Stop Sign Detection

Shuang Li

Cpr E 575 Spring 2018

April 26, 2018

Content

Abstract

In this paper I will talk about the initial steps I took to approach the problem of detecting stop signs using MATLAB, as well as three different proposed algorithms, point feature matching, HAAR feature cascade training, and Convolutional Neural Networks combining with Color Transformation (not implemented). I will compare the result for each algorithm and analyze their advantages and disadvantages.

1. Introduction

An automatic traffic sign detection system can assist drivers on better understanding their surrounding situations through reading the information from every traffic sign the vehicle has previously encountered. It can help reduce the traffic accident rates by helping drivers remember useful information such as speed limits, warnings for possible dangers, and more, therefore improving driving quality and safety. This system can also be utilized in assisting drivers with disabilities, developing autonomous vehicles, and more.

Nowadays, Traffic Sign Detection is considered a well-researched topic in Computational Perception and Image Processing. Many algorithms are used or developed for this purpose. Some of the popular ones include Support Vector Machines (SVM), Histogram of Gradient (HOG), convolutional neural network [1], Joint Transform Correlator [3], and Color Segmentation [5]. In this project, I will try out some basic image processing approaches using MATLAB and OpenCV, as well as other higher-level machine learning algorithms to improve the accuracy and efficiency of the detection.

2. Statement of Problem

For detection purpose in this project, all US traffics signs are divided into four categories: warning signs, temporary traffic control signs, regulatory signs, and other signs. Warning signs consist of a yellow background, a black border around the sign, and a primary symbol at the middle representing different content of each sign. Temporary traffic control signs consist of an orange background, a black border around the sign, and a black symbol or texts at the middle. Regulatory signs do not have a uniformed appearance, but some commonly seen ones are stop sign, yield sign, No xxx sign, and speed limit signs. The first three either have a red background or a red forbidden sign that could be used for detection. Speed limit signs only consist of black and white colors with the number in the middle indicating the speed limit on it. For this project, stop signs will be used to test the algorithms because of two reasons. First, red background with white words makes it relatively more distinguishable from the surrounding environment compare to other signs, and second, it is the best represented type of signs in the dataset.

Above all, traffic signs all consist of a big portion of bright colors on it. It attracts drivers' attention and informs drivers about their current situations in an efficient way. This also makes them stand out from the background. Although in the real world there are many factors that could affect the actual color of the signs received by the camera, detecting signs by detecting a big group of pixels consisting of similar colors in an image is probably still the most effective way to start this project. This special feature of traffic signs offers me a starting point to develop some basic techniques for traffic signs detection. However, it is necessary to consider other changeable factors in real world and make improvements to the current system as the project proceeds. Later in this project I attempted to consider those factors by using different machine learning algorithms including HAAR Cascade Classifier Training and Convolutional Neural Network(CNN).

In related work section I will talk about my experience of using basic image processing techniques in MATLAB to attempt to solve this problem. Then I will talk about some limitations that I have found out throughout the process. Later in Proposed Method and Experimental Result section I will talk about my attempts to adopt higher level machine learning techniques such as HAAR Cascade Classifier Training and Convolutional Neural Networks to increase the accuracy of the results. This project will serve as a learning experience for me to further explore different algorithms in machine learning and a starting point for me to pursue my interest in the field of Artificial Intelligence.

3. Related Work

3.1 Color detection using MATLAB

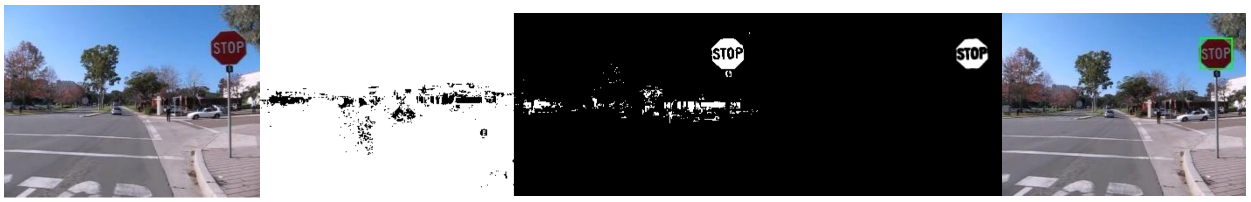
When trying to detect the stop sign, I used basic thresholding for three channels in MATLAB separately.

Fig. 1 includes two examples that use normal structuring element object (ones(n)) dilation.

