# Real Time Detection and Classification of Vehicles and Pedestrians Using Haar Cascade Classifier with Background Subtraction

*********************************************

## Implementation of the Project:

The detection phase of the proposed system is that, mainly a window of the target size is moved over the input image, and for each subsection of the image the Haar-like feature is calculated. This difference is then compared to a learned threshold that separates non-objects from objects.

The biggest advantage of this method is the calculation speed. Due to the use of integral images that quickly and efficiently generates the sum of values in a rectangular subset of a grid, henceforth a Haar-like feature of any size can be calculated in constant time (approximately 60 microprocessor instructions for a 2-rectangle feature).

The Fig. 1 shows a single frame of a sample video. The image contains pedestrians as objects. Fig. 2 is a snapshot of the output after using MOG2 Background Subtraction. The Gray section under a pedestrian represents the shadow of the person.

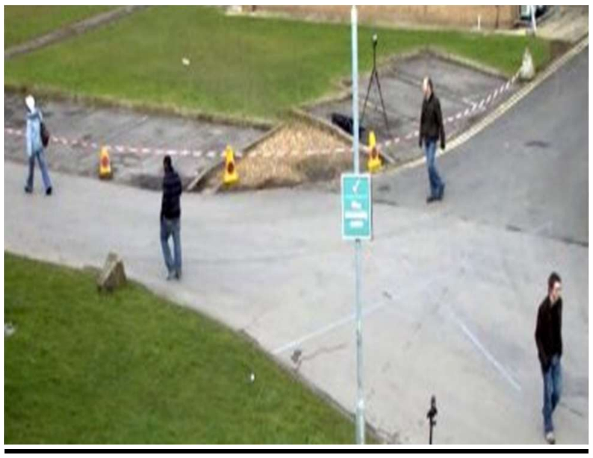


Fig. 1 Sample frame of the video

As a pre-processing step to improve the performance, Background Subtraction Using MOG2 was performed with the help of OpenCV library functions. The results for MOG2 subtraction for pedestrian detection is shown in Fig 2.

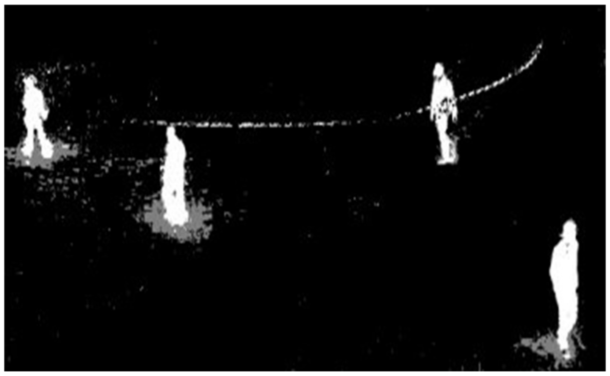
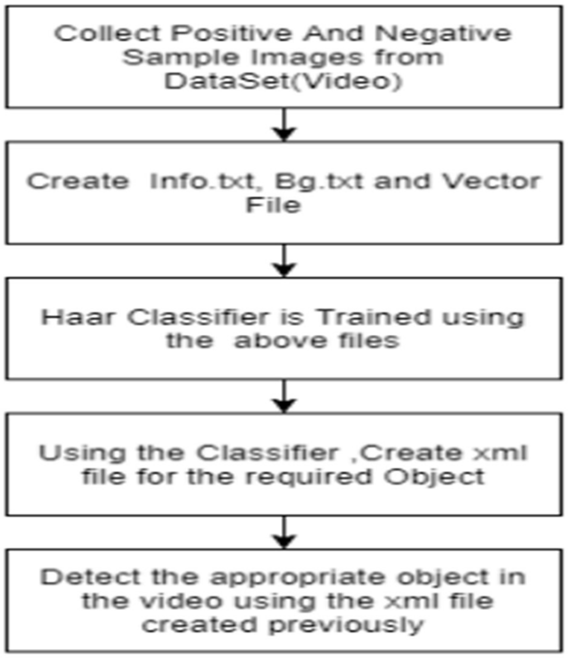


