

Assignment 2

PARKING LOT VACANCY DETECTOR

Saikiran Ambati

PSID: 1524311

Abstract:

The Goal of this assignment is to detect the vacancies in a parking lot for a given image of a public parking infrastructure.

The steps followed to complete the task:

- A) Creation of positive and negative samples out of data set and preparing them for training
- B) Extracting features and training using a classifier
- C) Detection and analysis to get the number of empty parking spots

Description of Data set:

The data set consists of images from three separate parking lots parking1a, parking1b and parking2. Each of the parking has data set sorted for three different scenarios cloudy, rainy and sunny. Each single day is represented by the folder, for example the folder contains multiple snapshot images (like 2012_12_12_10_00_05.jpg) and the corresponding ground truth file (like 2012_12_12_10_00_05.xml).

Ground Truth(.xml)

Ground truth file corresponding to each image in each parking lot folder will have its occupancy data along with its location. The ground truth file for each corresponding image has following parameters:

Space: each parking spot

Id: Space ID

Occupied: 0 – unoccupied, 1-occupied

Rotatedrect:

Center: Of the rectangle

Size: of the rectangle

Angle: orientation of the rectangle

```
<space id="4" occupied="1">
  <rotatedRect>
    <center x="502" y="348" />
    <size w="61" h="130" />
    <angle d="-76" />
  </rotatedRect>
  <contour>
    <point x="453" y="369" />
    <point x="543" y="386" />
    <point x="573" y="334" />
    <point x="446" y="304" />
  </contour>
</space>
```

Now, for data creation we will use these parameters from .xml, training a sample will require the image location, number of samples, top left corner of bounding box and size of the each sample by its width and height and store it in a info.txt file to create a vector for training in the following format.

Example:

parking1b\sunny\2013-02-23\2013-02-23_17_05_11.jpg 4 61 130 453 369 39 80 278 90 66 117 1141 292 52 101 1065 276

parking2\sunny\2013-02-23\2013-02-23_17_10_11.jpg 2 61 130 453 369 39 80 278 90

Part A: Sample creation and Training the Cascade Classifier

Training steps to create a Haar-like Classifier:

- Collection of positive and negative training images
- Marking positive images using *xml parser*
- Creating a *vec* (vector) file based on positive marked images using *Opencvcreatesamples*
- Training the classifier using *OpenCv_haartraining*
- Running the classifier using *cvHaarDetectObjects()*

Collection of Samples from Data base:

The original images from each parking lot looks as follows:



Using `rapidxml\rapidxml_utils.hpp` in OpenCV we will create samples based on the occupancy data in the above-mentioned format.



After creating positive samples, we need create a positive vector for training. Negative samples are handpicked and are combined with empty parking spaces and made them to a list of text file “bg.txt”.

So, the files ‘info.txt’ has positive samples information whereas ‘bg.txt’ has negative sample images for proceeding to train the cascade classifier, we have Haar and LBP.

Creating a Vector file of Positive Samples:

For training a cascade classifier we need to pack the collected samples from info.txt to a vector file by specifying the parameters such as location of info.txt, output vector file storage location, number of samples that are to be packed into a vector file, width and height of the objects.

```
D:\Masters\Spring'17\Computer Vision\Assignment 2\Haar-Training\Haar Training\training>createsamples.exe -info positive/info.txt -vec vector/carHaar.vec -num 550 -w 48 -h 24
```

```
Info file name: positive/info.txt
Img file name: (NULL)
Vec file name: vector/caarhaar.vec
BG file name: (NULL)
Num: 700
BG color: 0
BG threshold: 80
Invert: FALSE
Max intensity deviation: 40
Max x angle: 1.1
Max y angle: 1.1
Max z angle: 0.5
Show samples: FALSE
Width: 48
Height: 24
Create training samples from images collection...
positive/info.txt(551) : parse errorDone. Created 550 samples
```

Main Parameters:

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

The batch file loads info.txt and packs the object images into a vector file with the name of e.g. carhaar.vec

After running the batch file, we will have the file carhaar.vec in the folder. \training\vector

Arranging Negative samples:

Put your background images in folder ... \training\negative and run the batch file

`create_list`

```
dir /b *.jpg >bg.txt
```

Running this batch file, we will get a text file each line looks as below:

image1200.jpg

image1201.jpg

image1202.jpg

Later, we need this negative data file for training the classifier.

```
D:\Masters\Spring'17\Computer Vision\Assignment 2\Haar-Training\Haar Training\training\negative>create_list.bat
```

```
D:\Masters\Spring'17\Computer Vision\Assignment 2\Haar-Training\Haar Training\training\negative>dir /b *.jpg 1>bg.txt
```

The list is saved in negative image folder with name 'bg.txt'. the negative images consist of handpicked images of empty parking lots, roads and trees

Sample Back Ground (negative) images:



Cascade Training:

The next step is the training of classifier. As mentioned `opencv_traincascade`, `OpenCv_haartraining` may be used to train a cascade classifier,

Command line arguments of `opencv_traincascade` application grouped by purposes:

Common arguments:

-data <cascade_dir_name>

- Where the trained classifier should be stored.

