SUPERVISED LEARNING FOR COLOR SEGMENTATION AND STOP SIGN DETECTION

Chun-Nien Chan
Department of Electrical and Computer Engineering
University of California, San Diego
chc030@eng.ucsd.edu

1. INTRODUCTION

Traffic sign detection is an essential part of autonomous driving, which has been researched for a few decades. Much research has been conducted with a front camera installed in front of the vehicle sensing the environment in real-time. The goal of traffic sign detection is to design a visual model to localize and classify traffic signs in an image. It requires the ability to robustly model invariance to brightness, occlusions, and other variations in the environment.

There are several challenges for traffic sign detection. First of all, the quality of the results must be consistent for input images in various environmental conditions aforementioned. Accuracy is the most basic requirement and evaluation metric for the algorithm. For high-level autonomous vehicles, accuracy must be appropriately monitored guaranteed in all environmental conditions, including the worst scenarios. The second challenge is the performance of inference speed. An autonomous vehicle must sense the environment in a high frequency to react to road emergencies when driving. Moreover, the computing resource installed on the vehicle is limited. Thus, the traffic sign detection must be efficient enough for full autonomous driving application.

Recently, deep convolution networks have significantly improved image classification and object detection accuracy [1][2]. However, the drawbacks of these approaches may not be neglected. The major one is the requirement of a massive amount of training data. The accuracy is only guaranteed when sufficient training data is provided when training the model.

In this paper, we aim to design an algorithm for stop sign detection, which is a subproblem of traffic sign detection. We propose an algorithm based on image segmentation with pixel color classifier and deterministic bounding box classifier with object shape detection. With the assistant of the N-way merge approach, this algorithm can detect stop signs in various environmental conditions, invariant to changes in brightness and color saturation.

Fig. 1. Illustration of x-y image coordinate system. The origin is located at the bottom left of the image, which is similar to the standard Cartesian coordinates. Figure made by Arash Asgharivaskasi posted on Piazza for UCSD ECE276A Winter 2020.

2. PROBLEM FORMULATION

A stop sign detector takes an entire image as input. The detector first processes the whole image with image segmentation. The image segmentation produces a binary mask that indicates pixels that may belong to a stop sign. The binary mask is then passed to the stop sign classifier, which detects region of interests (ROIs) and decides if an ROI contains a stop sign.

2.1. Image Segmentation

Image segmentation aims at finding out pixels that may belong to the object we are interested in. The input of the image segmentation an $H \times W$ RGB image, which can be written as a 3-dimensional matrix $\mathbf{G} \in \mathbb{N}_0^{H \times W \times 3}$, where the values in the matrix are integers between 0 and 255. The output is a binary mask, which can be written in the matrix form as $\mathbf{M} \in \mathbb{N}_0^{H \times W}$. The values in the matrix are either 0 or 1, where 1 indicates that the corresponding pixel in the input image belongs to a stop sign, and 0 otherwise.

2.2. Stop Sign Classifier

The goal of a stop sign classifier is to form region of interests (ROIs) by grouping stop sign candidate pixels and select ROIs contain a stop sign. In the proposed architecture, the input of the stop sign classifier is the binary produced by image segmentation. The output is a list of rectangular ROIs that contain a stop sign. The coordinates of the bottom-left corner and top-right corner of the bounding box are used to represent a ROI, which can be written as $(x_{BottomLeft}, y_{BottomLeft}, x_{TopRight}, y_{TopRight}) \in \mathbb{Z}^4$. The coordinates are the pixel indices in the x-y image coordinate system. Fig.1 is an illustration of x-y image coordinate system.

3. TECHNICAL APPROACH

3.1. Color Enhancement

The pixels can be represented in different color spaces. In our experiments, three different color spaces are used: RGB, HSV, and YCrCb[3].There are several color enhancement approaches to adjust the saturation and brightness after converting the image to the desired color space. The approaches are listed as follows:

- **Histogram Equalization(channel_id)**: Given index of an channel and perform histogram equalization[3] on that channel.
- **Mul&Clip(channel_id,C,Vmax)**: Given index of an channel, multiple values in that channel by constant C, clip the values greater than $Vmax$ to $Vmax$.

