Table of Contents

Introduction

The objective and the purpose of this project was to train an image recognition software (haar training kit) to teach the machine to recognize cars in images or live streams. Labeling the positive images and understanding the whole training process is the main goal of this project.

Methodology

This was done by downloading and installing an open-source image recognition software titled OpenCV and using images of cars to train the software. Each member identified cars in 450 images each for a total of 1800 images. Once we collected this data, we gathered all the images together and trained the machine isolating a small portion of 1800 as training data then testing our accuracy against the remaining images in testing data. The software generated the AdaBoostCascadeClassifiers for 20 stages of training. The classifiers identify whether there were vehicles in the images, in order to generate a .xml file for vehicle detection.

Training process

1. We began by creating negative images. We added 1102 images of bmp format in a “negative” folder which did not contain any images of cars. We then ran the dos command “create_list.bat” in order to get a list of negative images as well as to register them into the infofile.txt.

2. We then created positive images. We added 9 sets of 200 positive images of cars into the temp/positive/rawdata folder. Then we split up the labeling work to 450 positive car pictures per person. After finishing the work that pertained to each member, we combined all of our results into a text file titled info.txt and took one picture from each set to create a text file titled test_sample.txt for testing purposes.

3. After, we created a vector.vec for haar training. We ran the command “createsamples.exe -info positive/info.txt -vec data/vector.vec -num 1800 -w 24 -h 24” to create a vector for haartraining.

4. The training began for 20 stages. Then we ran the command “haartraining.exe -data data/cascade -vec data/vector.vec -bg negative/infofile.txt -npos 1681 -nneg 1102 -nstages 20 -mem 1000 -mode ALL -w 24 -h 24 -nonsym” to start the training process. In our training process, we labeled 1681 valid positive images of cars and 1102 negative images. We changed the npos and nneg parameters accordingly.

5. An output.xml file was generated. After training the software we generated the “AdaBoostCARTHaarClassifier” files. Next, we copied them into a folder titled cascade2xml to generate the output.xml file using the DOS command: “haarconv.exe data output.xml 24 24”. An xml file was generated.

6. We then began to implement our trained vehicle detection software to a webcam. In this instance, we were using the webcam for real-time image detection instead of using static images. We first opened the OpenCV folder and found the haarcascade data and replaced the haarcascade_frontalface_alt.xml with output.xml. Next, using Visual Studio, we changed the load file from “haarcascade_frontalface_alt.xml” to “output.xml”. Then we ran the object_detect.cpp to configure the webcam. Lastly, we placed an image of a car that was not part of the positive images we used for training and let the program mark the vehicles inside the picture. However, in the end, we tried using the facedetect.exe as an alternative, and it worked successfully as well.

Results

Detection rate at the 20th stage is: 91.3%
False Alarm rate at the 20th stage is: 2.00875e-005

Analysis of the theoretical results

Overall detection rate of the detector: 91.3%
Overall false alarm rate of the detecto: 2.008e-005

Here is the partial cmd output pasted from the cmd window

Parent node null
1.000000 (detection rate)
1 (false alarm rate)

Parent node 0
0.997620 (detection rate)
0.531328 (false alarm rate)