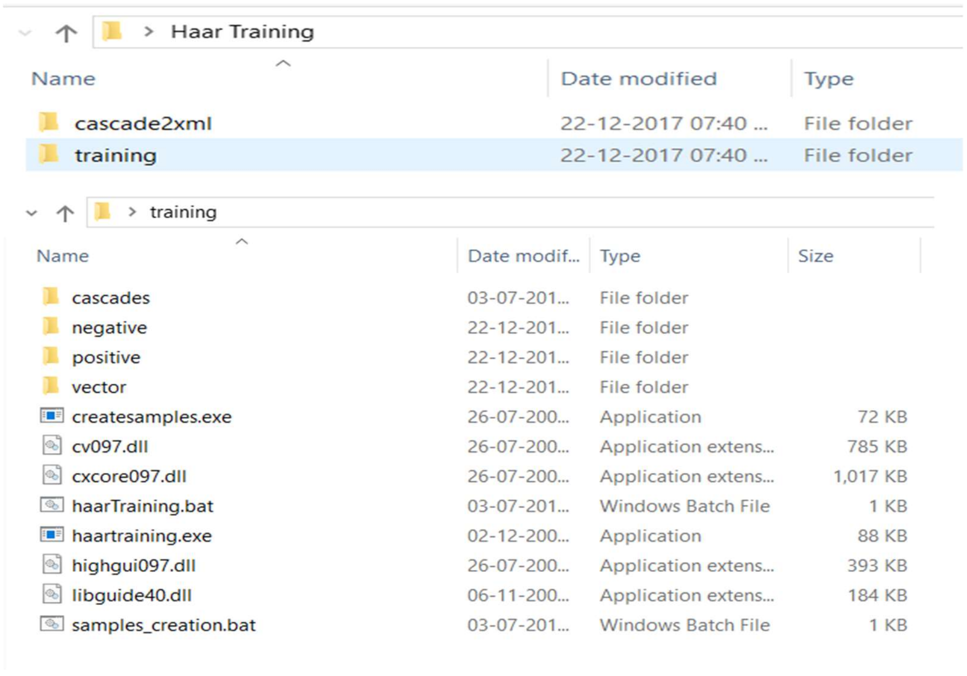
Fig. 2 Result after Background Subtraction

Next we create the XML files of the various objects (target object) to be detected ie one XML file each for bus, car, two-wheeler and pedestrians. The steps involved in creating the XML files is shown below


Collect Positive And Negative Sample Images from DataSet(Video)
Create Info.txt, Bg.txt and Vector File
Haar Classifier is Trained using the above files
Using the Classifier ,Create xml file for the required Object
Detect the appropriate object in the video using the xml file created previously

Flow Diagram of the various processes involved

To implement the above steps, we have used a special tool called Haar Training tool. This tool consists of various files and folders used for creating XML files as shown in the image below.



> Haar Training

| Name | Date modified | Type |
|---|---|---|
| cascade2xml | 22-12-2017 07:40 ... | File folder |
| training | 22-12-2017 07:40 ... | File folder |