3.2. Image Segmentation

Image segmentation is done by classifying the color of each pixel individually. Since a significant part of a stop sign is red, a color classifier is used to determine if the color of a given pixel is "stop sign red" or other colors. The value in the output binary mask will be 1 if the corresponding pixel in the input image is classified as "stop sign red" and 0 otherwise.

The input image is first transformed to the desired color space and prepare features for the classifier. The difficulty for classifiers to distinguish one color from another varies, and some classifiers may achieve better accuracy in one color space than the other. Moreover, values in different color spaces can be combined to form a large feature vector for a single pixel. This may provide the classifier with more information on the relationship between color channels to achieve better performance.

3.2.1. Supervised Learning

Supervised learning algorithms are used to define the pixel color classifier. The goal of the supervised learning is to find a classifier function $h : \mathbb{R}^d \to \mathbb{R}$ given a training data set $D := \{\mathbf{x}_i, y_i\}_{i=1}^n$ which minimize the value of the loss function: $h = \min_h loss(h)$, where d is the number of features of a single pixels, n is the number of pixels, $\mathbf{x}_i$ is the i^{th} pixel, and y_i is the color label for the i^{th} pixel. The training data can also be written in matrix notation $D = (\mathbf{X}, \mathbf{y})$, where $\mathbf{X} \in \mathbb{R}^{n \times d}, \mathbf{y} \in \mathbb{R}^n$

To apply classifier h to the input image $\mathbf{G} \in \mathbb{Z}^{H \times W \times 3}$, the image is first resized to a 2 dimensional matrix $\mathbf{X} \in \mathbb{R}^{(H*W) \times d}$. The rows of $\mathbf{X}$ are feature vector for each pixel in the image.

3.2.2. Logistic Regression

Logistic Regression is a discriminative model that uses a logistic function to model a binary dependent variable. The following equation is the conditional probability of label $\mathbf{y} \in \{-1, 1\}^n$ given training data $\mathbf{X} \in \mathbb{R}^d$ and model parameters $\omega \in \mathbb{R}^d$:

$$p(\mathbf{y}|\mathbf{X}, \omega) = \prod_{i=1}^{n} \sigma(\mathbf{y}_i \mathbf{x}_i^T \omega) = \prod_{i=1}^{n} \frac{1}{1 + exp(-\mathbf{y}_i \mathbf{x}_i^T \omega)} \quad (1)$$

The training step is to optimize the parameters ω regarding maximum likelihood estimation (MLE). The following is the optimization steps,:

$$\omega_{MLE}^{(t+1)} = \omega_{MLE}^{(t)} + \alpha \sum_{i=1}^{n} \mathbf{y}_i \mathbf{x}_i (1 - \sigma(\mathbf{y}_i \mathbf{x}_i^T \omega_{MLE}^{(t)})) \quad (2)$$

where α is the learning rate, and σ is the sigmoid function. The prediction made by the model using test example $\mathbf{x}_* \in \mathbb{R}^d$ and optimized parameter $\omega^* \in \mathbb{R}^d$ is the following equation:

$$y_* = \begin{cases} 1 & \mathbf{x}_*^T \omega^* \geq 0 \\ -1 & \mathbf{x}_*^T \omega^* < 0 \end{cases} \quad (3)$$

3.2.3. One vs All Logistic Regression

One vs all logistic regression is a model composed of k distinct binary classifiers for the purpose of classifying k different classes. For each label $l \in \{1, ..., K\}$ in the training data, a binary logistic classifier is trained by reassigning labels:

$$y_i' = \begin{cases} 1 & y_i = l \\ -1 & y_i \neq l \end{cases} \quad (4)$$

Given a test sample $\mathbf{x}*$, the inference is performed with following equation:

$$\begin{aligned} y* &= argmax_{l \in \{1,...,K\}} p(y = 1|\mathbf{x}^*, \omega_l) \\ &= argmax_{l \in \{1,...,K\}} \mathbf{x}_*^T \omega_l^* \end{aligned} \quad (5)$$

where ω_l is the model parameter for logistic regression of class l.

3.2.4. Kary Logistic Regression

Kary logistic Regression is another variant of logistic regression. Kary uses softmax function to classify k classes.