-vec <vec_file_name>

- vec-file with positive samples (created by `opencv_createsamples` utility).

-bg <background_file_name>

- Background description file.

-numPos <number_of_positive_samples>

-numNeg <number_of_negative_samples>

- Number of positive/negative samples used in training for every classifier stage.

-numStages <number_of_stages>

- Number of cascade stages to be trained.

-precalcValBufSize <precalculated_vals_buffer_size_in_Mb>

- Size of buffer for precalculated feature values (in Mb).

-precalcIdxBufSize <precalculated_idx_buffer_size_in_Mb>

- Size of buffer for precalculated feature indices (in Mb). The more memory you have the faster the training process.

-baseFormatSave

- This argument is actual in case of Haar-like features. If it is specified, the cascade will be saved in the old format.

-acceptanceRatioBreakValue

- This argument is used to determine how precise your model should keep learning and when to stop. A good guideline is to train not further than $10e-5$, to ensure the model does not overtrain on your training data. By default, this value is set to -1 to disable this feature.

Cascade parameters:

-stageType <BOOST(default)>

- Type of stages. Only boosted classifier are supported as a stage type at the given moment.

-featureType<{HAAR(default), LBP}>

- Type of features: HAAR - Haar-like features, LBP - local binary patterns.

-w <sampleWidth>

-h <sampleHeight>

- Size of training samples (in pixels). Must have the same values as used during training samples creation (opencv_createsamples utility).

Boosted classifier parameters:

-bt < {DAB, RAB, LB, GAB(default)}>

- Type of boosted classifiers: DAB - Discrete AdaBoost, RAB - Real AdaBoost, LB - LogitBoost, GAB - Gentle AdaBoost.

-minHitRate <min_hit_rate>

- Minimal desired hit rate for each stage of the classifier. Overall hit rate may be estimated as $(\text{min_hit_rate} \wedge \text{number_of_stages})$

-maxFalseAlarmRate <max_false_alarm_rate>

- Maximal desired false alarm rate for each stage of the classifier. Overall false alarm rate may be estimated as $(\text{max_false_alarm_rate} \wedge \text{number_of_stages})$

-weightTrimRate <weight_trim_rate>

- Specifies whether trimming should be used and its weight. A decent choice is 0.95.

-maxDepth <max_depth_of_weak_tree>

- Maximal depth of a weak tree. A decent choice is 1, that is case of stumps.

-maxWeakCount <max_weak_tree_count>

- Maximal count of weak trees for every cascade stage. The boosted classifier (stage) will have so many weak trees ($\leq \text{maxWeakCount}$), as needed to achieve the given -maxFalseAlarmRate.

Haar-like feature parameters:

-mode <BASIC (default) | CORE | ALL>

- Selects the type of Haar features set used in training. BASIC use only upright features, while ALL uses the full set of upright and 45-degree rotated feature set

Local Binary Patterns does not have any parameters

```
D:\Masters\Spring'17\Computer Vision\Assignment 2\Haar-Training\Haar Training\training>haartraining.exe -data cascades -vec vector/carHaar.vec -bg negative/bg.txt -npos 500 -nneg 500 -nstages 15 -mem 1024 -mode ALL -w 48 -h 24 -nonsym
```

```
Data dir name: cascades
Vec file name: vector/carHaar.vec
BG file name: negative/bg.txt
Num pos: 500
Num neg: 500
Num stages: 15
Num splits: 1 (stump as weak classifier)
Mem: 1024 MB
Symmetric: FALSE
Min hit rate: 0.995000
Max false alarm rate: 0.500000
Weight trimming: 0.950000
Equal weights: FALSE
Mode: ALL
Width: 48
Height: 24
Max num of precalculated features: 178956
Applied boosting algorithm: GAB
Error (valid only for Discrete and Real AdaBoost): misclass
Max number of splits in tree cascade: 0
Min number of positive samples per cluster: 500
Required leaf false alarm rate: 3.05176e-005
```

The size of `-W` and `-H` in `haartraining.bat` should be same as what you defined while creating a vector file

Haartraining collects a new set of negative samples for each stage, and `-nneg` sets the limit for the size of the set. It uses the previous stages' information to determine which of the "candidate samples" are misclassified.

