Vehicle Detection Project

The goals / steps of this project are the following:

- Perform a Histogram of Oriented Gradients (HOG) feature extraction on a labeled training set of images and train a classifier Linear SVM classifier
- Optionally, you can also apply a color transform and append binned color features, as well as histograms of color, to your HOG feature vector.
- Note: for those first two steps don't forget to normalize your features and randomize a selection for training and testing.
- Implement a sliding-window technique and use your trained classifier to search for vehicles in images.
- Run your pipeline on a video stream (start with the test_video.mp4 and later implement on full project_video.mp4) and create a heat map of recurring detections frame by frame to reject outliers and follow detected vehicles.
- Estimate a bounding box for vehicles detected.

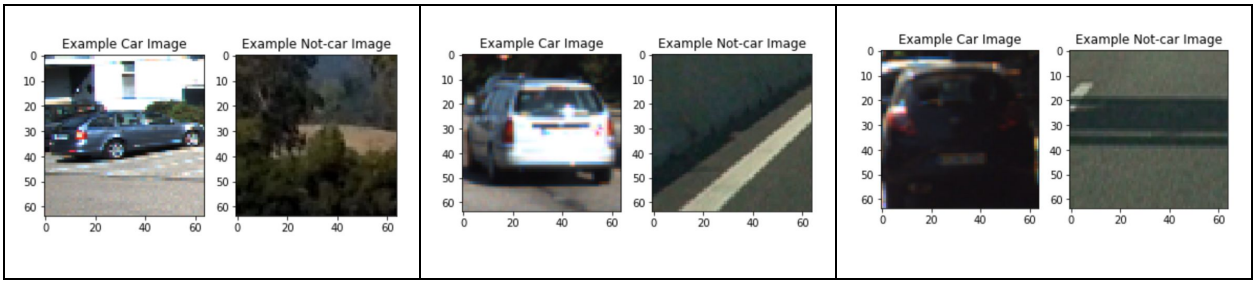
Histogram of Oriented Gradients (HOG)

Explain how (and identify where in your code) you extracted HOG features from the training images.

All useful functions were extracted to back.py file, just to make all simpler. The method get_hog_features is responsible for HOG features transformation.

The first step was to explore dataset. I used vehicles and non-vehicles datasets which were provided by Udacity, the images folder contains these datasets, I used the read_images method to read them and explore.

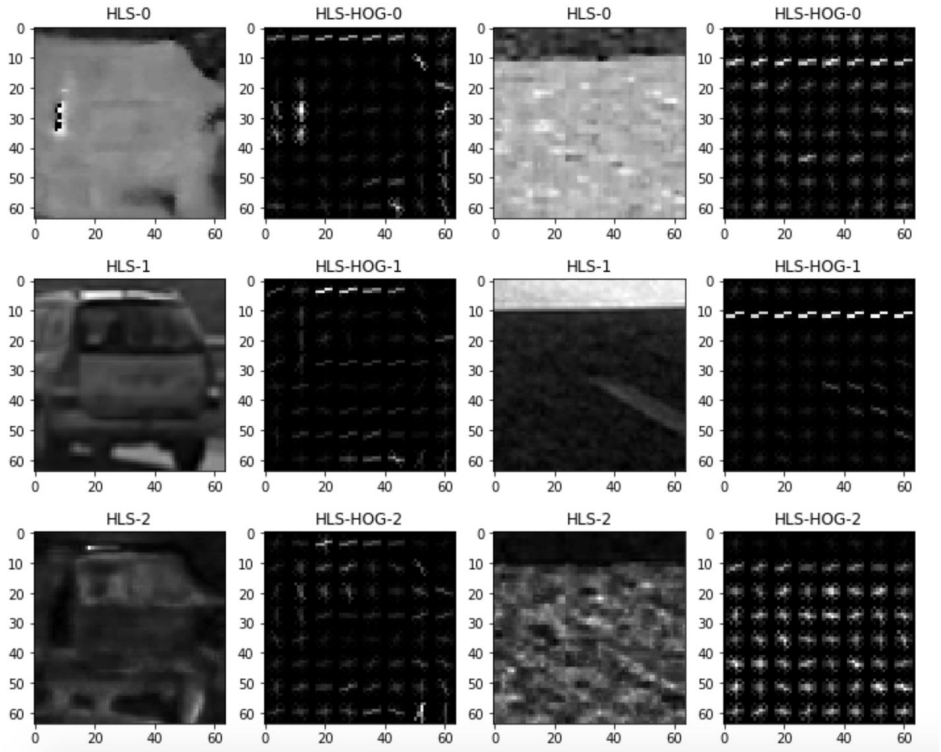
Here some examples.