The conditional probability of label $\mathbf{y} \in \{1, ..., K\}^n$ given data $\mathbf{X} \in \mathbb{R}^d$ and model parameters $\mathbf{W} \in \mathbb{R}^{K \times d}$ is the following equation:

$$p(\mathbf{y}|\mathbf{X}, W) = \prod_{i=1}^{n} \mathbf{e}_{y_i}^T \mathbf{s}(W\mathbf{x}_i) = \prod_{i=1}^{n} \mathbf{e}_{y_i}^T \frac{exp(W\mathbf{x}_i)}{\mathbf{1}^T exp(W\mathbf{x}_i)} \quad (6)$$

where $\mathbf{e}_j$ is the j-th standard basis vector and $s(z)$ is the softmax function.

The parameters W can be optimized via maximum likelihood estimation with gradient descent. What follows is the update equation:

$$W_{MLE}^{(t+1)} = W_{MLE}^{(t)} + \alpha(\sum_{i=1}^{n} (\mathbf{e}_{y_i} - \mathbf{s}(W_{MLE}^{(t)}\mathbf{x}_i))\mathbf{x}_i^T) \quad (7)$$

The the inference given a test sample $\mathbf{x}*$ is done by finding the maximum conditional probability:

$$y* = argmax_{y \in \{1,...,K\}} p(y|\mathbf{x}^*, W) \quad (8)$$

3.2.5. Gaussian Naive Bayes

Gaussian Naive Bayes is based on the assumption that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution. The model parameter ω includes μ and σ^2, where μ is the mean of the distribution and σ^2 is the variance. The conditional probability of a label $\mathbf{y}_i$ given parameter θ is modeled by Categorical distribution as the following equation:

$$p(\mathbf{y}_i|\theta) = \prod_{k=1}^{K} \theta_k^{\mathbf{1}\{\mathbf{y}_i = k\}} \quad (9)$$

and the conditional probability of the distribution $\mathbf{x}_{il} \in \mathbb{R}$ given label $\mathbf{y}_i$ and parameter ω is modeled by Gaussian distribution as the following equation:

$$p(\mathbf{x}_{il}|\mathbf{y}_i = k, \omega) = \phi(\mathbf{x}_{il}, \mu_{kl}, \sigma_{kl}^2) \quad (10)$$

The maximum likelihood estimation of the given parameters θ, μ, and σ^2 are the following:

$$\theta_k^{MLE} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{\mathbf{y}_i = k\} \quad (11)$$

$$\mu_{kl}^{MLE} = \frac{\sum_{i=1}^{n} x_{il} \mathbf{1}\{\mathbf{y}_i = k\}}{\sum_{i=1}^{n} \mathbf{1}\{\mathbf{y}_i = k\}} \quad (12)$$

$$\sigma_{kl}^{MLE} = \sqrt{\frac{\sum_{i=1}^{n} (x_{il} - \mu_{kl}^{MLE})^2 \mathbf{1}\{\mathbf{y}_i = k\}}{\sum_{i=1}^{n} \mathbf{1}\{\mathbf{y}_i = k\}}} \quad (13)$$

The prediction made by the model using test example $\mathbf{x}_* \in \mathbb{R}^d$ and parameters θ, μ, and σ^2 is the following equation:

$$\begin{aligned} y_* =& argmax_{y \in \{1,...,K\}} log\theta_y^{MLE} \\ &+ \sum_{l=1}^{d} log\phi(x_{*l}; \mu_{yl}^{MLE}, (\sigma_{yl}^{MLE})^2) \end{aligned} \quad (14)$$

3.3. Stop Sign Classification

After the image segmentation step, a binary map of stop sign candidate pixels is generated. Stop sign classification step then finds stop sign candidate ROIs by grouping and labeling 1s in the binary mask. These candidate ROIs must pass a set of hand-made rules to be considered as positive ROIs which contain stop signs.

3.3.1. Region Labeling

Region labeling is the first step processing the binary mask. LABEL in SKIMAGE.MEASURE is used to label connected pixels and regions. Two pixels are connected when they are neighbors in a 1-connected sense and both have value 1 in the binary mask. Each connected region is then assigned an unique positive integer as the label of the region. The output of this step is an integer mask $L \in \mathbb{N}_0^{H \times W}$, where each value in the mask is the region label of its corresponding pixel, and 0 when the pixel is not classified as stop sign candidate pixel in image segmentation.

