Color Segmentation for Stop Sign Detection

Shakeel Ahamed Mansoor Shaikna

Mechanical and Aerospace Engineering
University of California San Diego
San Diego, CA, US
samansoo@uscd.edu

***Abstract*—This report presents a study of Color Segmentation using Probabilistic models for training a given dataset and using it to detect stops signs. For Color Segmentation, training using Probabilistic color models may not be efficient as recently many Deep Learning models have been developed which has better performance than probabilistic models. After training, we detect the stop signs and return a bounding box for the stop signs in a test image.**

I. Introduction

In recent years, collection of image and video information for perception and processing of the image content has been of paramount importance in the fields like biomedical imaging, facial recognition and so on. Processing of these huge data by humans is not an effective way, so we are in need of systems which can automatically process these images once trained in an artificial environment.

The process of dividing an image into multiple regions is called Segmentation. Segmentation of the image makes the analysis of each regions easier. In this study, I used semantic or pixel-wise segmentation where each pixel in an given image are labelled and regions with the same label share the same characteristics.

Object-detection has become an essential part of many advanced tasks like Autonomous Vehicles, Face-Recognition and its significance has grown exponential over the decades. The detection of Stop signs is a crucial task for autonomous or self driving cars, as the car has to come to complete stop at the signs. So, here we use the trained model to detect stop signs in the testing image.

The workflow of the project goes from labelling of the dataset for supervised learning i.e., by hand labelling of images using RoiPoly. The positive samples are taken as pixels with Red Stop signs and negative samples are taken as pixels with Not Stop signs. Having created a masks for the dataset, I split the data into training and validation, as it reduce any over-fitting of data and also we can validate our trained models. Now, having a training data and labels, supervised learning can be done for color classification using probabilistic models such as Logistic Regression, Gaussian Generative Models etc. In this project, I have used Logistic Regression for training of the images.

Once the training is completed, we have the model which we use to detect the Red stop signs in the validation set. But, using logistic regression the model finds red color pixels which are not a Stop sign. So, I used metrics like aspect ratio and area of the regions to remove the bounding box which do not have the Stop signs.

Section II discusses the problem which I am trying to solve and Section III explains in detail the approach taken to solve the problems stated in Section II. Section IV presents the results from training the model, testing image and bounding box of the detected Stop Sign.

II. Problem Formulation

A. Color Segmentation

Color segmentation problem requires segmenting the image pixel-wise into a set of discrete regions which share the same characteristics (color). Each pixel x has 3 channels either RGB, HSV, YUV etc and can be formed as a 3D vector and discrete labels y can be ranging from 1,2...N. For the color segmentation I chose the color space to be RGB and I am trying to classify the pixels into two categories Red(1) and Not red(0).

Input pixel $x \in \mathbb{R}^3$

Output labels $y = \{0, 1\}$

Our goal is to define a function h which maps from $\mathbb{R}^3 \rightarrow \mathbb{R}$ i.e., assigning a label to given data. There are many methods to model a classifier, here we use Probabilistic models p(y| x) for the classification. I used Logistic Regression for classification.

$$h(x) := \arg\max_y p(y \mid x) \tag{1}$$

For training the model, we have Data D=$\{(x_i, y_i)\}$ of independent and identically distributed (i.i.d) samples and optimize the model using parameter learning p(y|x,ω)i.e., introducing extra parameters to improve the learning. There are different parameters like $\omega_{MLE}, \omega_{MAP}$. As the chosen model for training is a discriminative model we take ω_{MLE} (Maximum Likelihood Estimate) as the parameter.

$$\omega_{MLE} := \arg\max_\omega p(y \mid x, \omega) \rightarrow \text{Training} \tag{2}$$

$$y_* \in \arg\max_y p(y \mid x, \omega_{MLE}) \rightarrow \text{Testing} \tag{3}$$

B. Bounding Box

The formulation of bounding box is required to detect the exact region of the Stop Sign in the test image. For this task,

we give the mask i.e a binary image as input as it is easier to find the regions and get the coordinates of the bounding box as output.

