Vehicle Detection Project

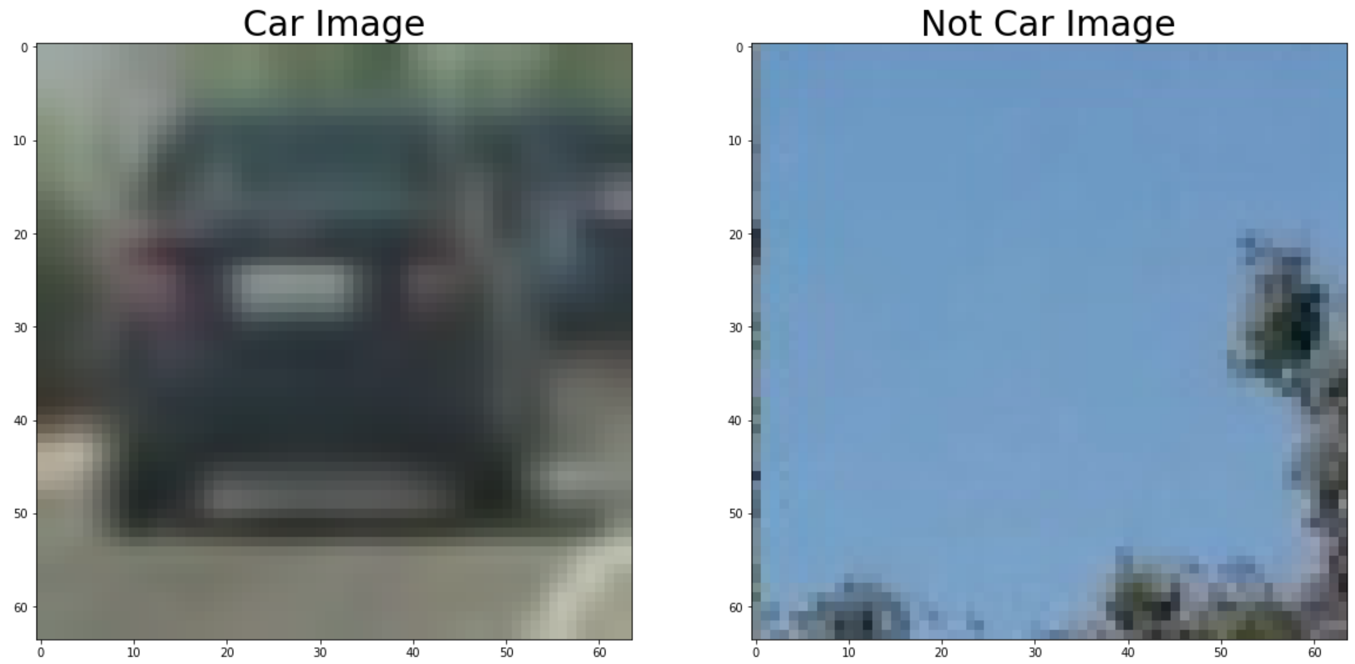
The goals / steps of this project are the following:

- Perform normalized Histogram of Oriented Gradients (HOG) feature and color feature extraction on a labeled training set of images and train and test a classifier Linear SVM classifier
- Implement a sliding-window technique and use trained classifier to search for vehicles in images.
- Run 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.

Histogram of Oriented Gradients (HOG)

1. Extracted HOG features from the training images

The code for this step is contained in the second code cell of the IPython notebook. I started by reading in all the `vehicle` and `non-vehicle` images. Here is an example of one of each of the `vehicle` and `non-vehicle` classes:

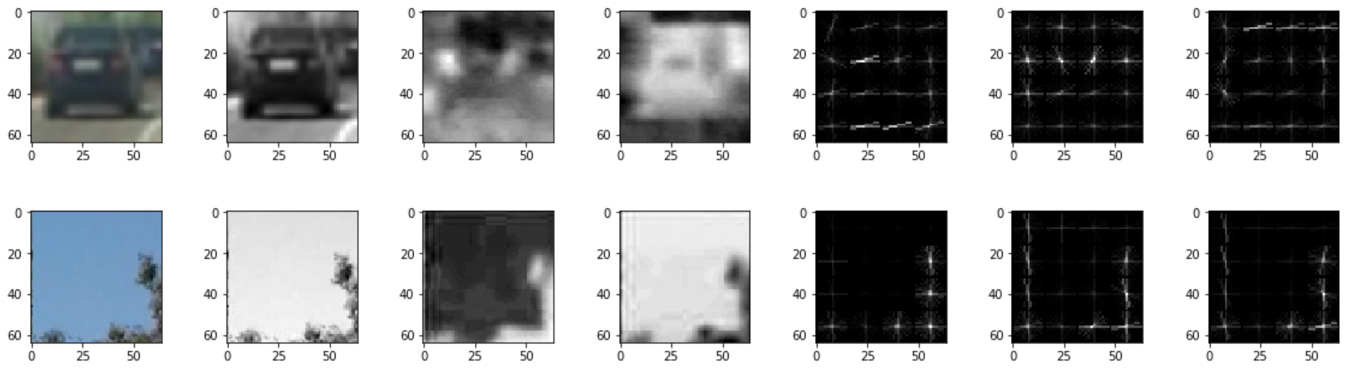


2. Final choice of HOG parameters

I then explored different color spaces and different `skimage.hog()` parameters (`orientations`, `pixels_per_cell`, and `cells_per_block`).

