Object Detection for Stop Signs

John Patrick Wisnewski
Graduate Student-EECS
Digitial Image Processing 7650 Fall 2021
University of Missouri-Columbia
ppwxmc@umsystem.edu

I. Abstract

This project is to designed to show how simple object detection methods can be used to solve important problems. The implementation of this program also allows for flexibility and change as you can swap out the pretrained weights for others and use that to detect other objects so long as you have quality pretrained data. The problem in this paper is detecting stop signs in images. This is not an overly complex problem, but is a crucial aspect of full self-driving or autonomous driving. The problem with stop sign detection is that it needs to be as close to 100% accuracy as possible and, and in this paper, we will discuss factors that will cause issues with achieving this.

II. Introduction

A. Motivation

The motivation for this problem is that it is a common and real-world problem. The principle of object detection can be applied to detecting all sorts of things. Stop signs are abundant and obtaining real data is simple and fairly easy. Tesla is a leader in full-self driving and has showed how leading in technology can help grow a large and successful business.

B. Scope

The scope of this project is to test the simple algorithm on test and real-world data. The importance of accuracy in the test data is not as high as the real-world data. The test data set is designed to see the limitations and abilities of the algorithm. The real data set is designed to show how proficient it is in real world examples where it would actually be implemented.

C. Description

This program is designed around using cascading classifiers and Haar features. The goal is to use the test data to establish a base line and test a wide variety, and then implement the same program with small changes to a real data set and see if the accuracy can be improved.

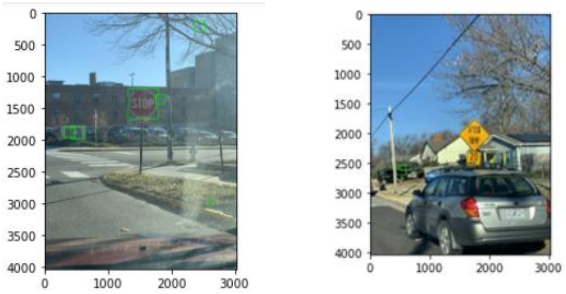
III. Approach: Algorithms and Module Implementation Details

The program is modeled after the Paul Viola paper on rapid object detection. This program was done in Jupyter Notebook using Python. The main libraries used were OpenCV, and matplotlib. The python library glob was also used for reading in the images from a folder and sorting them so that they appeared in an ordered way each time. This was mainly for organization and simplicity. The program uses cascade classifiers to detect the stop signs. the cascade classifiers are pretrained values and the program scans a box that goes across the image and finds a best match for the classifier. if it thinks it finds one then it will leave a green box around the part of the image it thinks is a stop sign [1].

[2]

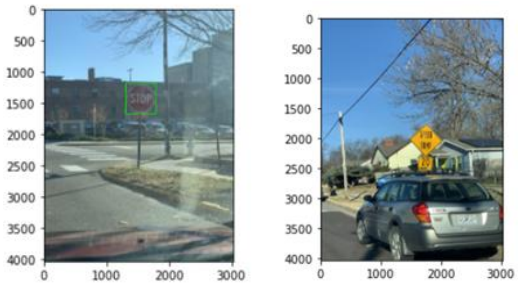
This gif shows how the box moved across the image. In the code you can change the size of the box that is used to detect features. I found that the size was very important in this program. For example, in my real data set I was using pictures that were very large, 4000x3000 pixels. I found out that using a smaller size box would cause more errors as it would detect small edges that could resemble a stop sign but were not actually a stop sign causing false positives.

IV. Experimental Results and Discussion



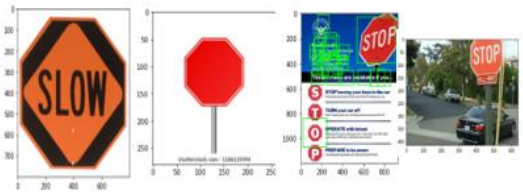
Both of these images detected stop signs in places where there were none. The minimum size of the box to find the Haar features was a 75x75 pixel box. I think that because these images are so large and the box is so small this caused the program to detect very minute edges that we cannot see with our eyes that caused these false positives.

