Vehicle detection

The goals / steps of this project are the following:

- Perform a Histogram of Oriented Gradients (HOG) feature extraction on a labeled training set of images and train a Linear SVM classifier.
- Optionally, you can also apply a color transform and append binned color features, as well as histograms of color, to your HOG feature vector.
- Note: for those first two steps don't forget to normalize your features and randomize a selection for training and testing.
- Implement a sliding-window technique and use your trained classifier to search for vehicles in images.
- Create heat map to remove the false positive detections.
- Run the pipeline on the video.
- Estimate a bounding box for vehicles detected.

1.1 Describe image pre-processing and pipeline:

The first step is to read the files containing images of cars and non-cars. Before using these images to train the classifier, the features should be extracted to be used for training. I have used the HOG feature combining with the spatial and color histogram data. Image below shows and example of hog image after HOG feature extraction from an image containing a car.

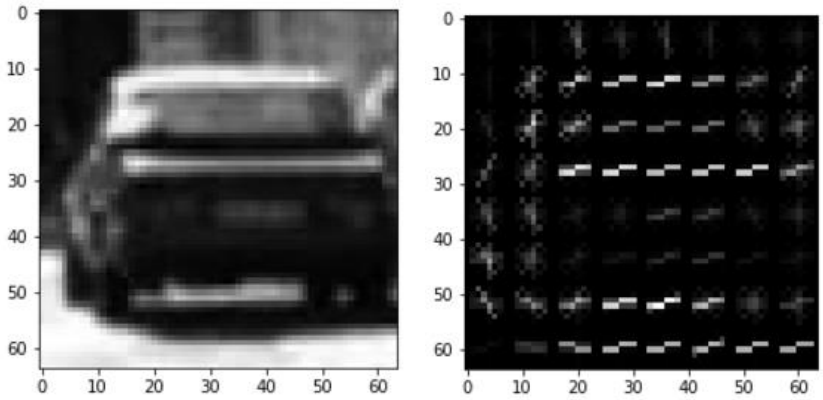

Figure 1. Left: Original car image image, right: HOG image with orientations = 9, pixel per cell = 8, and cell per block = 2.

The extracted features can be concatenated to be used for training the SVM classifier. The linear SVM classifier is the best due to the fact that it is much faster for both training and prediction comparing to using polynomial or rbf kernels for the SVM. And besides, the linear SVM classifier can reach reasonable accuracy. I have used LUV color-space for HOG, spatial, and histogram features. Spatial features contains 16x16 feature points, histogram has 16 bins and HOG has orientation of 10, pixel per cell of 8, cell per block of 2 and the color channel was involved in the HOG feature extraction was channel 0. To choose the best classifier that has good accuracy and also high speed I tried different combination of variable used to extract features that led to choosing the parameters mentioned above. Some of the results of using various parameters are shown in the table below. 80 percent of the dataset was used for the training and 20% for test:

Colorspace	Orient	Pix_per_cell	Cell_per_block	HOG_channel	Spatial	Histbin	Accuracy	Time (s)
LUV	10	8	2	0	16	16	98.5	7.3
LUV	10	4	2	0	32	32	98.1	78.8
RGB	10	8	2	ALL	16	16	97.4	187
RGB	10	4	2	ALL	32	32	Memory error	-

1.2 Searching in the image for area containing car:

After training the classifier, it can be used to find the area containing car in an image. To do so have to use sliding window to sample the different area of the image resize it to 64x64 and use classifier to predict the class of that area inside the window. Figure below shows area in the main image where windows are created:

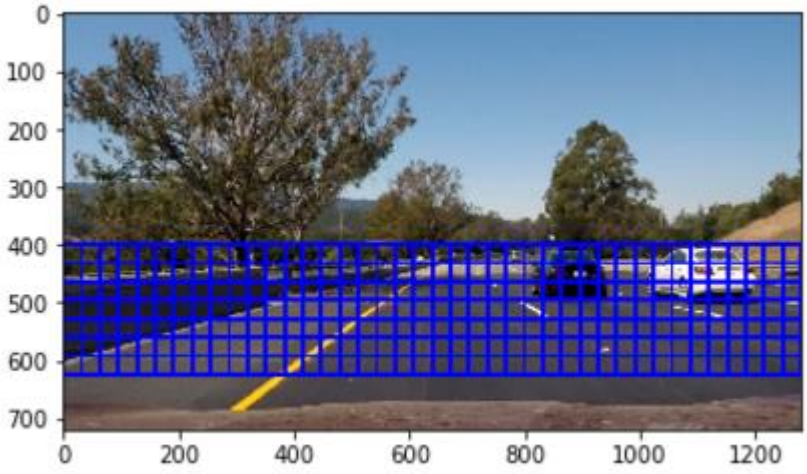

Figure 2. Area in the main image that windows are created.