Training ends when the ratio of misclassified samples to candidate samples is lower than $FR^{stage_{no}}$. So:

Regardless of the number of stages (`nstages`) that you define in `haartraining.bat`, the program may terminate early if we reach above condition. Although this is normally a good sign of accuracy in our training process, however this also may happen when the number of positive images is not enough (e.g. less than 500).

After training is initiated by specifying the number of stages, we will see the information as shown below

N	%SMP	F	ST.THR	HR	FA	EXP. ERR
---	------	---	--------	----	----	----------

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

For training this data set I have chosen 15 stages and the following are the images while training the classifier.

For stage 0:

```

Tree Classifier
Stage
+---+
| 0 |
+---+

    0

Number of features used : 138694

Parent node: 0

*** 1 cluster ***
POS: 200 200 1.000000
NEG: 200 0.440529
BACKGROUND PROCESSING TIME: 0.01
Precalculation time: 7.22
+---+---+---+---+---+---+---+---+
| N |%SMP|F| ST.THR | HR | FA | EXP. ERR|
+---+---+---+---+---+---+---+---+
| 1|100%|-|-0.967033| 1.000000| 1.000000| 0.060000|
+---+---+---+---+---+---+---+---+
| 2|100%|+|-1.653970| 1.000000| 1.000000| 0.060000|
+---+---+---+---+---+---+---+---+
| 3|100%|-|-1.332503| 1.000000| 0.430000| 0.042500|
+---+---+---+---+---+---+---+---+
Stage training time: 3.27
Number of used features: 3

```


Parent node: 0

*** 1 cluster ***

POS: 200 200 1.000000

NEG: 200 0.453515

BACKGROUND PROCESSING TIME: 0.00

Precalculation time: 6.69

	N	%SMP	F	ST.THR	HR	FA	EXP. ERR
1	100%	-	-0.885714	1.000000	1.000000	0.112500	
2	100%	+	-1.168176	1.000000	1.000000	0.112500	
3	100%	-	-0.639284	1.000000	0.470000	0.100000	

Stage training time: 2.92

Number of used features: 3

Parent node: 0

Chosen number of splits: 0

Total number of splits: 0

Tree Classifier

Stage

	0	1
0	1	

0---1

Tree Classifier

Stage

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	2	3	4	5	6	7	8	9	10	11	12	

0---1---2---3---4---5---6---7---8---9---10---11---12

Cascade performance

POS: 200 200 1.000000

NEG: 200 2.04408e-005

BACKGROUND PROCESSING TIME: 43.82

```

+---+---+---+---+---+---+---+---+---+
|  N  |%SMP|F|  ST.THR  |    HR    |    FA    |  EXP. ERR|
+---+---+---+---+---+---+---+---+---+
|   1  |100%|-|-0.441442| 1.000000| 1.000000| 0.331034|
+---+---+---+---+---+---+---+---+---+
|   2  |100%|+|-0.617569| 1.000000| 1.000000| 0.331034|
+---+---+---+---+---+---+---+---+---+
|   3  |100%|-|-0.772293| 1.000000| 1.000000| 0.283908|
+---+---+---+---+---+---+---+---+---+
|   4  | 90%|+|-0.841465| 1.000000| 1.000000| 0.265517|
+---+---+---+---+---+---+---+---+---+
|   5  | 93%|-|-0.668476| 1.000000| 0.779310| 0.237931|
+---+---+---+---+---+---+---+---+---+
|   6  | 85%|+|-0.662270| 1.000000| 0.781609| 0.237931|
+---+---+---+---+---+---+---+---+---+
|   7  | 88%|-|-1.066653| 1.000000| 0.836782| 0.227586|
+---+---+---+---+---+---+---+---+---+
|   8  | 84%|+|-1.137655| 1.000000| 0.843678| 0.197701|
+---+---+---+---+---+---+---+---+---+
|   9  | 86%|-|-0.946299| 0.997701| 0.666667| 0.170115|
+---+---+---+---+---+---+---+---+---+
|  10  | 82%|+|-1.082658| 1.000000| 0.673563| 0.152874|
+---+---+---+---+---+---+---+---+---+
|  11  | 82%|-|-0.922154| 1.000000| 0.572414| 0.128736|
+---+---+---+---+---+---+---+---+---+
|  12  | 81%|+|-0.997592| 1.000000| 0.577011| 0.137931|
+---+---+---+---+---+---+---+---+---+
|  13  | 81%|-|-0.847164| 0.997701| 0.498851| 0.116092|
+---+---+---+---+---+---+---+---+---+
Stage training time: 115.26
Number of used features: 13

