Final Project Write-Up

Traffic Sign Identifier

Liam Brown 101076657
Nick Truong 101070517

Abstract:

We created a program that can train a series of Haar cascade classifiers, and use them to simply and efficiently identify traffic signs on given images. We used said program to train our own haar cascade classifiers for a selection of german traffic signs provided by the German Traffic Sign Recognition Benchmark (GTSRB for short) for testing.

Introduction:

With the advent of self driving cars, it is becoming extremely important to be able to identify surroundings, such as the road, other cars, pedestrians and traffic signs quickly. It is also important for them to be able to identify new things later on easily (such as new car models, different types of vehicles, signs). That is why we made our program able to create a series of Haar cascade classifiers for any object image set that meets the requirements of the program (GTSRB format). Haar cascade classifiers are an object detection tool based on machine learning. They are created by being fed positive and negative images (images with and without the target) which can then be used for object detection. They are quick and effective, so we use them to detect traffic signs. Furthermore, we made our program easily able to take in other Haar cascade classifiers from other sources, such as the stop sign classifier in the program. This program also identifies objects in the provided image as fast as opencv's built in functions allow.

Background:

1: Object detection using Haar-cascade Classifier,
- by Sander Soo, Institute of computer science, University of Tartu
- Gave us more understanding about how haar cascades work in our context

Approach:

Step One (Completed):
- We set up an image identification system that can identify a stop sign using an already established cascade classifier
- It works with a folder of cascade classifiers, and a folder of images to view

Step Two (Completed*):
- Set up a haar cascade program that would generate a classifier based on the GTSRB standard
- It works for any image set that follows the aforementioned standard

Step Three (Unfinished):
- Add functionality for a live camera to identify signs

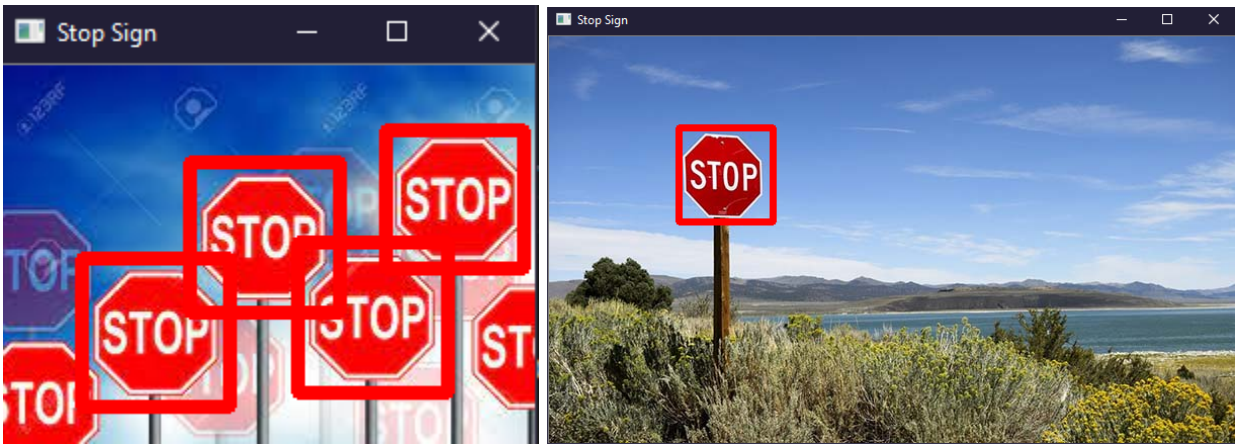
* Not as accurate as we'd like, due to time constraints

Results:

We were able to create a program that can take in a series of images and some cascade classifiers and outputs a new image with the classified object outlined.

With a proper cascade classifier, the program can effectively find the right signs on an image.