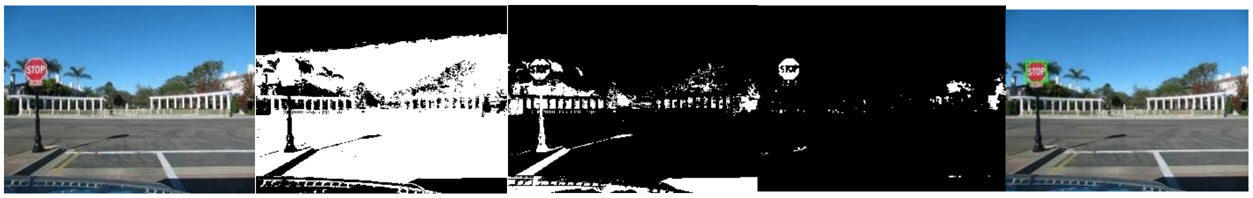
Fig. 2 below shows one example that needs to have manually modified structuring element object (eye(n)) dilation in order to successfully detect the sign.

1(a) is the original image; 1(b) shows all pixels that have a value bigger than 40 in red channel; 1(c) shows all pixels that have a value less than 30 in green channel; 1(d) is the logical sum of 1(b) and 1(c) after noise elimination; and 1(e) is the original image with a bounding box drawn based on the result of 1(d).

1(f)-(j) used the same techniques as demonstrated by (a)-(e) with different original image. 1(g) and 1(h) both used threshold value of 70 (red>70, green<70).

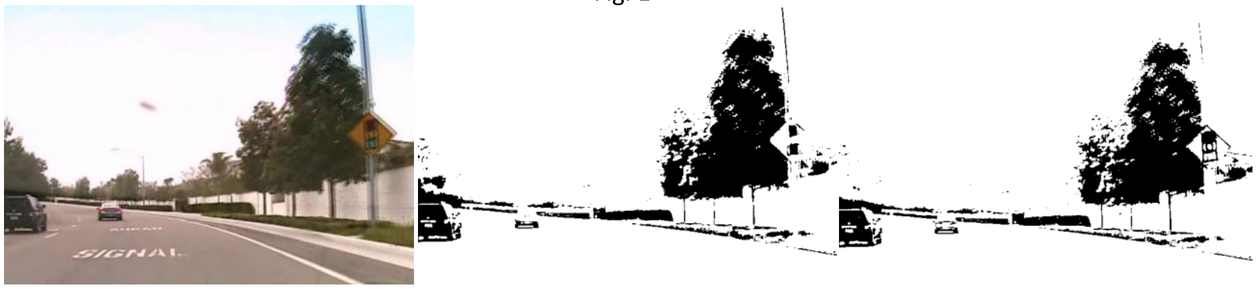


(a) (b) (c) (d) (e)

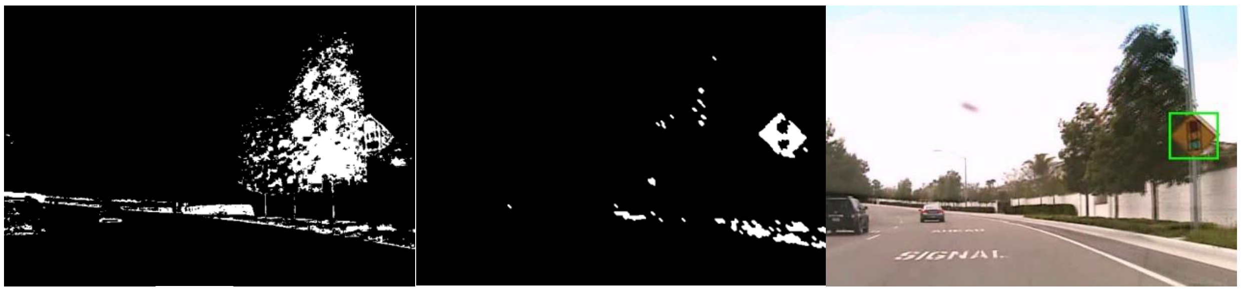


(f) (g) (h) (i) (j)

Fig. 1



(a) (b) (c)



(d) (e) (f)

Fig. 2

2(a) is the original image; 2(b), (c), (d) are the red channel, green channel, and blue channel threshold by certain threshold values tested manually; 2(e) is the logical sum of the three modified by a specific kind of dilation that was found manually; and 2(f) is the original image with the bounding box drawn based on the result of 2(e).

Limitations:

1. Thresholding values and eroding/dilating objects must be manually set up based on brightness, angle, and other changing aspects of the image.
2. This detection is only accurate when the sign is the biggest red object in the image. Surroundings can easily affect detecting results. It has a high possibility of getting false positive. So, there will be an issue when detecting signs far away from the camera. They will appear to be small in the image obtained hence this method will not function as expected.

Possible improvements:

1. Use color contrast (difference of the values of the three channels) to compute possible region of interests [9].
2. Detect Shape and Color at the same time.
3. For different lighting conditions, Color Segmentation techniques with images prepared by running through Histogram Equalization could be a potential solution for this problem [7]. The color of certain area of the image can be better enhanced using these techniques to increase the accuracy of detections in different lighting conditions and visibilities.
4. Could consider using color spaces other than RGB to make the results more accurate.

