Proposal for Traffic Sign Detection Project

Shuang Li

March 2018

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.

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, while the speed limit signs only consist of black and white colors with the number in the middle indicating the speed limit on it. For all testing, the stop sign will be used first to test the algorithms due to the fact that it is the best represented type of signs in the dataset. Later I may need to take pictures from the street myself to obtain more test cases for signs other than stop signs as well, but for now, detection of other signs will be a stretch goal for this project and hence will not be discussed in detail here.

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 many varied factors 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.

Objectives

The goal for this project is to first familiarize myself with basic image processing techniques using MATLAB and OpenCV(c++), then find out the limitations of these software/techniques through the first two stages of this project. Later in stage 3 I will try to adopt higher level processing techniques to solve problems from the previous two stages. In this project, I will explore the usage of the one of the machine learning algorithm, Convolutional Neural Networks, and replicate the process of training the model as shown in paper [2]. 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.

Plan of Action

Stage 1: Use basic image processing tools and techniques to detect traffic signs and indicate their locations on the image

1.1 Color detection

Using basic thresholding for three channels separately.

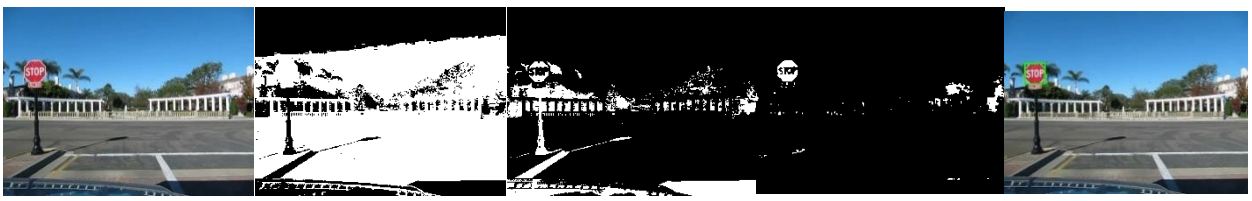
Fig. 1 shows two examples that use normal structuring element object (ones(n)) dilation.

Fig. 2 below shows one example that needs to have manually modified structuring element object (eye(n)) dilation.

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

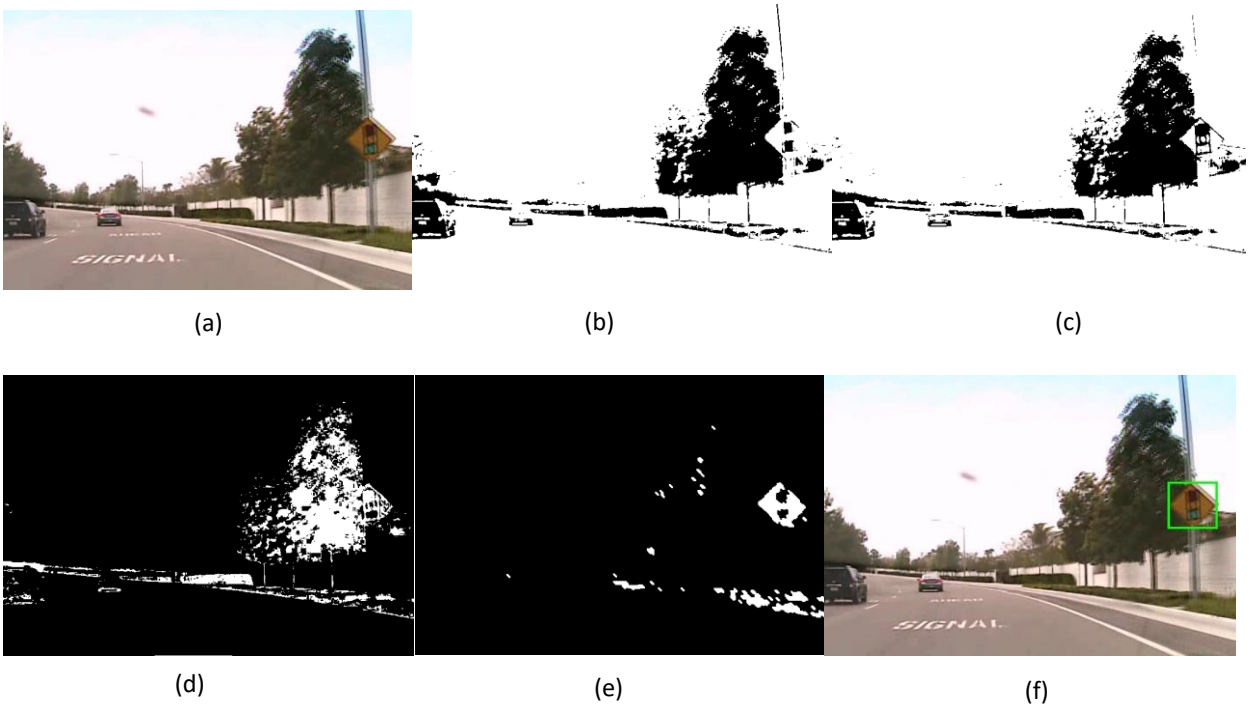


Fig. 2

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).

Problems noticed/occurred during the process:

1. Thresholding values and eroding/dilating objects have to 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 very high possibility of getting a false positive.

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.

1.2 Shape Detection

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;

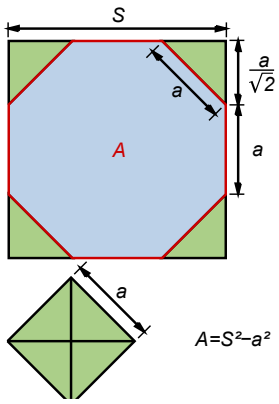


Fig. 3 Area of Octagon [11]

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 then it is marked as an octagon. In 4(d)-(f), due to massive background noises and over dilation, many false positive were detected.