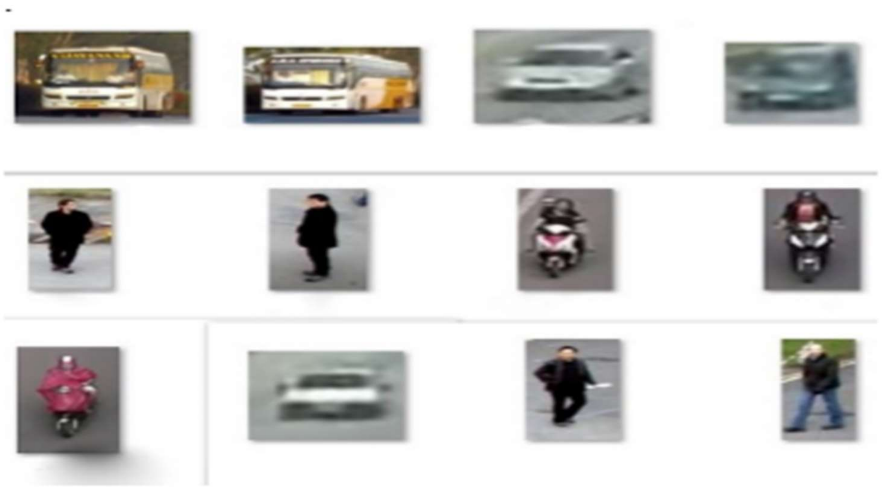
> training

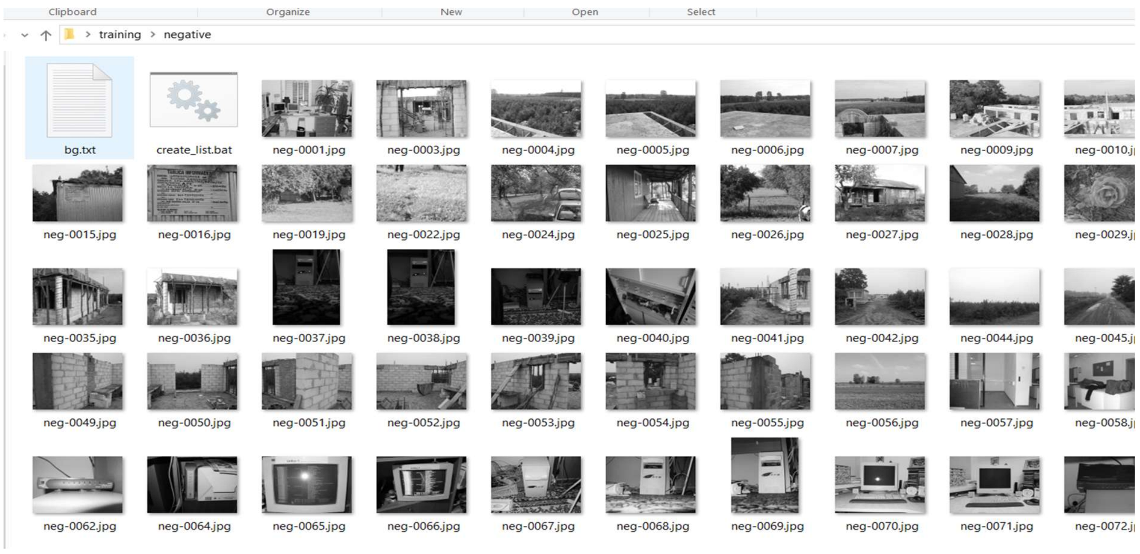
| Name | Date modif... | Type | Size |
|---|---|---|---|
| cascades | 03-07-201... | File folder | |
| negative | 22-12-201... | File folder | |
| positive | 22-12-201... | File folder | |
| vector | 22-12-201... | File folder | |
| createsamples.exe | 26-07-200... | Application | 72 KB |
| cv097.dll | 26-07-200... | Application extens... | 785 KB |
| cxcore097.dll | 26-07-200... | Application extens... | 1,017 KB |
| haarTraining.bat | 03-07-201... | Windows Batch File | 1 KB |
| haartraining.exe | 02-12-200... | Application | 88 KB |
| highgui097.dll | 26-07-200... | Application extens... | 393 KB |
| libguide40.dll | 06-11-200... | Application extens... | 184 KB |
| samples_creation.bat | 03-07-201... | Windows Batch File | 1 KB |

**Step 1: Collection of Positive and Negative Image Set**

For positive images dataset, some of the images were manually cropped out of the video and rest of them were downloaded from the internet. The negative images were downloaded through ImageNet and were made gray-scale as speed and performance of processing these images are high and simple when compared to its coloured image counterpart.

All the negative images are then placed in a folder called negative and all the positive images are then placed in a folder called rawdata within the positive folder of the training tool folder.


Positive images of bus, car, pedestrians and two-wheeler

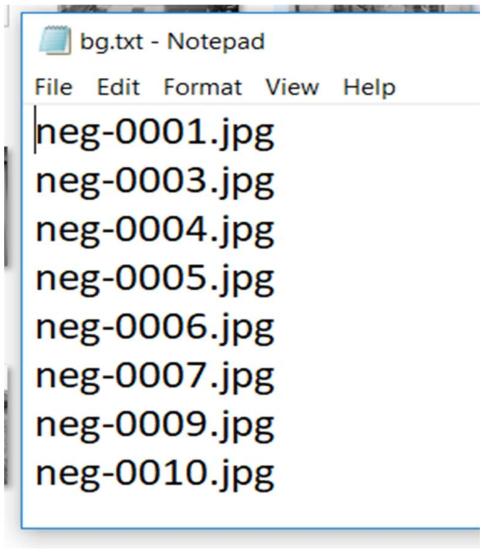


Gray-scaled Negative images placed in Negative Folder

Positive images are those that contain the target object and negative images are the ones which don’t have the target objects.