```

Generating a Cascade XML:

After finishing Haar-training step, in folder. /training/cascades/ you should have catalogues named from “0” up to “N-1” in which N is the number of stages you already defined in Harrtraining

In each of those catalogues there should be AdaBoostCARTHaarClassifier.txt file. 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”.

```
D:\Masters\Spring'17\Computer Vision\Assignment 2\Haar-Training\Haar Training\training>haarconv.exe data carHaar.xml 48 24
```

The .xml file will be generated, which must be able to detect the cars in parking lot and there by finding the number of vacant spots in parking lot.

LBP:

Performance of LBP training, below are images of few stages while training.

```
Parent node: 4

*** 1 cluster ***
POS: 4000 4094 0.977040
NEG: 3500 0.0340179
BACKGROUND PROCESSING TIME: 0.75
Precalculation time: 0.00
+---+---+---+---+---+---+---+---+
|  N  |%SMP|F|  ST.THR  |    HR    |    FA    | EXP. ERR|
+---+---+---+---+---+---+---+---+
|   1  |100%|-|-0.138808| 1.000000| 1.000000| 0.408933|
+---+---+---+---+---+---+---+---+
|   2  |100%|-|-0.316316| 1.000000| 1.000000| 0.324400|
+---+---+---+---+---+---+---+---+
|   3  | 93%|-|-0.704385| 1.000000| 1.000000| 0.340800|
+---+---+---+---+---+---+---+---+
|   4  | 93%|-|-0.885808| 1.000000| 1.000000| 0.288933|
+---+---+---+---+---+---+---+---+
|   5  | 91%|-|-1.125476| 1.000000| 1.000000| 0.244133|
+---+---+---+---+---+---+---+---+
|   6  | 97%|-|-0.973974| 0.995250| 0.875714| 0.221200|
+---+---+---+---+---+---+---+---+
|   7  | 88%|-|-0.790823| 0.996250| 0.740000| 0.239867|
+---+---+---+---+---+---+---+---+
|   8  | 87%|-|-0.944348| 0.998250| 0.758857| 0.215200|
+---+---+---+---+---+---+---+---+
|   9  | 85%|-|-1.006085| 0.995250| 0.600286| 0.183067|
+---+---+---+---+---+---+---+---+
|  10  | 83%|-|-0.809691| 0.995250| 0.602571| 0.171867|
+---+---+---+---+---+---+---+---+
|  11  | 83%|-|-0.995727| 0.996000| 0.610286| 0.146533|
+---+---+---+---+---+---+---+---+
|  12  | 82%|-|-0.984030| 0.995500| 0.622571| 0.139200|
+---+---+---+---+---+---+---+---+
|  13  | 81%|-|-1.071313| 0.995250| 0.528286| 0.136667|
+---+---+---+---+---+---+---+---+
|  14  | 81%|-|-0.915469| 0.995500| 0.509429| 0.125333|
+---+---+---+---+---+---+---+---+
|  15  | 80%|-|-0.865239| 0.995250| 0.482000| 0.110800|
+---+---+---+---+---+---+---+---+
Stage training time: 4638.64
```

*** 1 cluster ***

POS: 4000 4207 0.950796

NEG: 3500 0.00310018

BACKGROUND PROCESSING TIME: 7.41

Precalculation time: 0.00

N	%SMP	F	ST.THR	HR	FA	EXP. ERR
1	100%	-	-0.295092	1.000000	1.000000	0.334133
2	100%	-	-0.875090	1.000000	1.000000	0.326133
3	100%	-	-0.702815	0.997250	0.896857	0.274533
4	95%	-	-1.034095	0.997250	0.896857	0.236000
5	96%	-	-1.017417	0.995750	0.777714	0.231600
6	90%	-	-1.177044	0.996500	0.800286	0.241467
7	85%	-	-1.113512	0.995750	0.841143	0.196000
8	84%	-	-1.335967	0.996500	0.793714	0.199867
9	84%	-	-1.024624	0.995750	0.705429	0.166800
10	82%	-	-0.950091	0.995250	0.630857	0.162933
11	81%	-	-0.884472	0.995250	0.546857	0.158000
12	81%	-	-0.776836	0.995750	0.495143	0.162933