I tried various combinations of parameters and I end up using the `YCrCb` color space and HOG parameters of `orientations=12`, `pixels_per_cell=(16, 16)` and `cells_per_block=(2, 2)` because these parameters gives me best result.

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.



3. Trained a classifier using both HOG features and color features

I trained a linear SVM using the combination of color features and HOG features. I found out that HOG features is quite accurate, however, it has some problem with certain color. For example if I only use HOG feature, I have some problem with detecting the white car in the project video. I also use HOG color from all channels to train my classifier because this can get me higher test accuracy(from 95% to 99%).

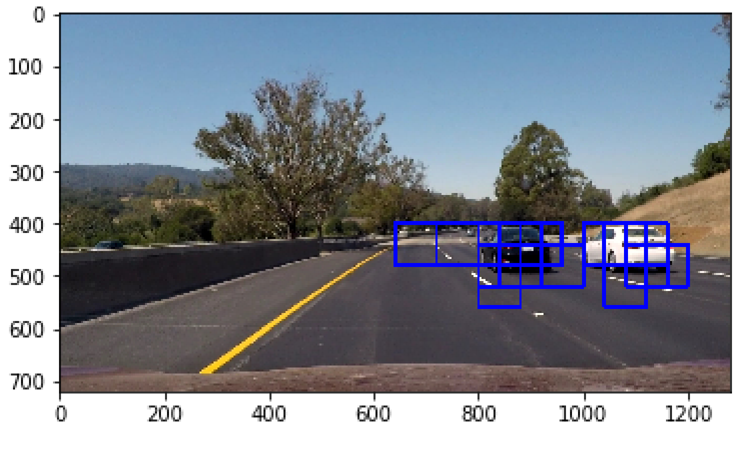
Sliding Window Search

1. Sliding Window Search Parameter

I search the region between y=[400,600] since this is the only region where cars shoud appear. I use window size 80x80 and overlap 50%, resize it to 64*64 and feed it to the classifier. By using a larger window, I got more accurate bounding box and less processing time.

2. Examples of Test Image

Ultimately I searched on two scales using YCrCb 3-channel HOG features plus spatially binned color and histograms of color in the feature vector, which provided a nice result. Here is example image:



Video Implementation

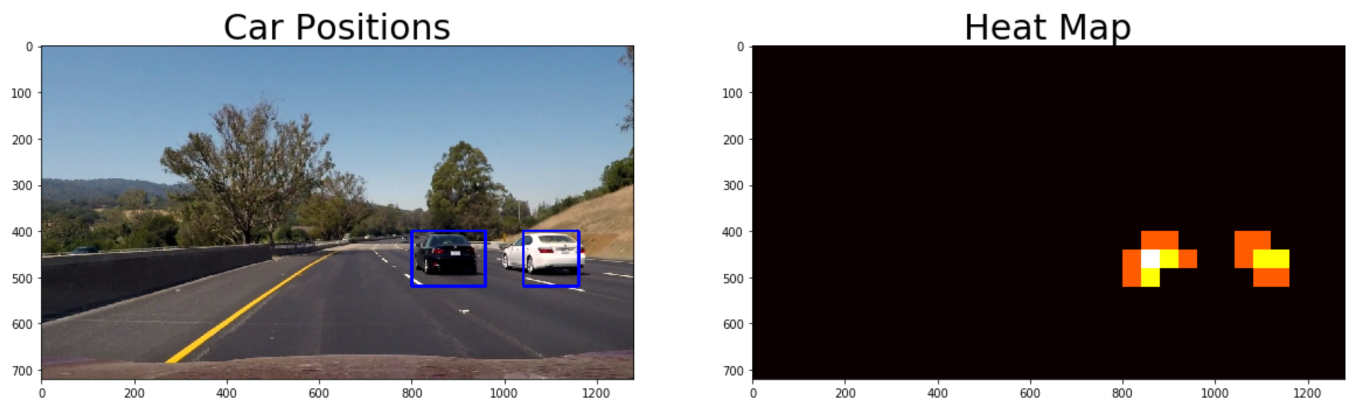
1. Final Video Output

Here's a link to my video result (https://youtu.be/4rLj6TLulTY)

2. False Positive

I recorded the positions of positive detections in each frame of the video. I use collections.deque() data type to record hot windows from previous n frames. From the positive detections I created a heatmap and then thresholded that map to identify vehicle positions. I then used scipy.ndimage.measurements.label() to identify individual blobs in the heatmap. I then assumed each blob corresponded to a vehicle. I constructed bounding boxes to cover the area of each blob detected.

Here's an example result showing heatmap from a frame of video on the right, the result of scipy.ndimage.measurements.label() and the bounding boxes then overlaid on the frame of video which shows on the right:



Discussion

For this project, I spend a lot of time on tuning the parameters. From the selection of interesting area select to windows sizes, each different parameter may cause some false positives. False positives have to be solved by tuning the buffer frame size and the threshold. Only when all parameters work good, the result is stable enough. With this approach, it's hard to tune the pipeline to adapt different road or weather conditions and even with larger window size, I got only 0.5 fps on my computer. I also tried this project video with faster R-CNN and Tiny yolo on my computer, which give me 7 fps and 30 fps result, so I think the CNN based solutions are much better and faster to solve this kind of problem.