Introduction

The project presents an efficient method for object detection and demonstrates its use for real-time vision on-board vehicles. The work is closely based on Paul Viola's face detection method [1], except that the system here was trained to detect objects related to traffic in order to assist the auto pilot on smart vehicles. The idea was adapted from D M Gavrila's paper [2] REAL-TIME OBJECT DETECTION FOR "SMART" VEHICLES.

Objective

The target is to create Haar feature-based cascade classifiers by training the system with an appropriate amount of traffic related information in order to detect vehicles, pedestrians and traffic signs in real-time. Thus, this system in addition to driving assistance/collision avoidance, could further find its application in tracking and recognition.

Haar-like Features

As described by Viola-Jones [1], Haar-like features are a simple and inexpensive image features based on intensity differences between rectangle-based regions that share similar shapes to the Haar wavelets and are defined as the difference of the sum of pixels of areas inside a rectangle and scale within the original image.

Cascading

Cascading is referred to a multistage learning based method with concatenation of many classifiers that use a collection of information. This information is fed from one to the next classifier in the cascade as an additional information. The method was introduced by Lienhart [3] as an addition to Viola's work. Each classifier uses k rectangular areas to make decision if the region of the image looks like the predefined image or not. Haar classifiers are organized in sequences called stages (classification stages). The stage value is the sum of its classifier values. During feature detecting stages are consequently applied to the region of the image until the stage value becomes less than the threshold value or all stages are passed. [4]

Datasets used

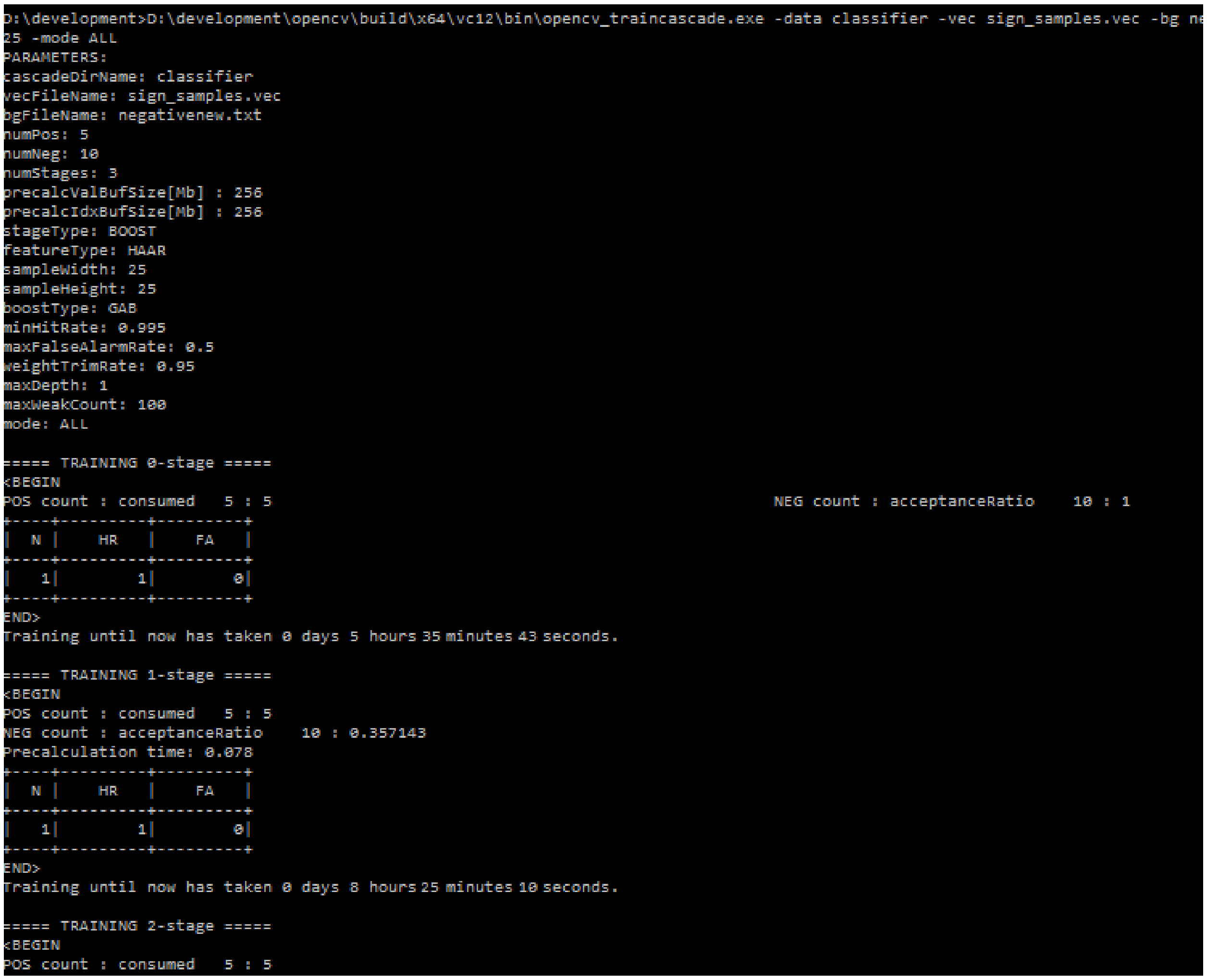
- Dataset for traffic signs and vehicles used from The Laboratory for Intelligent and Safe Auto-mobiles (LISA), CVRR, University of California, San Diego.
  http://cvrr.ucsd.edu/LISA/lisa-traffic-sign-dataset.html
- Dataset for Pedestrian used from CalTech Pedestrian Database, California Institute of Technology, Pasadena.
  http://www.cvc.uab.es/adas/site/?q=node/7
- CVC-07 DPM Virtual-World Pedestrian Dataset
- CVC-05 Partially Occluded Pedestrian Dataset