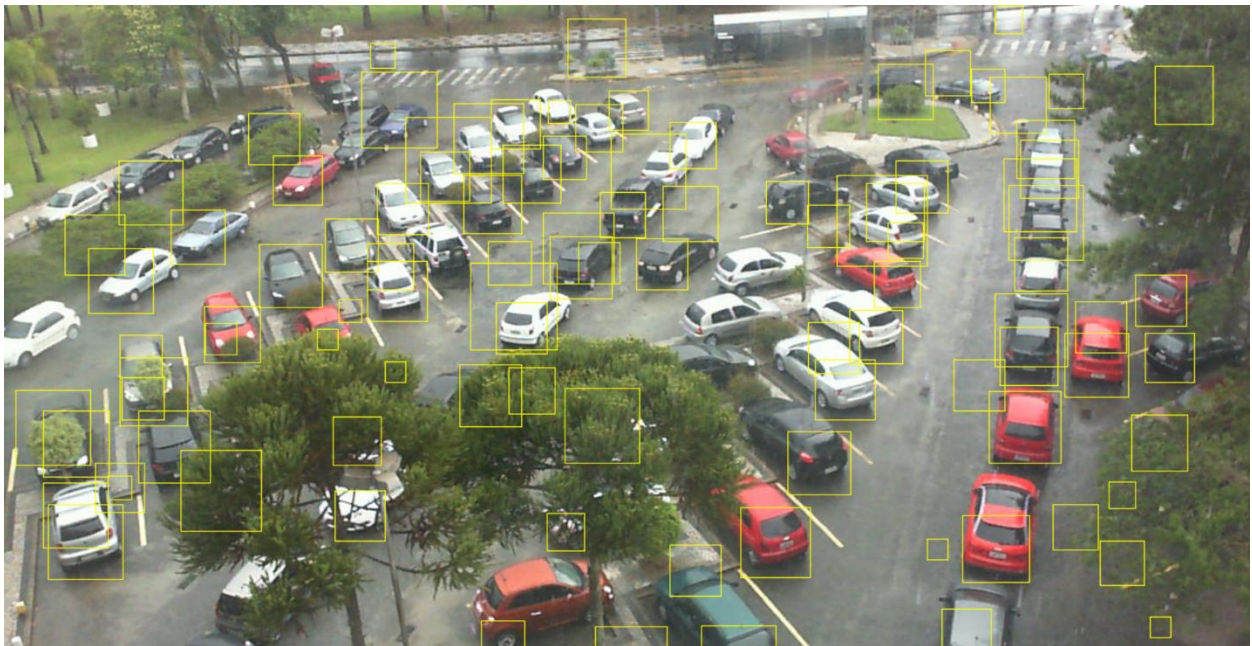
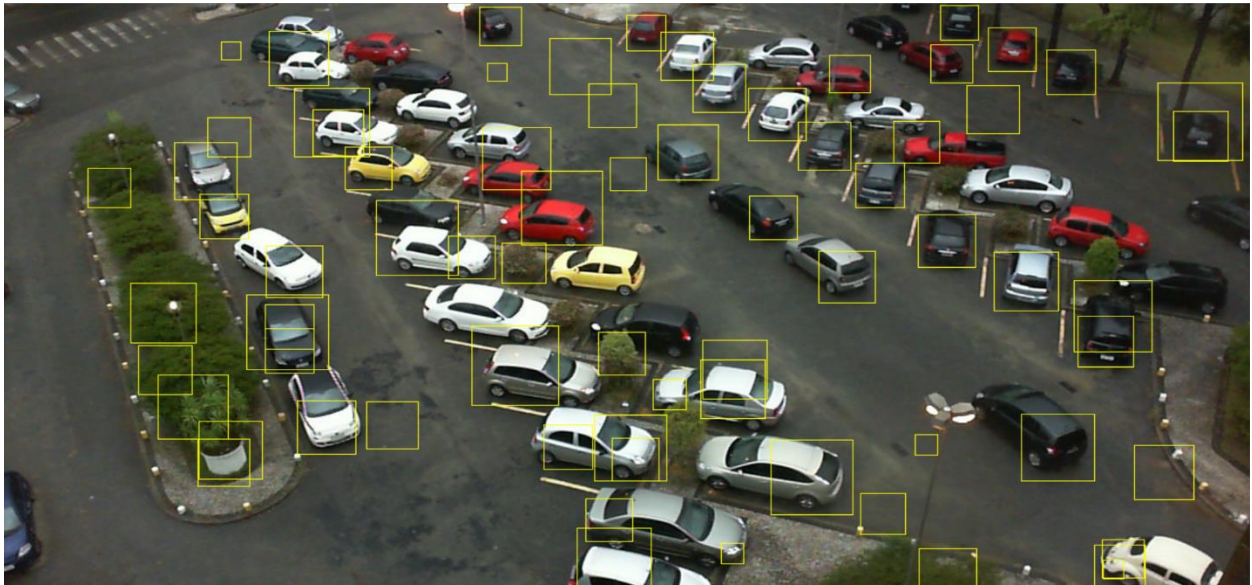
Stage training time: 3522.75

