TERM 1, PROJECT 5

Udacity Self Driving Car Nano Degree

Vehicle Detection and Tracking

Sri Harsha Kunda

1. **Writeup / README**
    a. This document mentions all the rubric points and how they were implemented in the Jupyter Notebook Code.
2. **Histogram of Oriented Gradients (HOG)**

    a. **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.**
        i. I implemented HOG features by using the get_hog_features function in cell 4. The function was taken from the lecture quizzes. I used YUV color space, with 11 orientations,2 cells per block,16 pixels per cell and 'ALL' channels of HOG.
        ii. After experimenting through RGB, HSV,HLS and LUV color spaces and 0,1,2 HOG channels, I have selected the above parameter to get 0.9802 accuracy. However, I have also found with 12 orientations sometimes I got slightly higher accuracy but the processing time was long for some unknow reason.

    b. **Describe how (and identify where in your code) you trained a classifier using your selected HOG features (and color features if you used them).**
        i. Cell 6 and 7 has the code for training the classifier and the parameter for extracting hog features. I trained my classifier with all hog channels in YUV color space. I tries with Spatial features and histogram method. However the accuracy stayed around 97% and 98% and the processing time was much more higher than just extracting the hog features only.
3. **Sliding Window Search**
    a. **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?**
        i. Cell 4 has the function find_cars which takes an image and identifies hog features in the image and draws a rectangular bounding box on a car. Cell 8-10 shows the implementation of this function and sample images with search area and bounding boxes. I have experimented between 380-400 for ystart and 500-750 for ystop and scale of 0.5-3.5 to understand the difference. The higher I go in scale, the bigger objects are being detected. A lower difference between ystart and y stop and lower scale will be useful to find far objects in the image while the opposite applies for objects that are bigger and near to

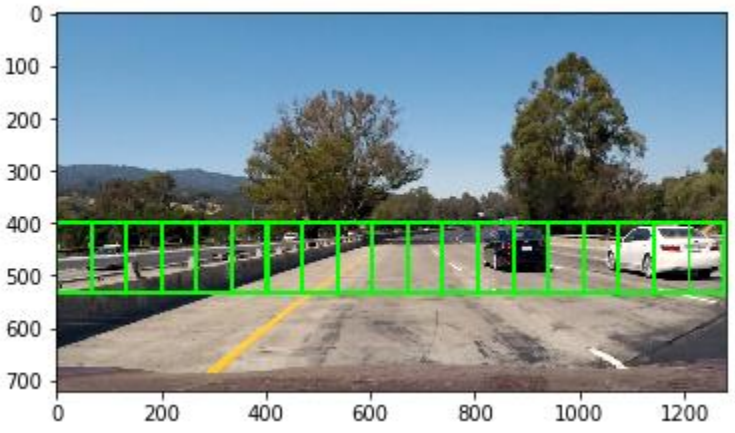
the camera, such as a car driving nearby. I created a list of values that worked on all the test images and applied them the image and video processing blocks.

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

i. I haven't spent much time optimizing the classifier, but I did play with it using different parameters such as color spaces, orient, pixels per cell, cell per block and hog channel. I tried my best for achieving a higher accuracy but at the same time decreasing the processing time for videos.

ii. I used the heat map technique mentioned in the lectures and thresholding the heat map image to remove the false positives. Much of the code was taken from lectures and modified slightly to achieve the best possible result.

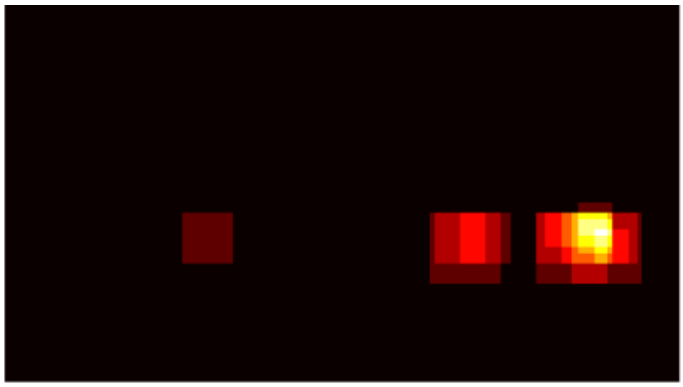
Test Image with Bounding Boxes



Search Area

TERM 1, PROJECT 5

Multiple Search Areas and Scales



Initial Heat Map

Thresholded Heat Map

TERM 1, PROJECT 5

Final Label Image

All 6 Test Images with Cars Found and Bounding Boxes

TERM 1, PROJECT 5

4. **Video Implementation**
    a. **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.)**
        i. Link to Video: https://www.youtube.com/watch?v=fX-jiOBSNs4&feature=youtu.be
    b. **Describe how (and identify where in your code) you implemented filter for false positives and some method for combining overlapping bounding boxes.**
        i. As mentioned in **3(b)(ii)** I have used heat map and thresholding techniques to remove overlapping windows and false positives. Also using few repetitive search areas with right scale values reduced too much effort into tuning the heat maps.
5. **Discussion**
    a. **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. One thing that I observed in project video is that when the white car is near a green sign board the detection fails, I am interested to know if this is due to few search windows that I created or because of the merging of features like a vehicle and non-vehicle nearby to each other confusing the classifier to classify the right way.
        ii. I would like to use TensorFlow and see if I can increase the accuracy of the classifier.