

# [Project Writeup] Vehicle Detection and Tracking

## PROJECT SPECIFICATION

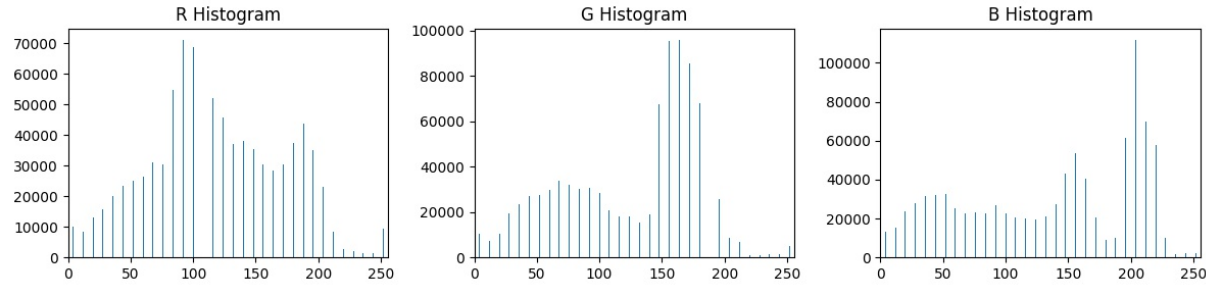
### Vehicle Detection and Tracking

Link to this document: [https://www.evernote.com/L/AC7Io0\\_R6Y9MJ5mrCPCNKaT\\_2rU-pbsZdv0](https://www.evernote.com/L/AC7Io0_R6Y9MJ5mrCPCNKaT_2rU-pbsZdv0)

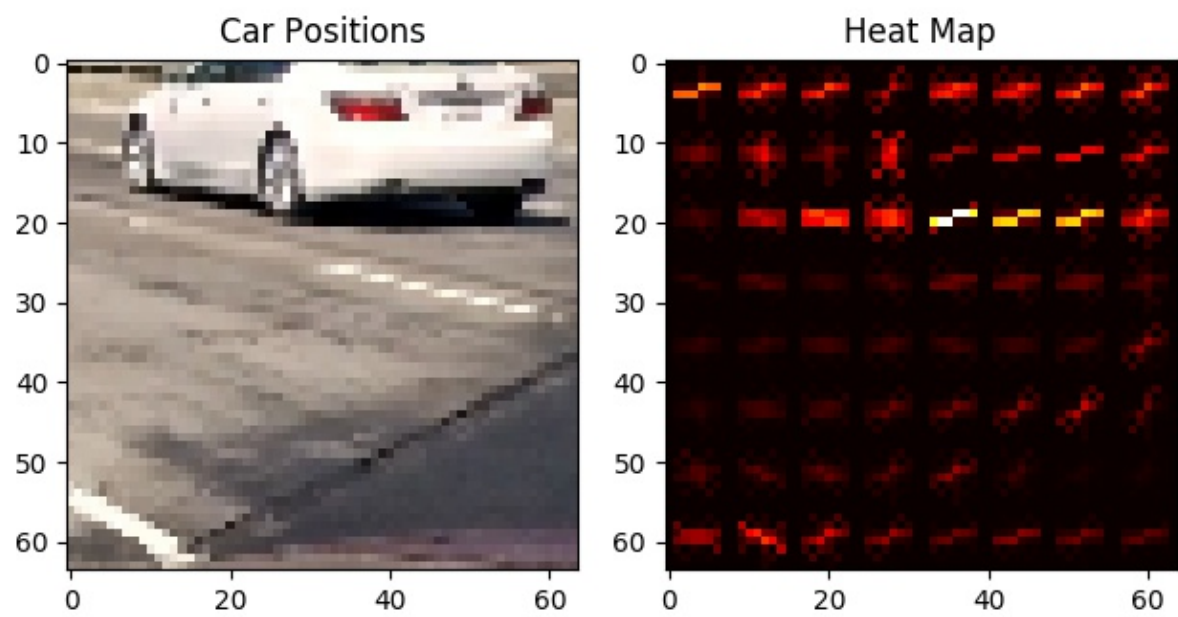
Writeup / Readme

CRITERIA	MEETS SPECIFICATION
Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. <a href="#">Here</a> is a template writeup for this project you can use as a guide and a starting point.	<p>The writeup / README should include a statement and supporting figures / images that explain how each rubric item was addressed, and specifically where in the code each step was handled.</p> <p>Important files:</p> <ul style="list-style-type: none"><li>- search_classify2.py # training happens here.</li><li>- main.py # detection and drawing happens here</li><li>- writeup.pdf</li><li>- LinearSVC_histbin_32_spacial_32_hog1506679413.pkl # trained model which gave 0.98 accuracy on test images.</li></ul>