When I increase the pixel box to 400x400, which is more relative to the size of the image we can eliminate these kinds of mistakes. The images shown below are correctly labeled as a true positive and a true negative.



It might not seem like a major change, but this could save someone's life by correctly detecting a stop sign and then completing a stop.

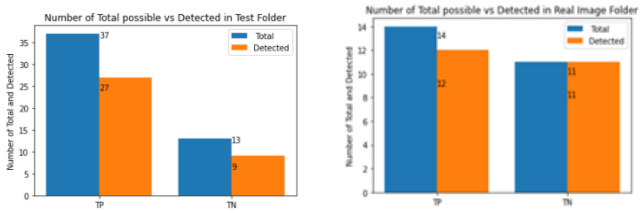
I originally tried to resize the images so that each image was the same size. This did not help the results however. With the test data set the size of the images varied greatly and so resizing them all by 1/3 for example proved to cause a lot of problems. I think that the larger the image with the more pixels the better the program can work. I tested this by consistently using the same size image in the real data set. Each image was 4000x3000. Each image was also taken on the same camera from roughly the same angle. These changes in implementation were to decrease the amount of variability. My thought process was if this program were to be implemented on a autonomous vehicle it would use the same camera to take pictures, they would have the same resolution and pixel count, and they would roughly be from the same distance and angle. So, I tried to replicate this in my real data set. I think that this is why my results were much better and consistent in my real data set opposed to the test data set which included images that were meant to give the program troubles.



Here are some examples of the images in the test image data set. They are all very different and different sizes. The first two on the left are good examples of no detections and were selected to try and fool the program, but the program succeed. The two on the right were bad examples. The image on the far right did not detect the stop sign and I think this was because the full stop sign was not in view which messed with the edge detection and the Haar features. Stop signs in the data set which were not fully included in the image did not register a true positive result. The second image from right registered the stop sign and then misfired a lot of times. I am not sure what went wrong and why it detected Haar features in those boxes, but this example would not be seen in a real-world scenario.

V. Results and Evaluation

My personal evaluation of this program is that it is not complete. In order for something like stop sign detection to be fully implemented it is going to have to be near 100% accurate. My results are promising but are not close enough to 100% that I would feel comfortable being in a vehicle driven by it.



The graph on the left is for the test data set while the graph on the right is with the real data I collected. The test data set had 37 true positives, of which 27 were detected. It also had 13 true negatives and 9 were detected in that. The real data set had 14 true positives with 12 being detected and 11 true negatives with all 11 being detected. I think that I was able to improve the results of the program by doing a few things differently. I already decided not to do resizing as that did not improve with results earlier but I included images that were all the same size originally. This then allowed me to use a larger box to detect the images. I thought that the program did a great job detecting real images which is what would be used if it were actually implemented. The two true positives that were not detected could have been avoided if they were a better-quality image. So if this program were to be implemented on an autonomous vehicle with a good camera I would expect very good results.

VI. Conclusion and Future Work

To conclude this paper, I think that the program worked as expected. Like most machine learning programs, proper data is necessary for accuracy and execution. If the data is poor you could expect the results to be poor. The data that I used was sufficient and along with quality images, good view and angle of the stop sign, and a high pixel count the program should work well. In the future I would like to use the same program to detect other objects. By changing the data.xml sheet I should be able to detect other objects. The .xml sheet has the pretrained weights, so if I had pretrained weights for another object the program should be able to detect them seamlessly. I enjoyed how the project was focused on real word applications, I was able to drive around Columbia and collect real images that were able to be detected. This an aspect I would like to explore in the future, implementing programs in my own experience.

REFERENCES

[1] Viola, P., and M. Jones. "Rapid Object Detection Using a Boosted Cascade of Simple Features." *Proceedings of the* 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, 2001, https://doi.org/10.1109/cvpr.2001.990517

[2] Bokade, Nikhil. "Face Recognition System." *Medium*, The Startup, 13 Jan.2021,https://medium.com/swlh/face-recognition-system d441e6d4b65c.

[3] Lee, Socret. "Understanding Face Detection with the Viola-Jones Object Detection Framework." Medium, Towards Data Science, 11Aug. 2021 https://towardsdatascience.com/understanding-face-detectionwith-the-viola-jones-object-detection-framework-c55cc2a9da14