3.2 Shape Detection using MATLAB

First, I took a binary image and found the boundary of each object. Then I used the difference of the relationship between their area and perimeter of each shape to calculate its possibility of being a certain shape(metric).

compute the roundness metric: metric = 4*pi*area/perimeter^2; [10]

compute the squareness metric: metric = 16*area/perimeter^2;

compute the octagoness metric (for stop signs):

area = 3*(a^2) (see Fig. 3), perimeter = 8*a,

hence metric = (64/3)*area/perimeter^2;

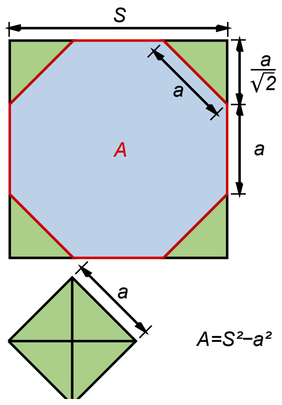
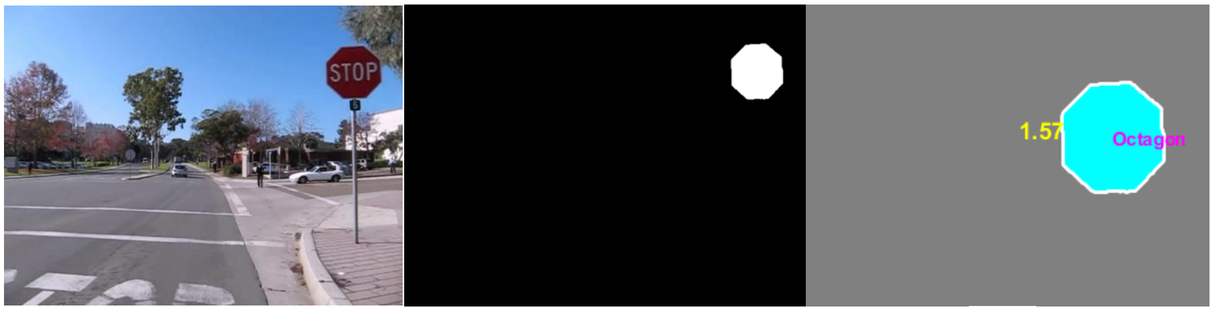


Fig. 3 Area of Octagon [11]

For the example shown in Fig. 4, 4(a) is the original image; 4(b) is 1(d) after holes being filled; and 4(c) is the image that's derived from the formula explained above(also zoomed in): taking the binary image 4(b), finding the boundaries of the object, calculating the perimeter and the area of the object, then plugging into the formula to get the octagoness metric (the yellow number on 4(c)). I compared the result number with the threshold value (1 in this case). If it is greater than the threshold value, it is marked as an octagon. In 4(d)-(f), due to massive background noises and over dilation, many false positives were detected.



(a) (b) (c)

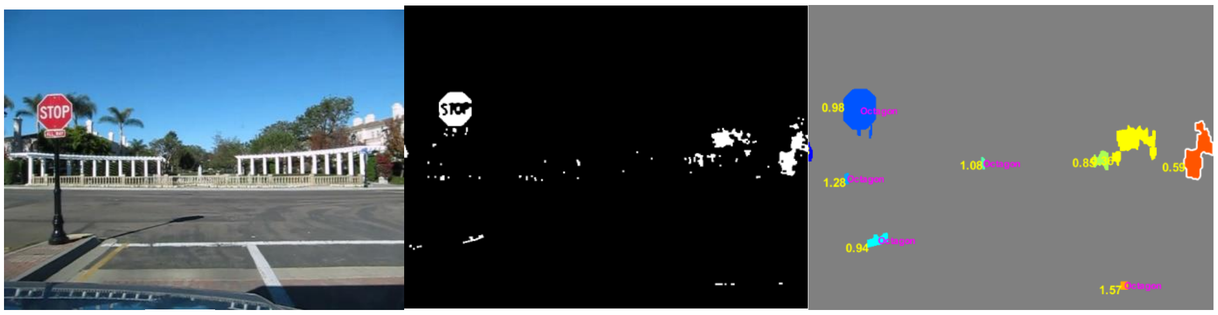

(d) (e) (f)

Fig. 4

Limitations:

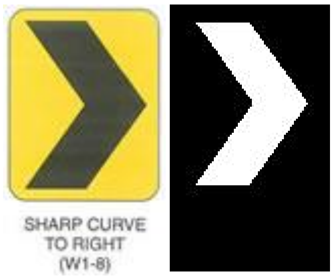
1. Must have perfect binary image for this algorithm to detect the shape. Could be easily affected by other noises in the background. (hence image from Fig. 2 does not work here)
2. Shapes must be **regular** polygons or circles on binary image, this algorithm will not be able to detect signs in pictures blurred or taken from different angles.

Possible Improvements:

1. Use other algorithms to eliminate noises from background.
2. Use math formula to calculate the possibility of certain object in the binary image being the shape wanted but in a different angle.
3. Assign probability to each pixel or each group of pixels from the result of both color detection and shape detection to calculate the probability of existing a sign in certain area. This can improve the accuracy when using two techniques simultaneously.
4. For motion blurred images, there are a lot of mathematical approaches that are developed to restore blurred images. I plan to explore this algorithm that is based on border deformation detection in the spatial domain [8].