Parent node 1
0.992861
0.302437

Parent node 2
0.988102
0.18093

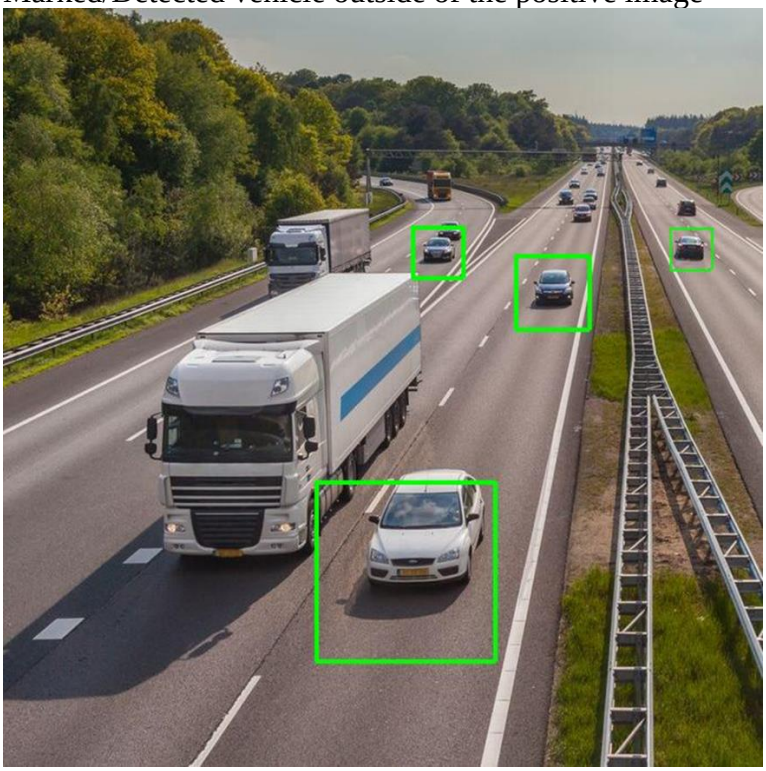
Parent node 3
0.983343
0.0572028
*
*
*
Parent node: 17
0.920880
0.000276272