3.3.2. Extract Region of Interest

After labeling connected pixels, rectangular ROIs are extracted for each of the region. REGIONPROPS in SKIMAGE.MEASURE is used to measure the properties of labeled regions. These properties include region area and convex hull image of the given region in binary image are used in later steps.

3.3.3. Region of Interest Classification

For each ROI extracted in the previous step, properties like region contour and shape are measured with OPENCV functions: CV2.FINDCONTOURS is used to find the contour of the convex hull image of the region and CV2.APPROXPOLYDP is used to approximate polygonal curve from the contour and extract the edges and corner angles of the polygon.

A candidate ROI must pass the following rules to be considered as a valid ROI contain stop sign:

- **Shape of Bounding Box**: The length of the longer side must mot exceed the length of the shorter side * $SideFactorThreshold = 1.5$. This rule can filter out bounding boxes that are too "thin" or too "fat".

- **Area of Region Compares to Image**: The ratio of the convex hull image area of the region must to the image area must be greater than $AreaRatioThreshold_{cvx2img} = 0.001$. This rule aims to filter out small ROI which is difficult to be observed by human eyes.

- **Area of Region Compares to Bounding Box**: The ratio of convex image area to the bounding box area must be greater than $AreaRatioThreshold_{cvx2box} = 0.2$.

- **Partial Octagon - Edges**: The number of edges of approximated polygon must between 7 and 10. A certain level of tolerance is allowed when detecting the shape. The ideal number of edges should be 8 since all the stop signs are octagons. However, due to occlusion, white spots, angle of the stop sign, and the defect in color segmentation, the approximated polygon might not perfectly have 8 edges.

- **Partial Octagon - Corner Angles**: There must be 2 set of four consecutive corners of the approximated polygon that angles sum between $540°$ to $580°$. A perfect octagon has 8 corners with angle $135°$, and the ideal sum of 4 consecutive corner angles is $560°$. However, the approximated polygon might not be a perfect octagon in real case. The goal of this rule is to decide if the region is a "partial octagon". Even if a stop sign is partially occluded by another object, it might still be able to be determined as a "partial octagon".

The bounding box of ROIs which pass the rules above are the output of stop sign classification. The coordinates of bottom-left and top-right corner of the bounding box $(x_{BottomLeft}, y_{BottomLeft}, x_{TopRight}, y_{TopRight})$ would be extracted to form a list of vector.

3.3.4. N-way Merge

The input images vary in brightness and color saturation. It is impossible to find a general image processing approach to guarantee all images after adjusted have the same level of brightness and saturation. An N-way merge approach is proposed to overcome this issue and improve the performance of our stop sign detector. Instead of processing the image with only one color enhancement approach, we make N duplicates of the image and enhance the color with N different approaches. Then each enhanced image went through the process of color segmentation and stop sign classification individually. The merge step of the N-way merge is to filter out those invalid regions in the binary mask and combine binary masks from N images together with element-wise OR operation. Finally, perform stop sign classification on the combined binary mask and output the bounding boxes.

We implement 4-way merge in our release code. The four color enhancement are listed as follows. The color spaces and operation of color enhancement are listed in .

1. **Normal Image**: The input image without color enhancement.

2. **Brightness Equalization**: Convert input image to YCrCb, then apply HISTOGRAMEQUALIZATION(CHANNEL_Y).

3. **Saturation Enhancement #1**: Convert the brightness equalized image to HSV, then apply MUL&CLIP(CHANNEL_S, 1.1, 255) and MUL&CLIP(CHANNEL_V, 1.1, 255).

4. **Saturation Enhancement #2**: Convert the brightness equalized image to HSV, then apply MUL&CLIP(CHANNEL_S, 2.3, 255) and MUL&CLIP(CHANNEL_V, 1.2, 255).

N-way merge can take advantages of multiple color enhancement approach. One drawback is that the merged binary mask may be noisy due to the OR operation and dwarf the performance of stop sign classification. However, by tunning parameters in 3.3.2, it is possible to filter out the false detections.