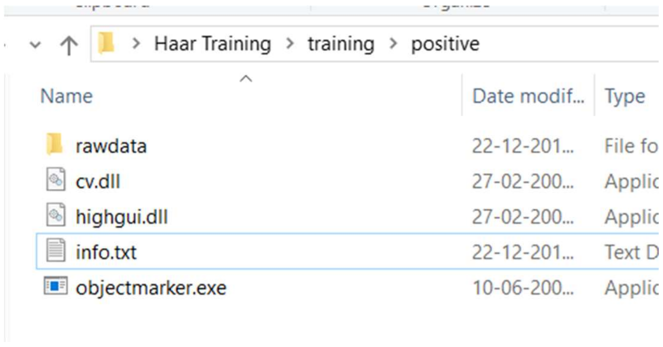
**Step 2: Creating the info.txt, bg.txt and Vector file**

In this step, we go to the negative folder and click on the batch file- create_list.bat. This in turn creates a file bg.txt which contains the list of names of the negative images as shown below



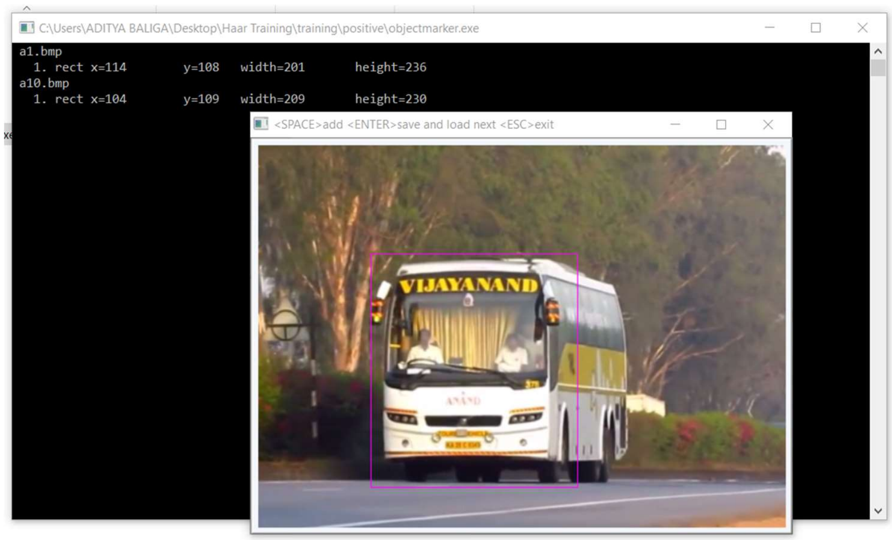
```
neg-0001.jpg
neg-0003.jpg
neg-0004.jpg
neg-0005.jpg
neg-0006.jpg
neg-0007.jpg
neg-0009.jpg
neg-0010.jpg
```

Next, we create the info.txt file which tells us about the exact position of object(s), no of object(s), its x co-ordinate, y co-ordinate etc. in the positive image. For this, we use a tool known as objectmarker.exe



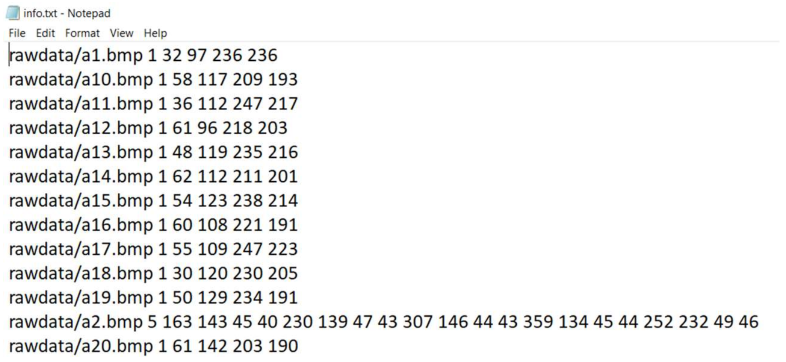
| Name | Date modif... | Type |
|---|---|---|
| rawdata | 22-12-201... | File fo |
| cv.dll | 27-02-200... | Applic |
| highgui.dll | 27-02-200... | Applic |
| info.txt | 22-12-201... | Text D |
| objectmarker.exe | 10-06-200... | Applic |

Using the object marker tool, we mark the object in each positive image given in the raw data folder by enclosing the object using rectangular box and then pressing <space>. If we have another object in the same image, then we select that object and press <space>. After all objects are selected, we go to the next image by pressing <enter> key. A line of detail about the object(s) detected in the image is then displayed in the objectmarker.exe as shown in the image given below.



```
C:\Users\ADITYA BALIGA\Desktop\Haar Training\training\positive\objectmarker.exe
a1.bmp
  1. rect x=114        y=108   width=201       height=236
a10.bmp
  1. rect x=104        y=109   width=209       height=230
```

