Vehicle Detection & Tracking

Files Submitted:

- ObjectDetection Package which contains all the code files described.
- Output_images directory has all the output of running the pipeline on test images
- Writeup_report.pdf for the project report
- demo.mp4 has the project video result
- various .npy and .pkl files contain the features and the model saved into files

Extracting HOG Features, Spatial & Color Features

The code for Color Features Extraction is present in ObjectDetection/FeatureExtraction/ColorHistogramFeatureExtractor.py

The code for Spatial Color Features Extraction is present in ObjectDetection/FeatureExtraction/SpatialColorFeatureExtractor.py

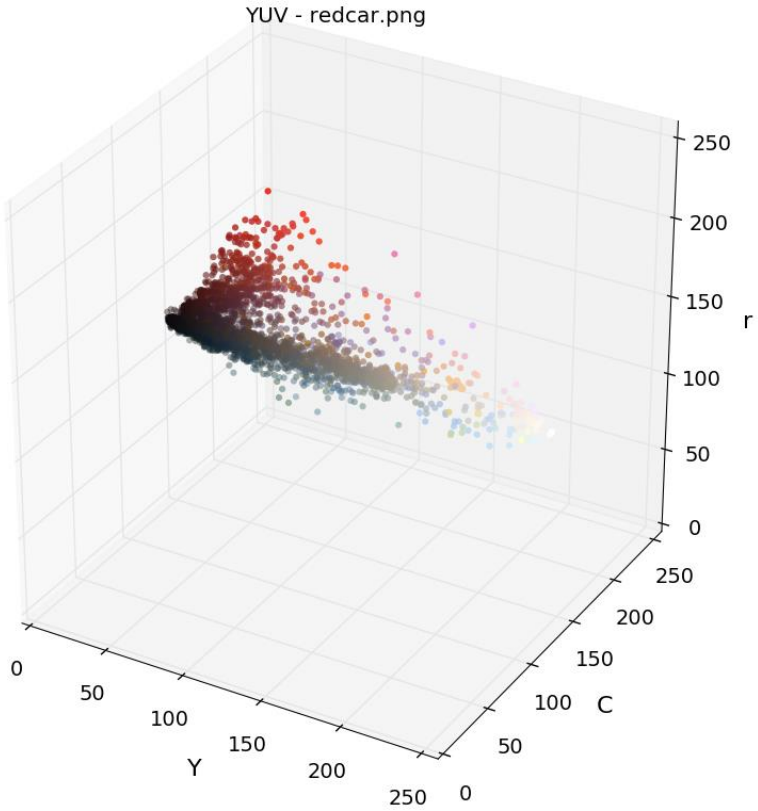
The code for HOG features extraction is present in ObjectDetection/FeatureExtraction/HogFeatureExtractor.py

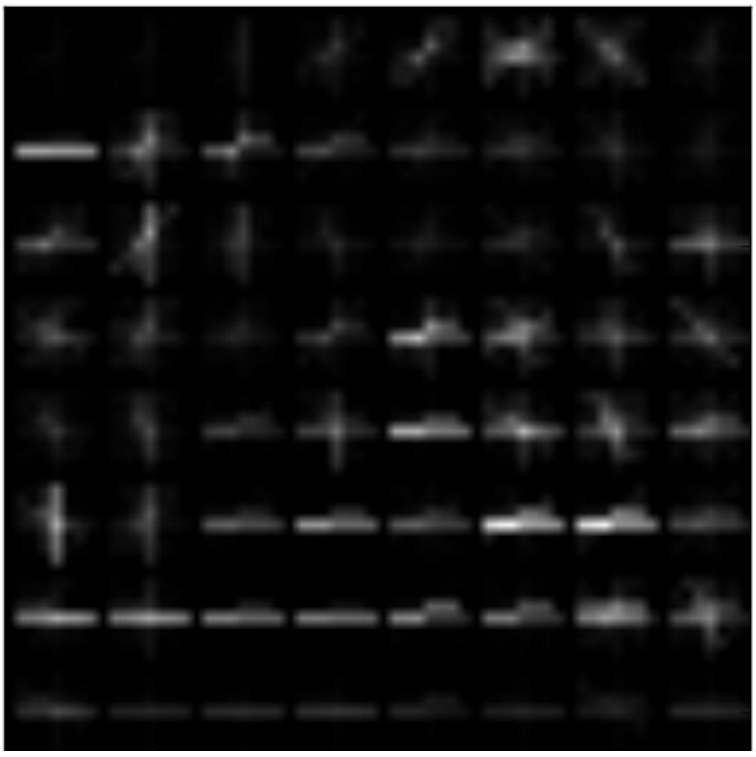
I started by reading in all the vehicle and non-vehicle images from ObjectDetection/ImageReader.py. Here is an example of one of each of the vehicle and non-vehicle classes:

