Vehicle Detection Project

By Tim Tobey

March 6th, 2017

Vehicle Detection Project

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 classifier 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.
- Run your pipeline on a video stream (start with the test_video.mp4 and later implement on full project_video.mp4) and create a heat map of recurring detections frame by frame to reject outliers and follow detected vehicles.
- Estimate a bounding box for vehicles detected.

Rubric Points

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

Vehicle Detection Project

By Tim Tobey

March 6th, 2017

Write-up / README

Criteria: **Provide a Write-up / README that includes all the rubric points and how you addressed each one. You can submit your write-up as markdown or pdf. Here is a template write-up for this project you can use as a guide and a starting point.**

This is it!

Histogram of Oriented Gradients (HOG)

Criteria: Explain how (and identify where in your code) you extracted HOG features from the training images. Explain how you settled on your final choice of HOG parameters.

The code of this step starts in file ***create_support_vector_classifier.py*** on line 8 where training data for the classifier names are loaded and saved as a numpy array. Here are examples of the training image used is:

Training Data: Car

Training Data: Non-Car

Vehicle Detection Project

By Tim Tobey

March 6th, 2017

The feature were extracted beginning on line 41 with the function ***all_image_process.*** The function takes in the variables to perform spatial imaging, a color histogram and a HOG.

The HOG features that were chosen on begin on line 21 and are as follows:

- orientations = 9
- pixels_per_cell = 8
- cells_per_block = 2
- color_space = **'YCrCb'**

These parameters were chosen through a process of trial and error. I began with trying different color spaces and see where and on what items were identified. After this I worked through the remaining parameters. I stopped when I arrived at a quality group of parameters which did a good job of identify cars with few errors.

Criteria: Describe how (and identify where in your code) you trained a classifier using your selected HOG features (and color features if you used them).

My model uses a Support Vector Machine (SVM). The data passed consisted of an array consisting of a spatial image array (32 x 32), a color histogram and a three channel HOG for each image. The code for these items can be found in the ***project_functions_2.py*** file on line 51, 60 and 24 respectively.

In the file ***create_support_vector_classifier.py*** on line 54 begins the process. The data for car and non-car data is stacked and scaled with the ***sklearn. StandardScaler***. The labels for the car and non-car data are created. Cars are labeled as one and non-cars are labeled is zero.
The data is then is shuffled and split into training and test sets. The test set was set at a size of twenty percent. On line 75 the model is trained.

Vehicle Detection Project

By Tim Tobey

March 6th, 2017

Sliding Window Search

Criteria: 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?

In the file ***project_functions_2.py*** the sliding window is implemented. It is located within the ***find_cars*** function on line120. The process begins on line 124 where the scale to be used is defined. I used a scale of 1, 1.25 and 1.5. I arrived at this range by using trial and error. I chose this because it provides good identification and few errors.

As the scale increases, the image size is reduced. This provides different proportions

Here is an example of the scaling effect on the image:

As for the overlap, I used a step of two, this again was trial and error which appears to have worked well. Two steps provided a 75% overlap given the cell size eight and the step was two.

I used a search area defined by Y coordinate starting at 400 and ending at 656. This was done primarily not to search the sky of the image.

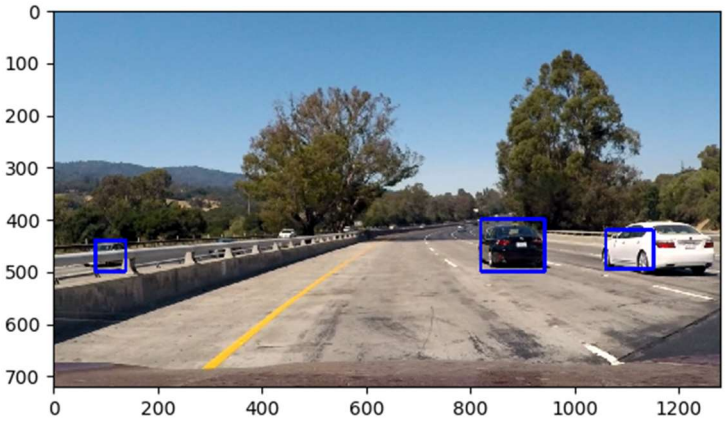
Criteria: Show some examples of test images to demonstrate how your pipeline is working. How did you optimize the performance of your classifier?

