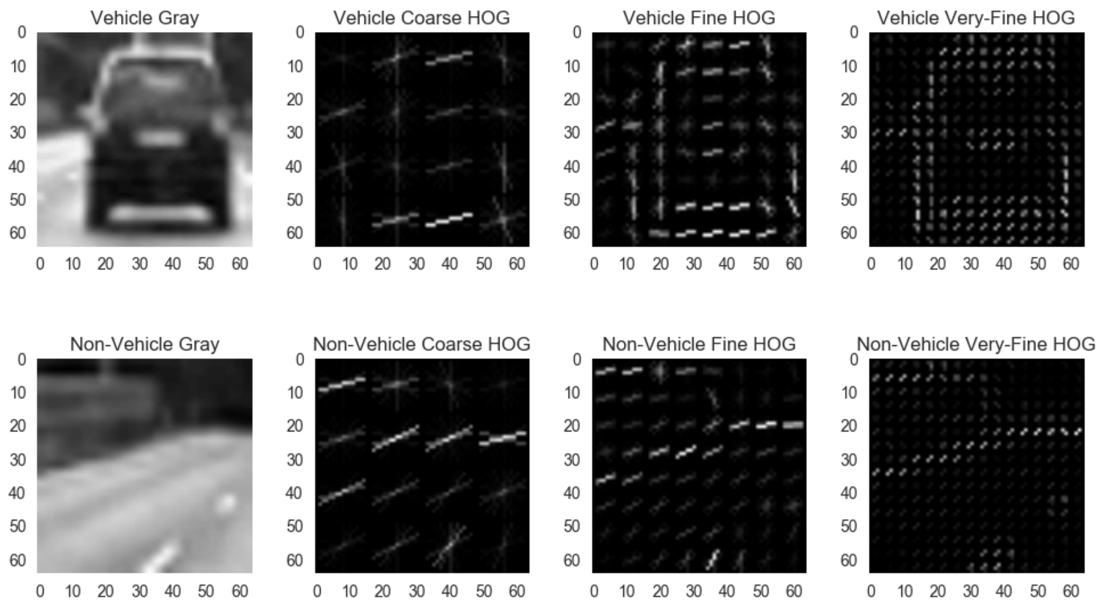
VEHICLE DETECTION PROJECT

The following explains the various components of an image processing pipeline for detecting vehicles in video frames. Specifically, this document considers

1. **Feature Extraction**: HOG and Color Histogram features.
2. **Model Development**: SVM for vehicle/non-vehicle image patch classification.
3. **Sliding Window Search:** Position and scales of windows used to search for vehicles in an image.
4. **Bounding Box:** Using a detection heat map from multiple frames to get vehicle bounding box.

