Vehicle Detection using SVM and OpenCV

The goals / steps of this project are the following:

1. Using Histogram of oriented gradients (HOG) and Color to extract image features
2. Train the Linear SVM classifier using the extracted features and labels
3. Sliding search of the cars using the trained SVM classier
4. Remove multiple detections and false positives using "heat"
5. Test on video
6. Combine with the lane detections pipeline

Histogram of Oriented Gradients (HOG)

I started by reading in all the cars and notcars images in step 0. Here is two examples of one of each of the cars and notcars classes shown in Figure 1:

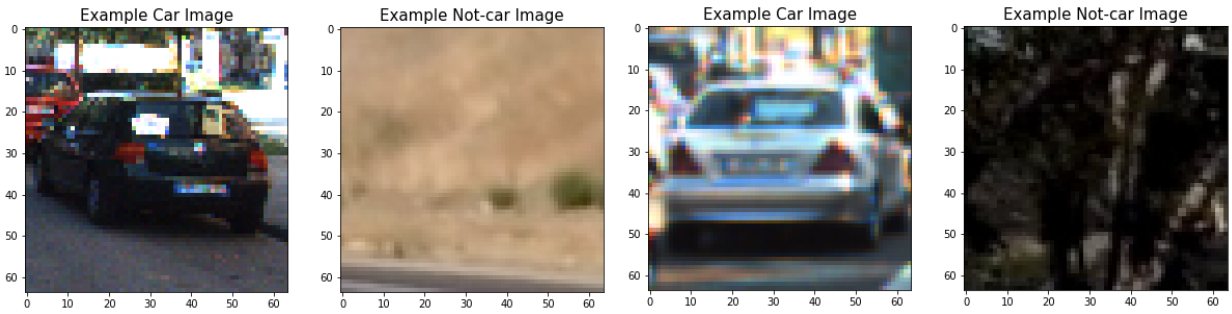
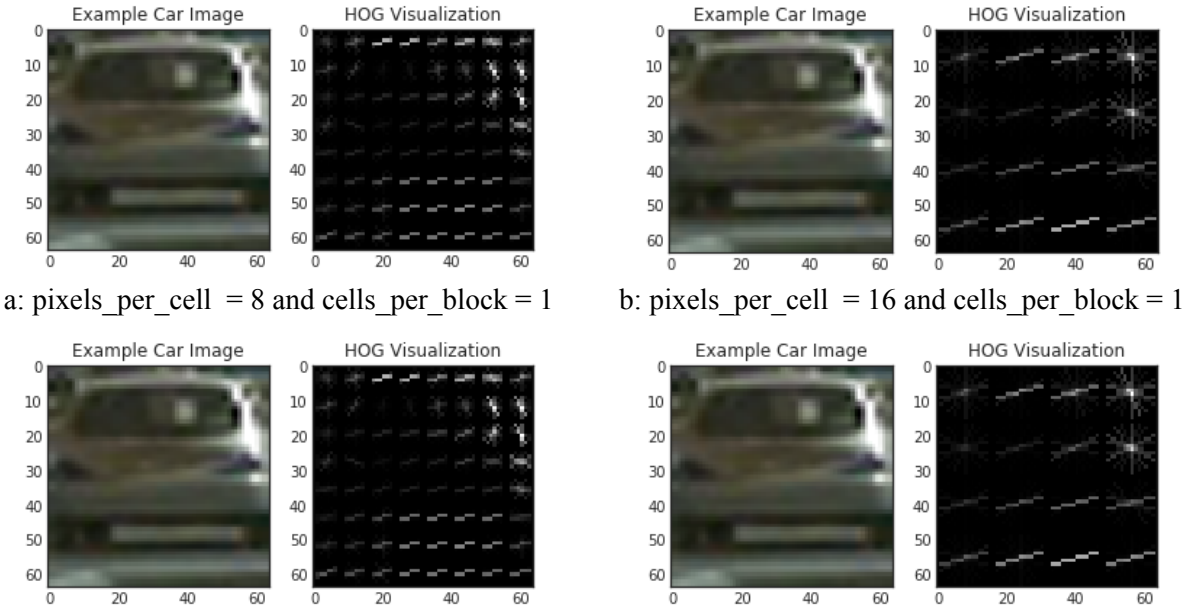


