Vehicle Detection

Writeup

General

I divided the project into 2 different implementations.
Once to test each feature in the following files:

- draw.py – methods for drawing
- features.py – extract the features (Spatial Binning of Color, Color histogram, hog features)
- search.py – slide window seach and find cars

The working pipeline was implemented in the following files:

- LaneDetector.py – Implementation of the project "Advanced Lane Finding"
- VehicleDetector.py – Implementation of the complete pipeline for vehicle detection. Combines the implementations of draw.py, features.py, and search.py in one class.

The file "CarND-Vehicle-Detection.py" contains the "main" method and runs the whole pipeline.

In the file "model.pickle" the "LinearSVM" Classifier and the "StandardScaler" were saved to prevent re-learn all the time.

Histogram of Oriented Gradients (HOG)

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

The code for this step is implemented in the features.py file in the "get_hog_features" method (beginning at line # 23).
Here is an example of a "car" and "not-car" image.

Car

not-Car

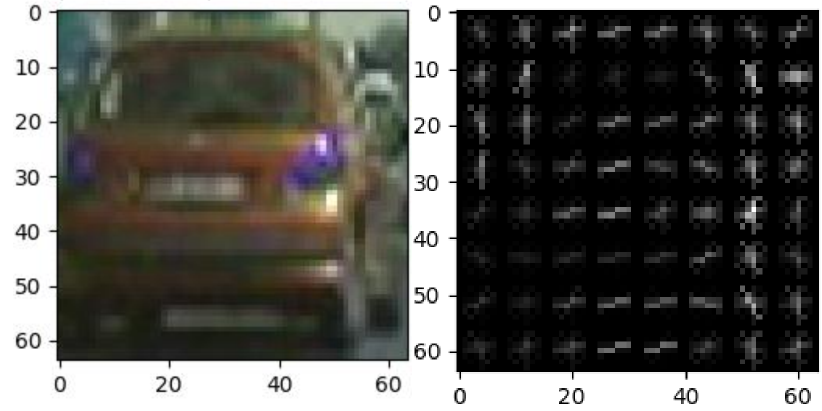
Extracting the features to train the LinearSVM is done in the following steps:

1. Import all images and divide into "car" and "not-car" images. (CarND-Vehicle-Detection.py, Method "get_training_data")
2. Call the "extract_features" method (implemented in features.py)
   This method executes each step of extracting the features:
   a. Color conversion of the picture
   b. extract HOG Features
   c. Spatial Binning of Colors
   d. Color Histogram
   e. Concatenate all single steps to a combined feature vector

For the HOG feature extraction I used the following parameters:

orient	11
pixels_per_cell	16
cells_per_block	2
channels	ALL

After testing with different parameters, these have turned out to be the best option for me.
Here is a sample image of an image with extracted HOG features.