FEATURE EXTRACTION: HOG FEATURES

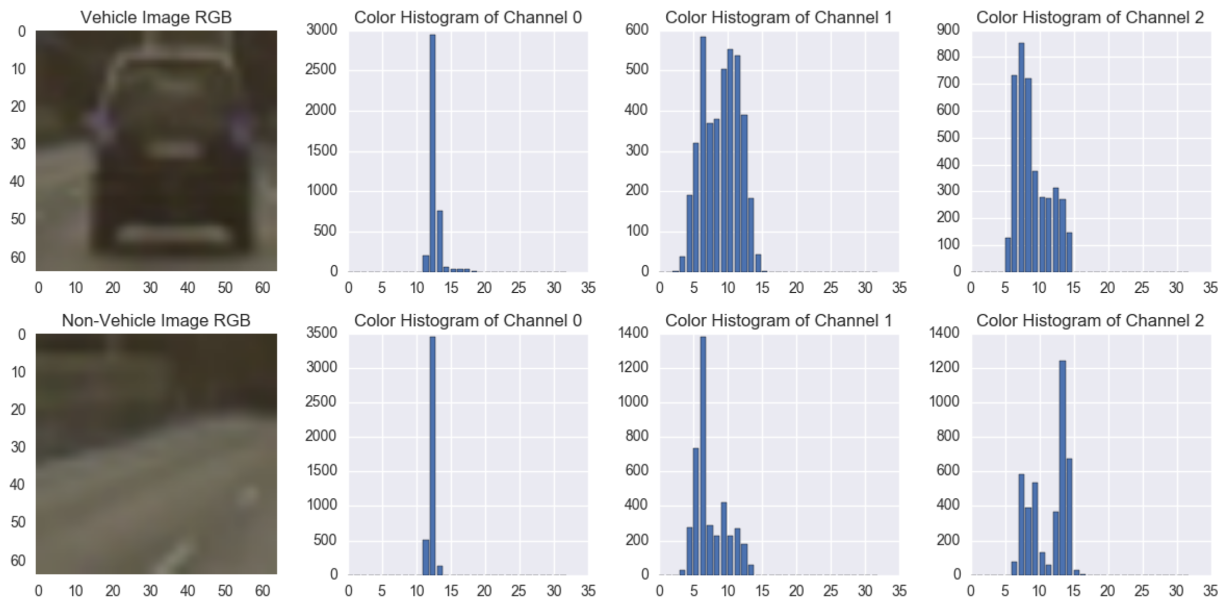
Since the edges associated with image patches containing vehicles are different than those containing other road imagery, the HOG feature is a good choice. The image below illustrates a visualization of HOG features for a vehicle and non-vehicle image.



The HOG features above all use 9 orientation bins, but different cell and block sizes. The *Coarse HOG* uses 16 pixels-per-cell and 2 cells-per-block and does not effectively distinguish the two examples. On the other hand, the *Very-Fine HOG*, which uses 4-pixels-per-cell and 4-cells-per-block, clearly distinguishes the two examples but results in large feature vectors that significantly increased the SVM classifier's training time. **The *Fine HOG*, which uses 8-pixels-per-cell and 2-cells-per-block, provided a good balance between the ability to discriminate between vehicle and non-vehicle examples as well as feature vector size.** In the code, the method ExtractHOGFeatures (cell *HOG Feature Extraction Methods* line 34) is responsible for HOG feature extraction.

FEATURE EXTRACTION: COLOR HISTOGRAM

Image patches with cars also have distinct coloring relative to patches of random road imagery. Consequently, using color histogram features is also a good choice. The plot below illustrates how different the histograms of vehicle and non-vehicle images in the HSV color space could be; note, the example vehicle and non-vehicle images are in RGB.



To select a color space, I compared the three-fold cross-validated test accuracy of SVM classifiers that use HOG and Color Histogram features extracted in the HSV, HLS, LUV, RGB, and YCrCb color spaces. The table below illustrates the performance of these models.

Color Space	Cross-Validated Test Accuracy
HSV	97.99%
HLS	97.92%
LUV	98.49%
RGB	97.41%
YCrCb	97.83%

As can be seen from the table above, the choice of color space did not have a significant impact on model's accuracy on the test images (all are within 1%). For the project, the YCrCb color space was selected because it provided better results for the project video. In the code, the method ExtractColorHistFeatures (cell *Color Histogram Feature Extraction Methods* line 1) is responsible for color histogram feature extraction.