Input: Mask $= \mathbb{R}^2$

Output: Box $= \mathbb{R}^4$

The goal of this task is to find the bounding box in terms of the bottom left and top right coordinates of x and y. $(x_{bottomleft}, y_{bottomleft}, x_{topright}, y_{topright})$.

The following section, gives a detailed explanation of solutions to the above stated problems.

III. Technical Approach

Color Segmentation for Stop Sign detection can be implemented through the following steps:

A. Labelling of Data
B. Training and Validation set
C. Supervised Learning
D. Stop Sign Detection

First I setup an python virtual environment and installed all the required packages such as;

- opencv-python
- matplotlib
- numpy
- roipoly

A. Labelling of Data

For Supervised Learning, we need to label each images given in the dataset. For creating the label, there are many approaches that can be used. I used $roipoly()$ function and hand-labeled regions of interest (Red Stop Sign). $roipoly()$ function helps us to choose a polygonal region and returns a boolean true value for the chosen region and boolean false for the other regions. As the Stop Sign region is an octagon, I chose the $roipoly()$ for labeling the dataset. After creating the labels, then the mask was created and I converted the boolean array to uint8 such that True becomes 1 and False becomes 0.

The difficulty faced using $roipoly()$ was that it could create a mask for only one Stop signs. The dataset contained images with multiple Stop signs. So, I had to choose multiple RoIs for those set of images in the dataset and append them into a single image.

Now, inspecting the type of each images in the dataset, some of the images was of type uint8 and some was float32. So, a normalization on the images was required. All the images was converted to float32 and the range was from 0 to 1. Then, both the images and the masks have been saved as a numpy array.

B. Training and Validation set

The partitioning of dataset is important as it can reduce overfitting of data and moreover we can evaluate the performance of our trained models. Now, the dataset was created and saved as a numpy array. Then, I partitioned the dataset to form training and validation sets. I took an 80:20 split of the dataset. While inspection of the dataset, I found that out of 200 images only 100 images contained Stop signs. So, while choosing the validation set, I randomly chose 20 images having Stop signs and 20 images without Stop sign using $np.random.permutation$ for validation set and the rest 160 images for training set.

C. Supervised Learning

The dataset has been labelled and partitioned, now we have the necessary input for Supervised learning. The pair of original image X and its binary classified mask y can be considered as the Data D=$\{(x_i, y_i)\}_{i=1}^{N}$, where N is the number of pixels present in the image X and x_i is the pixel, y_i is the label and parameterized by ω.

For training the model, I used logistic regression which is a discriminative model. Now, I try to implement logistic regression with the help of equation 2 to get the trained model. Since, X$\in \mathbb{R}^3$ and the label y $\in \{0, 1\}$, we can use sigmoid as the mapping function.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{4}$$

As y is a binary label $\in \{0, 1\}$, we can model the PMF of y as:

$$p(y = 1 \mid x, \omega) = \sigma(\omega^T x) \tag{5}$$

For p(y$\in x, \omega$) to be a valid PMF, it needs to sum to 1 over $\{0, 1\}$:

$$p(y = 0 \mid x, \omega) = 1 - \sigma(\omega^T x) \tag{6}$$

Hence, we can combine the two equation to get a final distribution of y.

$$p(y \mid x, \omega) = \sigma(\omega^T x)^y (1 - \sigma(\omega^T x))^{1-y} \tag{7}$$

As each of the data (X,y) is an i.i.d. we can model the joint data likelihood as:

$$p(y \mid X, \omega) = \prod_{i=1}^{N} p(y_i \mid x_i, \omega) \tag{8}$$

$$p(y_i \mid x_i, \omega) = \sigma(\omega^T x_i)_i^y (1 - \sigma(\omega^T x_i))^{1-y_i} \tag{9}$$

As log is a monotonically increasing function, we can write ω_{MLE} as;

$$\omega_{MLE} = \arg\max_{\omega} logp(y \mid X, \omega) \tag{10}$$

$$logp(y \mid X, \omega) = \sum_{i=1}^{N} y_i log(\sigma(\omega^T x_i) + (1 - y_i) log(1 - \sigma(\omega^T x_i))) \tag{11}$$

For finding the argmax value of ω we take the negative of $logp(y \mid X, \omega)$ and do gradient descent. The gradient descent of -$logp(y \mid X, \omega)$ is:

$$\nabla_\omega(-logp(y \mid X, \omega)) = -\sum_{i=1}^{n} (y_i - \sigma(x_i^T \omega) x_i) \tag{12}$$

where X is the original image containing N pixels and y contains the label of N pixels.