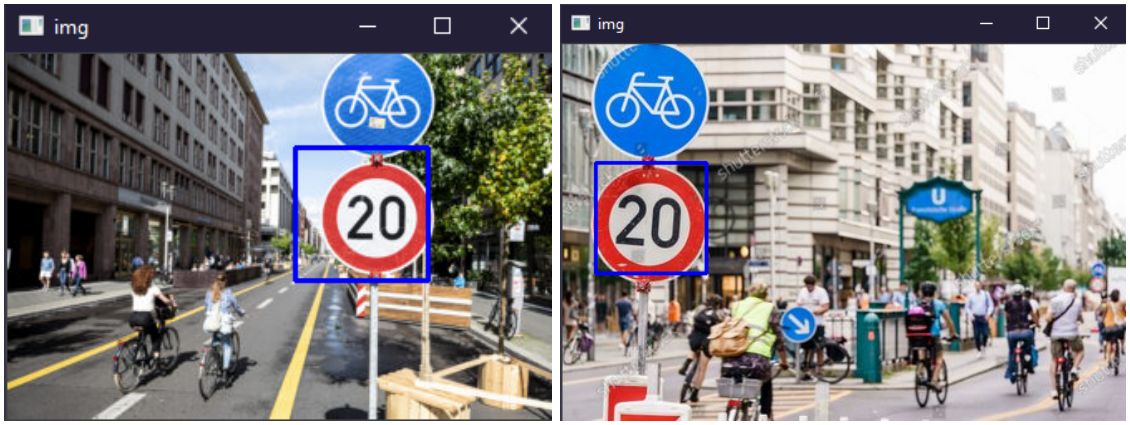
Stop Sign (default params)



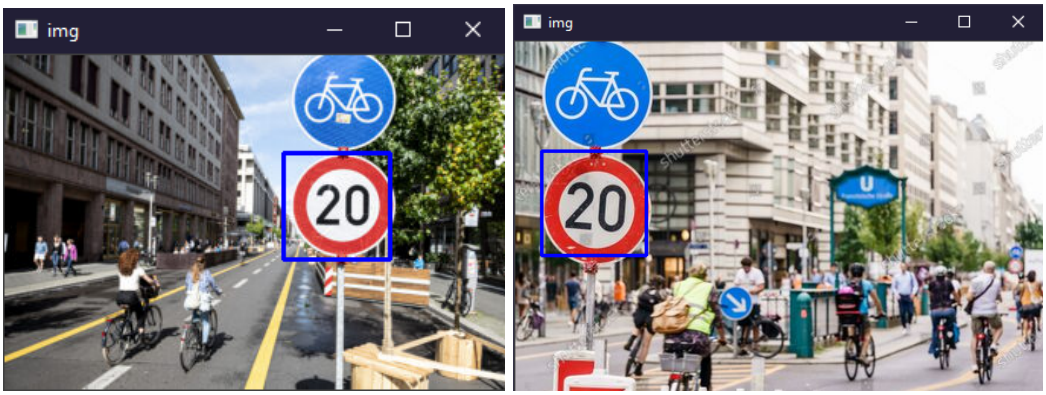
We attempted to generate cascade classifiers to identify certain signs, and had partial success.

With certain modifications to the scale factor and minimum neighbours, it can detect the sign required on a test image accurately, however, if, for example, we tested it for the 30km/hr sign, it would still identify the 20 km/hr sign as one of its own. Note that this is when applying only **one** Haar cascade classifier. The end goal has a different cascade classifier for each type of traffic sign (43 types total).

