitted. It takes into consideration slightly different cases as well


(a) STOP sign with bounding box

(b) Image mask

Fig. 11. Successful recognition of slightly dull image


(a) STOP sign with bounding box

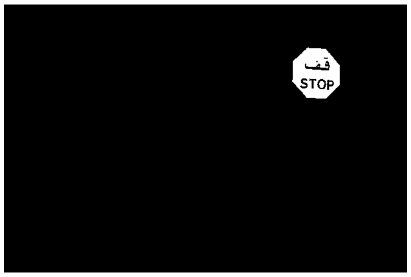
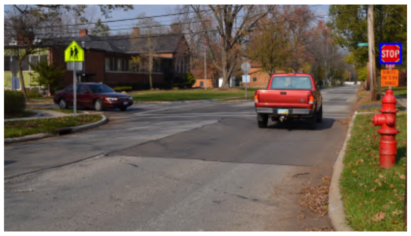

(b) Image mask

Fig. 10. Successful recognition of STOP sign in different language. As the model doesn't use Optical character recognition (OCR) and relies solely on shape and color, such signs are easy to detect


(a) STOP sign with bounding box

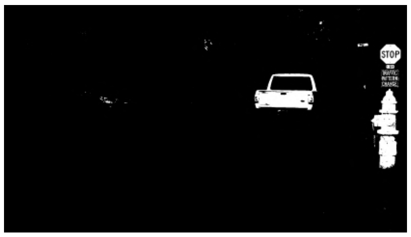
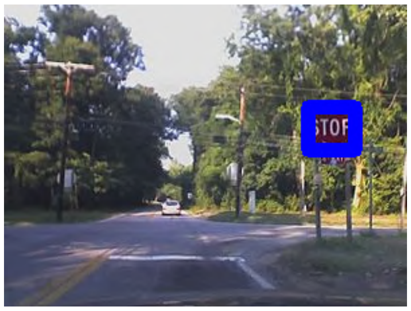
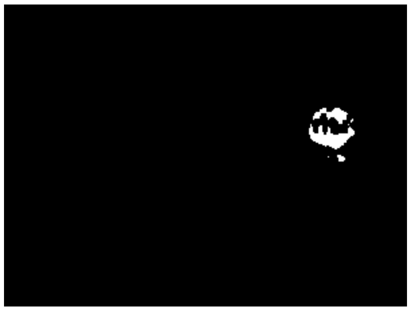

(b) Image mask

Fig. 12. Successful recognition of STOP sign in cluttered environment. Even though many other red objects are seen in the segmented image, STOP sign is accurately identified

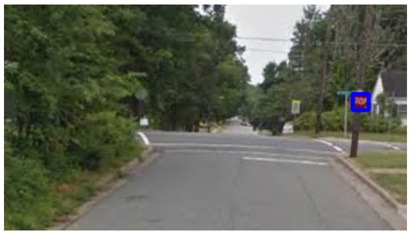


(a) STOP sign with bounding box

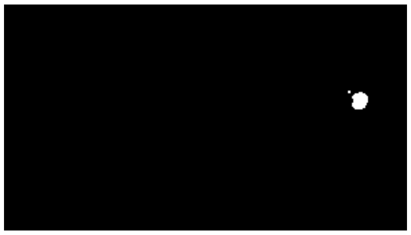


(b) Image mask

Fig. 13. Successful recognition of a low resolution poor image. Even though the image quality is poor, the model identifies the STOP sign demonstrating the robustness of the model