3.3 Use templates to recognize different meaning of the traffic signs

After Stage 1 is mostly completed, I will be able to detect potential region in an image that has a high possibility of containing a traffic sign. In this stage, I will also use basic image processing techniques that are similar to the ones in Stage 1 but with more detailed templates extracted from other images or obtained from other sources. With binary templates, I should be able to identify regions in an image that contain similar graphics as the templates, such as 'sharp curve to right/left', 'merging traffic', 'railroad crossing ahead', etc. In Fig. 5, 5(a) is an image of the original sharp curve to right sign; 5(b) is the template extracted from 5(a) using image erosion to eliminate noises.



(a) (b)

Fig. 5: templates extraction

Potential problems that I may encounter for this stage is that the binary templates I have are all in fixed sizes. It will be hard to use these templates on images with different sizes. However, from

Stage 1 I should be able to find region of interest prior to templates matching. From there I will need to find (or develop) an algorithm that can proportionally change the size of the templates based on the size of the region of interest. This way the accuracy of this detection will be greatly improved. And it will be applicable for much more images than the group of images with a fixed size.

Instead of resizing the templates, another way to matching symbols with different sizes is to preprocess input images. I can extract the bounding boxes drawn on each image and match them with templates to reduce processing time.

4. Dataset obtained

In this project I will be using the LISA Traffic Sign Dataset [6]. It contains 47 types of US traffic signs with over a thousand images for pedestrian crossing, stop, and signal signs separately. The resolution of the images ranges from 640x480 to 1024x522. This dataset contains both color and grayscale images, which is not very well represented when working with color detection but will not affect the project when using shape-based detection algorithms.

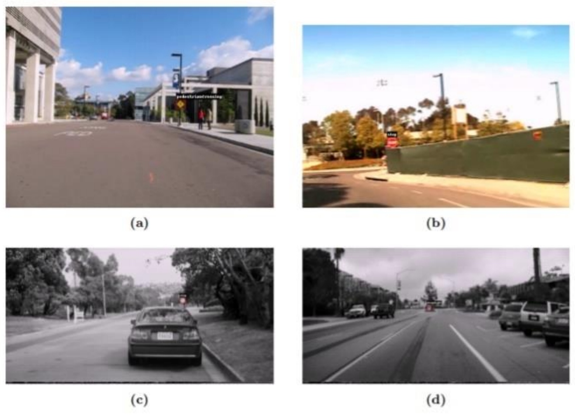


Fig. 6

Images from this dataset are all taken in real world situations. Many images are blurred (Example see Fig. 2(a)) or do not have a clear color contrast due to different reasons, which may cause some difficulties when being used.

For this project, I will mainly use the images containing stop signs for testing purposes. This dataset contains 1716 stop signs in total and is enough number of testing cases for this project.

Fig. 6: Randomly chosen examples of annotated signs from the LISA dataset.

5. Proposed Method

5.1 MATLAB Point Feature Matching

5.1.1 Speeded Up Robust Features (SURF)

SURF can be used to track objects and extract points of interests. It works without being affected by scale change or in-plane rotation but does not work well with uniformly-colored objects.

5.1.2 Implementation

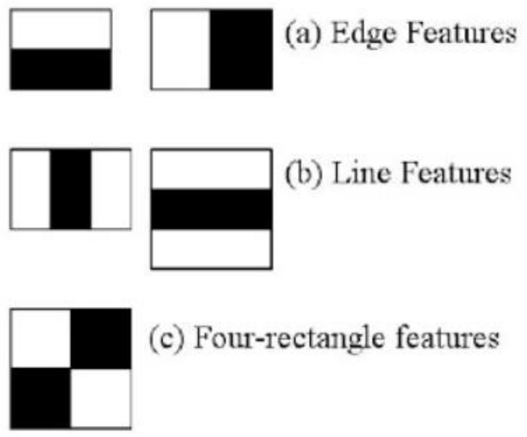
In this implementation, I used MATLAB built in function `detectSURFFeatures()` to find the SURF of the stop sign and match the features with testing pictures.

The process includes:

1. Read both object image and scene image to gray scale
2. Detect feature points using detectSURFFeatures() on both images
3. Extract feature descriptors at the interest points in both images
4. Match the features using their features using function matchFeatures()
5. Localize the object in the scene using putative matches with function estimateGeometricTransform()

5.2 HAAR Cascade Classifier Training

5.2.1 Concept



The algorithm takes lots of positive images (images with stop signs) and negative images (images without stop signs) to train the classifiers. Image on the left shows a basic idea of HAAR features. “Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.” [17]

From method described above, many features are calculated (around 160,000 for a 24x24 image). And among them the majority is irrelevant. Hence, we need to select features that can best represent the classifier. This is achieved by Adaboost.