MODEL DEVELOPMENT

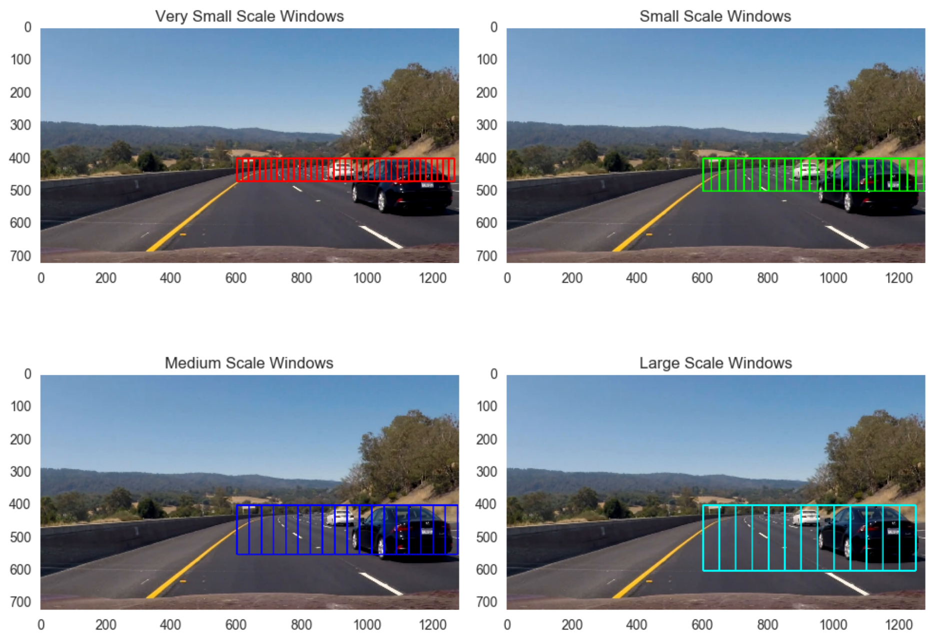
I extracted HOG and Color Histogram features in the YCrCb color space from all vehicle and non-vehicle examples using the method ExtractFeatures (cell *Feature and Model Construction Methods* line 1). This method loops over all images in a specified directory and concatenates the extracted HOG and Color Histogram features (line 19-24). The method ExtractFeatures is called for each vehicle and non-vehicle image directory in the cell *Model Performance and Colorspace* lines 11 and 22.

Next, I used the extracted features to train and evaluate a binary linear SVM classifier using the method TrainAndEvaluateModel (cell *Feature and Model Construction Methods* line 28). This method performs the following steps.

1. Randomly shuffle all examples (line 44).
2. Split examples into train and test set using an 80/20 split (line 49).
3. Assemble sklearn pipeline consisting of a StandardScaler and LinearSVC (line 53).
4. Grid search and 3-fold Cross-Validation on the training set to choose SVM parameter C (line 57).
5. Evaluate best model from gird search on test set (line 62).
6. Return model (line 69).