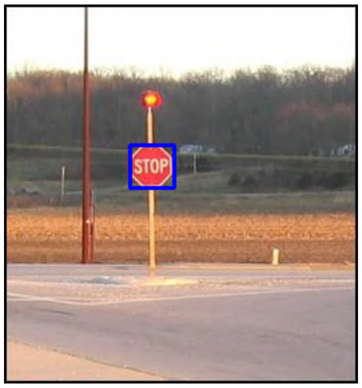


(a) STOP sign with bounding box

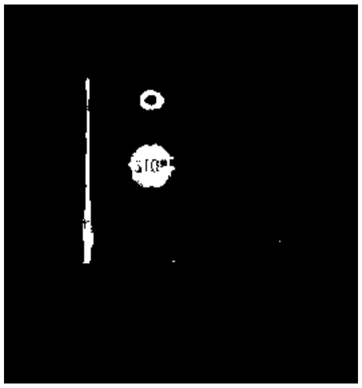


(b) Image mask

Fig. 14. Successful recognition of a low resolution poor image. Again, even though the image quality is poor, the model identifies the STOP sign demonstrating the robustness of the model

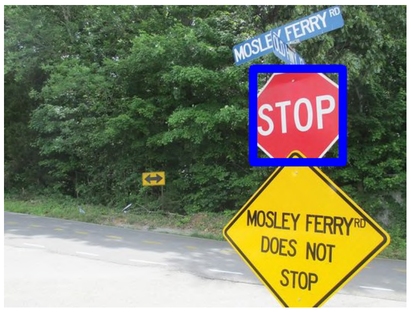


(a) STOP sign with bounding box

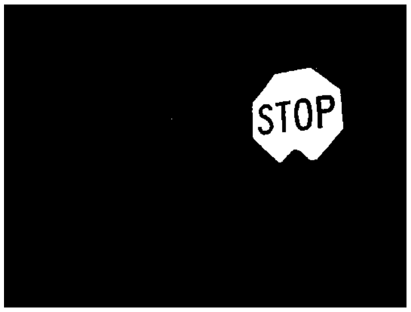


(b) Image mask

Fig. 15. Successful recognition of a low resolution poor image. Again, even though the image quality is poor, the model identifies the STOP sign demonstrating the robustness of the model

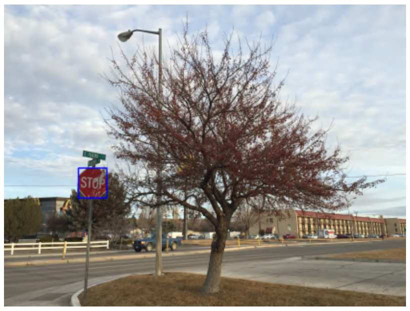


(a) STOP sign with bounding box

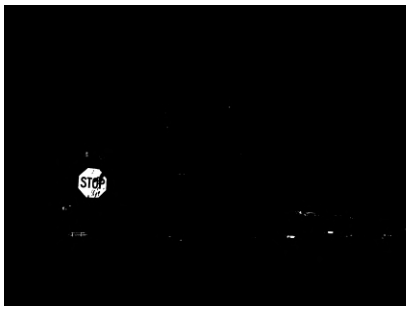


(b) Image mask

Fig. 16. Successful recognition of non-octagonal STOP sign. 9 edges can be seen of this STOP sign, still it is detected accurately.

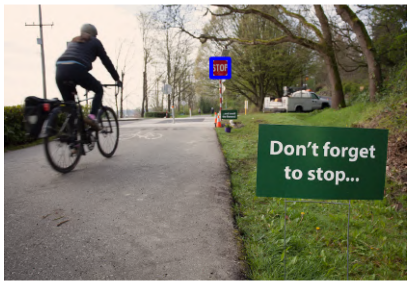


(a) STOP sign with bounding box

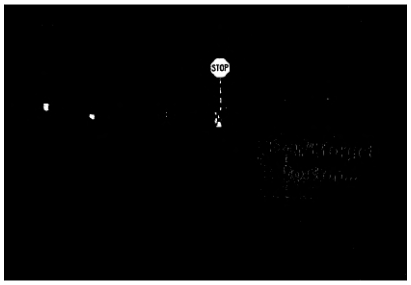


(b) Image mask

Fig. 17. Successful recognition of worn out STOP sign. Even though the red color has worn out, detection is accurate.