To select the features that are more relevant to the aiming object, the algorithm takes each feature and test them on the training images. The features with minimum error rates during the testing are selected. These features are called weak features, because each of them cannot classify images alone, but together they make a very strong classifier. 200 features can achieve an accuracy of 95% when doing face detection.

Even after decreasing the number of features, it is still very time-consuming to apply all features to every fixed-size window of an image. To solve this problem, they introduced the concept of Cascade of Classifiers. The idea is to group the features to several stages. The first few stages contain fewer features than later stages. Then apply each stage to the window in order. If the window failed in the first stage, it will be disregarded so not all features are applied on it. This makes the classifier much more efficient.

*This section is a simplified explanation of the concept. For more details check out the paper [17] in resources section.

5.2.2 training

Preprocessing of the images is implemented on my PC and the actual training process is completed on a Linux Virtual Private Server purchased from Digital Ocean.

5.2.2.1 Training images collecting

In the training process two sets of images are needed, positive and negative.

For negative images, I went to the website image-net.org and downloaded the athlete, people, and car three categories of pictures with the assumption that they will not contain stop signs. After eliminating the ugly images (placeholders for expired images), converting them to grayscale, and resize them to the same size, 100x100, I obtained 2101 negative samples.

I used OpenCV create_samples to create positive samples using the negative samples and a single positive image. This function superimposes the positive image onto the negative images with different angles and other factors and output the images as well as an info file containing information about all positive samples generated. Here is an example of what is in the info file: "`0001_0014_0045_0028_0028.jpg 1 14 45 28 28`", where the "`0001_0014_0045_0028_0028.jpg`" is the positive sample file name, "1" is how many objects the image contains, followed by the x, y coordinates of the object in the image and the width and height of it. After that I generated a vector file for the training later also using OpenCV create_samples function.

5.2.2.2 Parameters Selection

Usually resulting model is recommended to have a size of less than 50x50 as the number of features grow exponentially when the size increases. I decided to train the result model in a 30px width and 30px height size considering the RAM and Disk I have on the server.

5.2.3 Results Evaluation

The way I evaluated the result is to let the program output and save all images that are detected as containing a stop sign to a folder. Then I manually checked every positive image to see if the detection result was correct. This method is very time consuming, but I was not aware of any other method that can be used at the time.

For experiment results and potential improvement analysis of this method see section 6.2

5.3 Combination of Machine Learning Algorithms (did not implement due to time constraint)

In this part, I will use cited work [2] as a reference and talk about the process of replicating the work done in this paper. Fig. 7 and 8 provides a general outline of what steps are taken in the process.

5.3.1 Color Transformation

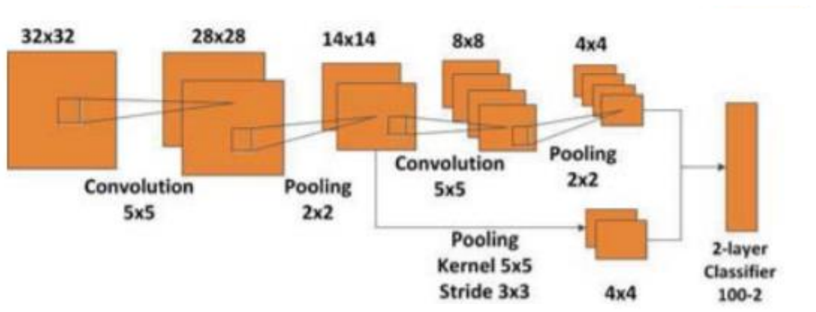


Fig. 8: CNN composition

As shown in Fig. 7, This algorithm will first process the original images to transform them from RGB to grayscale. Instead of manually setting threshold value as mentioned in 1.1 to solve the problems of lighting condition, weather condition, or natural fade, this algorithm uses Support Vector Machine(SVM) to avoid the sensitivity to color differences in different lightening conditions. First, they extract pixels from training data, then classify them as positive or negative. For example, in Fig. 5(a), the yellow background would be positive, and all other pixels are negative pixels. After that they train a classifier [12] and use the offset value as the map between RGB and grayscale value. The resulting gray scale images will be fed into CNN in later steps.

5.3.2 Convolutional Neural Networks (CNN)

The images processed then will go through several layers in the CNN. There are two types of layers in this network. The first one is fixed layer, containing only one layer. It takes the grayscale image processed by color transformation and find the potential regions of interest (Fig. 9). The second group of layers, learnable layers, is used to recognize the specific features for the classifier to find certain types of traffic signs. As shown in Fig. 8, there will be two learnable layers and results from both of them will be fed into the classifier to improve accuracy of the algorithm.

This algorithm is developed to distinguish different types of signs from each other. The training process includes separate trainings for each type of signs. In their paper they chose to include four types of signs: “Prohibitive”, “Danger”, “Mandatory”, and other. For simplicity, in this project I will start with one specific kind of regulatory sign, the stop sign, and proceed to other kind of warning signs depending on the time remaining for this project.