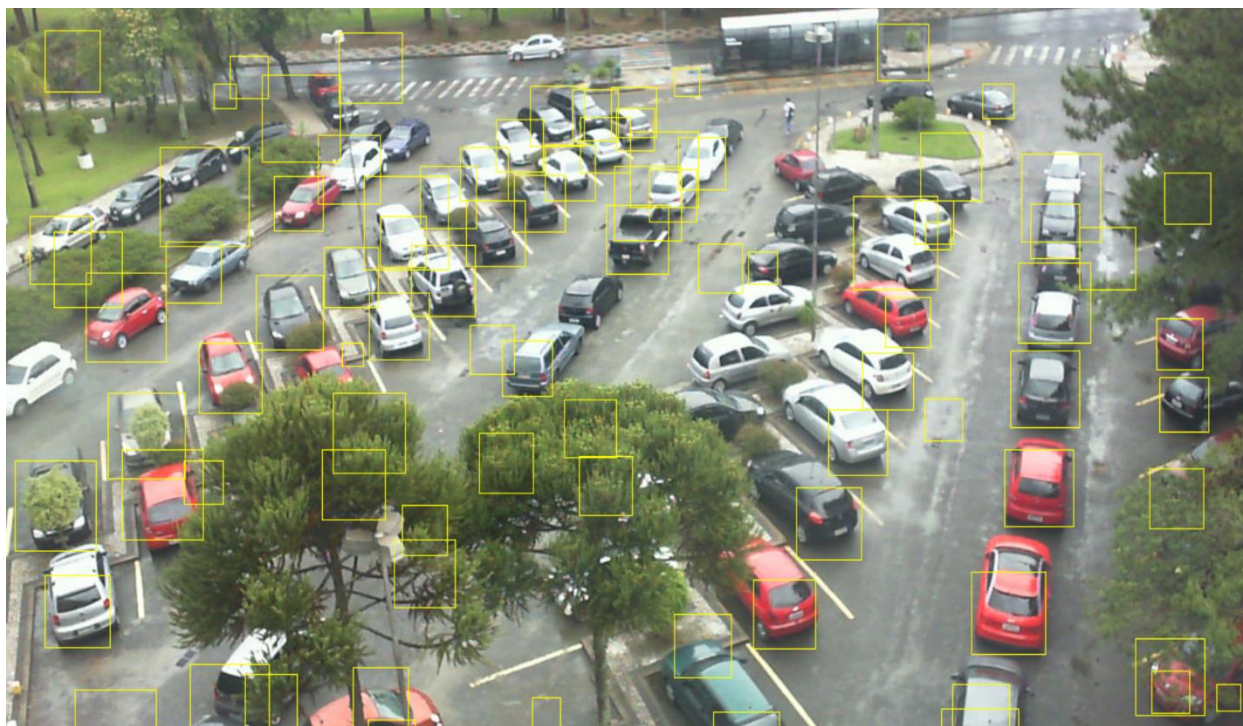
Number of used features: 12

Part B: Car Detection using the Trained Cascade Classifier

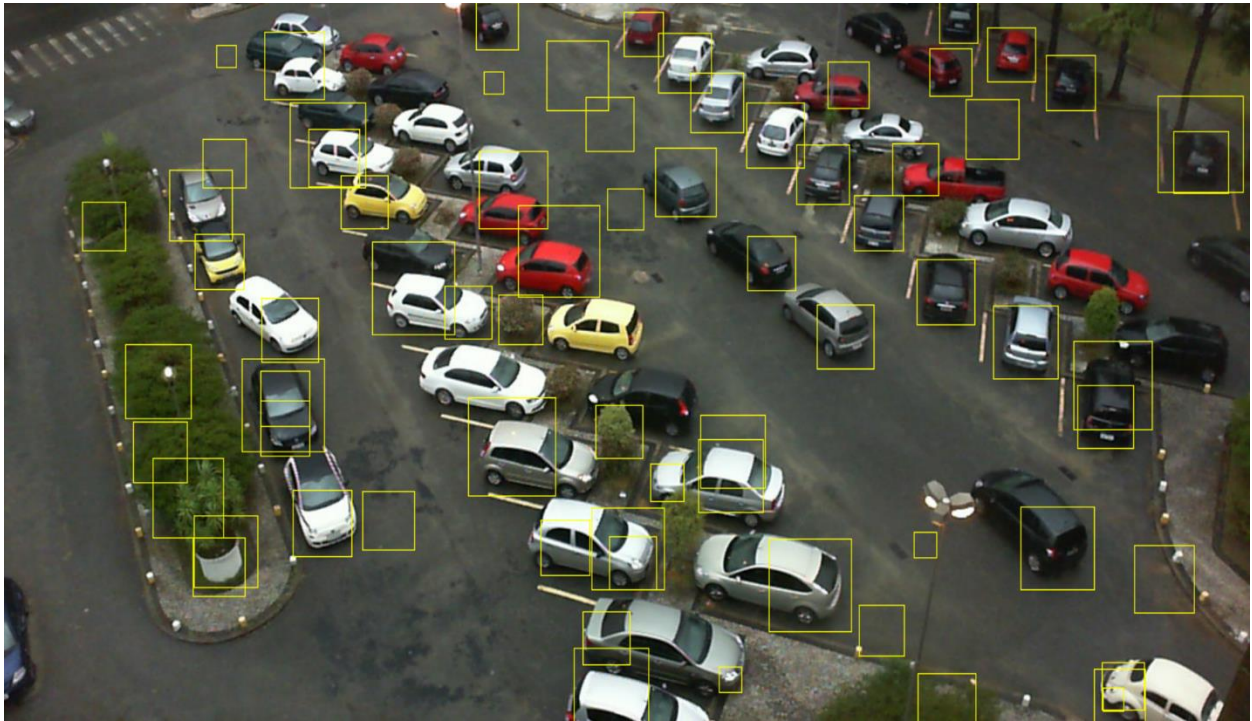
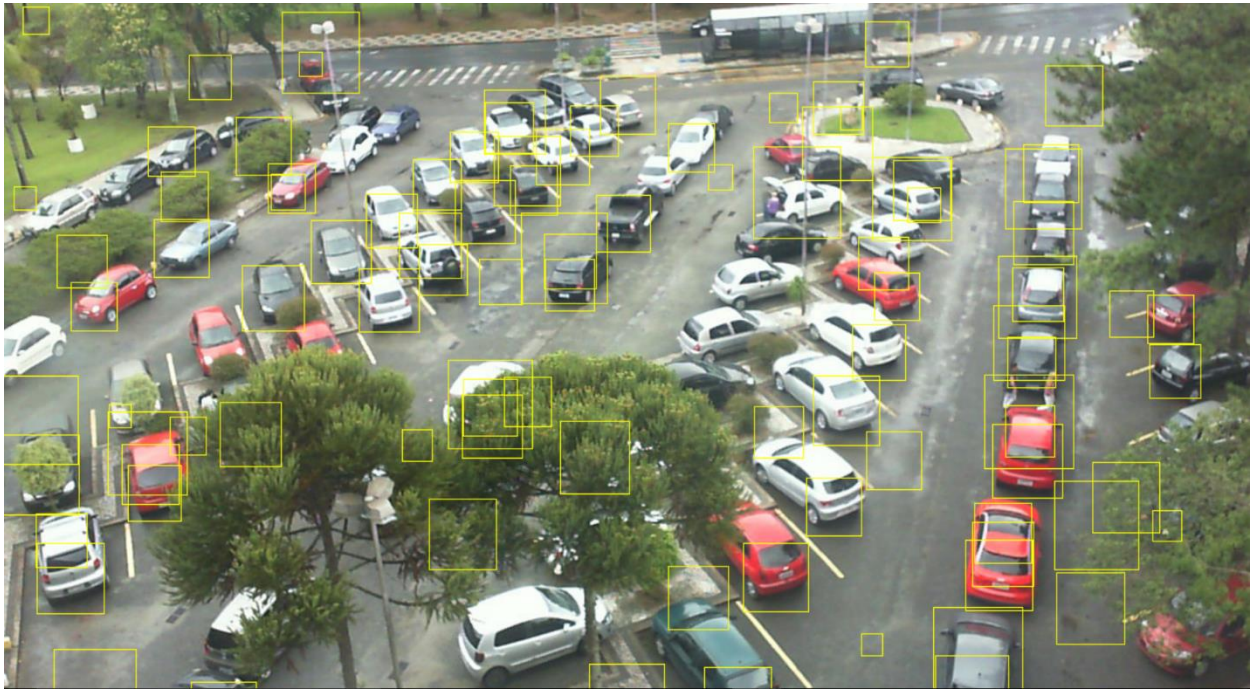
Now after training we have a xml file generated which will be used to detect the objects, the objects detected will depend up on the collection of samples, the back-ground data have random images other than cars so it is obvious that the classifier also detects few roads and trees which are not taken into consideration while analyzing empty spaces, the annotated spaces given in ground truth file are compared and analyzed in next part, here are few output images after detection. The images used are from rainy and cloudy folders