The area contains the sky or very close to the bottom of the image will not have any car inside then there is not any window there to predict the class of the region, which can make process faster. To make even faster can you multi-size windows, means for the area further to the camera smaller windows be created and for regions closer to camera bigger windows.

Figure below shows the area detected as having the car inside:

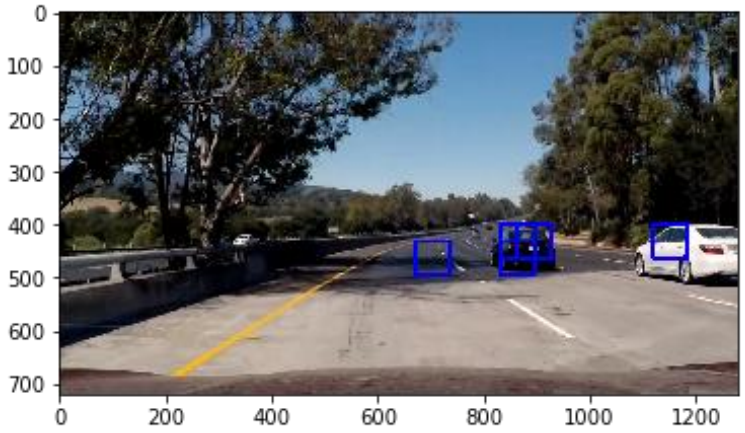

Figure 3. Area in the main image that detected as having car inside, which has a false positive detection too.

The detected area has a false positive in it. To remove the false positive samples have to use the sub-sample windows. In this method the some windows with finer spatial resolution are assigned to the area little bit bigger than the areas detected in the last step. Then these regions of the image inside these windows are fed into the classifier to detect the car there. By this process the detected windows in the area that really has car will be increased and reversely the number of true positive windows in the area does not have car will be decreased, which means false positive windows in the previous step are decreased.

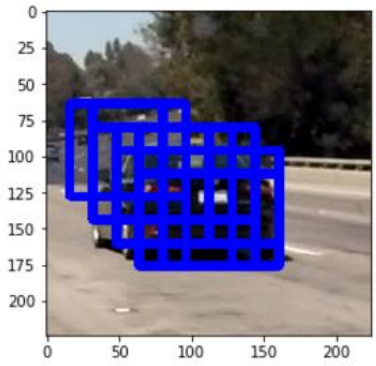

Figure 4. Detected are in one of the sub-sampling windows.

1.3 Create heat map:

After this step the image containing the detect area were used to create the heat map. As the area has false positive detection are decreased by choosing the proper threshold for the heat map can eliminated the false positive samples.

2. Process the video file:

Each frame the video can be process as describe above and the final output can be saved in the output file. To make the process little bit faster, as I mentioned earlier can use the multi-size sliding windows.

The time for processing the videos can be from few minutes to an hour depends on how fast the classifier is and how big are the feature vectors fed to the classifier for classification.

3. Conclusion:

In this project the vehicle detection implemented that can be used beside other previous projects as advanced lane detection, to first detect the vehicles and for instance fill the detected area with black rectangle and then feed it to lane finding algorithm to prevent it from distraction from the cars in the road.

The most challenging part in the project is finding a fast and accurate classifier. To improve the accuracy of the linear SVM classifier can you use more data, and also add augmentation to the training data to provide different lighting condition of the car and not car image to the model, which can be very important in different road image lighting.

The pipeline to process the video is not very good at detecting cars very far from camera, and improve this situation have to add car images that contains cars which are further from camera that the model can be able to distinguish a small car from the background.

The rectangle assigned to detected cars can be unstable from frame to frame and jump around. To improve it and get more smooth bonding boxes can store the heat maps over few previous frames and add all of the to the current image and apply the threshold to detect the cars