Eventually I ended up on two scales using YCrCb 3-channel HOG features plus spatially binned color and histograms of color in the feature vector. These provided a quality result. As mentioned earlier this was a trial and error process. Here are some example test images from a single image. (Note the effect of the image averaging, so the boxes are smaller than in the video):

Vehicle Detection Project

By Tim Tobey

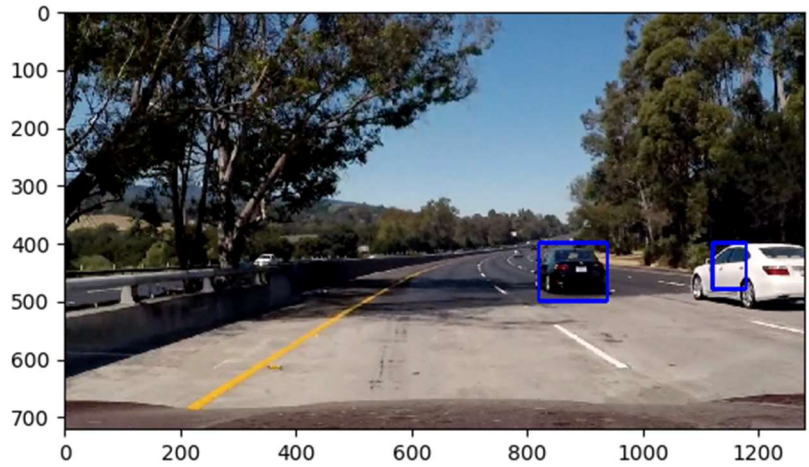
March 6th, 2017



Vehicle Detection Project

By Tim Tobey

March 6th, 2017



Vehicle Detection Project

By Tim Tobey

March 6th, 2017

Video Implementation

Criteria: **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.)**

See video in attachment. 😊

Criteria: **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.**

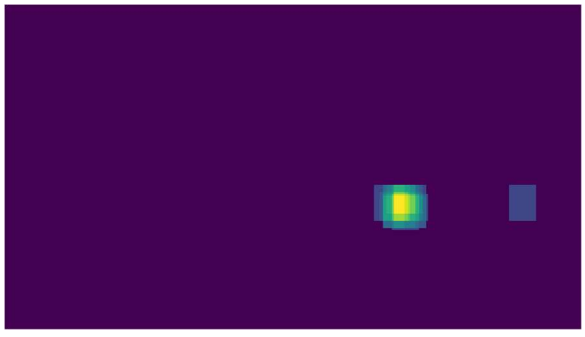
In the file ***project_main.py*** on line 40 my filtering and box averaging process begins. My code take the identified positive detections over 12 images. From the positive detections, I created a heatmap and then thresholded that map to identify vehicle positions. I then used scipy.ndimage.measurements.label() 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.

Vehicle Detection Project

By Tim Tobey

March 6th, 2017

Here's an example result showing the heatmap from a test photo, the result of scipy.ndimage.measurements.label() and the bounding boxes then overlaid on the last the image:



Test Image and Heat Map

Vehicle Detection Project

By Tim Tobey

March 6th, 2017

Discussion

Criteria: 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?

I originally did not implement multiple sliding scales and an average over multiple frames. This created errors and incomplete bounding boxes. My model has not been tested on trucks and other larger vehicles so my model my not pass this. It has not been tested on motorcycles or other small vehicles so the model could fail there as well. I could make my model more robust by using a more robust project video. The project video has limited traffic and may not be data intensive enough to create a reliable model. Obviously, I could also add more training data. I could also use multiple cameras to enhance the data to search for vehicles.