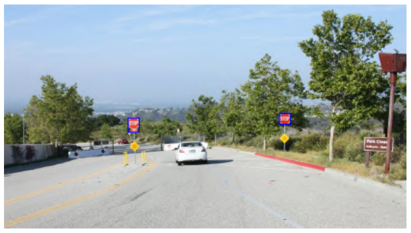


(a) STOP sign with bounding box

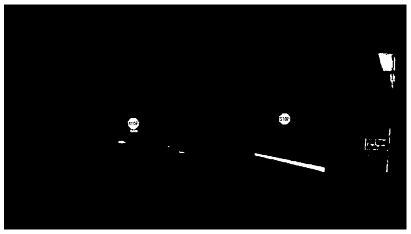


(b) Image mask

Fig. 18. Successful recognition of STOP sign far away in the image

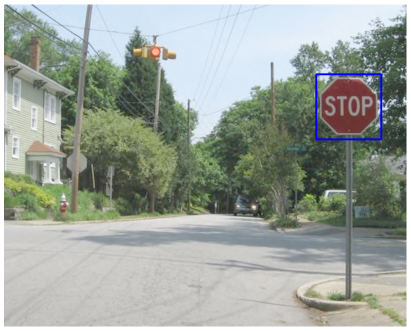


(a) STOP sign with bounding box

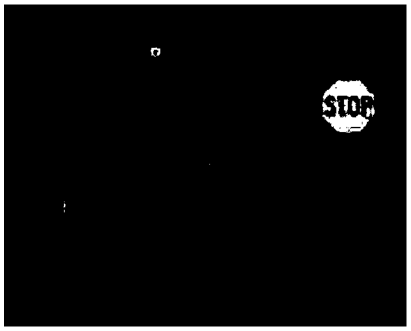


(b) Image mask

Fig. 19. Successful recognition of multiple STOP signs far away in the image



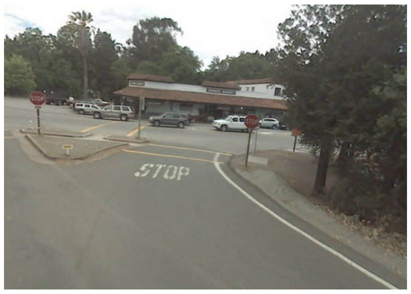
(a) STOP sign with bounding box



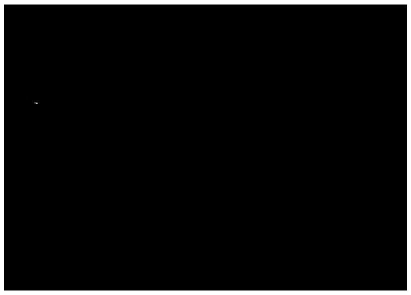
(b) Image mask

Fig. 20. Successful recognition of STOP sign which is in shadow region. The shadow changes the tint of red color but still results in good segmentation. Refer Fig. 4 (a) for Original image

B. Failed Tests

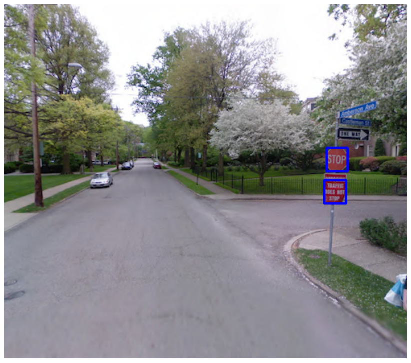


(a) Wrong detection

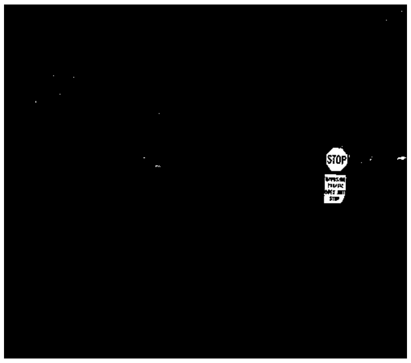


(b) Image mask

Fig. 21. The model fails in this case, since the image is too hazy. As it can be seen the classifier couldn't classify red pixels correctly.