After doing the above process with all the images in the rawdata folder, the objectmarker tool closes and creates a file called info.txt as shown below which gives details about positive image and the objects detected in it.



```
info.txt - Notepad
File  Edit  Format  View  Help
rawdata/a1.bmp 1 32 97 236 236
rawdata/a10.bmp 1 58 117 209 193
rawdata/a11.bmp 1 36 112 247 217
rawdata/a12.bmp 1 61 96 218 203
rawdata/a13.bmp 1 48 119 235 216
rawdata/a14.bmp 1 62 112 211 201
rawdata/a15.bmp 1 54 123 238 214
rawdata/a16.bmp 1 60 108 221 191
rawdata/a17.bmp 1 55 109 247 223
rawdata/a18.bmp 1 30 120 230 205
rawdata/a19.bmp 1 50 129 234 191
rawdata/a2.bmp 5 163 143 45 40 230 139 47 43 307 146 44 43 359 134 45 44 252 232 49 46
rawdata/a20.bmp 1 61 142 203 190
```

For Example, the line “rawdata/a1.bmp 1 32 97 236 236” in the info.txt means that the positive image named “a1.bmp” is stored in the path “rawdata/a1.bmp” is having 1 object in it, placed at position given by :
x co-ordinate= 32, y co-ordinate= 97, width=236, height=236.

Next we created the vector file for positive images. In folder ..\training\there is a batch file called samples_creation. bat. The content of the bath file is shown below.

createsamples.exe -info positive/info.txt
 –vec vector/objectvector.vec –num 200 -w 24 -h 24

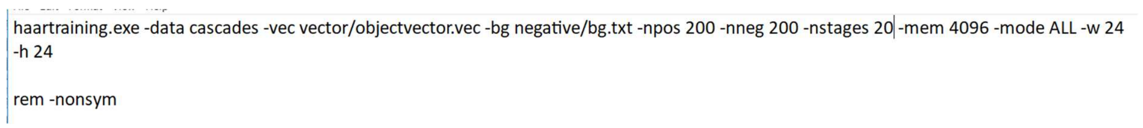
<u>Main Parameters:</u>

| | |
|---|---|
| -info positive/info.txt | Path for positive info file |
| -vec vector/objectvector.vec | Path for the output vector file |
| -num 200 | Number of positive files to be packed in a vector file |
| -w 24 | Width of objects |
| -h 24 | Height of objects |

This batch file loads info.txt and packs the object images into a vector file with the name of e.g. objectvector.vec . After running the batch file, we will get the file objectvector.vec in the folder..\training\vector.

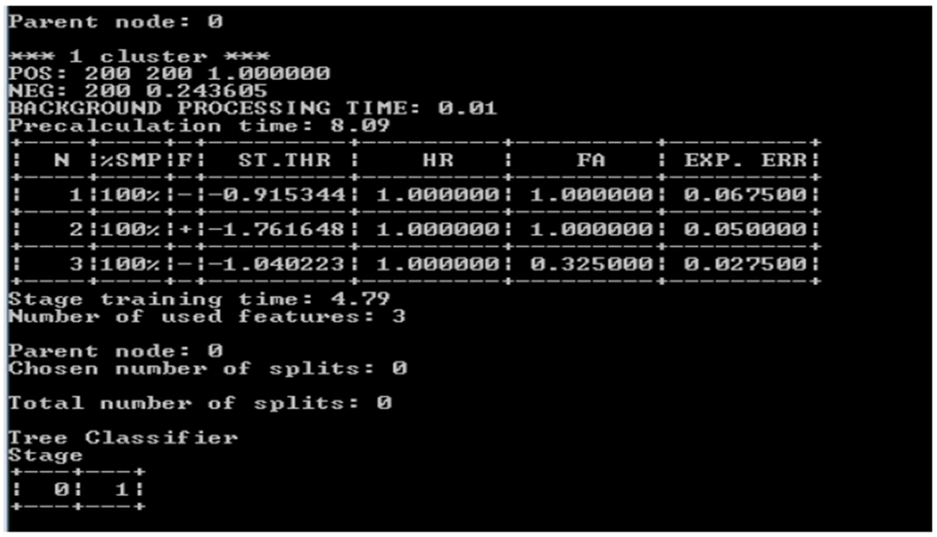
**Step 4: Training the Haar Classifier**