20km/hr classifier, (scaleFactor = 1.01, minNeighbours=200)



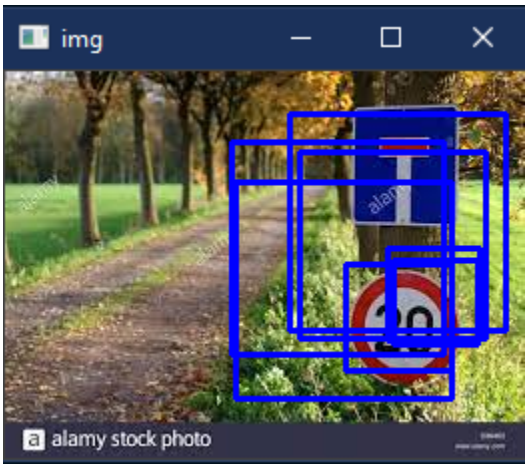
30km/hr classifier,(scaleFactor = 1.01, minNeighbours = 400)



Note: We use about 500 positive images and 250 negative images per sign, which is much less than we would have liked. This is because training would take at least a week of straight computing to complete, and because time was a big constraint, this was not feasible.

Note: Interestingly, it was only when we used above 150 minimum neighbours, and lowered the scale factor an equally extreme amount, did the program start to single out the signs properly in an image. This is very different from other cascade requirements, because they normally need a scale factor of ~1.4 and a minimum neighbours parameter of ~1-6.

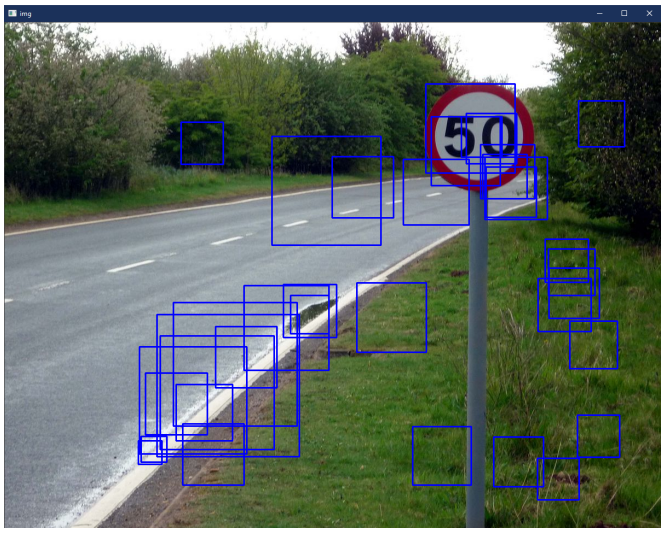
If we had gotten to train more, our accuracy would be a lot higher, and the Haar cascade classifiers would actually also be more efficient in runtime (would run faster). With more training, we would have strict/tight bounding boxes around all the traffic signs. Since we didn't train enough, the accuracy isn't perfect. While the program does tend to detect traffic signs fairly well, it also mistakes objects that are not traffic signs for traffic signs (which is a problem). This could be fixed by using a lot more negative images in the training process, and a lot more images in general. It's worth noting that the program cannot exactly identify the type of traffic sign detected (not great at telling the difference between some traffic signs), and the change above may help towards this but would likely not be a full solution for identification.



With the issues explained above and our final program running all the sign cascade identifiers at once, we end up with the following result: a program that successfully detects signs, but also detects elements that are not traffic signs (but usually contains a traffic sign in the detected region in some way). The high number of boxes is caused by the fact that the program runs many cascade classifiers

at once without a sharp enough accuracy. Thus the program can find the sign but often struggles to determine where the sign starts and ends. The saving grace is that at least one bounding box tends to be tightly fitted around the sign.

This could be resolved by training with a lot more negative images so that the cascade classifier would ignore elements that explicitly do **not** look like traffic signs (e.g. grass, dirt, etc.).



In the image above, we see a less graceful result. It seems as though we would need a lot more negative images of grass and road to make sure that the program does not confuse them when the features may be similar.

List of Work:

Equal work has been performed by both team members.

GitHub Page:

https://github.com/hoZer347/Traffic-Sign-Classifier

References:

1: OpenCV GitHub repository
- Open source
- Helped us learn how to use OpenCV for our project

2: Source for Testing Stop Sign Classifier
- GitHub Link
- stop_sign_classifier_2.xml specifically