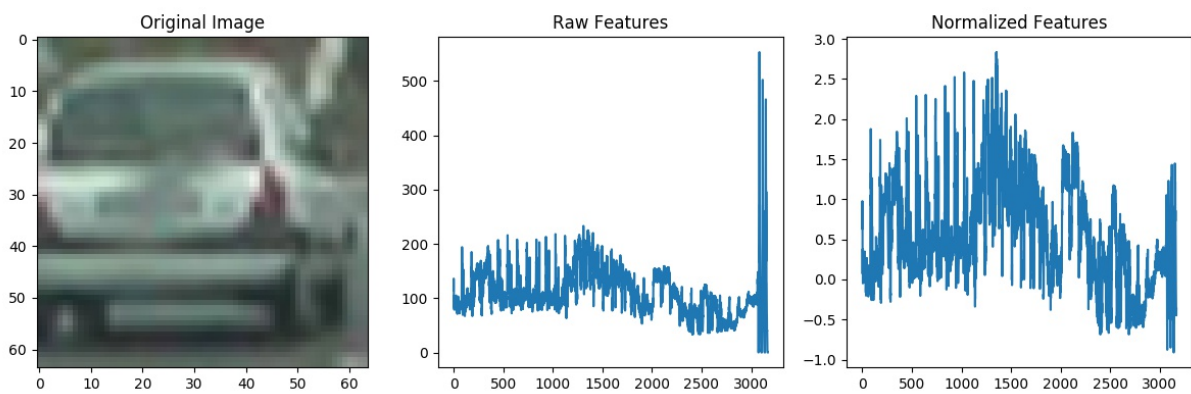
### Histogram of Oriented Gradients (HOG)

CRITERIA	MEETS SPECIFICATION
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.	<p>Explanation given for methods used to extract HOG features, including which color space was chosen, which HOG parameters (orientations, pixels_per_cell, cells_per_block), and why.</p> <p>We converted the png images to jpg. Color Space was changed to YcrBr from RGB. HOG was taken on first channel which is Y. Spatial Binning of 16x16 was done. Converting the full resolution image to 16x16 seize reduced the memory usage, reduces the parameters to compute on while retaining the features that identify the car. We take a histogram of the resulting images for each color channel. That would look like the below. We used bin size of 16.</p> <div></div>

Of this result we take the HOG of first channel, with these variables:  
orient = 9 # HOG orientations  
pix\_per\_cell = 8 # HOG pixels per cell  
cell\_per\_block = 2 # HOG cells per block  
hog\_channel = 0 # Can be 0, 1, 2, or "ALL"



Using StandardScalar transform we transform we normalise the values



Describe how (and identify where in your code) you trained a classifier using your selected HOG features (and color features if you used them).	<p>The HOG features extracted from the training data have been used to train a classifier, could be SVM, Decision Tree or other. Features should be scaled to zero mean and unit variance before training the classifier.</p> <p>LinearSVC was used to train the model. LinearSVC showed better performance that SVC. The prediction was faster that SVC. A test accuracy of 0.9814 was achieved. StandardScalar was used to normalise the data before training.</p>
---	--

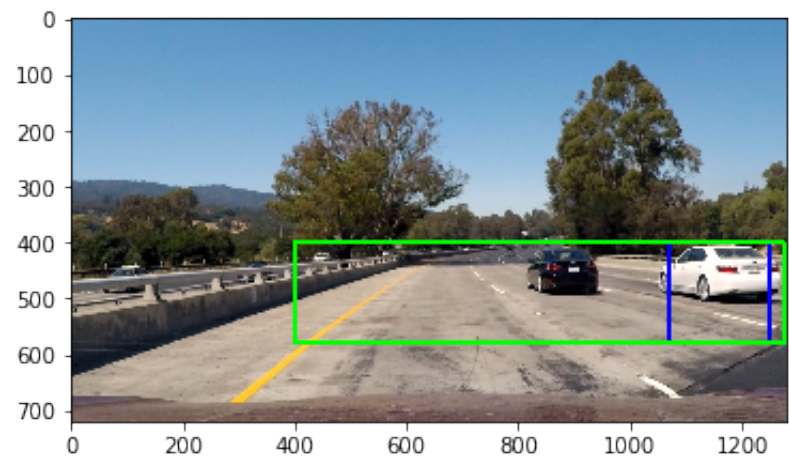
Sliding Window Search

CRITERIA	MEETS SPECIFICATION
Describe how (and identify where in your code) you	A sliding window approach has been implemented, where overlapping tiles in each test image are classified as vehicle or non-vehicle. Some justification has been given for the particular implementation chosen.