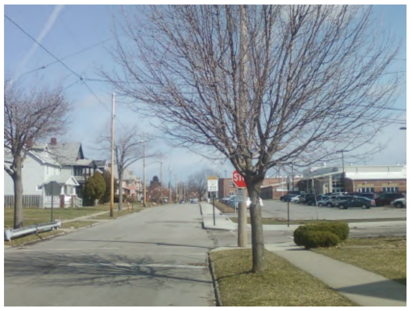


(a) Wrong detection

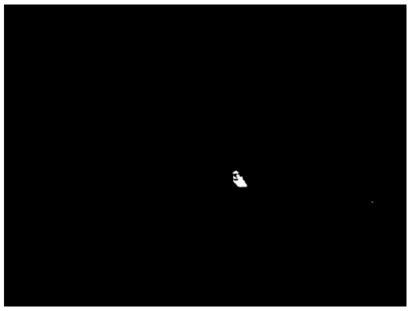


(b) Image mask

Fig. 22. The model detects the STOP sign correctly but also detects other sign board which is similar in shape in the segmented image. The segmentation is accurate though.

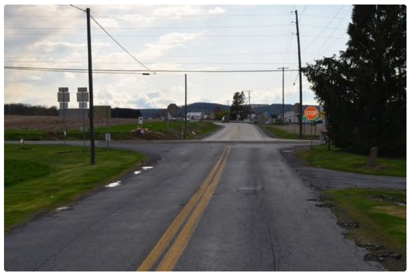


(a) No detection

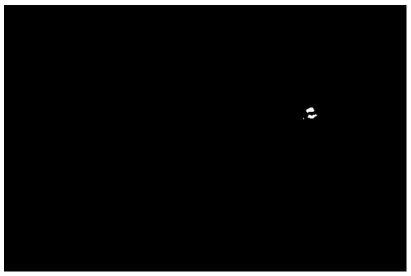


(b) Image mask

Fig. 23. No STOP sign is detected as only partial sign is visible in the image. The tree obscures the view and hence results in wrong detection. Again, segmentation is accurate.

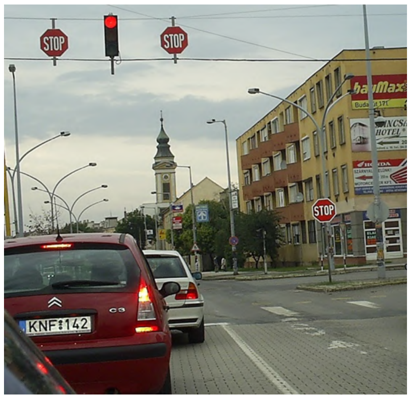


(a) Wrong detection



(b) Image mask

Fig. 24. Here the segmentation fails due to different shade of red. It is more towards orange side and hence the model fails to classify.