I then explored different color spaces and chose the YCrCb color space as I felt it was the one which could consistent differentiate car and non car images. I also explored 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 YCrCb color space and HOG parameters of orientations=8, pixels_per_cell=(8, 8) and cells_per_block=(2, 2):


YUV - redcar.png
250
200
150
r
100
50
0
250
200
150
100
C
50
0
0
50
100
150
200
250
Y



Final Choice of HOG Parameters

For choosing the right HOG parameters I have experimented a lot until I got a very good accuracy on both the training and validation set during the training phase.

The best parameters I choose which were giving the highest accuracy are:

- orient = 12
- pix_per_cell = 8 # HOG pixels per cell
- cell_per_block = 1 # HOG cells per block
- hog_channel = "ALL" # Can be 0, 1, 2, or "ALL"

The parameters I chose for the spatial color features are:

- spatial_size = (48, 48)

The parameters I chose for the Color Histogram features are:

- hist_bins = 32

Training a Classifier:

I trained using a Linear SVC. The code for the training is present in ObjectDetection/TrainPhase.py

I split the data into training and validation sets – Training set was 80% of the data and validation was 20%. I used a standard scaler in order to normalize the data. I played around with the parameters for the LinearSVC using GridSearchCV and found the best paramets were C = 1 and loss = ‘hinge’.

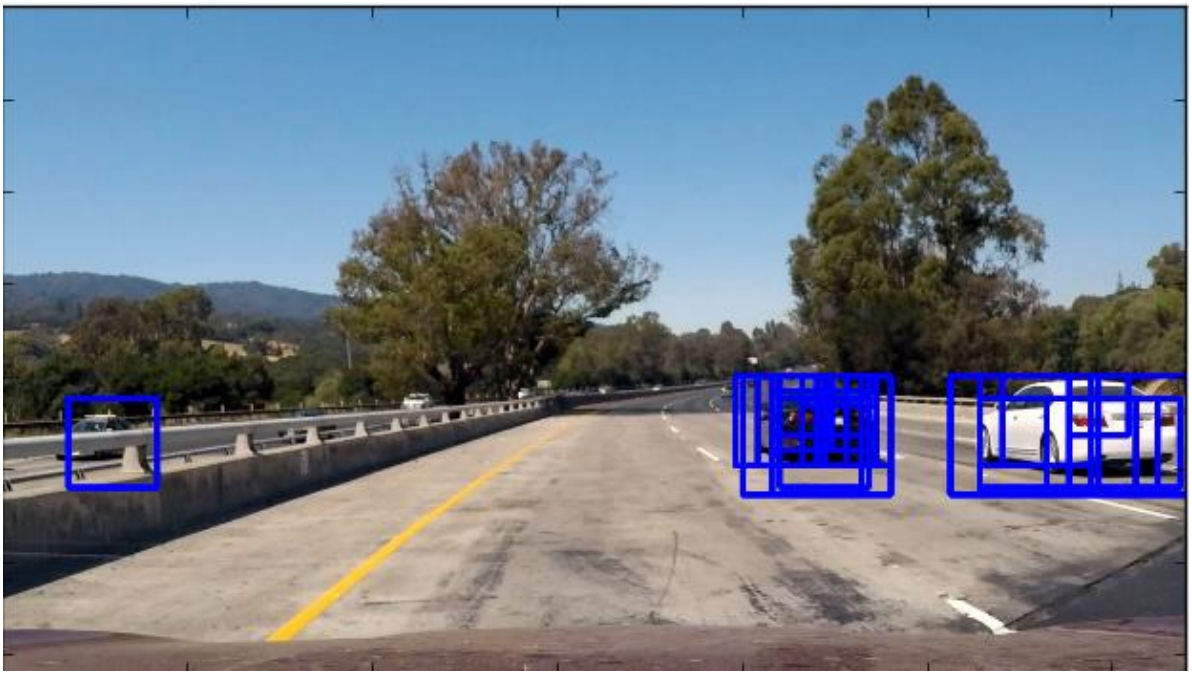
The model was trained and it achieved a test accuracy of 98.81 %.

