Writeup Template

You can use this file as a template for your writeup if you want to submit it as a markdown file, but feel free to use some other method and submit a pdf if you prefer.

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.

Rubric Points

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

Writeup / README

1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. Here is a template writeup for this project you can use as a guide and a starting point.

You're reading it!

Histogram of Oriented Gradients (HOG)

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

I extract the HOG of the features in multiple places. First in get_hog_features in the SVM training. Then inside find_cars. The code was mostly from the Udacity lessons with some modifications.

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

I iterated a lot on the parameters until I just found something that more or less works. I am using:

color_space = 'YCrCb' # Can be RGB, HSV, LUV, HLS, YUV, YCrCb

orient = 11

pix_per_cell = 16

cell_per_block = 2

hog_channel = 'ALL' # Can be 0, 1, 2, or "ALL"

spatial_size = (32, 32) # Spatial binning dimensions

hist_bins = 16   # Number of histogram bins

spatial_feat = True # Spatial features on or off

hist_feat = True # Histogram features on or off

hog_feat = True # HOG features on or off

y_start_stop = [380, None] # Min and max in y to search in slide_window()

3. Describe how (and identify where in your code) you trained a classifier using your selected HOG features (and color features if you used them).

I trained a linear SVM using:

color_space = 'YUV' # Can be RGB, HSV, LUV, HLS, YUV, YCrCb

orient = 11

pix_per_cell = 16

cell_per_block = 2

hog_channel = 'ALL' # Can be 0, 1, 2, or "ALL"

spatial_size = (32, 32) # Spatial binning dimensions

hist_bins = 64   # Number of histogram bins

spatial_feat = True # Spatial features on or off

hist_feat = True # Histogram features on or off

hog_feat = True # HOG features on or off

y_start_stop = [380, None] # Min and max in y to search in slide_window()

Sliding Window Search

1. 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?

I used the find_cars algorithm to find the cars in the images, but it is still lacking. Hence i am looking for hints in this submission

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

Examples are available in the attached notebook

Video Implementation

1. 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: https://youtu.be/WzbgZtzRC1k

2. 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.

I used the heat map to filter out false positives. I then compared the rectangles across frames. I also did hard negative mining (I am including the extra images in my submission)

Here are six frames and their corresponding heatmaps:

They can be seen in the attached notebook and html

Here is the output of `scipy.ndimage.measurements.label()` on the integrated heatmap from all six frames:

This can be attached notebook

Here the resulting bounding boxes are drawn onto the last frame in the series:

Can be seen in the attached notebook

Discussion

1. Briefly discuss any problems / issues you faced in your implementation of this project. Where will your pipeline likely fail? What could you do to make it more robust?

I am still having some minor issues with this submission. I am hoping to get tips as to how to improve this submission and then I will complete the write up for a (hopefully) final submission