4. RESULTS AND DISCUSSION

4.1. Image Segmentation Performance

Supervised learning algorithms in 3.2 are examine with different feature sets. The training data includes 100,000,000+ labeled pixels from 197 non-enhanced images. Each pixel is assigned one of six color labels: STOP-SIGN-RED, OTHER-RED, ORANGE, BROWN, BLUE, and OTHERS. In the experiment, 200,000 pixels from each color class (1,200,000 in total) is uniformly chosen for cross validation from data set, independent to which image the pixel came from.

For all experiments afterward, 5-fold cross-validation is used to calculate accuracy, precision, and recall. The hyper-parameters for logistic regression, 1-vs-all logistic regression, and Kary logistic regression are listed as follows: 0.005 for learning rate α, 500 for maximum iteration, $3,000$ for batch size in mini-batch gradient descent. A bias term is added to the decision for all logistic regressions.

When measuring the performance of the learning algorithm, how we decide a prediction is accurate is different from traditional approaches. In the stop sign detection application, we only care about color STOP-SIGN-RED and OTHERS. Even though more than two classes of color are labeled in the training data and used in training and testing, we treat the prediction incorrect if and only if STOP-SIGN-RED is predicted as any other color or other color is predicted as STOP-SIGN-RED. The performance of learning algorithms are listed in Table. 1

Fig.4.2-4.2 are stop sign detection results from 8 different test images. The image segmentation is performed using Gaussian naive Bayes classifier with 3 classes. The training pixels are selected from the remaining 189 images which are not testing images. Even though most of the red pixels of stop

signs are correctly detected, there are many false alarms come from red and orange objects like roadblock and brick walls.

4.1.1. Data with Different Number of Classes

Even though only 2 different classes STOP-SIGN-RED and OTHERS need to be distinguished in stop sign detection, there is a hypothesis that split pixels into more classes would help the learning algorithms find the "boundary" between colors. To test this hypothesis, three different labeling approaches are listed as follows:

- **2 Classes**: STOP-SIGN-RED and OTHERS
- **3 Classes**: STOP-SIGN-RED, OTHER-RED (including OTHER-RED, ORANGE, BROWN), and OTHERS
- **6 Classes**: STOP-SIGN-RED, OTHER-RED, ORANGE, BROWN, BLUE, and OTHERS

The result shows that split the pixels into more classes does not help in improving the accuracy.

4.1.2. Learning Algorithms

In the experiment, we tested 4 different learning algorithms: logistic regression (for 2 classes), one vs all logistic regression, K-ary logistic regression, and Gaussian naive Bayes. The result shows that Gaussian naive Bayes achieves the best performance in terms of accuracy, precision, and recall. Kary logistic regression outperforms one vs all logistic regression. Moreover, when only two classes are used, binary logistic regression has similar performance as one vs all logistic regression and Kary logistic regression. The reason is that multiclass logistic regression would be equivalent to binary one when there are only 2 classes.

4.1.3. Color Space as Features

In the experiment, we compared the performance of learning algorithms using pixel values in different color spaces. We tested 3 different color spaces: RGB, HSV, and YCrCb. The result shows that learning algorithms on HSV features perform worst. Since the value of the H channel of red pixels is not linear, which are distributed at two ends of the range, either close to 0 or 180. It makes it difficult for a linear classifier to fit the data correctly.

We also tried to combine values from distinct color spaces to form a larger feature vector. The hypothesis is that the classifier would perform better if more information is provided in the data. However, the result shows that this approach does not help in this application. The classifiers do not take advantage of extra information when distinguishing colors.

4.2. Stop Sign Detection Performance

Based on the result shown in Sec.4.1, Gaussian naive Bayes classifier with 3 classes is used in the image segmentation. Total 1,200,000 pixels uniformly chosen from training images are used to train the classifier.

Fig.4.2 and Fig.4.2 shows the shows the stop sign detection results of stop signs viewed in different angles and occlusion. The green boxes in the original images are detected stop signs, and blue circles and curves in the convex images are corners and edges of approximated polygons. The results shows that the proposed approach can successfully detect stop signs in different angles and occlusion.

Fig.4.2-4.2 are stop sign detection results from 8 different test images. Our approach is able to detect stop signs in different environmental conditions.