Parent node: 18
0.916716
0.000259135

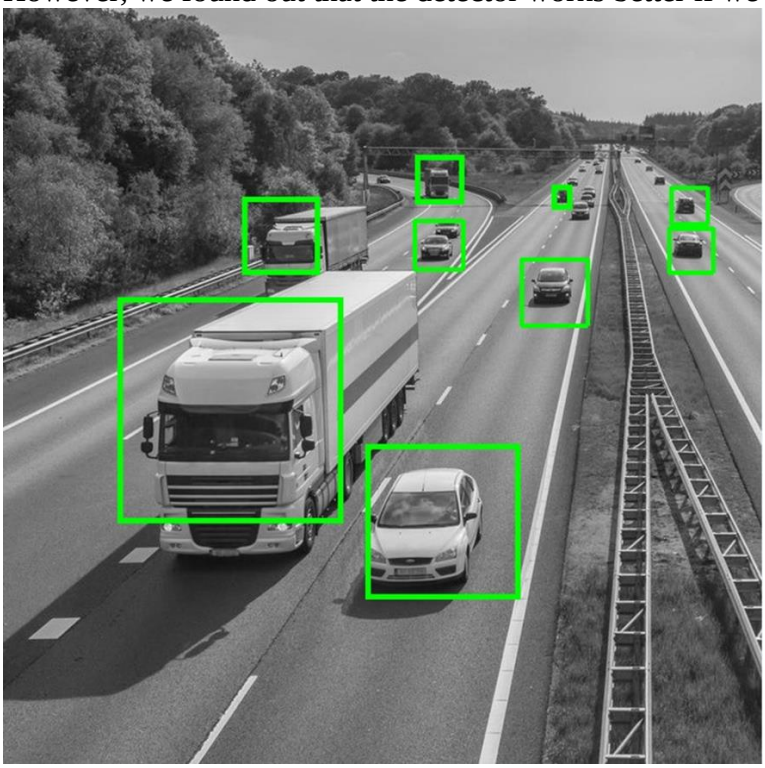
Parent node: 19
0.912552
0.000200875

Results on testing data

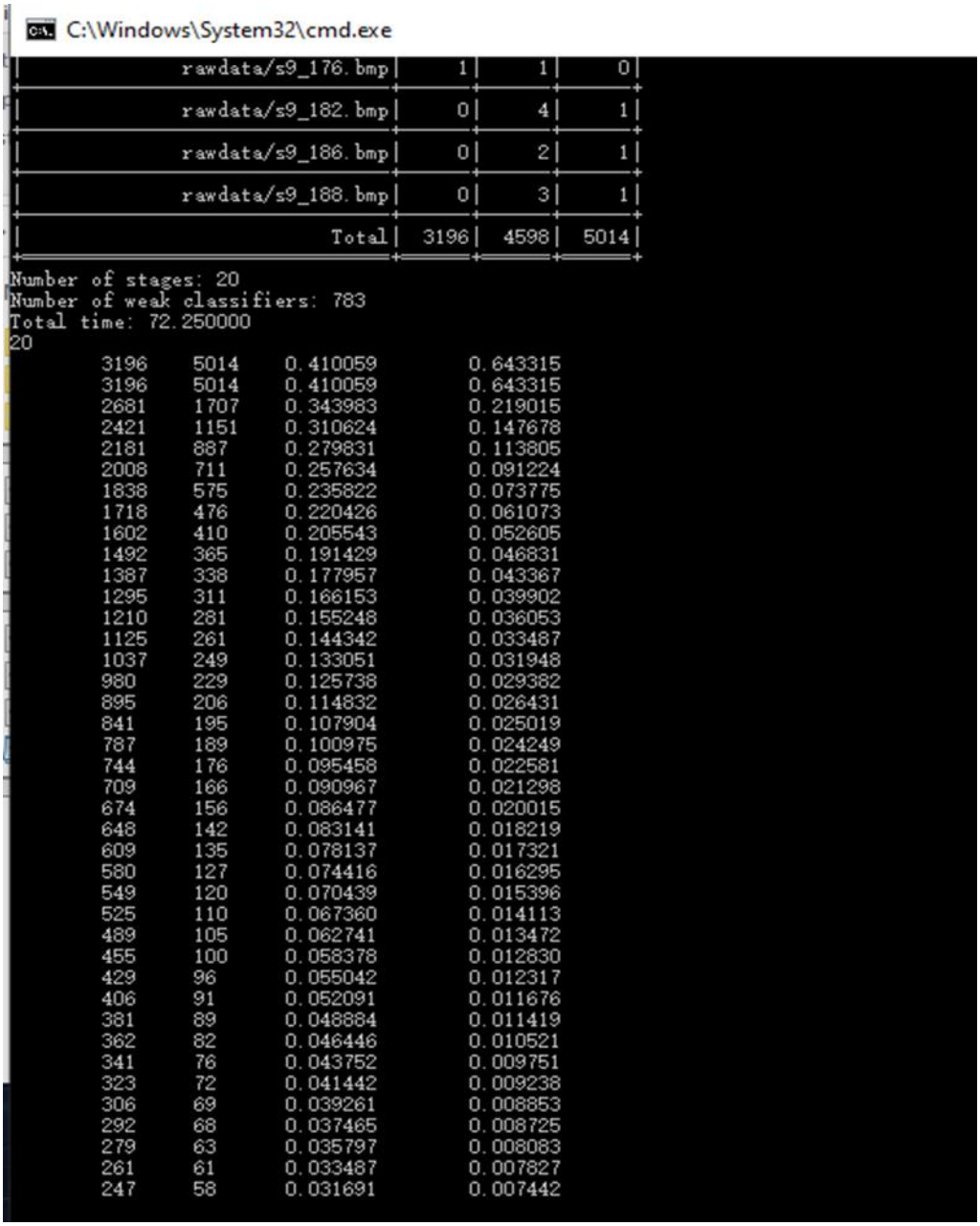
Marked/Detected vehicle outside of the positive image



However, we found out that the detector works better if we use grey images.



Result of the detection of testing data:
The command in the READ_ME_FIRST.txt is WRONG, it won't work! However, I modified the command to "performance.exe -data data/cascade -w 24 -h 24 -info positive/info.txt -ni". Then it started to work.



Analysis of results

Add more negative images probably would help improve the overall detection rate as well as lowering the false alarm rate. We were using 1102 negative/background images in this case, which is way too small in terms of the quantity. That's how we should improve our project in terms of detection accuracy.

Conclusion

As a result, our project was successfully able to detect vehicles from both webcam and static image sources. It recognized that the objects within the image were vehicles and circled them accordingly. Our false alarm rate is a very low number indicating the learning is accurate against the testing model. The interesting fact that we found during the testing phase, is that the factedetec.exe works a lot better if we convert a picture with color to a grey picture. However, the instructions of this project are really outdated and impossible to follow, a lot of DOS commands need to be modified. For instance, the command to run perfromance.exe should be "performance.exe -data data/cascade -w 24 -h 24 -info positive/info.txt -ni". Otherwise, the program won't recognize the output.xml file.

References

OpenCV forum
Online tutorials