Figure 1. Random cars and notcars examples

We then used the HOG to extract features used to train the SVM classifier in step 1. The HOG is also referred as the HOG feature descriptor, in which the distribution (histograms) of directions of gradients (oriented gradients) are used as features. Gradients (x and y derivatives) of an image are useful because the magnitude of gradients is large around edges and corners (regions of abrupt intensity changes) and we know that edges and corners pack in a lot more information about object shape than flat regions. [1]

The HOG features was extracted using the scikit-image ***hog()*** function. The function takes in parameters include orientation, ***pixels_per_cell*** and ***cells_per_block***. The number of orientations represents the number of orientation bins that the gradient information will be split up into in the histogram. Typical values are between 6 and 12 bins. The ***pixels_per_cell*** parameter specifies the cell size over which each gradient histogram is computed. The ***cells_per_block*** parameter specifies the local area over which the histogram counts in a given cell will be normalized.

These parameters were tweaked by trial and error, the orientation was set to 9 as same as the class, ***pixels_per_cell*** and ***cells_per_block*** were tested by 8, 16 and 1, 2 combinations. The HOG features can be visualized in Figure 2. It looks like the 8 & 2 combinations will give the best results.



a: pixels_per_cell = 8 and cells_per_block = 1

b: pixels_per_cell = 16 and cells_per_block = 1

c: pixels_per_cell = 8 and cells_per_block = 2

d: pixels_per_cell = 16 and cells_per_block = 2

Figure 2. HOG features visualizations

Train the Linear SVM classifier using the extracted features

A linear SVM was trained using ***sklearn.svm.LinearSVC()*** with the extracted car features to learn to classify car and not car. The 8792 cars and 8968 notcars images were comprised of images taken from the KITTI vision benchmark suite, and examples extracted from the project video itself.

The parameters were set as: ***color_space*** = 'LUV', ***hog_channel*** = 'ALL', ***spatial_size*** = (16, 16), ***hist_bins*** = 16.

Sliding search of the cars

The sliding window search method is performed by sliding windows of different size across the interest area of the image. The window size of $(64, 64)$, $(96, 96)$, $(128,128)$ and $(256,256)$ were tested in the lower half size ($[380, 650]$) of the test images as in Figure 3.

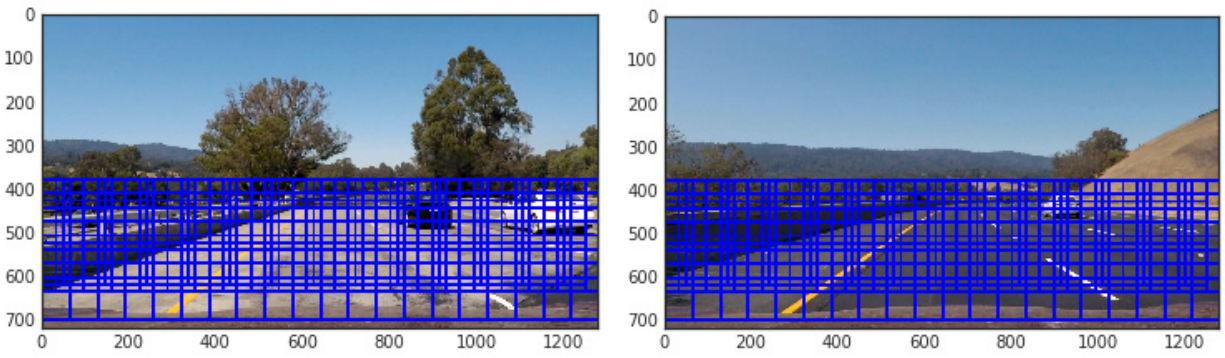


Figure 3. Sliding window search using different size

Remove multiple detections and false positives using "heat"

We built a heat-map to combine overlapping detections and remove false positive detections. To make a heat-map, we simply added "heat" (+=1) for all pixels within windows where a positive detection is reported by your classifier. The multiple detections and heat-maps for the testing images are shown in Figure 4.