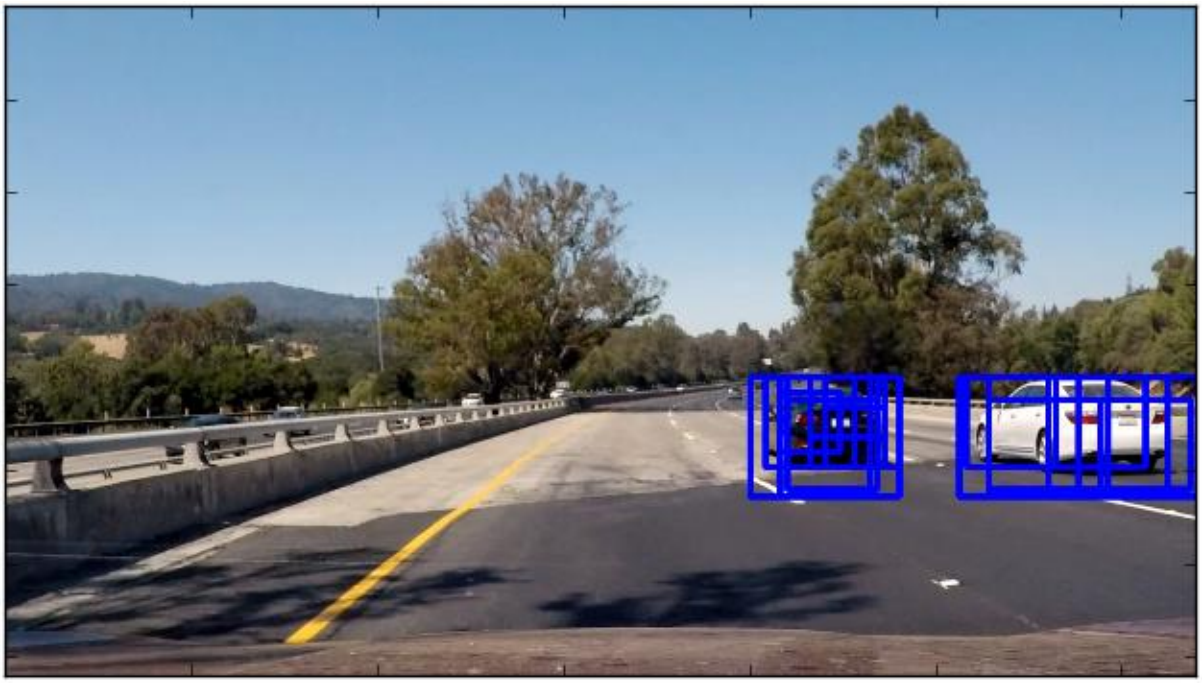
Sliding Window Search

The code for the sliding window search is in ObjectDetection/SlidingWindowHelper.py lines 36 – 75 and in ObjectDetection/Detector.py lines 28-30

I decided to search in various window sizes of 64, 96, 126 and 256. I used a constant overlap of 0.75 between the windows as it was giving the most useful result and detections. I also limited the search to lower half of the image between (400, 656) as the other objects like sky etc. were irrelevant and need not be searched.

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 are some example images:





Video Implementation

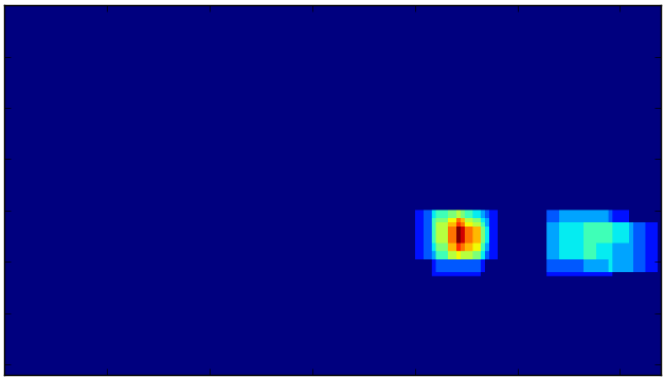
Here's a link to my video result -
https://drive.google.com/open?id=0Bwh773at_cohS2J1eW1ld0RVV0k