Detection Using Haar Features:





Detection using LBP Features:










To Detect and put a bounding box around a object we will use the detect multi scale function.






```
cascade.detectMultiScale(img, cars, 1.1, 2, CV_HAAR_SCALE_IMAGE, Size(10, 10), Size(100, 100));
```






Where we can specify the minimum and maximum size of the bounding box along with the scaling factor






Part C: Parking Lot analysis (Vacancy Detection) Using Detected Cars

Now we analyze the detected cars to find out the number of vacant spots in the lot by comparing it with its corresponding ground truth file by considering the area covered by bounding box around a detected car. Then find the Accuracy from True positive and True negative with in annotated spaces. For testing purpose, I have taken 20 set of images from cloud and rainy of 10 each

Test Image	Classifier Feature	Training			True positive	False Positive	Accuracy
		Stages	No. of Positives	No. of Negatives			
	Haar	15	3000	3500	34	0	86
	LBP	15	4000	3800	32	0	80
	Haar	15	3000	3500	10	4	82
	LBP	15	4000	3800	12	4	85
	Haar	15	3000	3500	12	0	84
	LBP	15	4000	3800	10	0	82
	Haar	15	3000	3500	0	6	94
	LBP	15	4000	3800	0	6	94
	Haar	15	3000	3500	18	0	74
	LBP	15	4000	3800	16	0	72

Test Image	Classifier Feature	Training			True positive	False Positive	Accuracy
		Stages	No. of Positives	No. of Negatives			
	Haar	15	3000	3500	0	1	97
	LBP	15	4000	3800	0	4	96
	Haar	15	3000	3500	15	0	80
	LBP	15	4000	3800	18	0	78
	Haar	15	3000	3500	3	6	88
	LBP	15	4000	3800	3	6	88
	Haar	15	3000	3500	30	0	82
	LBP	15	4000	3800	32	0	80
	Haar	15	3000	3500	15	2	76
	LBP	15	4000	3800	18	2	78

Test Image	Classifier Feature	Training			TP	FP	Accuracy
		Stages	No. of Positives	No. of Negatives			
	Haar	15	3000	3500	0	4	96
	LBP	15	4000	3800	0	6	94
	Haar	15	3000	3500	74	0	74
	LBP	15	4000	3800	78	0	76
	Haar	15	3000	3500	12	0	92
	LBP	15	4000	3800	16	0	94
	Haar	15	3000	3500	34	0	88
	LBP	15	4000	3800	38	0	90
	Haar	15	3000	3500	1	4	96
	LBP	15	4000	3800	1	4	96

Test Image	Classifier Feature	Training			True positive	False Positive	Accuracy
		Stages	No. of Positives	No. of Negatives			
	Haar	15	3000	3500	0	4	96
	LBP	15	4000	3800	0	6	94
	Haar	15	3000	3500	0	4	96
	LBP	15	4000	3800	0	6	94
	Haar	15	3000	3500	28	0	78
	LBP	15	4000	3800	32	0	82
	Haar	15	3000	3500	70	0	84
	LBP	15	4000	3800	68	0	82
	Haar	15	3000	3500	0	4	94
	LBP	15	4000	3800	0	6	96

Conclusion:

Hence found the number of empty parking spaces by training the images of parking from different lots by collecting samples of both positive and negative and using two features namely LBP and Haar to train the cascade classifier and later tested and analyzed them to find empty spaces.