Classifier generation

Four different Haar cascade classifiers were created using the frames from above mentioned datasets and Naotoshi Seo's tutorial [5] was used to learn the training method. In order to train a classifier we need hundred sample views of cars, signs, pedestrians called positive examples, these are then scaled to a particular size for training purpose. The following sizes were used in the different classifiers that were created. Four different classifiers were created for cars, pedestrians, rectangular traffic signs like speed limits, turns restrictions and the last one for octagonal and rhombus signs like stop sign, pedestrian crossing etc., the built-in tools in open cv like opencv_traincascade.exe, opencv_createsamples.exe along with an object marker tool to crop the objects for training were used.

Classifier for Object	Training size		No. of Positive images (approx.)	No. of Negative images (approx.)	Stages used	Time required for training (approx.)
	Width	height				
Vehicle	20	20	800	1100	12	~2 D
Pedestrians	26	74	650	900	22	1.5 D
Traffic sign1	24	24	1150	2000	19	2 D
Traffic Sign2	25	25	300	1000	16	15 H

Table 1. Specifications of the classifiers used



```
D:\development>D:\development\opencv\build\x64\vc12\bin\opencv_traincascade.exe -data classifier -vec sign_samples.vec -bg ne
25 -mode ALL
PARAMETERS:
cascadeDirName: classifier
vecFileName: sign_samples.vec
bgFileName: negativenew.txt
numPos: 5
numNeg: 10
numStages: 3
precalcValBufSize[Mb] : 256
precalcIdxBufSize[Mb] : 256
stageType: BOOST
featureType: HAAR
sampleWidth: 25
sampleHeight: 25
boostType: GAB
minHitRate: 0.995
maxFalseAlarmRate: 0.5
weightTrimRate: 0.95
maxDepth: 1
maxWeakCount: 100
mode: ALL

===== TRAINING 0-stage =====
<BEGIN
POS count : consumed   5 : 5                                            NEG count : acceptanceRatio    10 : 1
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        0|
+----+---------+---------+
END>
Training until now has taken 0 days 5 hours 35 minutes 43 seconds.

===== TRAINING 1-stage =====
<BEGIN
POS count : consumed   5 : 5
NEG count : acceptanceRatio    10 : 0.357143
Precalculation time: 0.078
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        0|
+----+---------+---------+
END>
Training until now has taken 0 days 8 hours 25 minutes 10 seconds.

===== TRAINING 2-stage =====
<BEGIN
POS count : consumed   5 : 5
```

Screenshot of training phase for traffic sign2 classifier

Classifiers Utilization

After the classifiers were trained, a python code was written which involved the usage of open cv module in order to detect the said objects with the help of the created classifiers. After a classifier is trained, it can be applied to a region of interest in an input image. The classifier outputs a "1" if the region is likely to show the object, and "0" otherwise. To search for the object in the whole image one can move the search window across the image and check every location using the classifier. The classifier is designed so that it can be easily "resized" in order to be able to find the objects of interest at different sizes, which is more efficient than resizing the image itself. So, to find an object of an unknown size in the image the scan procedure should be done several times at different scales. [5]

