Vehicle Detection Project

Report based on the standard template.

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.

HOG feature extraction

I combined all the feature extraction methods proposed in the lectures:

- Color space
- HOG Channel
- Spatial
- Histogram

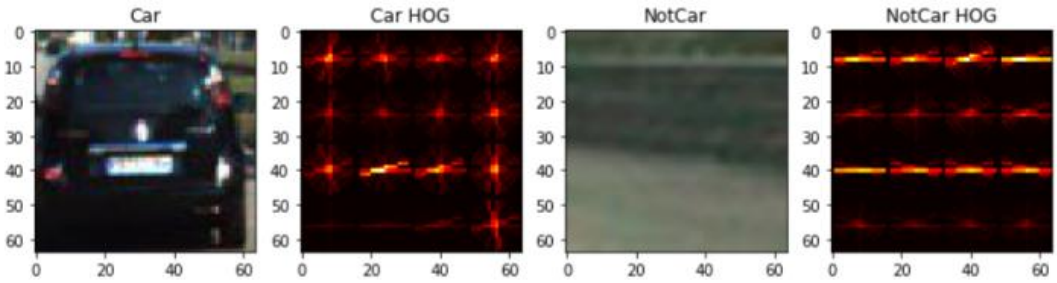
1. Color space

I explored the following color spaces (RGB, HSV, LUV, HLS, YUV, YCrCb), the best result were obtained with the YCrCb and LUV in that order.

2. HOG Channel

I tested all channel options separately (0, 1, 2, or ALL) and selected the ALL option in combination with:

- Orient: 8, 9, 11, 16
- Pixels per cell: 8, 16
- Cell per block: 2, 4



3. Spatial

I enable the spatial feature selection using a spatial binning dimension of (16, 16); other dimension tested were (32, 32).

4. Histogram

I enable the histogram feature selection using 32, 16 bins; finally selected using the 32 option.

5. Parameters Used

The final parameters are:

```
colorspace = 'YCrCb'
orient = 11
pix_per_cell = 16
cell_per_block = 2
hog_channel = 'ALL'
spatial_size = (16, 16)
hist_bins = 32
spatial_feat = True
hist_feat = True
hog_feat = True
```

Parameters where compared with this thread and this one.

The HOG feature extraction is ready to be trained on a labeled training set of images.

5. Classifier

The first step was to load all image data provided to create a classifier, car and non_cars separately.

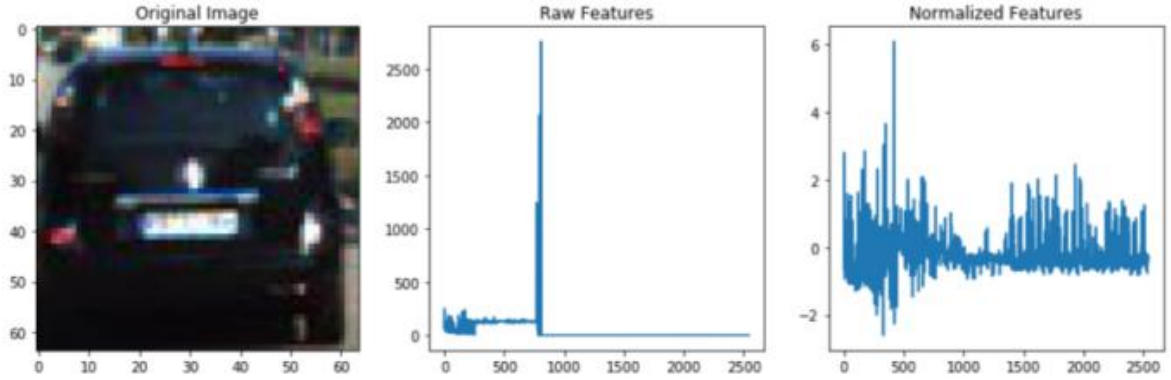
The testing was done with all data vs 1/3 of all data to filter the time-series data on the GTI* folder to try to skip images to break this pattern.

No significant changes were noticed so I continued with the whole dataset. The data was shuffled and divided; 80% for training and 20% for test.