During their training process, they used two classes of training data: 1-class contains images that have traffic signs, and 0-class contains all images that have no traffic signs present. I will use the same training method to replicate their results. The goal for this stage is set to complete the replication of this project, but my main focus will be to understand how the algorithm covered in this paper actually works.

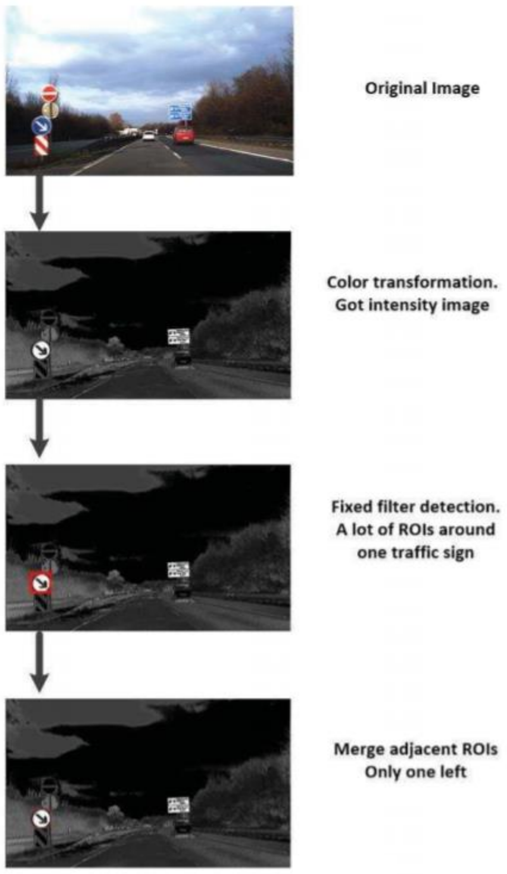
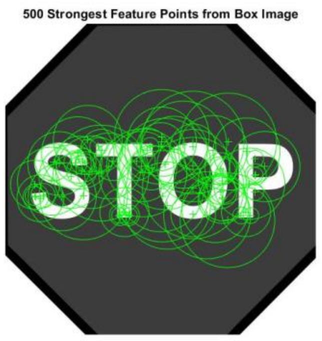

Original Image
Color transformation. Got intensity image
Fixed filter detection. A lot of ROIs around one traffic sign
Merge adjacent ROIs Only one left

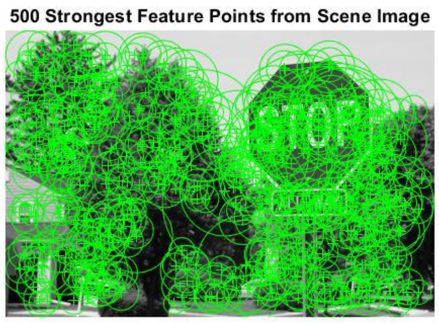
Fig. 9

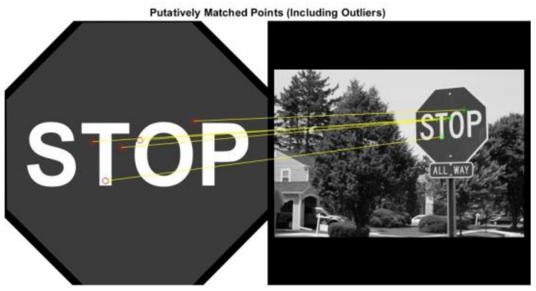
6. Experimental Result

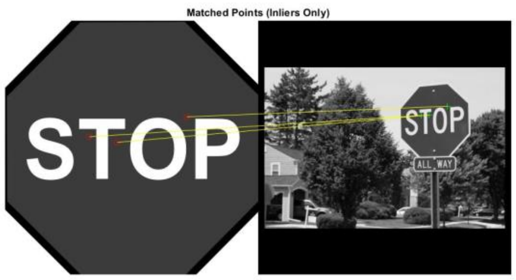
6.1 MATLAB Point Feature Matching

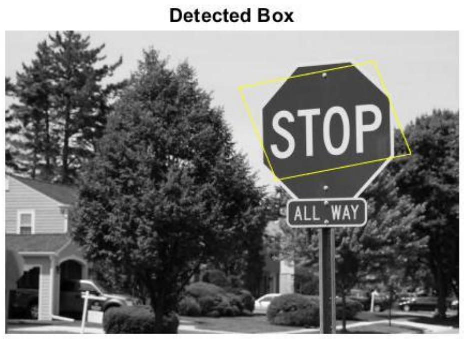
This method was tested on 3 different images containing stop signs, and the one shown below is the only one that is successful. From the result, this method is not suitable for stop sign detection as stop signs containing relatively simple patterns and the images are sometimes blurry and this method is not successful in detecting images that has insufficient details, for example, low resolution or blurry images.