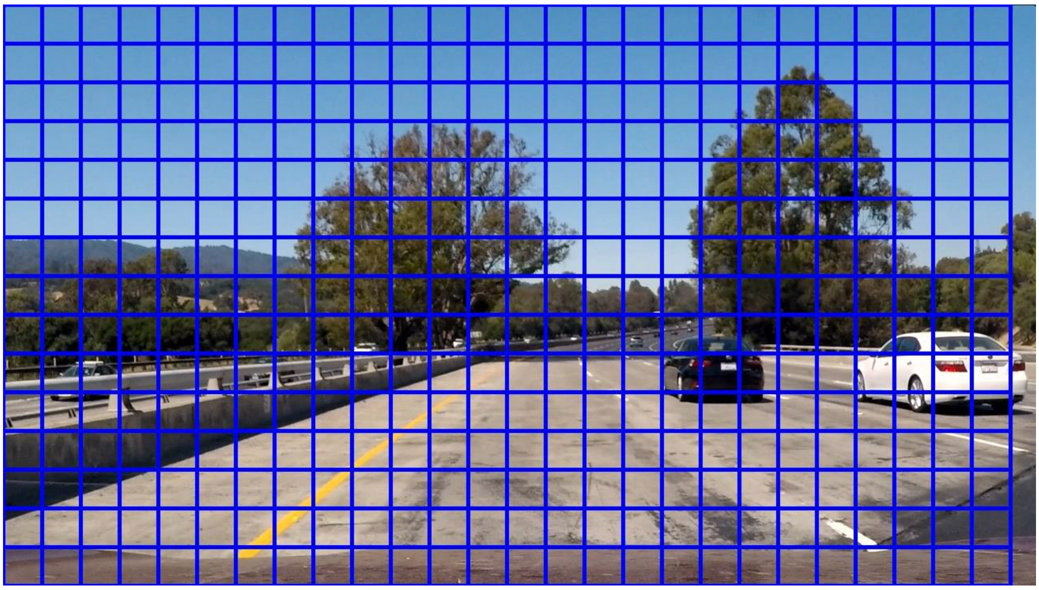


2. Training the LinearSVM Classifier

In the file "CarND-Vehicle-Detection.py" in the "main()" method the file "model.pickle" is loaded with the classifier and scaler. If the file does not exist, a LinearSVM Classifier is trained.
To do this, first load the images ("car", "not-car") and extract the features.
Standardize the features by StandardScaler.
Splits of the training (80%) and test data (20%) via "train_test_split" and train the LinearSVM.

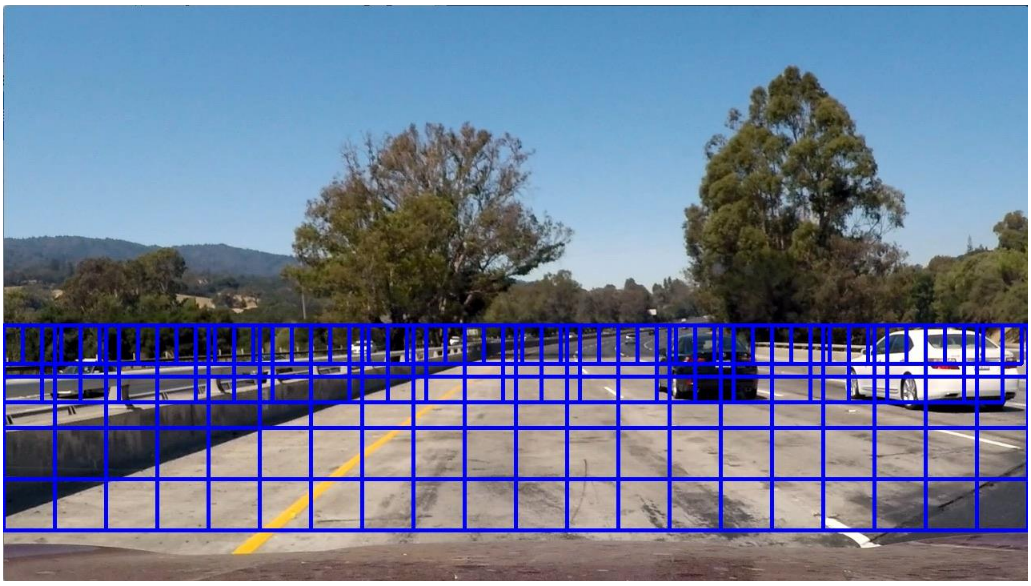
3. Window Search

The function "slide_window()" has been implemented in the file "search.py". Here is a sample picture of the search over the whole picture.



For the actual implementation in the pipeline a "find_cars()" method was implemented. The implementation can be found both as a test function in the "features.py" file and in the "VehicleDetector.py" file for the video.

Here is an example image of the "Hog Sub-sampling Window Search":

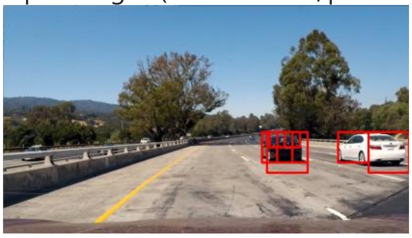


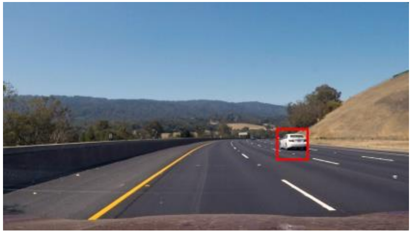
3. Pipeline

I've used the following features for the pipeline:

- Color space (YUV)
- Spatially binned color (size=(64, 64))
- Histograms of color (nbins=32)
- HOG features („ALL“ Channels“)
- Different Scales and ROIs

Here are 2 example images (without false/positive removal):

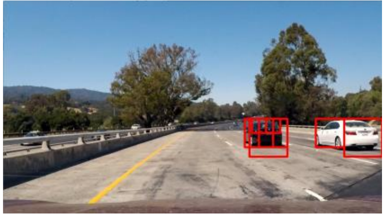


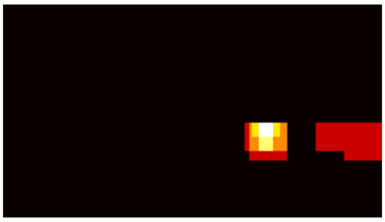


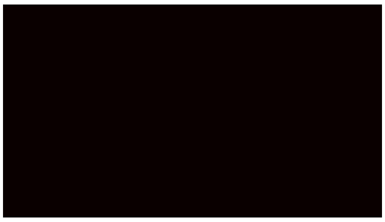
A full video demonstrating the function of the pipeline can be found in the "output_videos" folder.

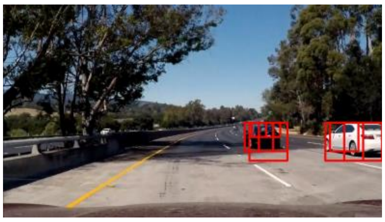
4. False/Positive Filter

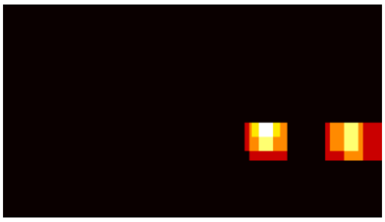
To remove the false/positives, I have created a heatmap of all positive detections.
Here are some examples of the heatmap:





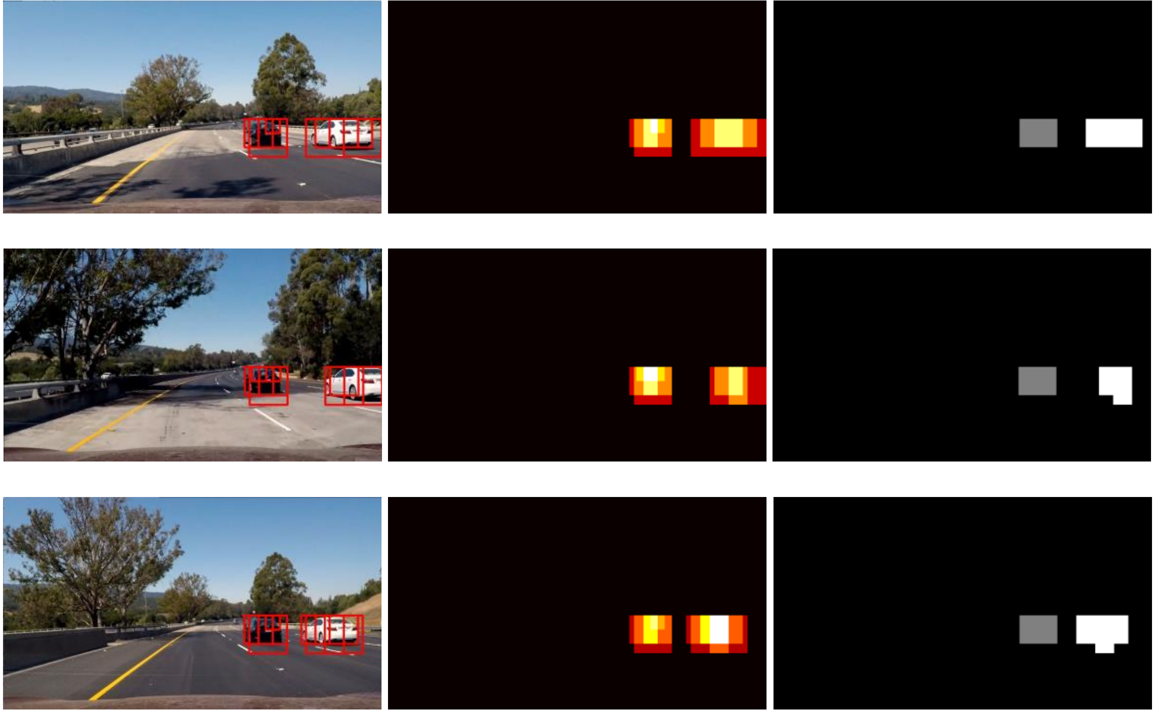




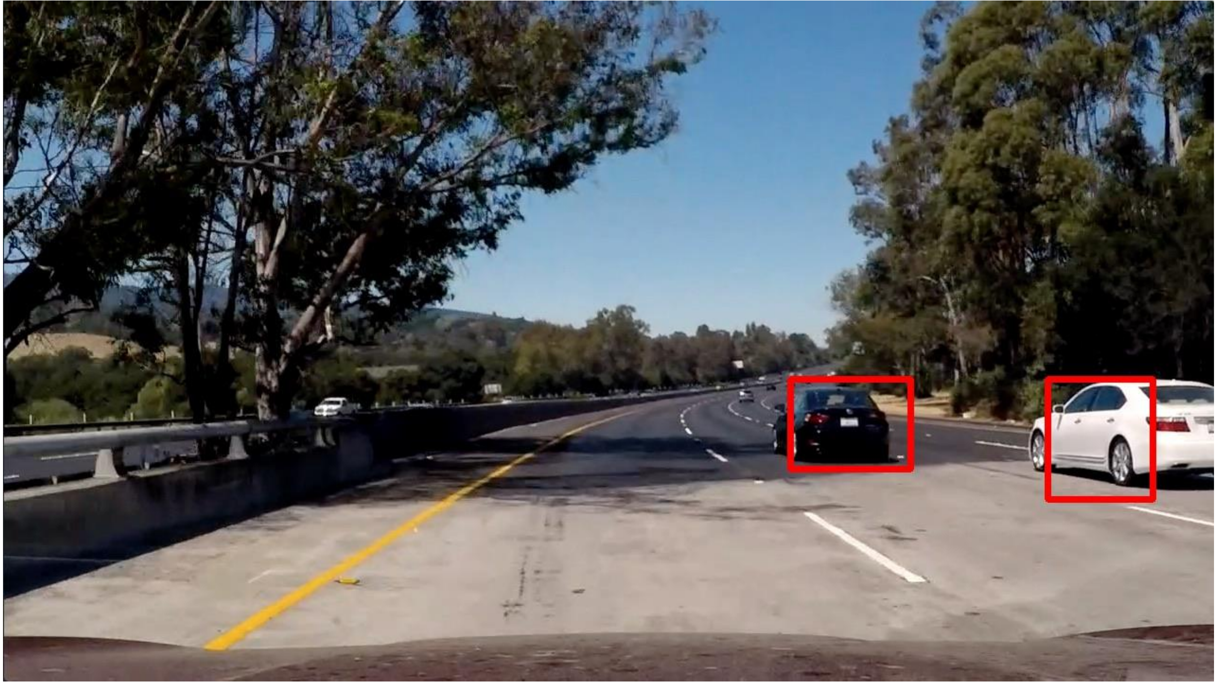


A threshold value is used to determine the vehicle positions. I then used „scipy.ndimage.measurements.label()“ to identify individual blobs in the heatmap. I then assumed each blob corresponded to a vehicle. I constructed bounding boxes to cover the area of each blob detected.

Here are some sample pictures:



Here's an example after the pipeline has been completely run:



Discussion

The biggest difficulties I had with the False / Positive detection. In particular, on the opposite lane on the left side detection has struck more often. To fix the problem, I set the "xstart" for the "find_cars ()" to 400px in the "VehicleDetector". In a productive implementation, the ROI should be determined more dynamically.

Another point is the performance. Currently, the pipeline is too slow to process images in real time. An improvement could be a method such as "YOLO" can be reached (https://pjreddie.com/darknet/yolo/).