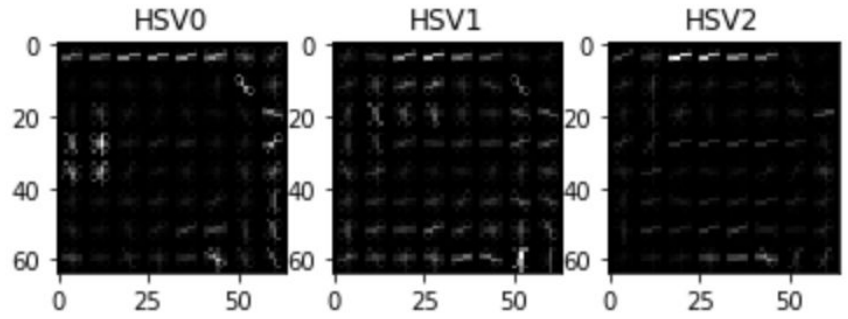
The code for HOG extraction contains in get_hog_features().

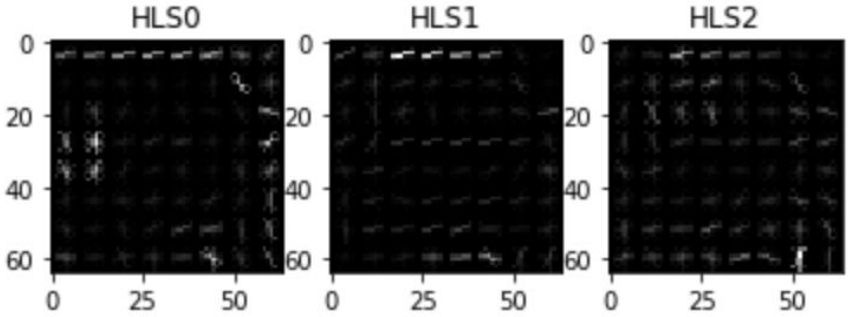
I then explored different color spaces and different skimage.hog() parameters (orientations, pixels_per_cell, and cells_per_block). I grabbed random images from each of the two classes and displayed them to get a feel for what the skimage.hog() output looks like.

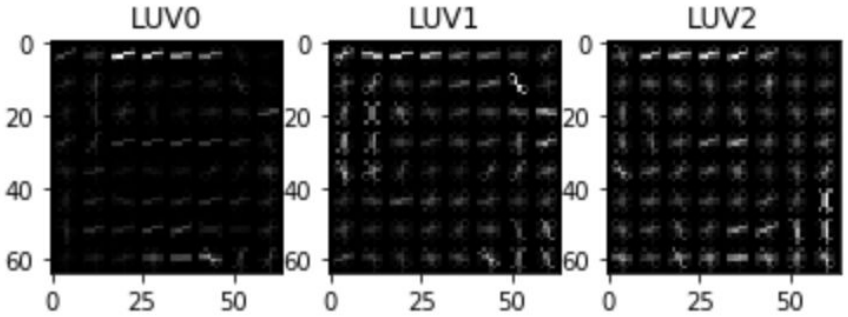
Here is an example using the HLS color space and HOG parameters of orientations=9, pixels_per_cell=(8, 8) and cells_per_block=(2, 2):



I decided to experiment with different color spaces also.







I choose LUV color space for further exploration.

Explain how you settled on your final choice of HOG parameters.

In order to set the final HOG parameters, I tried to find such ones that give best SVM results and do not explode dimension of the feature vector.

The method extract_features() contains HOG transform and color features extractions. I turned off color and tested the performance of SVM classifier. The lecture parameters were chosen as default ones. I tested them on first LUV channel because it provides the best contours of the car among others.

Default orientations number was the best, see table below.

orientations	feature vector length	extraction time(sec)	Training time(sec)	Test accuracy	delta
9	324	21.12	4.58	0.949	0
8	288	23.49	5.03	0.9445	-0.0045
4	144	21.78	3.83	0.9279	-0.0211
16	576	32.84	6.51	0.9471	-0.0019

I decided to increase pixels per cell in order to speed up extractraction of features.

Pixels per cell	feature vector length	extraction time(sec)	Training time(sec)	Test accuracy	delta
8	1764	29.09	7.1	0.9496	0
4	8100	67.36	45.68	0.9524	+0.0028
16	324	21.12	4.58	0.949	-0.0006

Changing number of cells per pixel will not improve performance a lot.

cells per block	feature vector length	extraction time(sec)	Training time(sec)	Test accuracy	delta
2	324	21.12	4.58	0.949	0
1	144	23.65	3.23	0.9423	-0.0067
4	144	25.97	3.94	0.9352	-0.0138

After these tests the final params for HOG transform are orients = 9, pix_per_cell = 16, cell_per_block = 2.