In this step, we click the haartraining.exe file which starts the haar classifier training. The following things have been set in the haarTraining.bat file.



```
haartraining.exe -data cascades -vec vector/objectvector.vec -bg negative/bg.txt -npos 200 -nneg 200 -nstages 20 -mem 4096 -mode ALL -w 24
-h 24

rem -nonsym
```

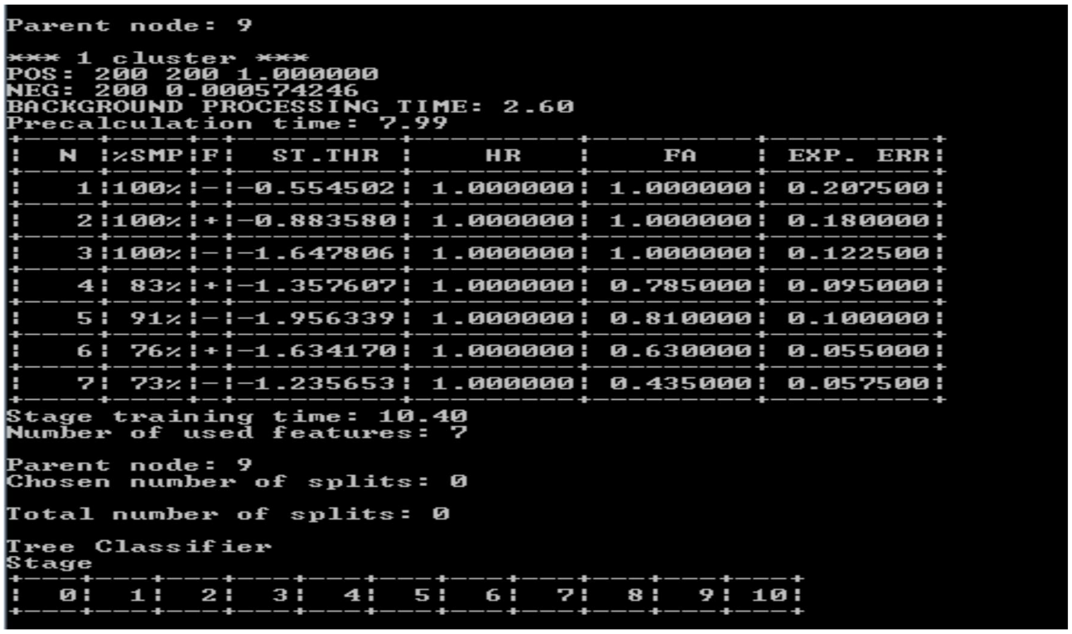
The bat file is used to define location of the vector files, negative files, the number of positive and negative images used- here 200 for both, the memory allocated to this training program in terms of MB- here its 4096MB, the width and the height of the sample images- here 24 for both , the number of training stages here 20. Here –data cascades means path and for storing the cascade of the classifiers.


```
Parent node: 0

*** 1 cluster ***
POS: 200 200 1.000000
NEG: 200 0.243605
BACKGROUND PROCESSING TIME: 0.01
Precalculation time: 8.09
+----+----+-+---------+---------+---------+---------+
|  N |%SMP|F|  ST.THR |    HR   |    FA   | EXP. ERR|
+----+----+-+---------+---------+---------+---------+
|   1|100%|-|-0.915344| 1.000000| 1.000000| 0.067500|
+----+----+-+---------+---------+---------+---------+
|   2|100%|+|-1.761648| 1.000000| 1.000000| 0.050000|
+----+----+-+---------+---------+---------+---------+
|   3|100%|-|-1.040223| 1.000000| 0.325000| 0.027500|
+----+----+-+---------+---------+---------+---------+
Stage training time: 4.79
Number of used features: 3

Parent node: 0
Chosen number of splits: 0

Total number of splits: 0

Tree Classifier
Stage
+---+---+
|  0|  1|
+---+---+
```

The above screen appears when the haartraining.exe is used. Here

Parent node: Defines the current stage under training process
N: Number of used features in this stage
%SMP: Sample Percentage (Percentage of sample used for this feature)
F: "+" if flipped (when symmetry applied) and "–" if not
ST.THR: Stage Threshold
HR: Hit Rate based on the stage threshold
FA: False Alarm based on the stage threshold
EXP. ERR: Exponential Error of strong classifier