I trained a linear SVM using a GridSearch to optimize the C parameter (selected value is 0.1). Then I combined it with the selected HOG features. The execution got an accuracy of 98%:

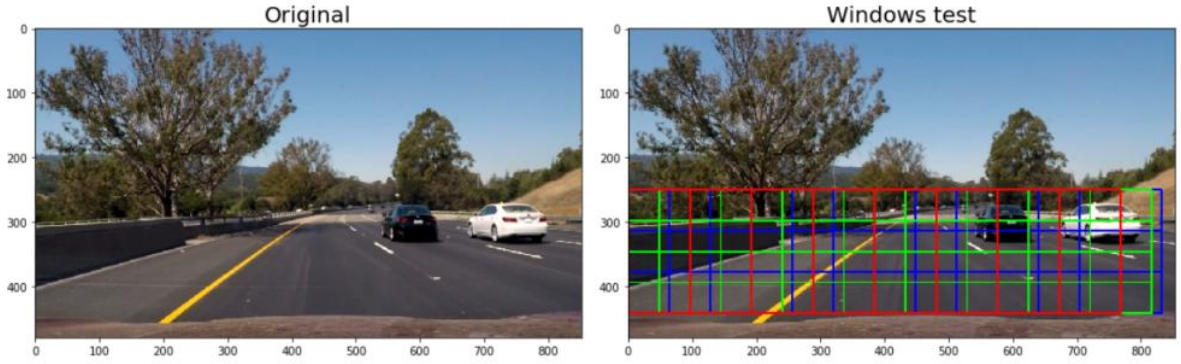
```
Training started
70.85 seconds to extract HOG features...
Min:0.0 Max:4096.0
GridSearchCV took 63.37 seconds for 8 candidate parameter settings.
Model with rank: 1
Mean validation score: 0.991 (std: 0.001)
Parameters: {'C': 0.001}
Model with rank: 2
Mean validation score: 0.991 (std: 0.001)
Parameters: {'C': 0.01}
Model with rank: 3
Mean validation score: 0.990 (std: 0.000)
Parameters: {'C': 0.1}
Best C value: 0.1
Classifier LinearSVC(C=0.1, class_weight=None, dual=True, fit_intercept=True,
    intercept_scaling=1, loss='squared_hinge', max_iter=1000,
    multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
    verbose=0)
Using: 11 orientations 16 pixels per cell and 2 cells per block
Feature vector length: 2052
5.4 seconds to train SVC
Test Accuracy of SVC = 0.9887
My SVC predicts: [1. 0. 0. 0. 1. 0. 0. 1. 1. 0.]
For these 10 labels: [1. 0. 0. 0. 1. 0. 0. 1. 1. 0.]
0.00059 seconds to predict 10 labels with SVC
Dataset: 8792 cars, 8968 non_cars, 14208 for training, 3552 for test
Training ended
```

An example of a random car and raw and scaled features:

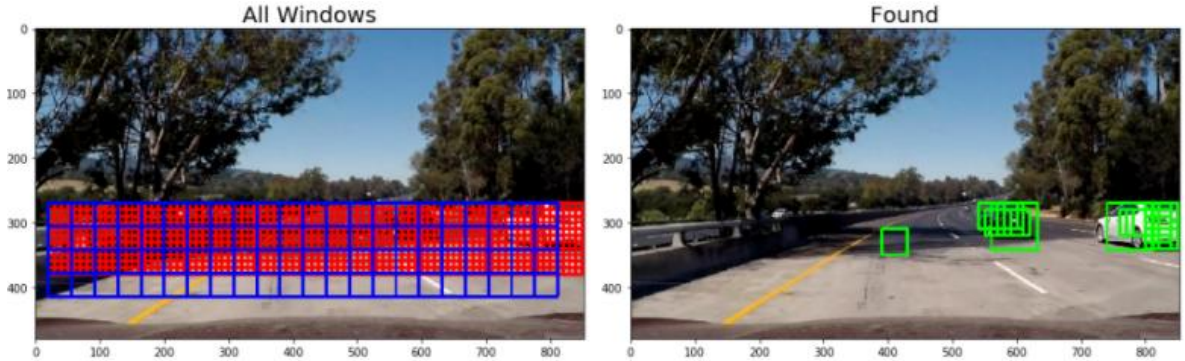


Sliding Windows

The sliding-window technique used will search for the lower half of the image with different patch sizes with a 1.5 scale size, other scale size tested but discarded were (0.75, 1, 1.25, 1.75, 2):



Later on when implemented on the pipeline two windows sizes were selected with different overlay values to maximize possible matches. For bigger image patches the overlay is 0.5 and for smaller patches the overlay is 0.75. The result is the following; left side the search space and right side the classifier matches:



The region of interest changes for each window size, for all it starts around half of the Y size (300th pixel) from the top and spans vertically to:

- bottom minus 80 pixels for bigger windows.
- 3/4 of its Y size for smaller windows.

Pipeline

The pipeline searches YCrCb 3-channel HOG features plus spatially binned color and histograms of color in the feature vector (complete_pipeline function):

- Get a list of all the windows (getROIWindows function).
- Use the classifier and check which window contains a car (search_windows function calling single_img_features and validating the prediction value).
- Add 1 to an empty heatmap image.
- Apply a threshold (4 value used) to only show areas which are 'warm' enough.
- Use the scipy label function to draw boxes (draw_labeled_bboxes function)

Below are images showing the different steps, applied on the test images.