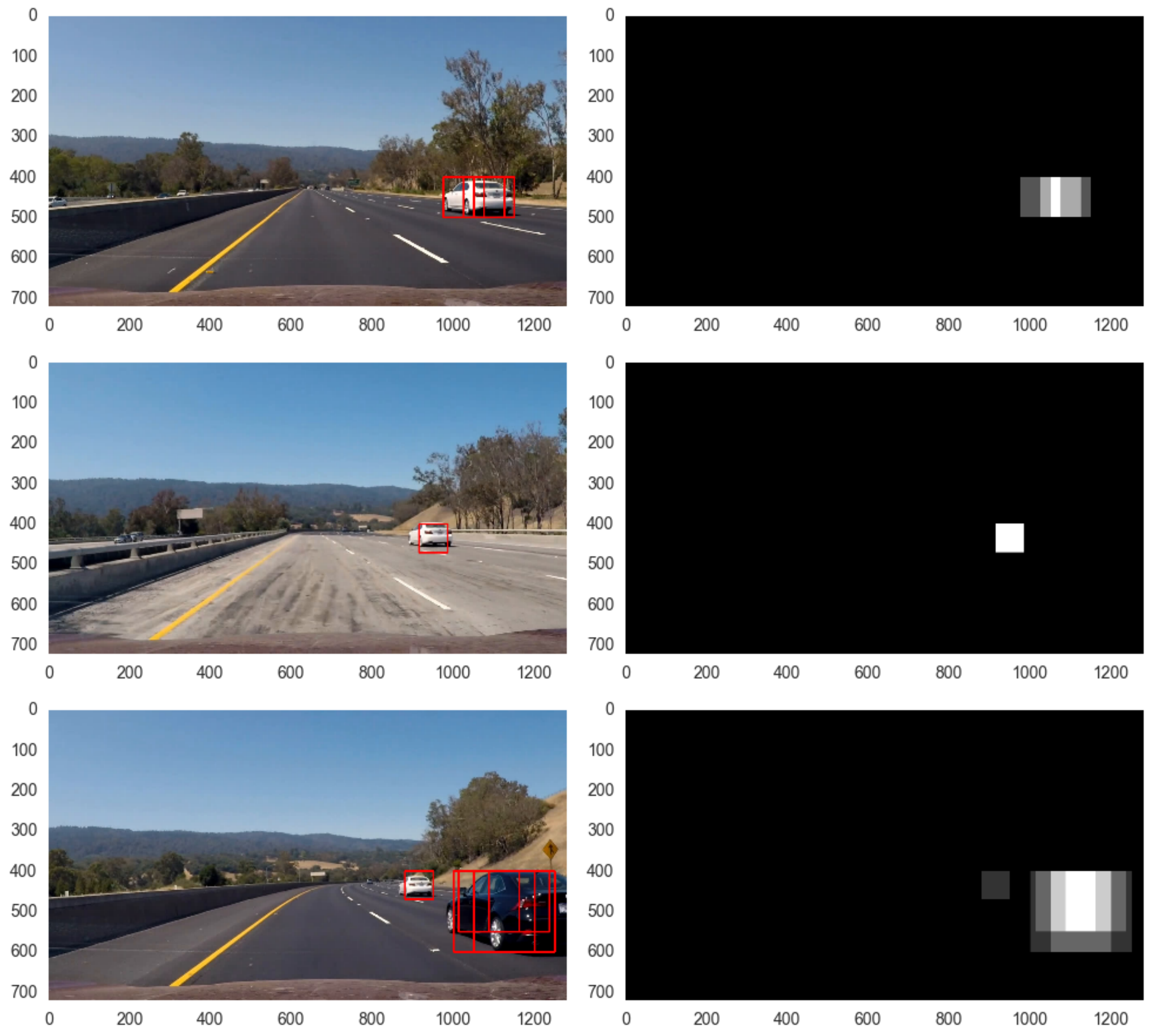
SLIDING WINDOW SEARCH

I search for vehicles using overlapping windows of sizes 70x70, 100x100, 150x150, 200x200. The top left corners of the windows span the range x=600-1280 and all have y=400. These limits were chosen so that only the driving portion of the road is covered (not the road shoulder or sky). Window sizes were chosen so that cars close and far away would fit in at least one the window size. Shown below are windows of each size and how vehicles fall within them (i.e. white car in red windows and blue car in cyan windows).