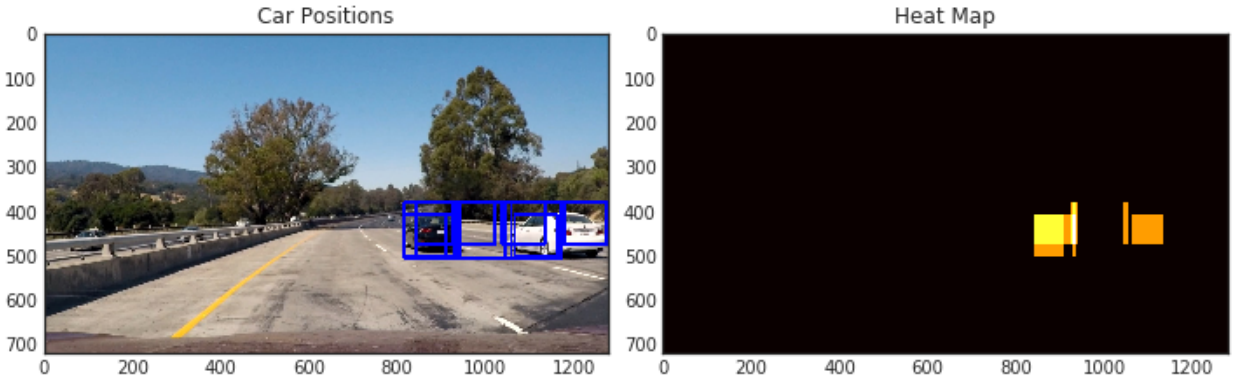


Figure 4. The multiple detections and heat-maps result of test image 1

Reflection

For simplicity and speed, we used linear SVM as a baseline classifier throughout the study as same as in the study [3]. In the final results, there are three mains issues exist:

1. The classifier. It's can be seen that the SVM give a reasonable classification between car and not car for most scenarios, but between 24 to 30 seconds in the video, it doesn't classify the white car well. Based on the first submission review, I tuned the feature extraction parameters to achieve a higher test accuracy of SVC (99.38%). I also conducted a multi-frame accumulated heatmap by using collections.deque in the def ***draw_windows()*** function. The white car no detection between 24 to 30 seconds issue got solved, but the heatmap threshold was hard to tune and there's still some false detections.
2. The sliding window technique. The sliding window is very computational expensive which make the techniques impossible for real-time application as well as for heavy traffic scenarios.
3. The multi-scale bounding boxes. The printed squared bounding boxes were obvious non-optimal. We picked some sizes trying to fit for the goal objects, but too many sizes will burn the computational dramatically.

I did some research and realized there are existing techniques for above issues. As mentioned in the class, the classifier performances can be improved by using advanced CNN architectures such as VGG [4].

For the sliding window, it's so time consuming due to it need to search the manually selected whole section step by step. The problem is well solved by Faster R-CNN [5]. The proposed object detection system Faster R-CNN is composed of two modules: a deep fully convolutional network that proposes regions and the Fast R-CNN detector that uses the proposed regions. The proposed regions make the sliding window search much faster. Moreover, in order to make the Region Proposal Network (RPN) to share computation with Fast R-CNN object detection network, they used VGG-16 CNN to share a common set of convolutional layers.

For the multi-scale bounding boxes issue, several papers have proposed ways of using deep networks for predicting object bounding boxes. The MultiBox methods [6] generate region proposals from a network whose last fully-connected layer simultaneously predicts multiple class-agnostic boxes. In [7], a multi-scale object detection techniques was proposed to represent object boundary shapes.

Future work will be very interesting if we combine the latest proposed Faster R-CNN [5] with the 2009 proposed object boundary shapes representation [7] techniques. A real-time object shapes detection results are predictable rewarding.

Reference

[1] scikit-image HOG (https://classroom.udacity.com/nanodegrees/nd013/parts/fbf77062-5703-404e-b60c-95b78b2f3f9e/modules/2b62a1c3-e151-4a0e-b6b6-e424fa46ceab/lessons/fd66c083-4ccb-4fe3-bda1-c29db76f50a0/concepts/d479f43a-7bbb-4de7-9452-f6b991ece599)

[2] Histogram of Oriented Gradients (http://www.learnopencv.com/histogram-of-oriented-gradients/)

[3] Dalal N, Triggs B. Histograms of oriented gradients for human detection. InComputer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on 2005 Jun 25 (Vol. 1, pp. 886-893). IEEE.

[4] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. 2014 Sep 4.

[5] Ren S, He K, Girshick R, Sun J. Faster R-CNN: towards real-time object detection with region proposal networks. IEEE transactions on pattern analysis and machine intelligence. 2017 Jun 1;39(6):1137-49.
[6] Erhan D, Szegedy C, Toshev A, Anguelov D. Scalable object detection using deep neural networks. InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2014 (pp. 2147-2154).
[7] Ommer B, Malik J. Multi-scale object detection by clustering lines. InComputer Vision, 2009 IEEE 12th International Conference on 2009 Sep 29 (pp. 484-491). IEEE.