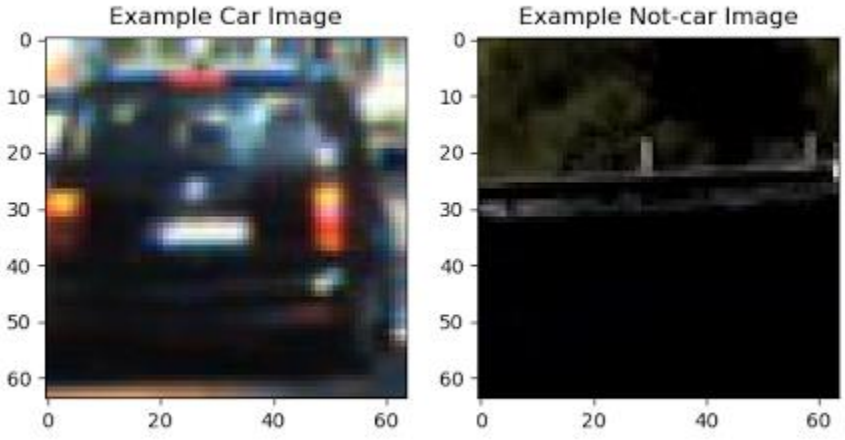
###Histogram of Oriented Gradients (HOG)

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

I created notebook Project 5-1.ipynb to show the extraction of HOG features. I first define the function to return the HOG features. I then define the extraction function for features from images. Once I am done defining the feature extraction function, I then get the images for training from the Udacity directory that I downloaded to my computer and append the two categories of “Cars” and “NotCars". Below is an example:



After I have separated the two sets of images, I run the feature extraction to train the classifier. Then I test on a test size of 20%. Upon testing, I get a test accuracy of 94%.

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

I experimented with numerous variables but I ultimately decided to use the parameters used in the course to maintain consistency with the rest of the program. I used an Orientation of 9, and 2 cells per block with 9 pixels per cell. I used Hog_Channel 0.

####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).

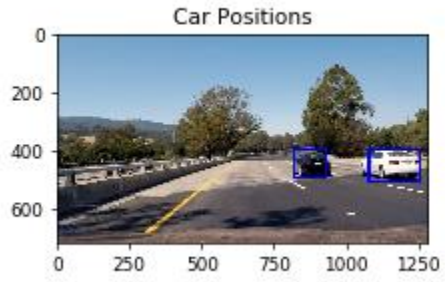
I train the data in the following two functions: Car_Features and notcar_features. They are both located in the project 5-1.ipynb notebook below the HOG parameters

###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 implement the slide_window function in the project 5-2.ipynb notebook. The overlap was set to 0.5.

####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?



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.)

The video is attached to the github repository

####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 heatmap to eliminate false positives in project 5-2.ipynb notebook

####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?

While the code is working, I believe that the problem is identifying cars when you do not have a clear picture of them. It is my belief that they key to identifying cars by camera is to look at relative motion of items in the video. Since relative motion to the background is an ideal indicator of items you want to recognize, developing code for identifying moving objects would be ideal to identify cars and other items we cannot train the classifier to identify such as people, motorcycles, and even non-traditional road vehicles such as recreational vehicles.