CarND Vehicle Detection Project Writeup

Tiffany Huang

September 6, 2018

1 Vehicle Detection Project

This project consists of the following steps/tasks:

- Perform a Histogram of Oriented Gradients (HOG) feature extraction on a labeled training set of images and train a Linear SVM classifier.
- Optionally, apply a color transform and append binned color features, as well as histograms of color, to the HOG feature vector.
- Normalize the features and randomize a selection for training and testing.
- Implement a sliding-window technique and use the trained classifier to search for vehicles in images.
- Run the pipeline on a video stream 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.

My project code can be found here: `https://github.com/tahuang/Vehicle_Detection`. Next, I will consider each of the rubric points individually and describe how I addressed each point in my implementation.

2 Histogram of Oriented Gradients (HOG)

2.1 HOG Feature Extraction

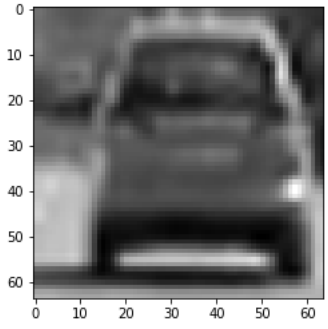
The code for extracting HOG features from the training images is in section 1 of `vehicle_detection.ipynb` in the `get_hog_features` and `extract_features` functions. I started by reading in all the `vehicle` and `non-vehicle` images. I then used `skimage.hog()` to extract the HOG features. Here is an example of one of each class:

(a) Car

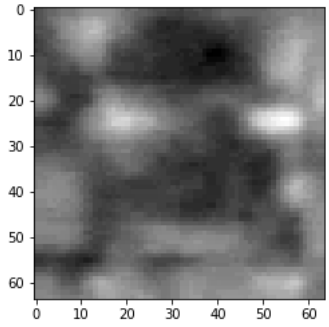
(b) Not a Car

2.2 HOG Parameter Selection

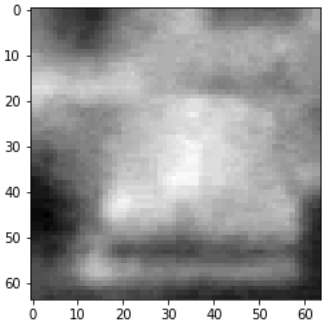
I settled on my final choice of HOG parameters of `orient=9`, `pix_per_cell=8`, and `cells_per_block=(2,2)` by trying various combinations of parameters and seeing which ones resulted in the highest score when testing the Linear SVM classifier. This set of parameters had the best test performance of 0.9924. Here is an example of what the HOG features look like with these parameters for a car image using the YCrCb color space:



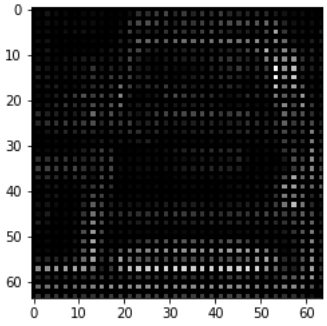
(a) Car YCrCb Ch 1



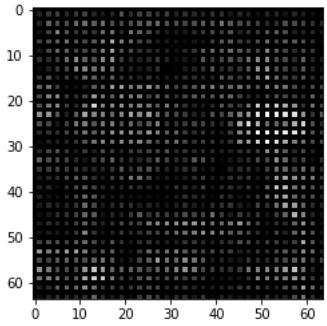
(b) Car YCrCb Ch 2



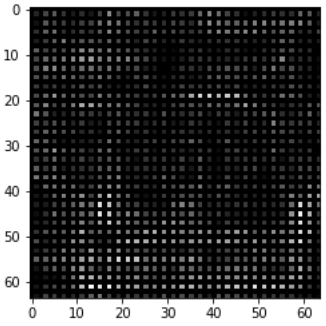
(c) Car YCrCb Ch 3



(d) Car HOG Ch 1



(e) Car HOG Ch 2

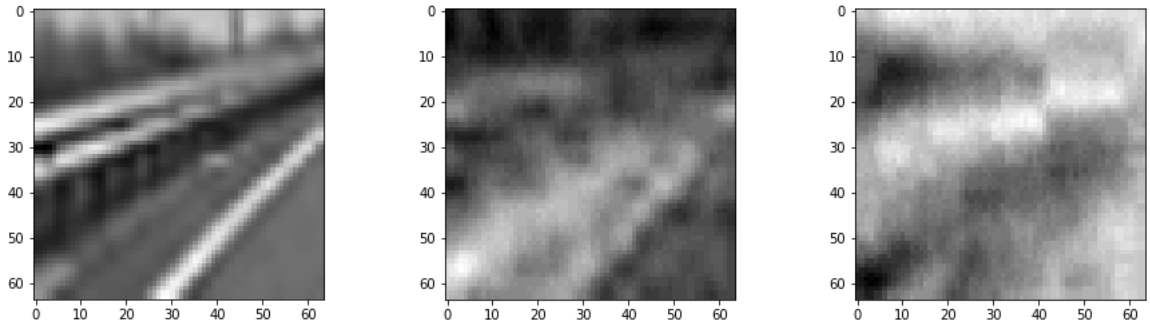


(f) Car HOG Ch 3

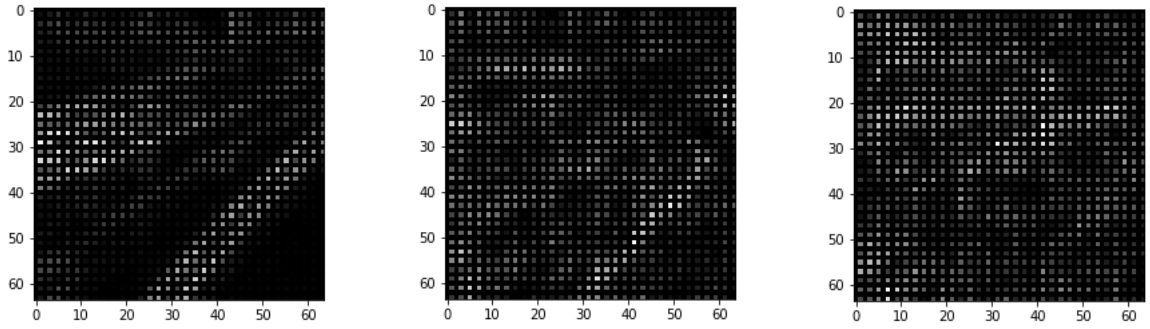
These are the HOG features for a non-vehicle image:

2.3 Training a Classifier

I trained a Linear SVM classifier in section 2 of `vehicle_detection.ipynb` where I used spatial color features, color histogram features, and HOG features. I randomly split the training