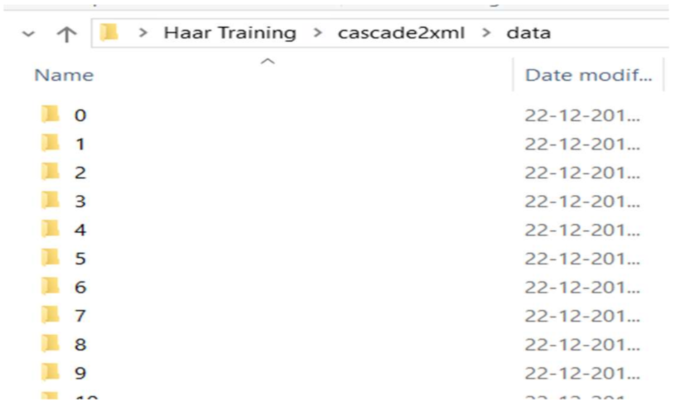
Image of the trainer during 10th stage is shown below. Here we observe that the overall false detection/alarm(F.A) has decreased,no of used features increases and also time required for training increases with the increase in stages.



```
Parent node: 9

*** 1 cluster ***
POS: 200 200 1.000000
NEG: 200 0.000574246
BACKGROUND PROCESSING TIME: 2.60
Precalculation time: 7.99
+----+----+-+---------+---------+---------+---------+
|  N |%SMP|F|  ST.THR |    HR   |    FA   | EXP. ERR|
+----+----+-+---------+---------+---------+---------+
|   1|100%|-|-0.554502| 1.000000| 1.000000| 0.207500|
+----+----+-+---------+---------+---------+---------+
|   2|100%|+|-0.883580| 1.000000| 1.000000| 0.180000|
+----+----+-+---------+---------+---------+---------+
|   3|100%|-|-1.647806| 1.000000| 1.000000| 0.122500|
+----+----+-+---------+---------+---------+---------+
|   4| 83%|+|-1.357607| 1.000000| 0.785000| 0.095000|
+----+----+-+---------+---------+---------+---------+
|   5| 91%|-|-1.956339| 1.000000| 0.810000| 0.100000|
+----+----+-+---------+---------+---------+---------+
|   6| 76%|+|-1.634170| 1.000000| 0.630000| 0.055000|
+----+----+-+---------+---------+---------+---------+
|   7| 73%|-|-1.235653| 1.000000| 0.435000| 0.057500|
+----+----+-+---------+---------+---------+---------+
Stage training time: 10.40
Number of used features: 7

Parent node: 9
Chosen number of splits: 0

Total number of splits: 0

Tree Classifier
Stage
+---+---+---+---+---+---+---+---+---+---+---+
|  0|  1|  2|  3|  4|  5|  6|  7|  8|  9| 10|
+---+---+---+---+---+---+---+---+---+---+---+
```

**Step 5 : Creating XML file**

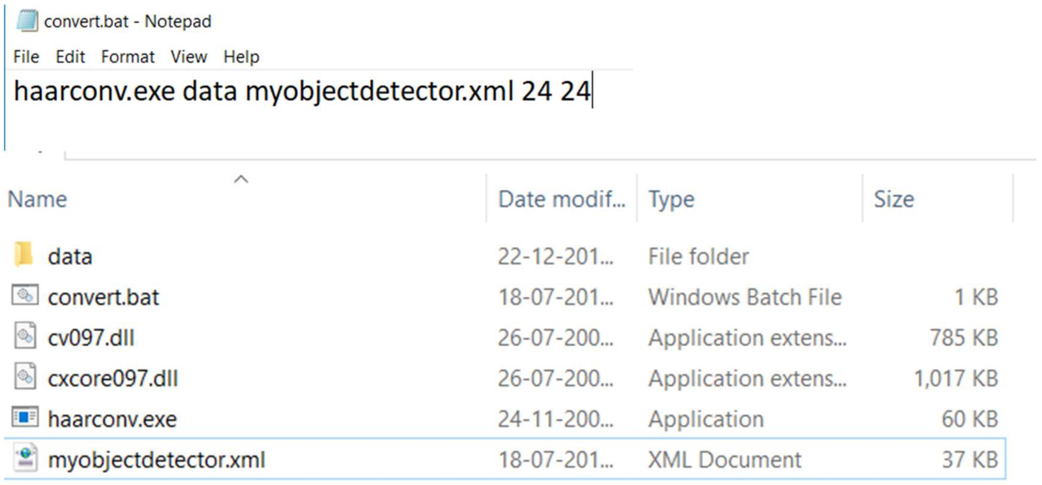
After the above process ends, the cascades folder is loaded up with 0 upto N-1 folders ,where N is the number of stages we inputed in the haarTraining.bat file. Each of these folder contain a file called AdaBoostCARTHaarClassifier.txt. Now we copy all these folders to cascade2xml/data folder as shown below.