6.2 HAAR Feature cascade training

I used the 1716 images containing stop signs from the dataset to test the HAAR cascade I trained. From the 0-14 stages, I tested cascades contain 9, 10, 11, 12, 13, 14, 15 stages of training.

Number of Stages	Detected as positive	Correctly detected	percentage of correct/total	percentage of correct/positive
9	1416	26	1.52%	1.84%
10	931	10	0.58%	1.07%
11	603	5	0.29%	0.83%
12	315	4	0.23%	1.27%
13	217	1	0.06%	0.46%
14	133	1	0.06%	0.75%
15	97	1	0.06%	1.03%

From the result above, this method is not successful. The best result out of the tested cascades is the cascade contains 9 stages. It detected 1416 positive out of 1716 images to have stop sign, in which only 26 are correctly detected. False positive rate is very high.

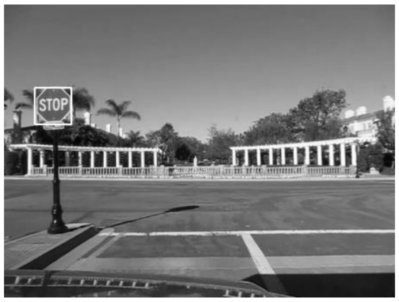
Below are some correct results from the 11-stage cascade.

1

2

3

4



5

False positive

False positive results from this HAAR cascade contains some similar features. Here are some of the examples.

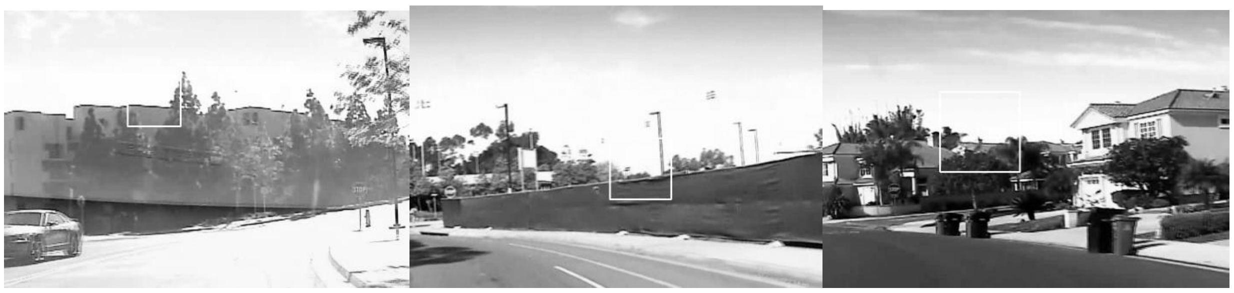


Fig. 9

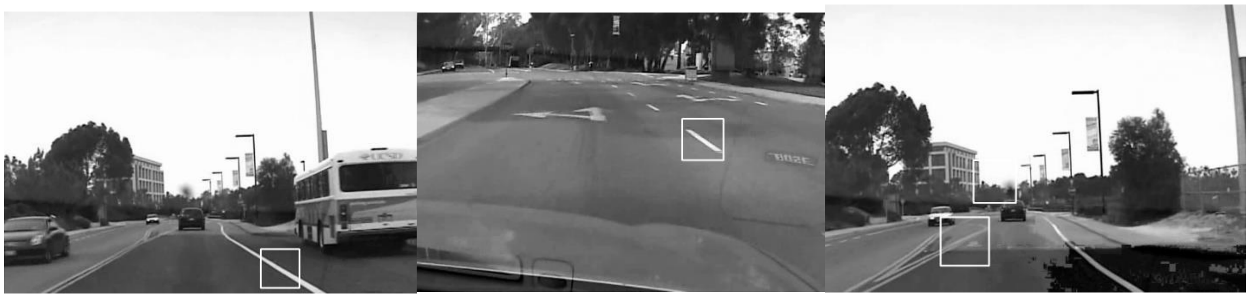


Fig. 10

For samples from Fig. 9, the top half of the boxed area are all brighter than the bottom half. For samples from Fig. 10, they all have a white line across the region. The possible reason that these areas are detected is because of the way HAAR features are defined. In section 5.2.1 I talked a little bit about the features that this algorithm is looking for. From these false positive, I can make an assumption that the HAAR feature does not work well with stop sign detections due to the way the features of an object is found.

6.3 Post-Experiment Discussion

From the results above, possible reasons that the Point Feature Matching method did not work as expected could be that the SURF does not work well with uniformly-colored objects. Since the stop sign is mostly covered with red, this might limit the number of features the algorithm can find. Hence in the future, other types of features can be used for stop sign detection.

For the HAAR Cascade Classifier training method, the possible reasons that it did not achieve the expected accuracy could be that the positive samples I used do not contain enough real-life scenarios, such as different weather, lighting conditions, etc. Also, a lot of testing images was blurry due to movement of the camera, and this training process did not take that into consideration.

7. Management Plan

7.1 Budget

This project did not require any expenses for software or hardware. Thought I used DigitalOcean Linux VPS to train the HAAR cascade, which is $0.06/hour for 6 days of use.