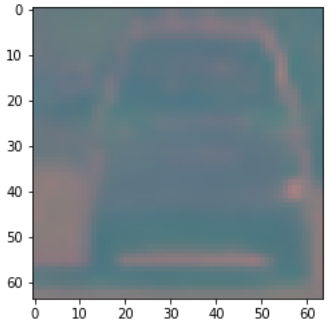


(a) Non-Car YCrCb Ch 1 (b) Non-Car YCrCb Ch 2 (c) Non-Car YCrCb Ch 3

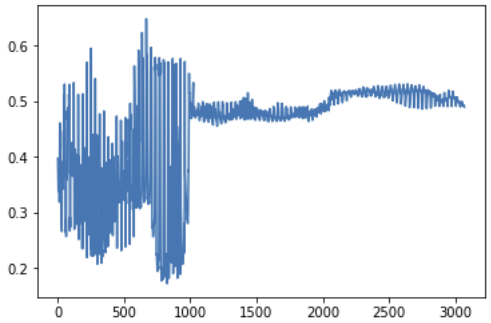


(d) Non-Car HOG Ch 1 (e) Non-Car HOG Ch 2 (f) Non-Car HOG Ch 3

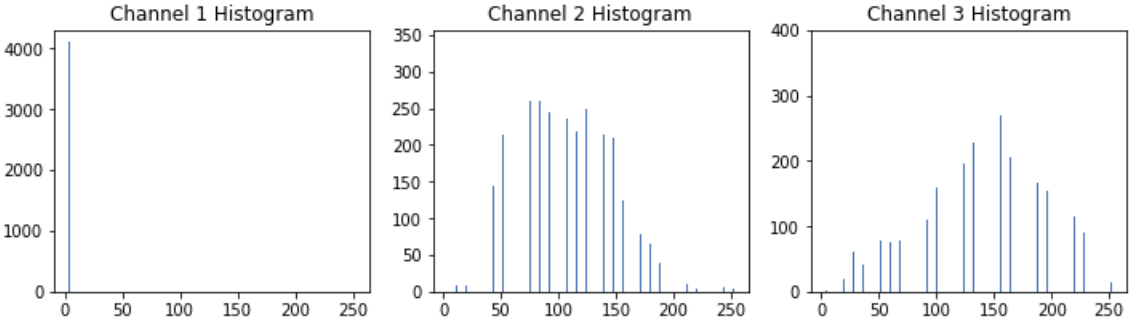
data into training and test sets, then normalized all the data by fitting a `StandardScaler` to the training data split. I then did a parameter search on the classifier to find the best `C` value and tested the classifier on the split off test set. The best C value was 0.1 and the classifier test accuracy was 0.9924. Below is an example of the spatial color features and the color histogram features:



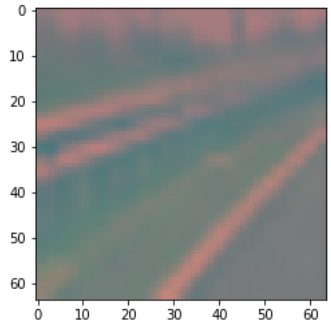
(a) Car YCrCb Image



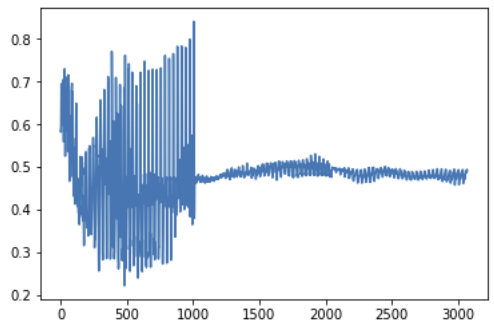
(b) Car YCrCb Spatial Color Features



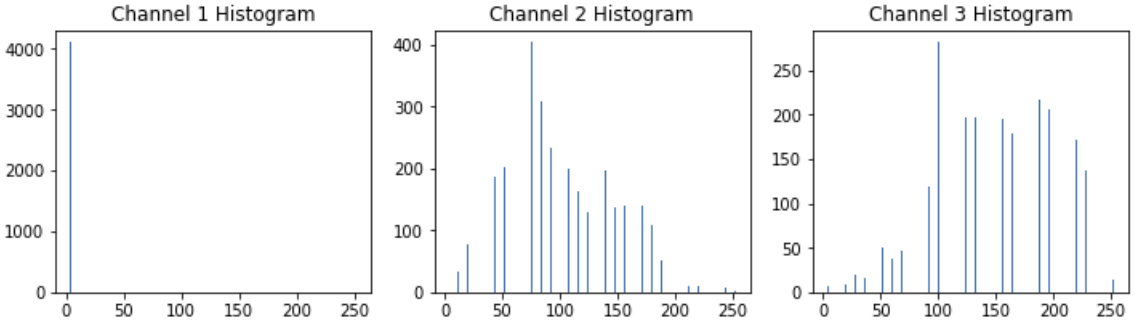
(c) Car YCrCb Color Histogram



(a) Non-Car YCrCb Image



(b) Non-Car Spatial Color Features



(c) Non-Car Color Histogram

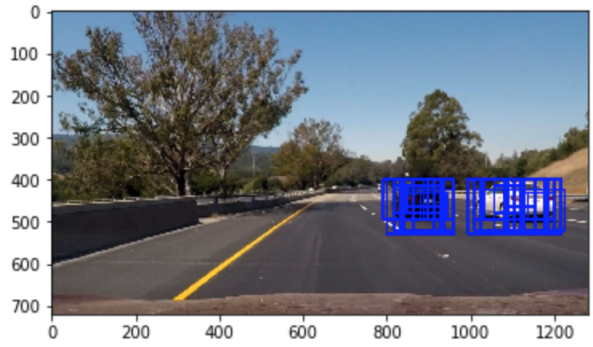
3 Sliding Window Search

3.1 Sliding Window

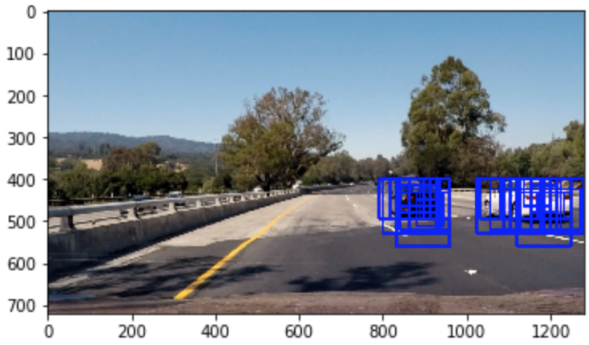
I implemented the sliding window search in section 3 of `vehicle_detection.ipynb`. I tested a couple different scales and window overlaps and picked the parameters that produced the best performance on the test images. The scales I used were 1, 1.5, and 2 and the window search was 8 pixels per cell and 8 cells per window with a cell step size of 2.

3.2 Working Pipeline

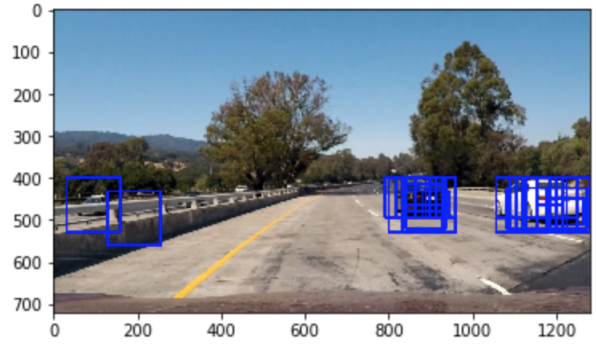
For my pipeline, I used the YCrCb color space, 3-channel HOG features, spatially binned color features, and color histogram features to detect vehicles. I optimized the classifier by trying different parameter sets and choosing the best one. I also added the spatial and histogram features in addition to the HOG features to increase robustness. Here are some example images of the pipeline working:



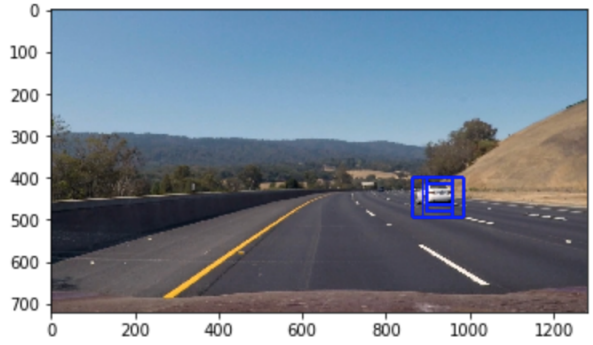
(a) Example 1



(b) Example 2



(c) Example 3



(d) Example 4

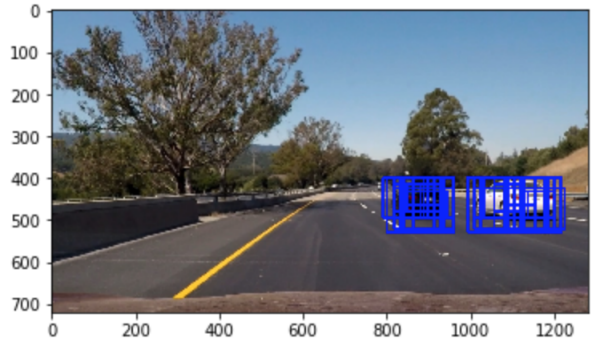
4 Video Implementation

4.1 Video

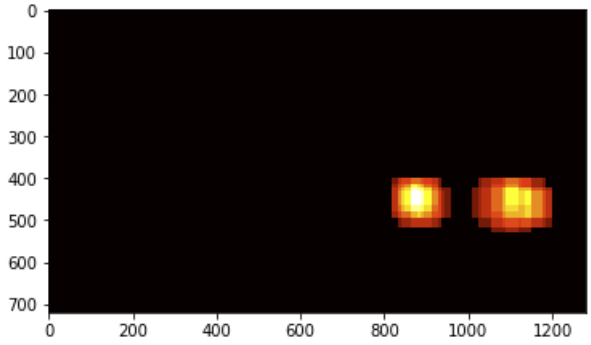
My video can be found in the `vehicle_detection_project_video.mp4` file.

4.2 False Positives

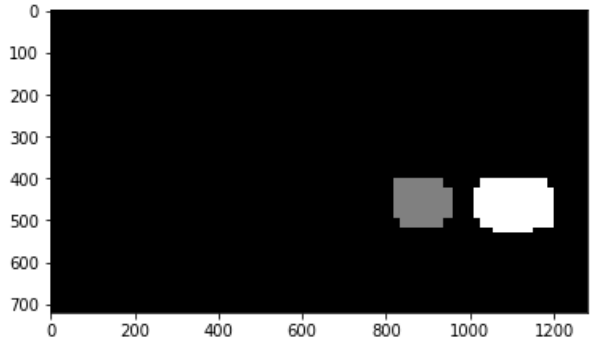
To filter out false positives and combine overlapping bounding boxes, I created a heatmap for detections and thresholded the heatmap to throw out weak detections (≤ 2 detections). The implementation of the heatmap functions can be found in section 4 of `vehicle_detection.ipynb`. I then used `scipy.ndimage.measurements.label()` to identify individual blobs in the heatmap. Once the vehicle blobs were detected, I drew bounding boxes over the blobs. Here is an example of the bounding boxes found, the heatmap generated, the labeled heatmap, and the final filtered result.



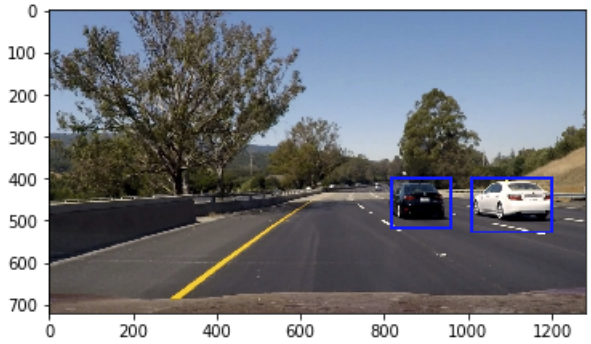
(a) Non-filtered Bounding Boxes



(b) Heatmap



(c) Labeled Blobs



(d) Filtered Result

5 Discussion

The approach I used was to convert the images into the YCrCb color space, extract spatially binned color features, color histogram features, and HOG features, and use a Linear SVM classifier to detect vehicles. I believe the YCrCb color space worked out well because the luminance (Y) component captures the idea that cars are metallic and usually brighter than the road surface and surrounding objects. The Cr and Cb components capture the blueish and reddish tints that many dark-colored cars contain. The HOG features really capture the shape of the car and help to distinguish it from similarly colored objects in the background. The pipeline might fail if we see cars that were not in our training set. For instance, if there was a dump truck on the road that we had never seen before in the training set, it is possible we would not be able to identify it if the HOG features were quite different. The pipeline

also fails for cars that are far away or occluded because it is no longer possible to distinguish their distinctive gradient features. The pipeline might also fail if we see a car from a different angle than we have seen before in our training data because the newly angled car could have different spatial color and HOG features that do not match other cars in the training set. To make the classifier more robust, we could augment the training data with all types of cars and angles of cars. If we did this, we would also have to add more non-car examples so that the training data is not biased towards a vehicle classification.