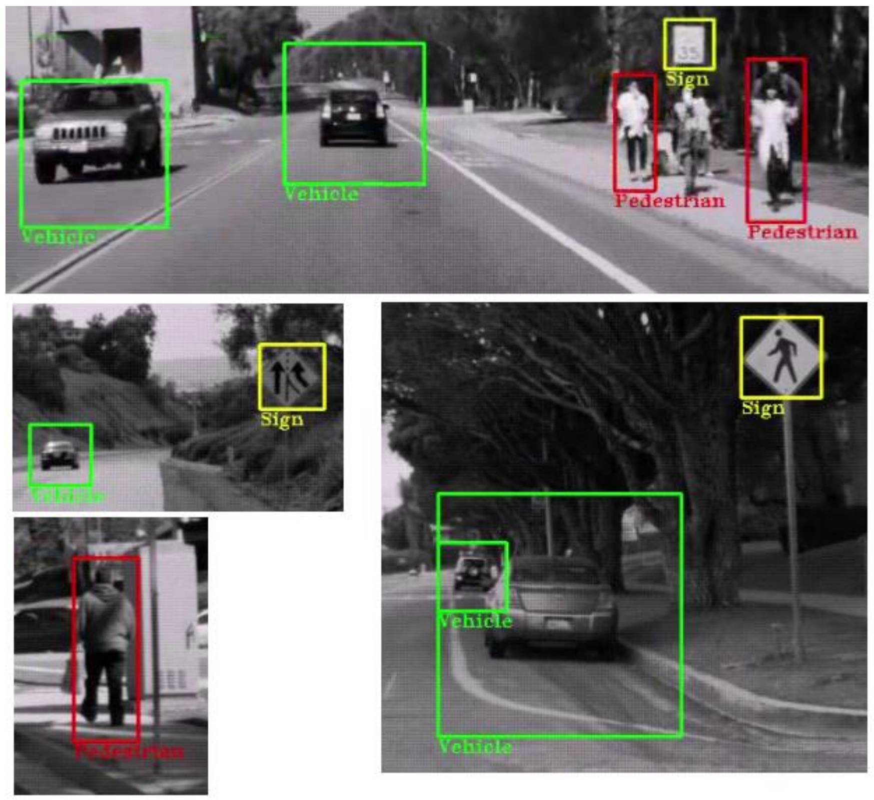
Annotations

The following annotations were used to classify the detected objects:

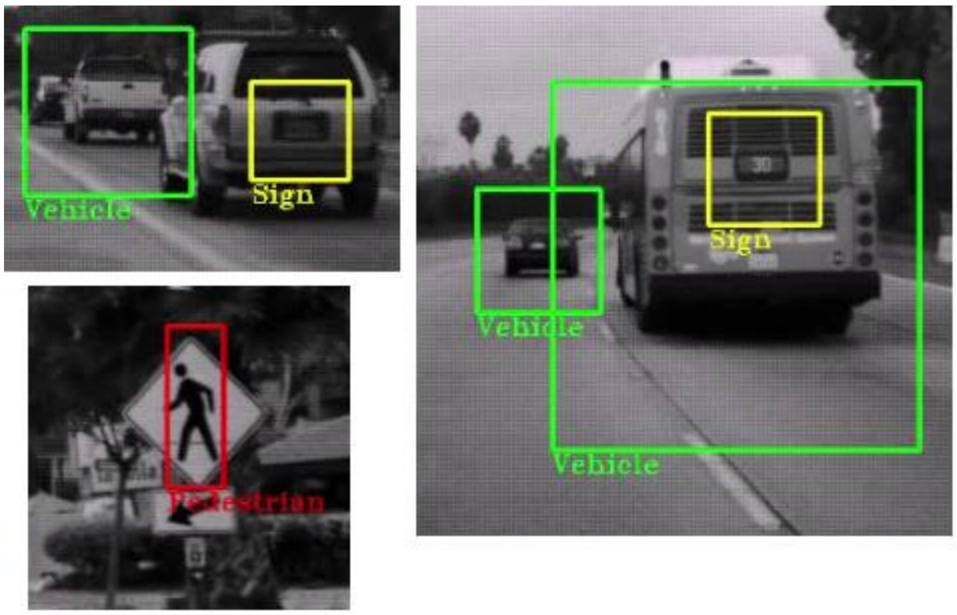
Object	Color	Annotation text
Vehicle	Green	Vehicle
Pedestrians	Red	Pedestrian
Traffic signs	Yellow	sign

Table 2. Annotations used

Results



Few samples of True Positive results



Few samples of False Positive results

As depicted in the above pictures, the application was able to successfully detect vehicles, pedestrians and traffic signals. The accuracy of the program was pretty good and very few false positives were noticed.

Snapshots: More snaps of the demonstration video are available *here*

Additional Work

In addition to this work, an algorithm can be created to actually recognize the detected signs which could prove to be even more helpful to the auto-pilot feature of intelligent vehicles. Also we can categorize the detected vehicles and train the auto-pilot to make way for the emergency/police vehicles. Another additional work would be to measure the distance between the vehicle and pedestrians, then write algorithms to implement safety breaks.

References

[1] P. Viola, M.J.Jones, "Robust Real-Time Face Detection", International Journal of Computer Vision 57, Vol.2, pp.137–154, 2004.

[2] Gavrila, D.M., Philomin, V., "Real-time object detection for 'smart' vehicles", IEEE International Conference of Computer Vision, Vol.1, pp.87–93, 1999.

[3] Rainer Lienhart and Jochen Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. IEEE ICIP 2002, Vol. 1, pp. 900-903, Sep. 2002.

[4] https://software.intel.com/en-us/node/504530

[5] http://note.sonots.com/SciSoftware/haartraining.html

[6] http://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html