Both spatial and histogram features add more information and increase the accuracy.

spatial	color	feature vector length	extraction time(sec)	Training time(sec)	Test accuracy
false	false	324	21.12	4.58	0.949
true	false	1092	23.67	3.22	0.9595
false	true	516	33.5	1.85	0.9747
true	true	1284	34.33	2.46	0.9806

Finally I extracted HOG features from all channels.

Channels	feature vector length	extraction time(sec)	Training time(sec)	Test accuracy
0	1284	34.33	2.46	0.9806
All	1932	57.32	3.51	0.9907

Sliding Window Search

Describe how (and identify where in your code) you implemented a sliding window search. How did you decide what scales to search and how much to overlap windows?

The find_cars only has to extract hog features once and then can be sub-sampled to get all of its overlaying windows. Each window is defined by a scaling factor where a scale of 1 would result in a window that's 8 x 8 cells then the overlap of each window is in

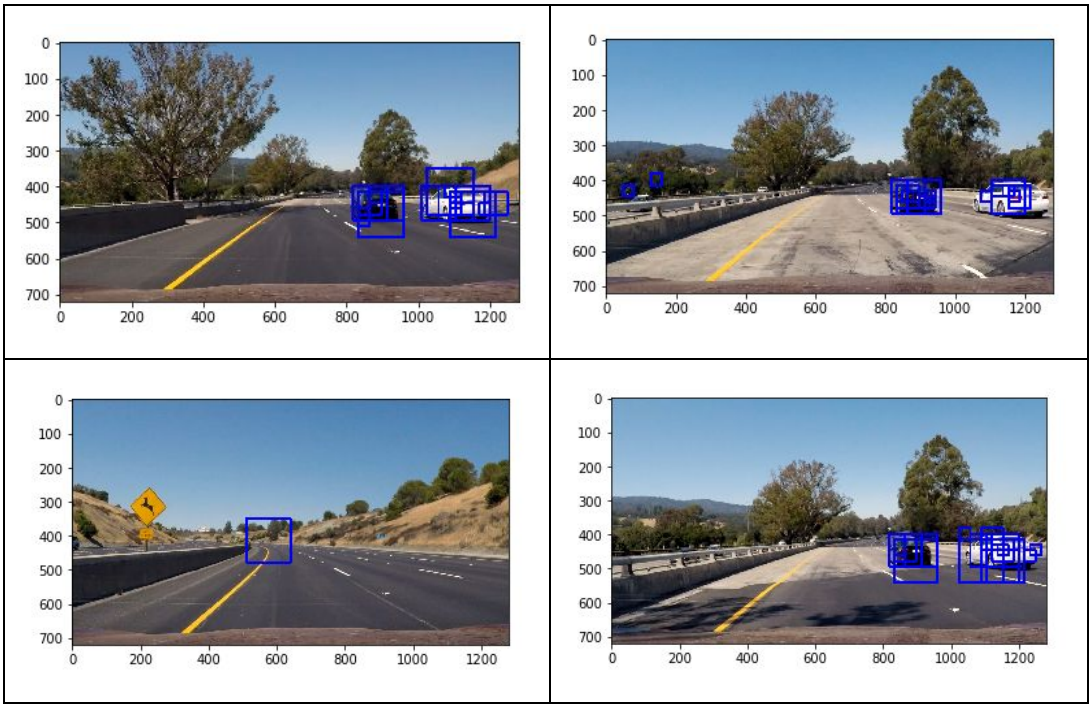
terms of the cell distance. This means that a cells_per_step = 2 would result in a search window overlap of 75%.

In order to detect cars with different sizes I added four scales to pipeline [0.5, 1.0, 1.5, 2.0] .

Show some examples of test images to demonstrate how your pipeline is working. What did you do to optimize the performance of your classifier?

You can find the pipeline method in the notebook. I have described optimization steps of classifier above already.

Here some examples.



Video Implementation

Provide a link to your final video output. Your pipeline should perform reasonably well on the entire project video (somewhat wobbly or unstable bounding boxes are ok as long as you are identifying the vehicles most of the time with minimal false positives.)

Here's a link to my video result .

Describe how (and identify where in your code) you implemented some kind of filter for false positives and some method for combining overlapping bounding boxes.

The pipeline method contains filtering and overlapping code. I decided to store last 10 heats and apply threshold method:

```
# save heat to global array
    indx = crt_ind % n
    heats[indx] = heat

smoothed = np.zeros_like(heat).astype(np.float)
    for heati in heats:
        if heati is not None:
            smoothed += heati

smoothed = apply_threshold(smoothed, 0.7 * n)
```

Description

The final bounds of the car are a smaller than actual ones, I could try other scaling parameter and try to change number of saved heats and threshold, but on my computer it took 51 minutes to process 51 seconds video. Extracting methods must be sped up.

I need also try out other videos and see how it goes.