Assignment 4

Manoj Alwani (109335757)

Algorithm Steps:

1). Initial Setup: I have used the parameters defined in the paper [1-2]. The **resolution ratio (alpha) is taken as 2** and scale ratio (**beta) as 1.2** The number of resolution levels are **R=4**. The other details are in the table below.

Resolution	Cell Size	Block Size	Detector Size	# orientations	Block stride	Detector Stride
1	(3, 3)	(6, 6)	(8, 16)	9	(2, 2)	(1, 1)
2	(4, 4)	(8, 8)	(16, 32)	9	(2, 4)	(2, 2)
3	(6, 6)	(12, 12)	(32, 64)	9	(4, 4)	(4, 4)
4	(8, 8)	(16, 16)	(64, 128)	18	(4, 8)	(8, 8)

Table1: Parameters at each resolution level of pedestrian detection system. All parameters except orientations expressed as (width, height) pair in pixels.

Window size at highest resolution chosen as (64, 128) and its size reduced at each resolution according to resolution ratio.

2). Training

A). I created the negative patches for training from INRIA TEST folder which contains the negative images at full resolution. To generate negative training data set from full resolution images, I randomly sampled 10 normalized images (160 X 96) from each image and collected around 2000 training and test images. The positive normalized training samples already provided in (INRIAPerson/train_64x128_H96) folder.

The **code** for creating these samples is in **(4.2_data_set_creation)** folder. I have used 200 positive and 200 negative images from training. The images used are listed in .lst files.

B). I have then trained the classifier at different resolutions. I have used 4 **(R)** resolutions and parameters for training used are described in Table 1.

I stored the **Weight and bais** of the classifier for different resolution in structure **train_mat** so that it can be used in the detection.

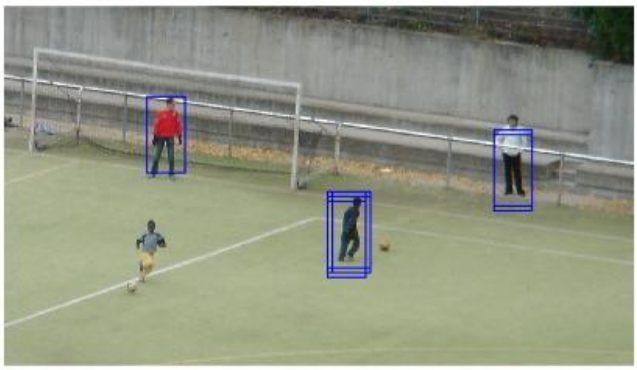
The code for training is in **4.2_training folder.** The code file training is **object_training.m** file. The code is generalized for number of training images, different positive and negative samples, margin and parameters shown in Table 1. I am describing below how by just changing the parameters we can use the code for different data set, different block size and number of images. (Which also completes extra credit problem 1)

Parameters:	effect
Images_to_train_pos :	**Number of positive images to train**
Images_to_train_neg :	**Number of negative images to train**
cell_size :	**For cell size at different resolution**
block_size:	**Block size at different resolution**
block_stride:	**stride for block at different resolution**
orientations:	**Orientation at different resolution**

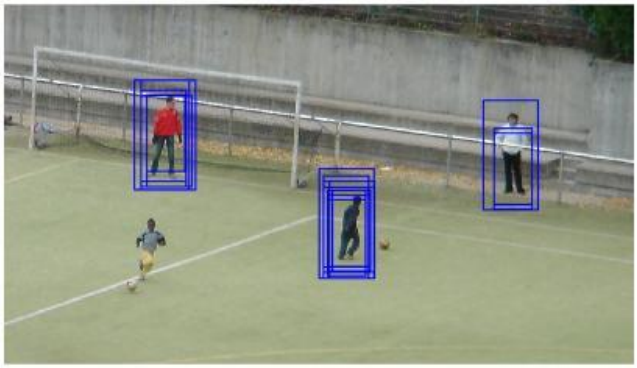
3). Detection : The code for detection is in **4.2_object_detection** folder. I have included two codes for detection in this folder. **object_detection_at_resolution.m which** detect objects only at resolution space the other code **object_detection_at_resolutionspace.m** works at both resolution and scale.