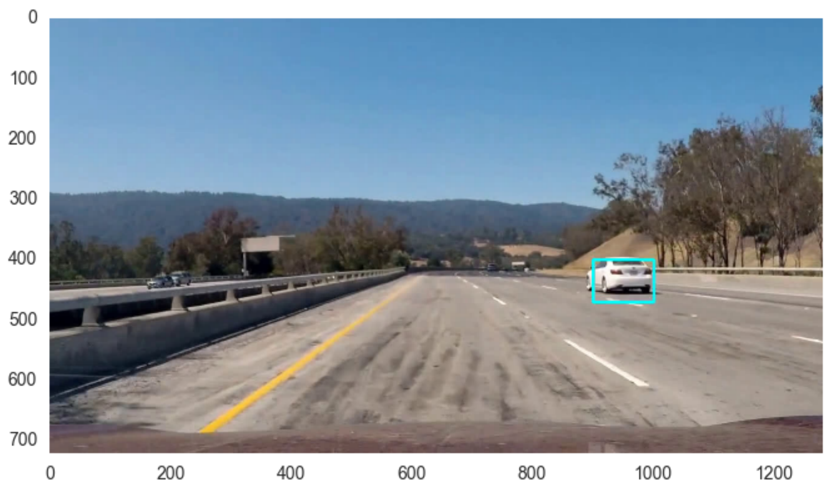
DETECTING VEHICLES

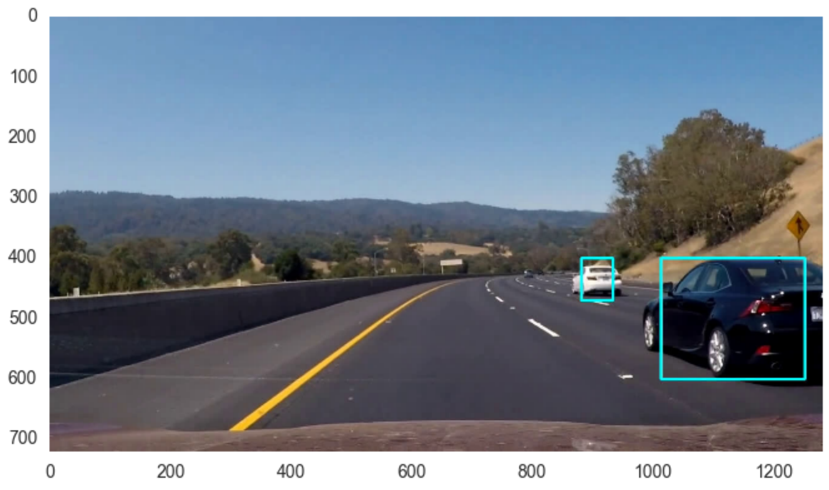
Vehicles are detected in a frame using the *VehicleDetector* class (cell *Vehicle Detection* line 1). This object applies the trained sklearn pipeline (StandardScaler and LinearSVC) to each one of the analysis windows shown above (see the method *detect_vehicles* in cell *Vehicle Detection* line 26). When a window is determined to contain a vehicle the value of the pixels that correspond to the window are incremented in a heat map; consequently, pixels corresponding to regions of overlapping detections appear brighter in the heat map. As an example, the images below show all the analysis windows flagged by the classifier as containing a vehicle and the accompanying heat map for frames from the project video. The image in the third row illustrates simultaneous detection of near and far vehicles in a single frame and the corresponding heat map is brighter for regions of significant overlap.



BOUNDING BOXES AND FALSE DETECTIONS

To reject false detections, we take advantage of the fact that vehicle detections will likely overlap within frames (as shown in the examples above) as well as across frames (since the frame rate is sufficiently high). To implement this observation, the heat maps from the last 10 frames are summed and thresholded (with a value of 10) to create a binary image (cell *Vehicle Detection* lines 70 and 73). The thresholding is what enforces that a pixel be a member of a detection within and across multiple frames (something that is not likely to be true for a false detection). Next, we extract the vehicle bounding box by applying connected components analysis to this binary image using the functions *skimage.measure.label* and *skimage.measure.regionprops* (cell *Vehicle Detection* lines 73-78). The images below illustrate the vehicle bounding boxes extracted from frames of the project video. The image in the second row illustrated two bounding boxes of different sizes bounding a near and far away vehicle.





IMPROVEMENTS

The vehicle classifier discussed here uses a fixed set of analysis windows that don't consider variation in the road or its curvature. So, if the road curves significantly, a fraction of the analysis windows will search for vehicles in inappropriate parts of the video frame and contribute to false detections. Another potential way to improve the system is to improve the accuracy of the underlying vehicle/non-vehicle classifier using other model classes or an ensemble of classifiers (e.g. fuse the decisions of an SVM, Decision Tree, and Random Forest models).