5. ACKNOWLEDGEMENT

The labeled data set is a joint work of following people:

- A53295675 Yunhsiu Wu
- A53308491 David Lu
- A53306916 Minhsueh Cheng
- A53295319 Yu-Tsun Yang
- A53317226 Sheng-Wei Chang
- A53316963 Chun-Yen Liou
- A53263730 Jui-Te Lin
- A53298402 Ya-Hsiu Hsieh
- A53314199 Chun-Nien Chan
- A53306080 Yu-Hao Liu

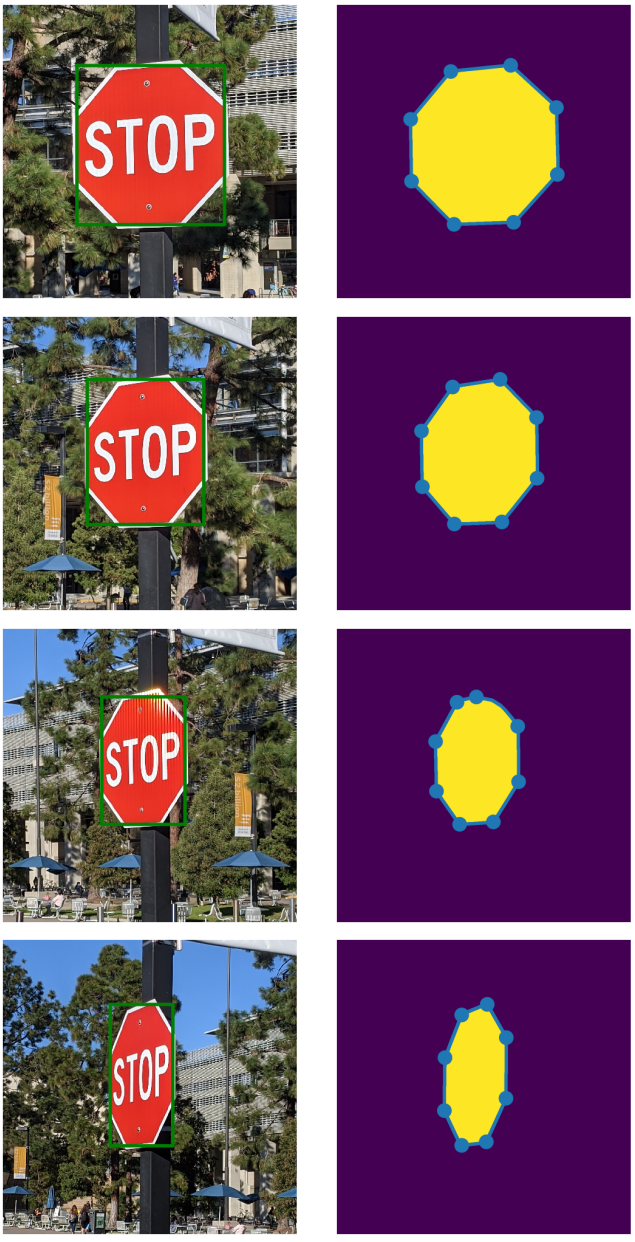
The training data includes 100,000,000+ labeled pixels from 197 non-enhanced images. Each pixel is assigned one of six color labels: STOP-SIGN-RED, OTHER-RED, ORANGE, BROWN, BLUE, and OTHERS. Thanks to people aforementioned for assuring the quality of label data, designing the format for easily use in python code, and labeling data.

6. REFERENCES

[1] Xiaolong Wang, Abhinav Shrivastava, and Abhinav Gupta, "A-fast-rcnn: Hard positive generation via adversary for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2606–2615.