The MLE update equation can be formulated as

$$\omega_{MLE}^{t+1} = \omega_{MLE}^{t} + \alpha \sum_{i=1}^{n} (y_i - \sigma(x_i^T \omega) x_i) \tag{13}$$

Training parameters:

- Learning rate α = 0.01
- Epochs = 45
- Initial weights $\omega_0 = np.random.randn(4, 1)$

The necessary parameters have been initialized, I have initialized the weights as (4,1) 3 for RGB channels and 1 for bias. And now I perform Stochastic gradient descent as I do gradient descent for each image on the training set separately. Image X is of size (N,4) and label/mask y is of size (N,1). At the end of each epoch, the calculated loss and weights have been saved as numpy array.

After performing training over epochs, I calculate the performance of the trained model in the validation set. The prediction of y is given by:

$$y^* = 1, (x^*)^T w^*) >= 0 \tag{14}$$

$$y^* = 0, (x^*)^T w^*) < 0 \tag{15}$$

D. Stop Sign Detection

Given a new image, I perform segmentation operation by creating a new mask for that image using equation 14&15. Now, a binary mask has been created and I perform morphological operations on the masked image like dilation. I perform these operations to make the mask sharper around the edges as it will be easier for the detection of Stop sign. For dilation operation I chose the kernel size as (3,3).

Now, I find the contours in the dilated image by using openCV function $cv2.findContours()$. This function returns all the contours with their hierarchy i.e., each contour is linked to another. Hierarchy will be a list of lists.

Next, I calculate the coordinates of the smallest bounding box for all contours in the dilated mask using $cv2.boundingRect()$ function. This returns the top left coordinates x,y and width and height of the bounding box w,h.

There can be other pixels containing red color which are not a Stop sign. For this I used metrics like aspect ratio (height/width) and ratio of bbox area to image area to discard the regions not having Stop signs.

IV. Results

From Fig. 1 we can see that the loss is decreasing as expected and it shows the training is proper.

The following cases was observed by me:

Case 1: Fig 2 shows that the bounding box has been formed perfectly and the mask confirms this. It is a successful case.

Case 2: Fig 4 shows that there is a stop sign clearly, but it is not detected. It is a failure case.

Case 3: In case of figure 6, there is no stop sign and none are detected. It is a successful case.

Case 4: In case of Fig. 8, there is no stop sign but a stop sign is detected, this is a failure case. This is due to the dilation of the image.

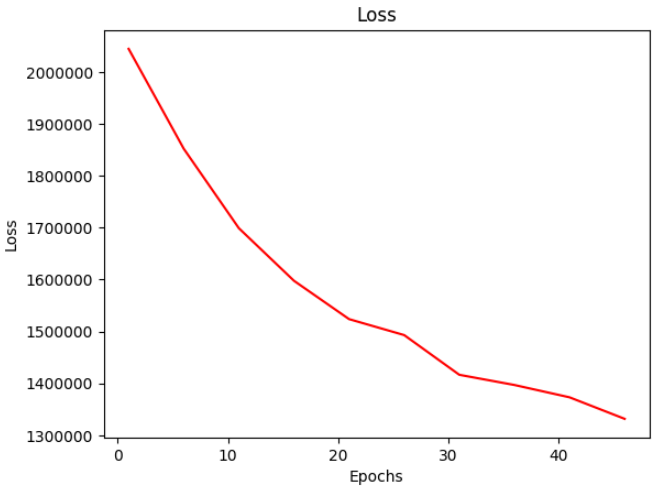

Fig. 1. Loss vs Epoch

Fig. 2. Stop Sign Detected

Acknowledgment

I wish to acknowledge here that I collaborated with the following students while working on the project.