(a) Wrong detection

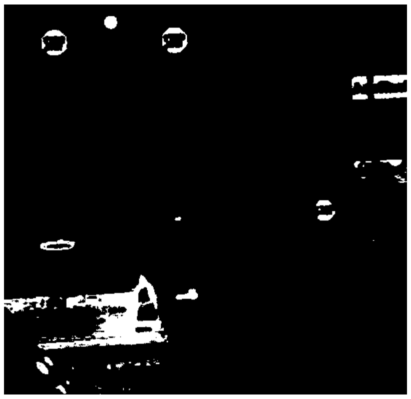

(b) Image mask

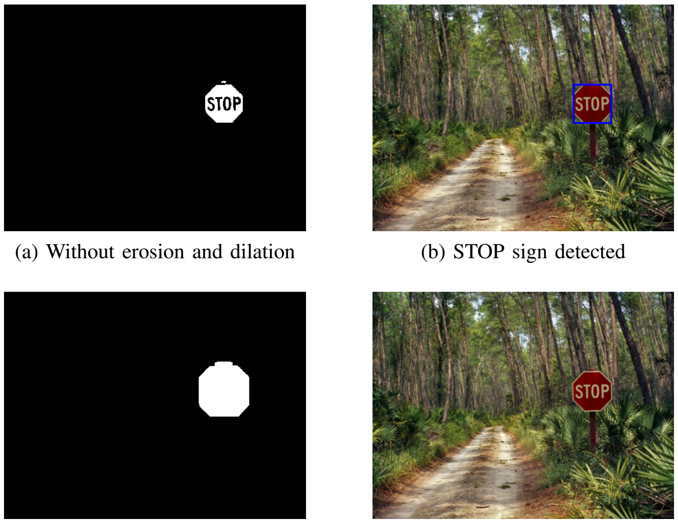
Fig. 25. 3 stops are present in the image of which non are detected. Even though the segmentation is good enough model fails due to breaks in segmented image. The text "STOP" is big enough to break the octagon into two parts. This results in wrong detection. Though the segmentation is fairly good, it fails due to STOP letters being too big.

It can be seen that the Gaussian Discriminative model is able to segment the image accurately almost every time. The reason behind some cases where the segmentation is not good is mainly the light in the image. Though gamma correction improves the lighting and did help perform better, it still is far from ideal.

Other pre-processing techniques like changing from BGR to HSV colors-space, increasing the saturation value and then converting back to BGR color-space have been tried and tested. This does improve the results in some cases but destroys the robustness of the model. Hence, this idea was discarded.

Sometimes, though the segmentation is good, the recognition module is unable to detect STOP signs mainly due to image being divided into two parts by STOP text. This can be improved using Morphological Transformations like erosion and dilation. This does improve such cases but again, destroys the robustness of the model For example see Fig. 26.

Histogram Equalization is another method for improving results by removing noise. This has been tested but didn't give satisfactory result. One reason behind receiving unsatisfactory results is that the model is not trained after histogram equalization. Contrast Limited Adaptive Histogram Equalization (CLAHE) is another method which can improve the results. We leave this for future work.



(a) Without erosion and dilation
(b) STOP sign detected
(c) With erosion and dilation
(d) No STOP sign detected

Fig. 26. (a) and (b) work pretty well without erosion and dilation. However, the results are ruined even by using erosion and dilation with small kernel size (3,3) which can be seen in (c) and (d)

VI. Conclusion and Future Work

The model works as expected and is able to detect STOP signs accurately almost always.

Morphological Transformations can be used to improve the results. However a smart way of applying these transformations must be used, i.e. not using it on all images but only on those where required, or else it will just ruin the results as seen in Fig. 26

However, this model can easily be fooled by placing red octagonal shaped objects in the image. To avoid this and guarantee of it being a STOP sign, Optical character recognition (OCR) can be used. Also, more focus needs to be drawn on pre-processing of images. All images are bound to be different due to different colors, light intensity, cameras etc. and hence should be made compatible to the model. Clear, sharp bright and vivid images would give the most accurate result.

Acknowledgment

I thank Mr. Shiladitya Biswas, Mr. Sushruth Nagesh, Ms. Asfiya Baig, Ms. Savitha Srinivasan for collaborating with me on this project. We held many whiteboard discussion sessions and also shared the labelled data with almost equal contribution.

References

[1] M. A. A. Sheikh, A. Kole, and T. Maity. Traffic sign detection and classification using colour feature and neural network. In *2016 International Conference on Intelligent Control Power and Instrumentation (ICICPI)*, pages 307–311, Oct 2016.
[2] Nikolay Atanasov. Ece276a: Sensing estimation in robotics.
[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.