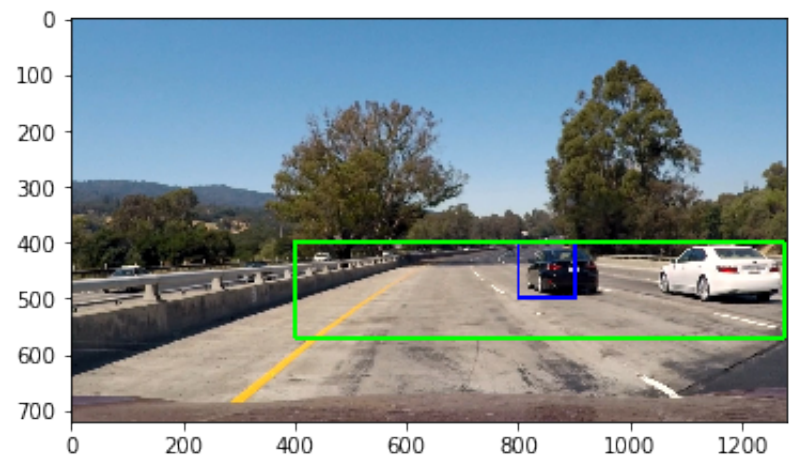
implemented a sliding window search. How did you decide what scales to search and how much to overlap windows?

Four Window sizes where used. Each bounding to a specific area in the road. Different sliding overlaps where also used.

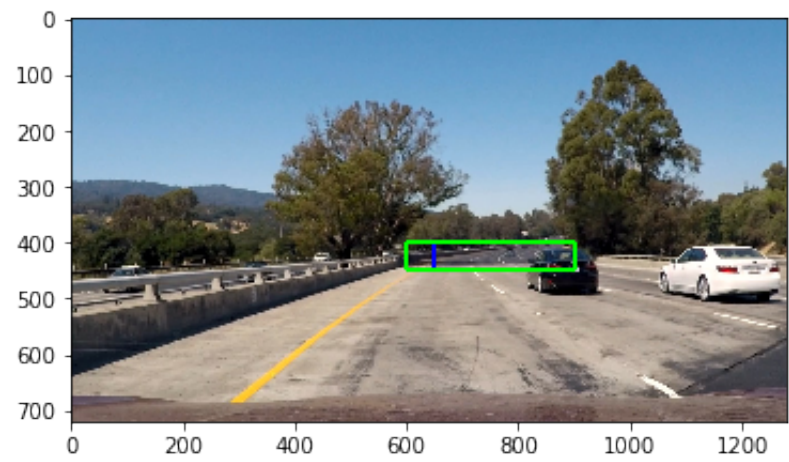
Large  
Bounding rectangle: (400, 400), (1275, 580)  
Window size : 180x180