(a) (b) (c)

(d) (e) (f)

Fig. 4

Problems noticed/occurred during the process:

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].

Stage 2: 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.

Stage 3: Using machines learning algorithms to further increase the accuracy of detection

After using basic image processing techniques, in Stage 3 I will proceed to explore a different approach for traffic sign detection: machine learning.

3.1 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: samples from dataset

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.

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

3.2 Machine Learning Algorithm

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

3.2.1 Color Transformation

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

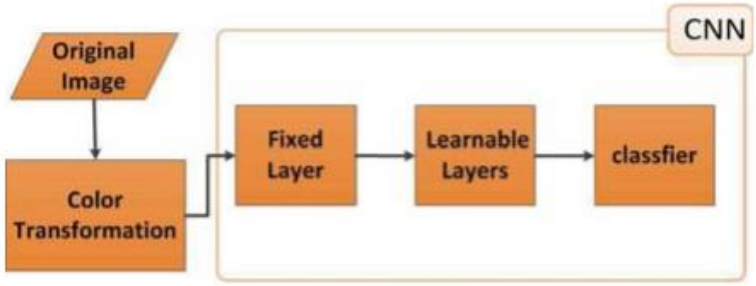


Fig. 7: Algorithm pipeline

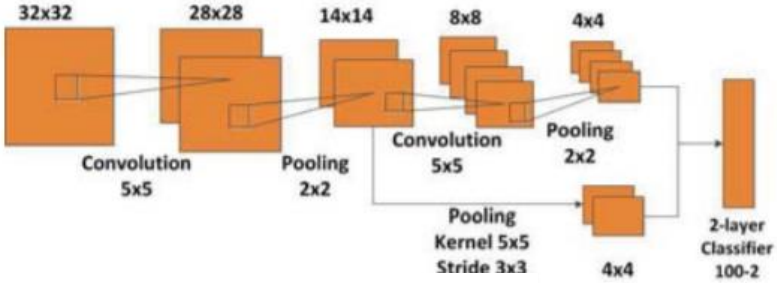


Fig. 8: CNN composition

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.

3.2.2 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.

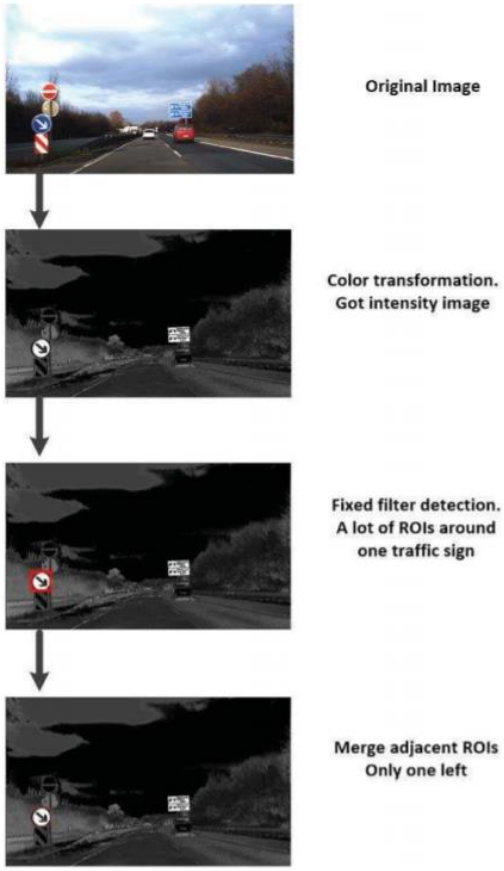


Fig. 9

Result Evaluation

With the dataset I currently have, I will divide them to 2 groups in a 3:1 scale. ¾ of the images will be used to train the network, and ¼ will be used to run tests on. The algorithm will take an array of predicted results and compare it with the actual results to output 3 percentages of correctness, false positive, and false negative. The way this project will be evaluated is through the percentage of correctness, false positive, and false negative. Success for this project is

defined as the following: partially or completely replicating the algorithm in paper [2] to achieve a 90% or higher accuracy with a 10% or lower false positive and false negative rate.

Management Plan

Proposed Budget

Presumably this project will not require any expenses for software or hardware. Though time may be taken to take images of traffic signs from street if I run out of test cases.

Weekly plan – 3/11/2018-4/19/2018

Week 1 (3/11-17)	Continue working on Stage 1.1 and 1.2; start researching relevant materials needed for implementing Stage 2
Week 2 (3/18-24)	Implement Stage 2 and write a tentative report to sum up both Stage 1 and 2
Week 3 (3/25-31)	Study background knowledge needed for understanding Convolutional Neural Networks and the algorithm itself using online and other resources; continue studying cited paper [2] and prepare necessary tools and knowledge for implementing Stage 3
Week 4 (4/1-7)	Proceed to Stage 3.1 and 3.2
Week 5 (4/8-14)	Finish and conclude relative works done for Stage 3; start drafting final report
Week 6 (4/15-19)	Complete final report using draft and sum-ups written earlier in the process

*Subject to change due to unexpected difficulties throughout the process

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. Hopefully in the process of completing this project I can be exposed to a variety of diverse perspectives on the field of Computational Perception and Artificial Intelligence, and eventually find my interest to grow as I explore further into the field.

Acknowledgement

Thank you to anonymous reviewers 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.