> Haar Training > cascade2xml > data

| Name | Date modif... |
|---|---|
| 0 | 22-12-201... |
| 1 | 22-12-201... |
| 2 | 22-12-201... |
| 3 | 22-12-201... |
| 4 | 22-12-201... |
| 5 | 22-12-201... |
| 6 | 22-12-201... |
| 7 | 22-12-201... |
| 8 | 22-12-201... |
| 9 | 22-12-201... |

Now we should combine all created stages (classifiers) into a single XML file which will be our final file a "cascade of Haar-like classifiers". We then set the convert.bat file with the name of the xml file to be created ,width and height. Then we create the XML file by clicking on haarconv.exe. After that we will get myobjectdetector.xml file as shown below.



```
haarconv.exe data myobjectdetector.xml 24 24
```

| Name | Date modif... | Type | Size |
|---|---|---|---|
| data | 22-12-201... | File folder | |
| convert.bat | 18-07-201... | Windows Batch File | 1 KB |
| cv097.dll | 26-07-200... | Application extens... | 785 KB |
| cxcore097.dll | 26-07-200... | Application extens... | 1,017 KB |
| haarconv.exe | 24-11-200... | Application | 60 KB |
| myobjectdetector.xml | 18-07-201... | XML Document | 37 KB |

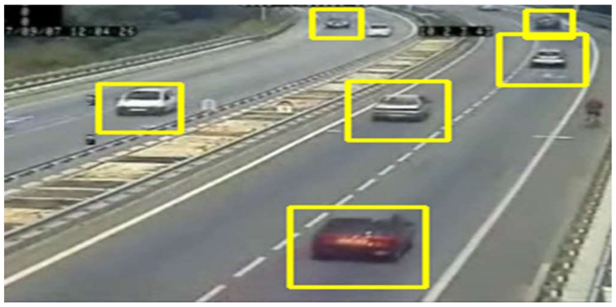
Similarly, we create XML files for car, pedestrians, 2-wheeler etc.

The  generated XML files is then handled with the Cascading Classifiers Command Of OpenCV and then each frame was converted into GrayScale as Coloured frames  lower the processing speed.

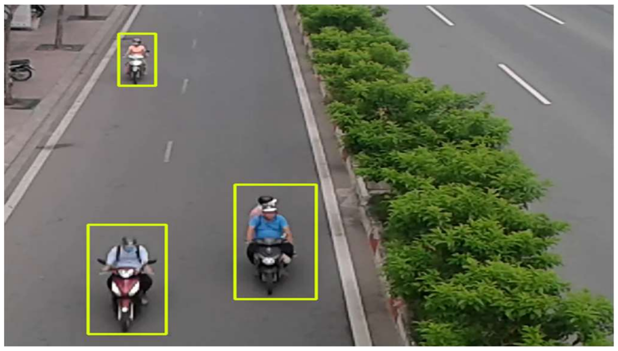
The next step involved detection of  objects which was accomplished by the detectMultiScale, the important parameter of this command is the scaling factor which plays an important role in avoiding false detection and miss detection. Various values of scaling factor were tried and the most suitable one was chosen for each car, bus, two-wheeler and pedestrian detection and hence rectangles were drawn onto the detected objects.

Sample images of the result obtained after using the various xml files used for detecting objects like bus, car, pedestrian and two-wheeler is shown below.

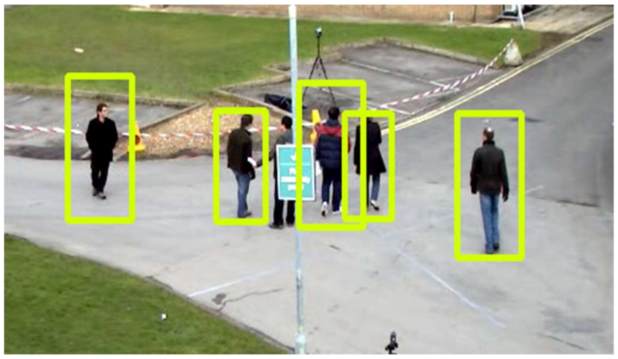
Detection of Bus



Detection of Car



Detection of Two –Wheeler



Detection of Pedestrian

****************************************************************************************