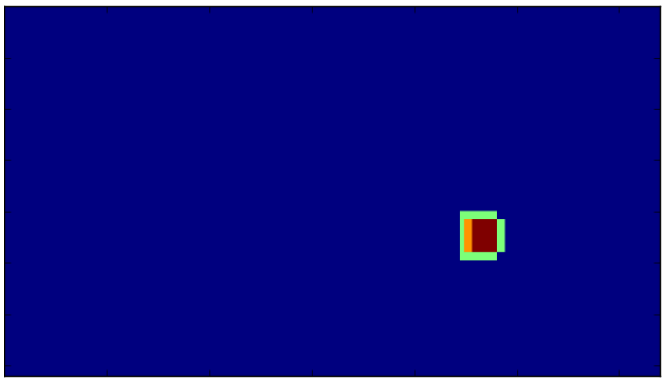
I recorded the positions of positive detections in each frame of the video. From the positive detections I created a heatmap and then thresholded that map to identify vehicle positions. I then used scipy.ndimage.measurements.label() to identify individual blobs in the heatmap. I then assumed each blob corresponded to a vehicle. I constructed bounding boxes to cover the area of each blob detected.

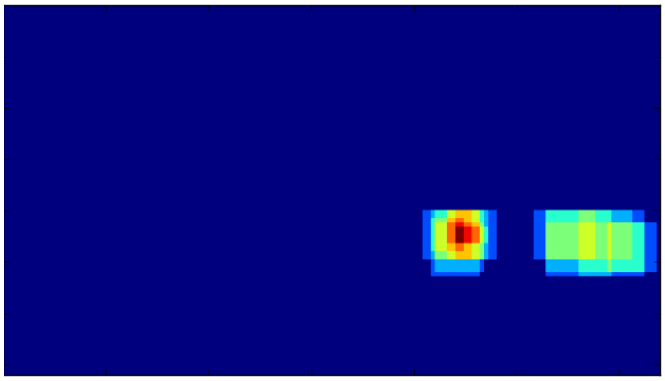
Here's an example result showing the heatmap from a series of frames of video, the result of scipy.ndimage.measurements.label() and the bounding boxes then overlaid on the last frame of video.

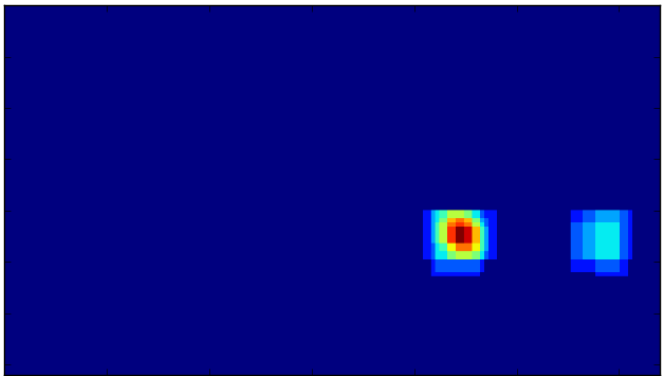
Here are the six frames:

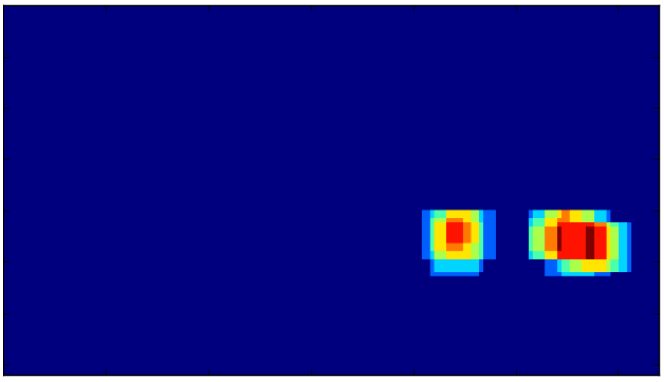
Here are the heatmaps of the six frames respectively:



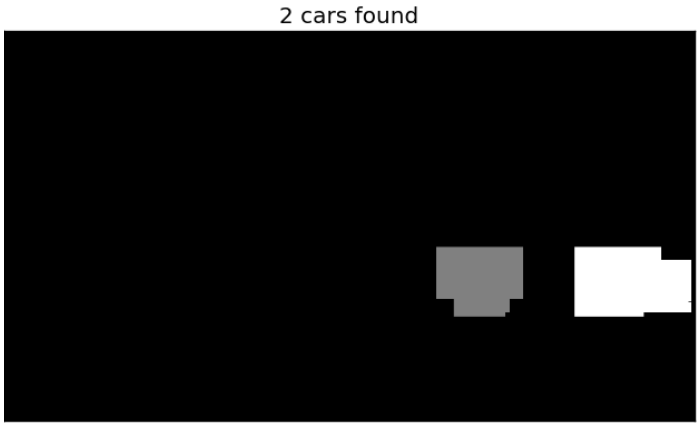


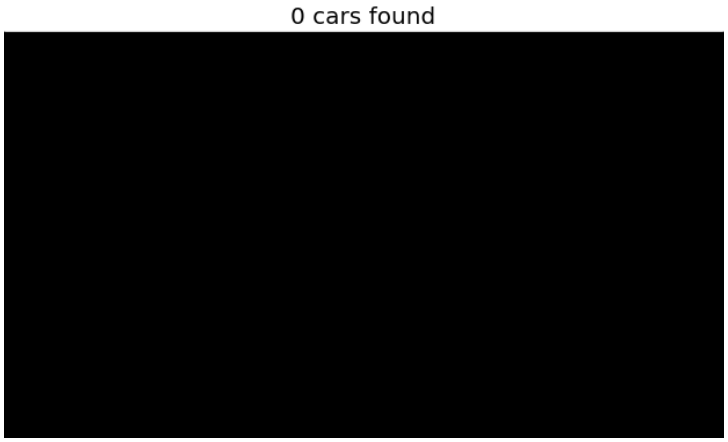






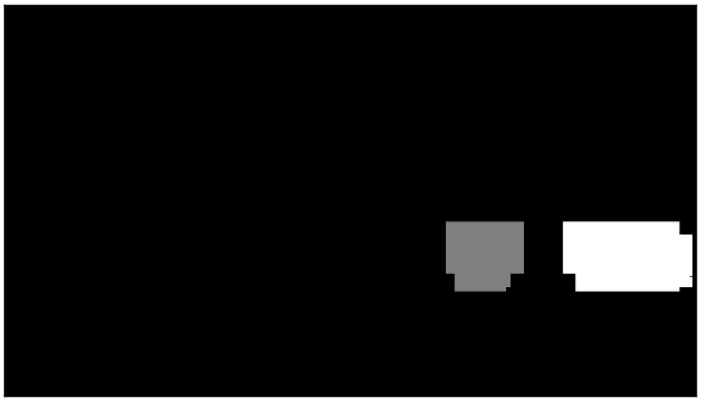
Here are the outputs of scipy.ndimage.measurements.label() on the integrated heatmap from all six frames respectively:



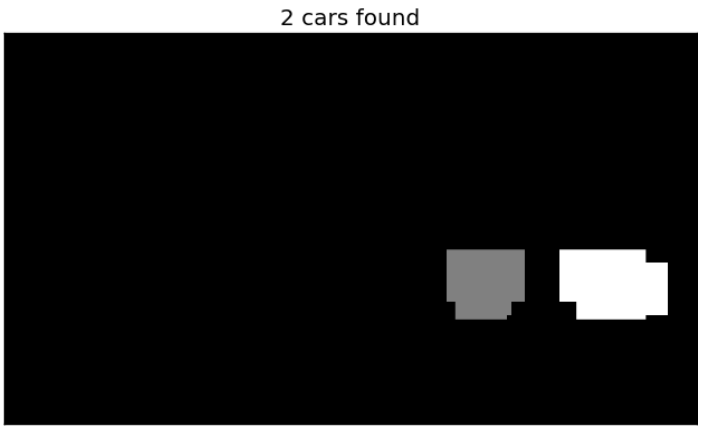


1 cars found

2 cars found



2 cars found



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

Discussion

Here I'll talk about the approach I took, what techniques I used, what worked and why, where the pipeline might fail and how I might improve it if I were going to pursue this project further.

I approach I took was similar to how it was taught in the lessons. I first experimented a lot with various color spaces and various parameters for the HOG features and spatial and color histogram features and finally picked the ones which felt good enough.
I initially had trouble with the training where the validation error was not close to the training error but I was able to resolve via tuning the parameters.

I think the pipeline might fail in other roads and videos and it has been strictly set to the project video.
The model can be improved by training it on a lot more data. I have not used the Udacity dataset

but training on it would help the model a lot. Also to make the model faster I could implement HOG subsampling.