7.2 Weekly plan – 3/11/2018-4/19/2018

Week 1 (3/11-17)	Continue working on MATLAB basic image processing techniques
Week 2 (3/18-24)	Start to explore MATLAB built-in image processing functions and compare the testing results
Week 3 (3/25-31)	Study background knowledge needed for training HAAR feature cascaded. Understanding Convolutional Neural Networks and the algorithm itself using

	online and other resources; Implement Point Feature Matching algorithm and analyze the result
Week 4 (4/1-7)	Setting up a server for HAAR cascade training and start the training process
Week 5 (4/8-14)	Test results on HAAR cascade. Find possible reasons of unsuccess and re-train the cascade
Week 6 (4/15-19)	Complete final report using draft and sum-ups written earlier in the process

8. Conclusion

There will be plenty rooms for improvement even after the implementing all three Stages, such as continuing increasing the accuracy of detection or reducing the processing time to achieve real-time detection. Though, the main purpose of this project is for me to learn about how high-level machine learning algorithms work and to gain hands on experiences on how to operate them. In the process of completing this project I was exposed to a variety of diverse perspectives on the field of Computational Perception and Artificial Intelligence and found my interest to further explore the field. I would like to continue to finish this project on my own to get a better understanding of the algorithms that I am interested in learning but have not have the time to do so.

Acknowledgement

Thank you to the reviewers of this paper for your support!

Works Cited

[1] Zhu Z, Liang D, Zhang S, Huang X, Li B, Hu S, “Traffic-Sign Detection and Classification in the Wild”, https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Zhu_Traffic-Sign_Detection_and_CVPR_2016_paper.pdf

[2] Yihui Wu, Yulong Liu, Jianmin Li, Huaping Liu, Xiaolin Hu, “Traffic Sign Detection based on Convolutional Neural Networks”, in Proc. Int. Joint Conf. Neural Netw., Dallas, TX, USA, 2013, pp. 747–753

[3] Vishal R. Deshmukh, G. K. Patnaik, M. E. Patil, International Journal of Computer Applications (0975 – 8887) Volume 83 (No3, December 2013), “Real-Time Traffic Sign Recognition System based on Colour Image Segmentation”, http://research.ijcaonline.org/volume83/number3/pxc3892575.pdf

[4] Chia-Hsiung Chen, Marcus Chen, Tianshi Gao, “Detection and Recognition of Alert Traffic Signs”, Stanford University Artificial Intelligence Laboratory, http://ai.stanford.edu/~kosecka/final_report_final_traffic.pdf

[5] Safat B. Wali, Mahammad A. Hannan, Shahrum Abdullah, Aini Hussain, Salina A. Samad, Universiti Kebangsaan Malaysia, Malaysia, “Shape Matching and Color Segmentation Based Traffic Sign Detection System”, http://www.pe.org.pl/articles/2015/1/6.pdf

[6] LISA Traffic Sign Dataset, http://cvrr.ucsd.edu/LISA/datasets.html

[7] Hasan Fleyeh, “Traffic Signs Color Detection and Segmentation in Poor Light Conditions”, Conference on Machine VIsion Applications, May 16-18, 2005 Tsukuba Science City, Japan, https://pdfs.semanticscholar.org/2867/39c5b609156b55f27e83b0b9785c297d6810.pdf

[8] Yiliang Zeng, Jinhui Lan, Bin Ran, Qi Wang, Jing Gao, “Restoration of Motion-Blurred Image Based on Border Deformation Detection: A Traffic Sign Restoration Model”, http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0120885

[9] Chunsheng Liu, Faliang Chang, Zhenxue Chen, and Dongmei Liu, “Fast Traffic Sign Recognition via High-Contrast Region Extraction and Extended Sparse Representation”, IEEE Transactions on Intelligent Transportation Systems, Vol. 17, No. 1, January 2016

[10] “Identifying Round Objects”, https://www.mathworks.com/help/images/examples/identifying-round-objects.html

[11] Inductiveload, https://commons.wikimedia.org/wiki/File:Octagon_in_square.svg

[12] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “LIBLINEAR: A library for large linear classification,” Journal of Machine Learning Research, vol. 9, pp. 1871–1874, 2008.

[13] Ankit Sachan, “Zero to Hero: Guide to Object Detection using Deep Learning: Faster R-CNN,YOLO,SSD”, http://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/

[14] MATLAB tutorial: Object Detection in a Cluttered Scene Using Point Feature Matching, https://www.mathworks.com/help/vision/examples/object-detection-in-a-cluttered-scene-using-point-feature-matching.html#btt5qyu

[15] “Creating your own Haar Cascade OpenCV Python Tutorial”, https://pythonprogramming.net/haar-cascade-object-detection-python-opencv-tutorial/

[16] P. Viola, M. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features”, IEEE, 8-14 Dec. 2001.

[17] “Face Detection using Haar Cascades”, OpenCV, https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html