1. Predict Cars

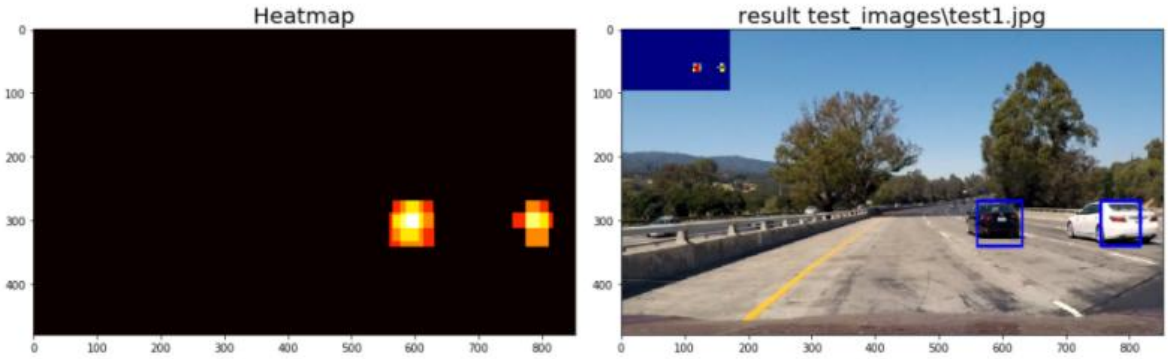
The LinearSVC includes a built-in decision_function method, which returns a confidence score based on how far a data point is from the decision boundary, so as higher values equate to higher confidence predictions that can be thresholded. I added a threshold value to further distinguish between positive and negative classifications. This change removed many of the false-positive detections.

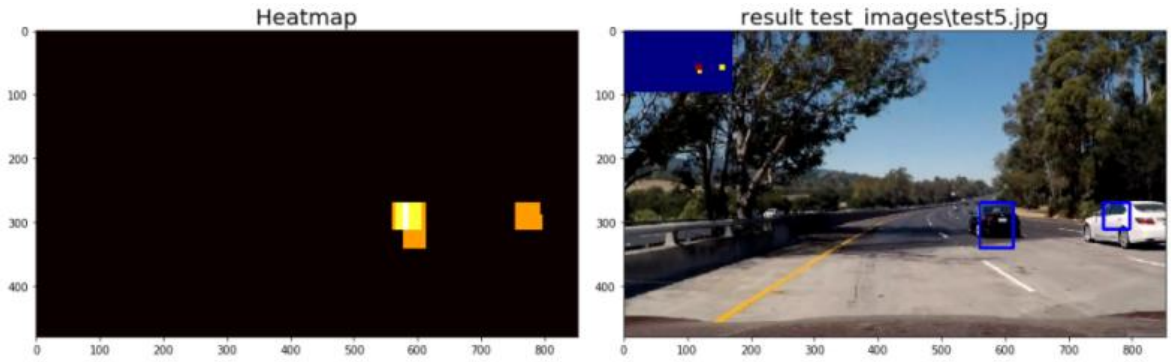
The predictions based on the sample images:

Notice still some false positive values:

2. Heatmap

In several cases windows are detected that do not contain any vehicles. I removed those areas (false positives) with a heatmap and the apply_threshold function. The add_heat function is used to accumulate evidence of detections; the draw_labelled_boxes function will display the aggregated results:





1. Video

The image processing pipeline applied to the first image worked fine. But, when the pipeline was applied to the second video, in some cases the detection was wrong; several frames were out of scope. I applied this idea discussed in the forum to get a better heatmap. I used a queue to keep the last 7 frames for detection and summed all, then changed the threshold accordingly (7*3=21) so I got better result.

Considerations

The removal of false positives is quite a challenge. I tested several combinations for the classifier, in combination with different windows sizes and thresholds.

I combined the following parameters but still could not get a better pipeline

- # of orientation bins
- grid of cells
- cell sizes
- adding overlap between cells
- block normalization

I feel that the HOG features parameters are not robust enough to the variations in shape and colors as for the second video it still has some issues with false positives not correctly filtered and in some cases the white car is not correctly detected. It can be verified for a fraction of time (from second 25 to 28).

I discarded the use of hard negative mining because when I added more samples to the test set and adjusted the threshold I got better results.

An alternative option would be to apply other techniques besides HOG features, such as a specific Neural Network targeted to vehicle detection. Based on the forum some options are YOLO, SSD.