- Anirudh Swaminathan
- Siddarth Meenakshi Sundaram
- Venkatesh Prasad Venkataramanan

We hand-labeled the images and created the dataset together.

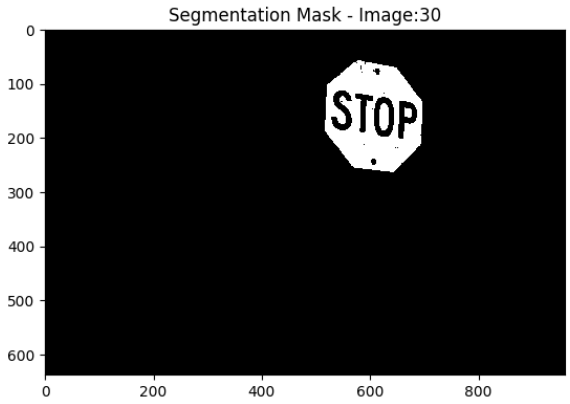

Fig. 3. Mask of the image

Fig. 6. No Stops detected

Fig. 4. Stop Sign not detected

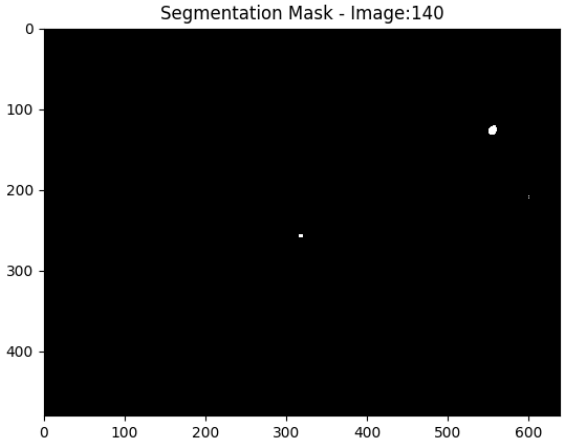

Fig. 7. Mask of the image

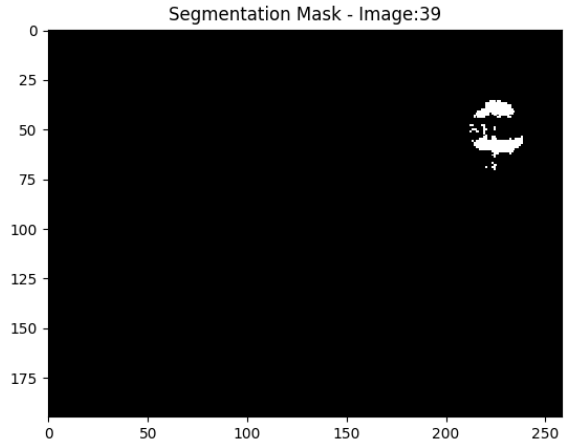

Fig. 5. Mask of the not detected image

Fig. 8. No Stops, but detected building

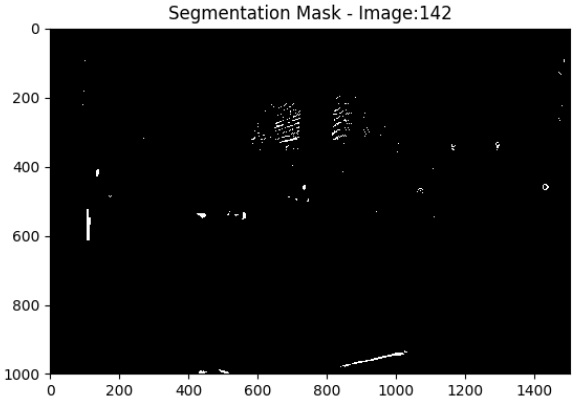

Fig. 9. Mask of the image