One of the issue in scale space is to decide scale ratio parameter. This parameter is totally dependent on image size. For different values it gives different results. As shown in figure below detection at scale 2 and 1.2.

Scale 2



scale 1.2

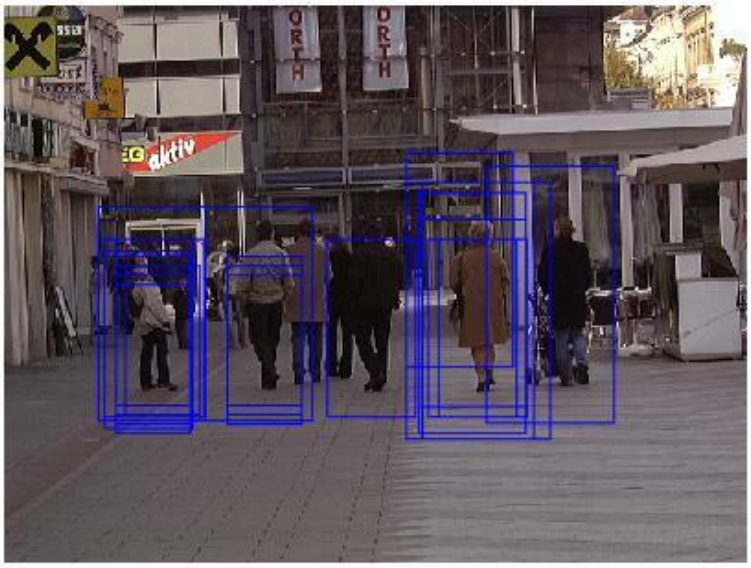


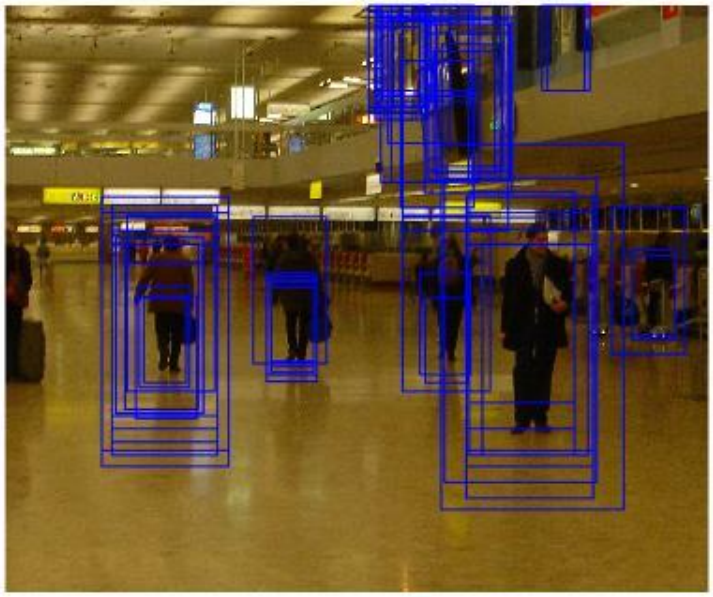
As we see that we get more accurate windows at scale 1.2 So I fixed the scale ratio at 1.2 and adding one more level in the pyramid for the detection untill

floor(ImageWidth/Scale)>64 and floor(ImageHeight/Scale)>128

which is also suggested by original paper from inria. [1]

Results on some other images shown below:

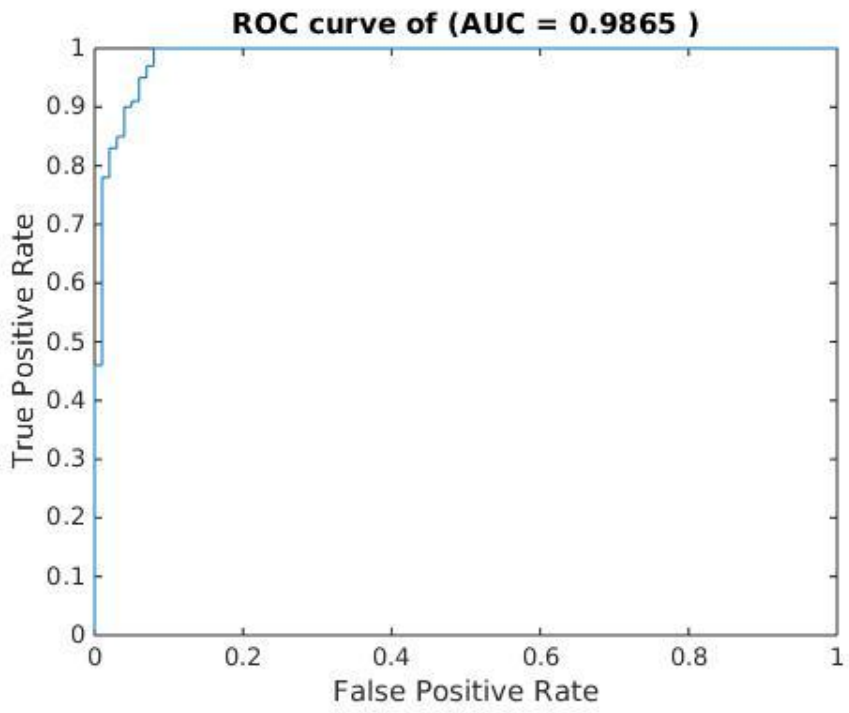




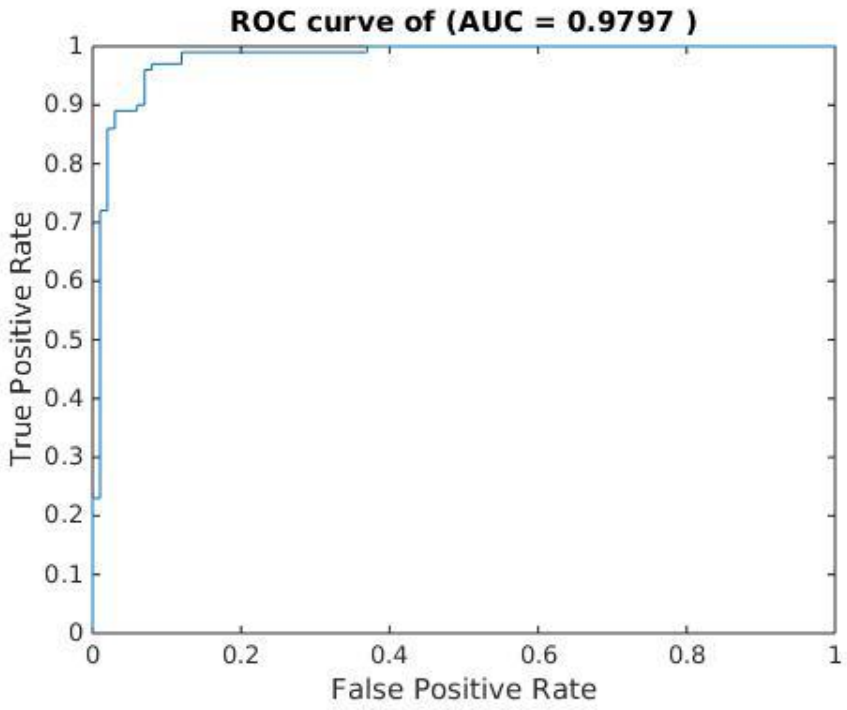
4). **Evalution:** I have evaluated my results for detection on the test images in **Inria/test_64x128_H96 folder.** I used 100 positive and 100 negative images for testing. I got accuracy of around **58.5% to 65%** for different set of images. While these images are small and have binary condition that person is there or not. For this kind of evaluation I included script along with detection. The codes for this evaluation is included in **4.3_detection folder**. There is flag in the code called **result_display** which shows the result along with evalution.