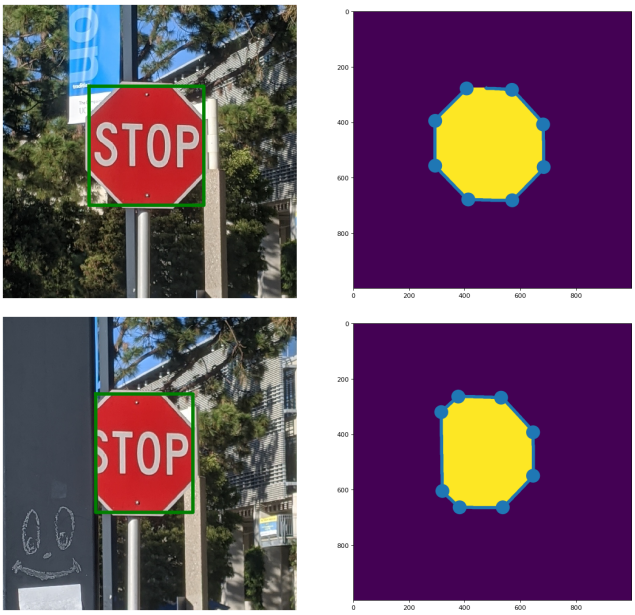
Learning Algorithm	Features	# of Class	Accuracy	Precision	Recall
Logistic Regression	RGB	2	$98\% \pm 0\%$	$93\% \pm 1\%$	$92\% \pm 2\%$
One-vs-All Logistic Regression	RGB	2	$98\% \pm 0\%$	$93\% \pm 1\%$	$92\% \pm 2\%$
K-ary Logistic Regression	RGB	2	$97\% \pm 1\%$	$93\% \pm 5\%$	$92\% \pm 7\%$
Gaussian Naive Bayes	RGB	2	$98\% \pm 0\%$	$92\% \pm 0\%$	$95\% \pm 0\%$
Logistic Regression	RGB	3	-	-	-
One-vs-All Logistic Regression	RGB	3	$81\% \pm 11\%$	$53\% \pm 38\%$	$89\% \pm 41\%$
K-ary Logistic Regression	RGB	3	$96\% \pm 6\%$	$88\% \pm 24\%$	$93\% \pm 11\%$
Gaussian Naive Bayes	RGB	3	$98\% \pm 0\%$	$93\% \pm 0\%$	$93\% \pm 0\%$
Logistic Regression	RGB	6	-	-	-
One-vs-All Logistic Regression	RGB	6	$95\% \pm 7\%$	$83\% \pm 24\%$	$94\% \pm 8\%$
K-ary Logistic Regression	RGB	6	$97\% \pm 2\%$	$95\% \pm 4\%$	$86\% \pm 16\%$
Gaussian Naive Bayes	RGB	6	$98\% \pm 0\%$	$94\% \pm 0\%$	$93\% \pm 0\%$
Logistic Regression	HSV	2	$79\% \pm 21\%$	$44\% \pm 58\%$	$56\% \pm 76\%$
One-vs-All Logistic Regression	HSV	2	$80\% \pm 18\%$	$46\% \pm 62\%$	$53\% \pm 78\%$
K-ary Logistic Regression	HSV	2	$77\% \pm 21\%$	$36\% \pm 46\%$	$76\% \pm 77\%$
Gaussian Naive Bayes	HSV	2	$95\% \pm 0\%$	$84\% \pm 0\%$	$88\% \pm 0\%$
Logistic Regression	HSV	3	-	-	-
One-vs-All Logistic Regression	HSV	3	$75\% \pm 25\%$	$50\% \pm 43\%$	$80\% \pm 49\%$
K-ary Logistic Regression	HSV	3	$82\% \pm 12\%$	$61\% \pm 36\%$	$51\% \pm 72\%$
Gaussian Naive Bayes	HSV	3	$95\% \pm 0\%$	$81\% \pm 0\%$	$88\% \pm 0\%$
Logistic Regression	HSV	6	-	-	-
One-vs-All Logistic Regression	HSV	6	$80\% \pm 15\%$	$39\% \pm 49\%$	$55\% \pm 64\%$
K-ary Logistic Regression	HSV	6	$77\% \pm 21\%$	$52\% \pm 41\%$	$71\% \pm 50\%$
Gaussian Naive Bayes	HSV	6	$95\% \pm 0\%$	$84\% \pm 0\%$	$87\% \pm 0\%$
Logistic Regression	YCrCb	2	$96\% \pm 2\%$	$91\% \pm 5\%$	$86\% \pm 19\%$
One-vs-All Logistic Regression	YCrCb	2	$97\% \pm 0\%$	$90\% \pm 1\%$	$90\% \pm 1\%$
K-ary Logistic Regression	YCrCb	2	$97\% \pm 0\%$	$90\% \pm 2\%$	$90\% \pm 2\%$
Gaussian Naive Bayes	YCrCb	2	$98\% \pm 0\%$	$92\% \pm 0\%$	$95\% \pm 0\%$
Logistic Regression	YCrCb	3	-	-	-
One-vs-All Logistic Regression	YCrCb	3	$74\% \pm 24\%$	$73\% \pm 28\%$	$98\% \pm 1\%$
K-ary Logistic Regression	YCrCb	3	$95\% \pm 5\%$	$87\% \pm 20\%$	$83\% \pm 37\%$
Gaussian Naive Bayes	YCrCb	3	$98\% \pm 0\%$	$93\% \pm 0\%$	$93\% \pm 0\%$
Logistic Regression	YCrCb	6	-	-	-
One-vs-All Logistic Regression	YCrCb	6	$80\% \pm 15\%$	$64\% \pm 58\%$	$62\% \pm 68\%$
K-ary Logistic Regression	YCrCb	6	$95\% \pm 3\%$	$89\% \pm 18\%$	$81\% \pm 25\%$
Gaussian Naive Bayes	YCrCb	6	$98\% \pm 0\%$	$94\% \pm 0\%$	$93\% \pm 0\%$
Logistic Regression	RGB+HSV+YCrCb	2	$98\% \pm 0\%$	$93\% \pm 2\%$	$93\% \pm 3\%$
One-vs-All Logistic Regression	RGB+HSV+YCrCb	2	$98\% \pm 0\%$	$94\% \pm 1\%$	$94\% \pm 1\%$
K-ary Logistic Regression	RGB+HSV+YCrCb	2	$98\% \pm 0\%$	$94\% \pm 1\%$	$93\% \pm 1\%$
Gaussian Naive Bayes	RGB+HSV+YCrCb	2	$95\% \pm 0\%$	$78\% \pm 0\%$	$98\% \pm 0\%$
Logistic Regression	RGB+HSV+YCrCb	3	-	-	-
One-vs-All Logistic Regression	RGB+HSV+YCrCb	3	$87\% \pm 11\%$	$69\%41 \pm \%$	$83\% \pm 60\%$
K-ary Logistic Regression	RGB+HSV+YCrCb	3	$98\% \pm 0\%$	$94\% \pm 2\%$	$93\% \pm 2\%$
Gaussian Naive Bayes	RGB+HSV+YCrCb	3	$97\% \pm 0\%$	$88\% \pm 0\%$	$97\% \pm 0\%$
Logistic Regression	RGB+HSV+YCrCb	6	-	-	-
One-vs-All Logistic Regression	RGB+HSV+YCrCb	6	$86\% \pm 18\%$	$65\% \pm 50\%$	$89\% \pm 18\%$
K-ary Logistic Regression	RGB+HSV+YCrCb	6	$96\% \pm 4\%$	$88\% \pm 19\%$	$93\% \pm 10\%$
Gaussian Naive Bayes	RGB+HSV+YCrCb	6	$98\% \pm 0\%$	$91\% \pm 0\%$	$95\% \pm 0\%$