Medium  
Bounding rectangle: (400, 400), (1275, 570)  
Window size : 100x100

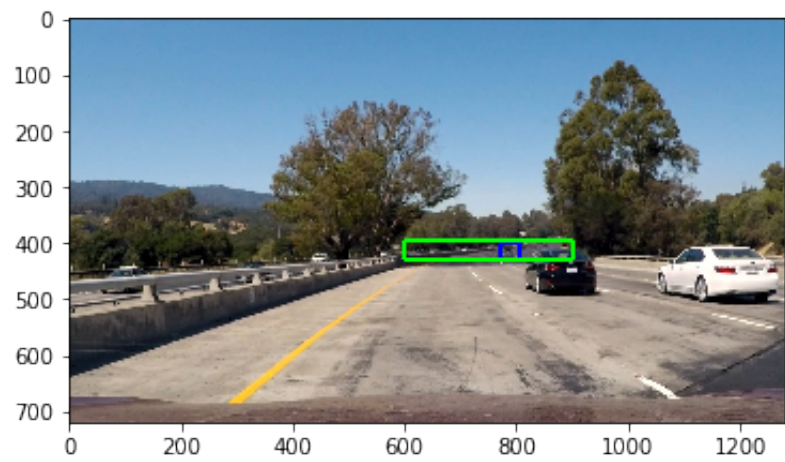


Small  
Bounding rectangle: (600, 400), (900, 450)  
Window size : 50x50

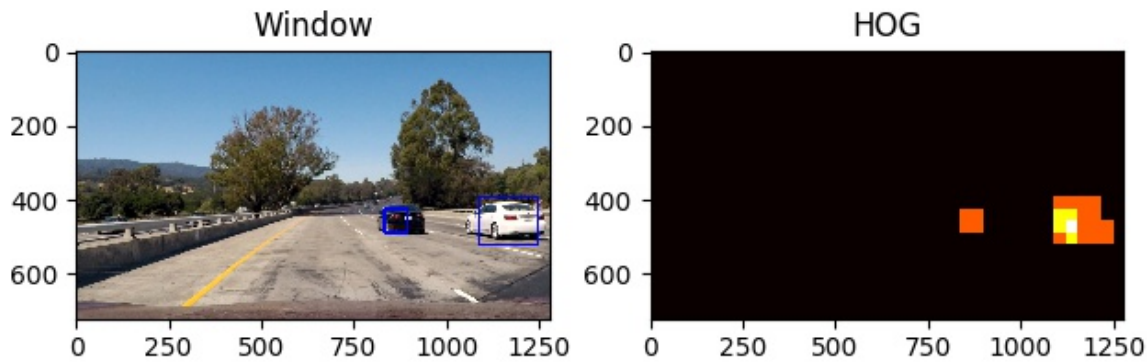


X-Small (cars at the extreme distance)\_ these cars couldn't be caught since the heat map was defined all window wide, and the heat threshold was not enough.

Bounding rectangle: (600, 395), (900, 430)  
Window size : 35x35



Heat map sample.




Show some examples of test images to demonstrate how your pipeline is working. How did you optimize the performance of your classifier?	<p>Some discussion is given around how you improved the reliability of the classifier i.e., fewer false positives and more reliable car detections (this could be things like choice of feature vector, thresholding the decision function, hard negative mining etc.)</p> <p>Good prediction reduced the number of false positives. Heat map threshold was set to 2 to avoid showing any false positives.</p> <p>The area where the road ends was omitted from window area to avoid detecting unnecessary cars.</p>
---	--

Video Implementation

CRITERIA	MEETS SPECIFICATION
Provide a link to your final video output. Your pipeline should perform	<p>The sliding-window search plus classifier has been used to search for and identify vehicles in the videos provided. Video output has been generated with detected vehicle positions drawn (bounding boxes, circles, cubes, etc.) on each frame of video.</p> <p>Please find the car video below.</p>



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.)	<div> Generic File</div> <p>I have not yet submitted the advanced lane line detection. That's why submitting without lane lines marked.</p>
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.	<p>A method, such as requiring that a detection be found at or near the same position in several subsequent frames, (could be a heat map showing the location of repeat detections) is implemented as a means of rejecting false positives, and this demonstrably reduces the number of false positives. Same or similar method used to draw bounding boxes (or circles, cubes, etc.) around high-confidence detections where multiple overlapping detections occur.</p> <p>Heat map threshold of overlapping detections was set to 2 to avoid showing any false positives.</p>

Discussion

CRITERIA	MEETS SPECIFICATION
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?	<p>Discussion includes some consideration of problems/issues faced, what could be improved about their algorithm/pipeline, and what hypothetical cases would cause their pipeline to fail.</p> <p>If the car is on the right most lane the window areas has to be reconfigured. Currently each from takes more that 4 seconds to render which is not feasible in realtime. Spacial binning the images might improve performance.</p>

---

**Suggestions to Make Your Project Stand Out!**

A stand out submission for this project will be a pipeline that runs in near real time (at least several frames per second on a good laptop) and does a great job of identifying and tracking vehicles in the frame with a minimum of false positives. As an optional challenge, combine this vehicle

detection pipeline with the lane finding implementation from the last project! As an additional optional challenge, record your own video and run your pipeline on it to detect vehicles under different conditions.

Rubrics template: <https://review.udacity.com/#!/rubrics/513/view>