The results for full resolution images are included in results folder.

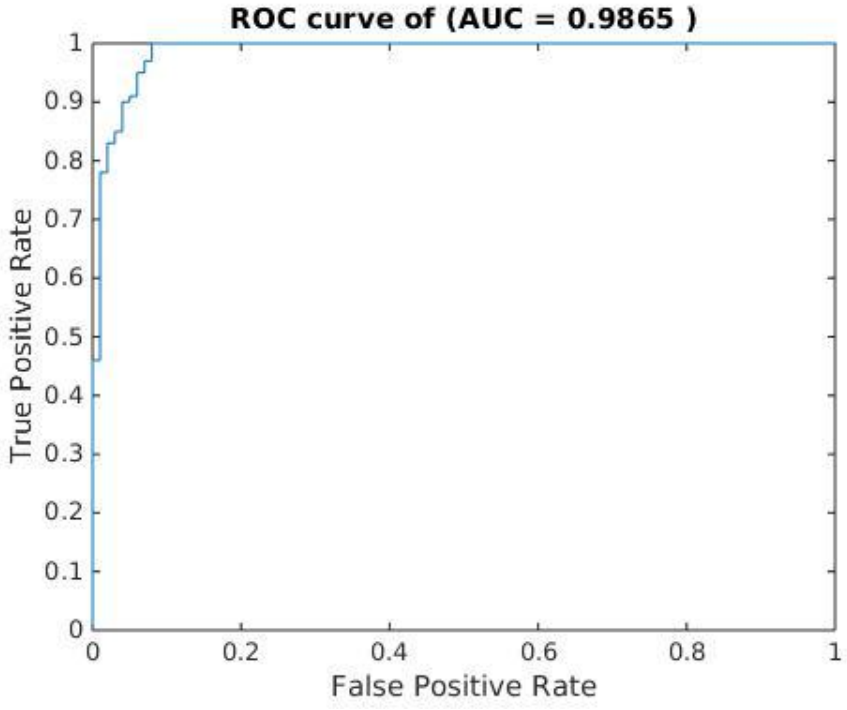
The ROC code is included in **4.3_ROC** folder. The results of ROC for different resolution is shown below. I generated the training hog features using VLFEAT library and passed these features to **svmtrain** function of libsvm library. **The ROC curves at different resolution gives the intuition that at which resolution false positive rate is high and we choose our threshold parameters according to that.**



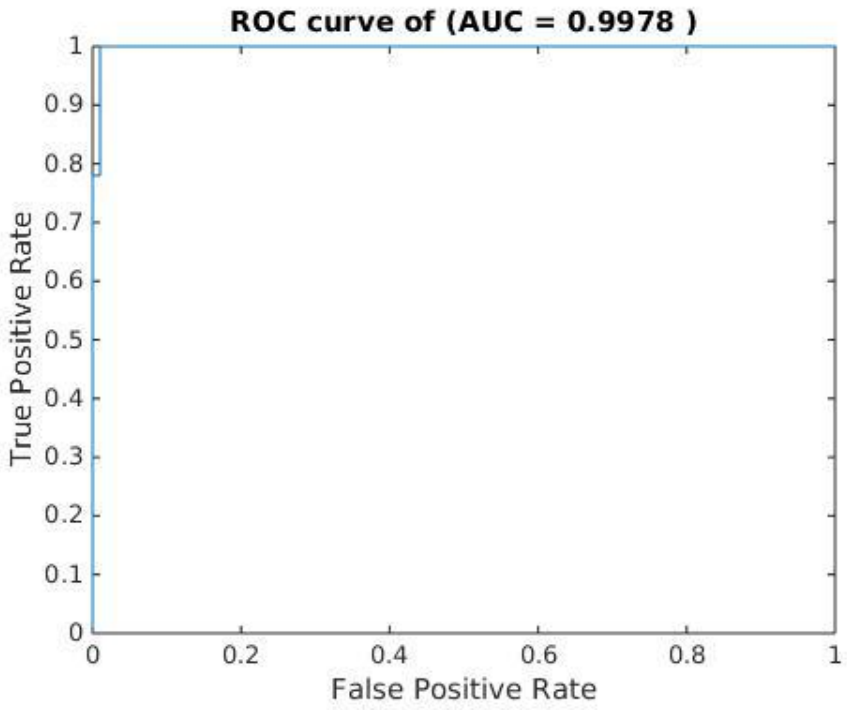
Resolution 2:



Resolution 3:



Resolution 4:



The different parameters used for evaluation are reported in table above. The resolution ratio **alpha** is 2 and **scale ratio** is 1.2. I have also done the testing on different scales as shown in the image above in detection section. The images used for test and train are reported in **.lst** files in the folders. Trained parameters are included in **train_parameters** folder. I have tested my code with different number of positive and negative images and for different set of images the trained parameters for these tests are in **train_parameters** folder.

Extra Credit:

1). Code for HOG extraction function so that it can control the amount of stride for blocks and the number of cells in each block?

As the code is already generalized for changing the number of cells and block stride as shown in the table above. We have used fixed set of parameters for these variables for pedestrian detection. By changing these values in both training and testing we can get the desired results for different stride and cell size.

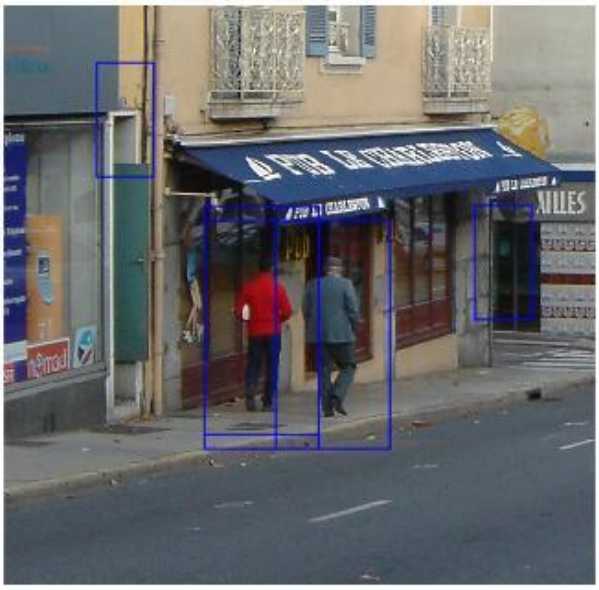
One of the version of HOG extraction function is in **extra_credit/part1** folder where we have make the stride of block size. In this case our evaluation results drops on the test images (as described in section 3 above) from 58.5 % to 40% the reason for this is that now there now no overlapping blocks so during detection features are not trained accurately.

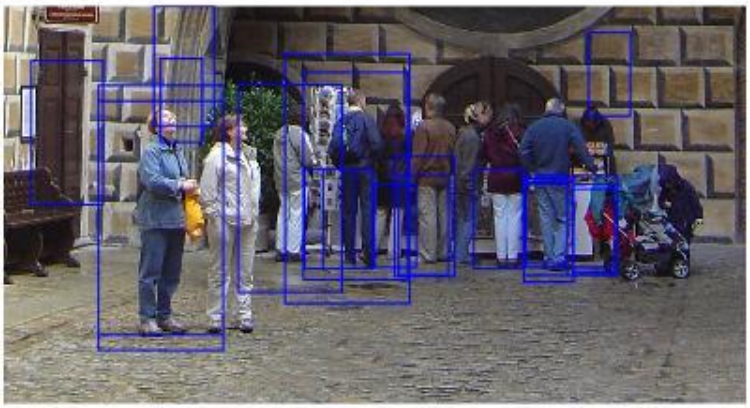
2). Adding hard negative samples:

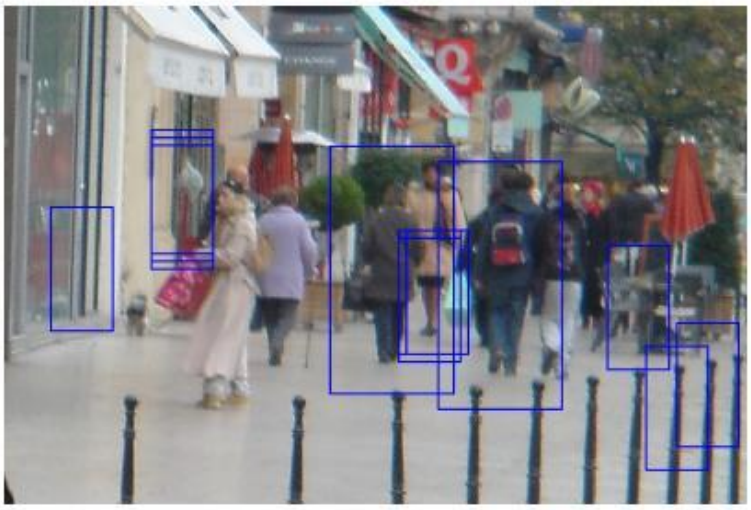
The code for adding hard samples in included in **extra_credit/part2** folder. At each resolution (>1) we detect our code on hard samples (in hard_samples.lst file) at one resolution down. If hard samples are detected at false positive than we train that sample in the current resolution. As described in [3].

Adding the hard samples increase our accuracy by good amount it reaches from **58.5% to 64%** on the images which were used earlier without hard samples. The score reaches maximum up to 75 % for different set of images. It can increase more if the network is trained with more number of images.

The results for some images are shown below:







3). **Car and Motor Bike Detection on pascal Visual Object Classes (VOB) challenge 2006 database**

I have tested the same model for car and motor bike detection. The code for car detection is in **extra_credit/part2/car** folder. The parameter used for car detection are shown in table below.

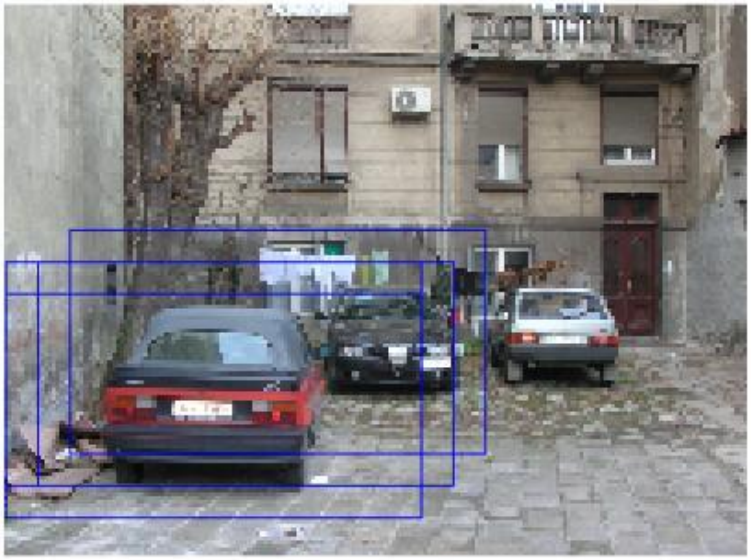
The parameters for car detection:

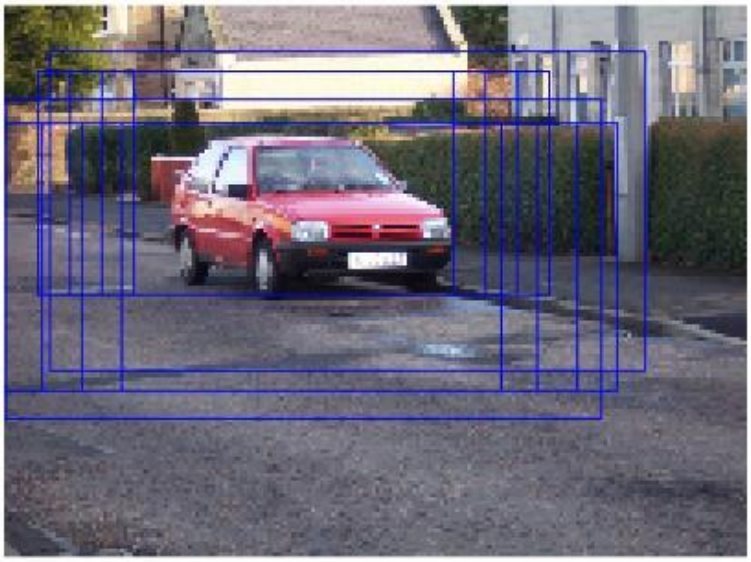
Resolution	Cell Size	Block Size	Detector Size	# orientations	Block stride	Detector Stride
1	(3, 2)	(6, 4)	(13, 7)	9	(2, 1)	(1, 1)
2	(4, 3)	(8, 6)	(26, 14)	18	(3, 2)	(2, 2)
3	(6, 4)	(12, 8)	(52, 28)	18	(4, 4)	(4, 4)
4	(8, 6)	(16, 12)	(104, 56)	18	(8, 4)	(8, 8)

Table2: Parameters at each resolution level of car detection. All parameters except orientations expressed as (width, height) pair in pixels.

The folder contains both training and testing code and parameters matrix which I got after training. The images used for train and test are listed in **.txt** files.

Results for car are shown below:





The code for car detection is in **extra_credit/part2/motorbike** folder. The parameter used for motorbike detection are shown in table below.

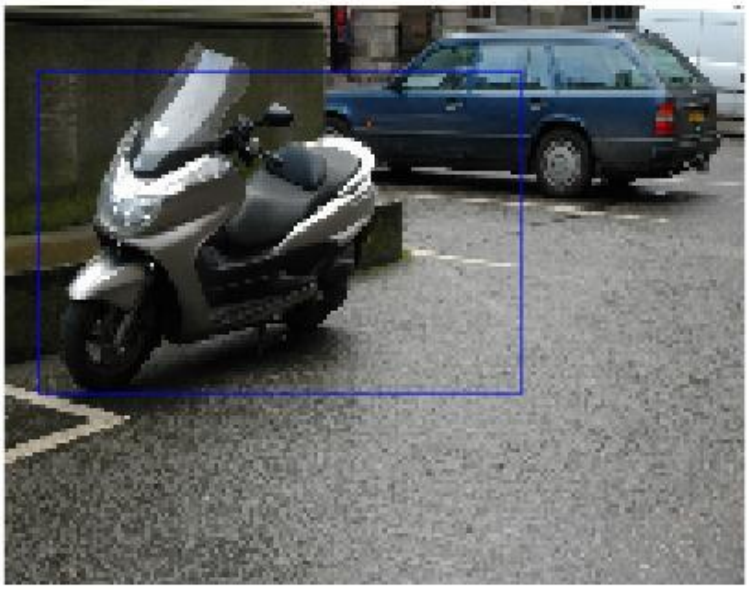
The parameters for Motorbike detection:

Resolution	Cell Size	Block Size	Detector Size	# orientations	Block stride	Detector Stride
1	(3, 3)	(6, 6)	(15, 10)	9	(3, 2)	(1, 1)
2	(5, 4)	(10, 8)	(30, 20)	15	(4, 3)	(2, 2)
3	(6, 6)	(12, 12)	(60, 40)	18	(6, 4)	(4, 4)
4	(8, 8)	(16, 16)	(120, 80)	18	(8, 4)	(8, 8)

Table3: Parameters at each resolution level of motorbike detection. All parameters except orientations expressed as (width, height) pair in pixels.

The folder contains both training and testing code and parameters matrix which I got after training. The images used for train and test are listed in **.txt** files.

Results for bike shown below:



I realized that motorbike give more accurate results because it has much more features like tyres/handle etc. as compared to car which is just a rectangle box with very less number of features.

References:

[1]. http://lear.inrialpes.fr/pubs/2005/DT05/

[2]. http://pascal.inrialpes.fr/data/human/

[3]. http://www3.cs.stonybrook.edu/~ial/content/papers/2007/wzhang-iccv2007.pdf