Table 1. Performance of image segmentation using learning algorithms



(a) Original image (b) Convex image

Fig. 2. Stop sign detection: various angles



(a) Original image (b) Convex image

Fig. 3. Stop sign detection: occlusion

(a) Bounding box (b) Segmentation result

Fig. 4. Stop sign detection: Sample 1

(a) Bounding box (b) Segmentation result

Fig. 5. Stop sign detection: Sample 2

(a) Bounding box

(b) Segmentation result

Fig. 6. Stop sign detection: Sample 3

(a) Bounding box

(b) Segmentation result

Fig. 7. Stop sign detection: Sample 4

(a) Bounding box

(b) Segmentation result

Fig. 8. Stop sign detection: Sample 5

(a) Bounding box

(b) Segmentation result

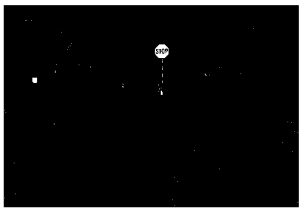
Fig. 9. Stop sign detection: Sample 6

(a) Bounding box

(b) Segmentation result

Fig. 10. Stop sign detection: Sample 7

(a) Bounding box



(b) Segmentation result

Fig. 11. Stop sign detection: Sample 8

[2] Fan Yang, Wongun Choi, and Yuanqing Lin, "Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2129–2137.

[3] Rafael C Gonzalez, "Richard e. woods," *Digital image processing*, vol. 2, pp. 550–570, 2002.