UDACITY

PROJECT

Vehicle Detection and Tracking

A part of the Self-Driving Car Engineer Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Requires Changes

4 SPECIFICATIONS REQUIRE CHANGES

I am happy that I could review your project and seeing your hard work behind. I would say you are almost there, just read the remarks and you will be able to upload a successful submission very soon. Keep up the good work.

Writeup / README

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.

You provided a writeup with the necessary information in it. Well done.

Histogram of Oriented Gradients (HOG)

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.

You correctly used the function `skimage.feature.hog(...)` to get the HOG feautres. Also you tried out different color spaces and HOG parameters and found that the combination below works well for you. In

Machine Learning the trial and error approach is an often used technique. Nice work.

```
color_space = 'YUV' # Can be RGB, HSV, LUV, HLS, YUV, YCrCb
orient = 9
pix_per_cell = 8
cell_per_block = 2
hog_channel = 'ALL' # Can be 0, 1, 2, or "ALL"
hog_feat = True # HOG features on or off
```

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.

You extracted color histogram features beside HOG features. You trained a Linear Support Vector Classifier based on the extracted and normalized features and you achieved >99% accuracy on the test set. Awesome.

However in the folder `./all/` there are only 2336 images. Originally Udacity provided a training set with more than 16000 images. Using less images during the training can result that the classifier is overfitted. I suggest using all the training images to have a better classifier. Also, using spatial binning features can help somewhat in this.

Sliding Window Search

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.

You implemented the sliding window approach here with five different sizes of widows (scales: 1.0, 1.25, 1.5, 1.75, 2.0). You carefully chose an appropriate overlap ratio and an y-axis range to narrow the search. Nice work.

However it seems that two different approach are mixed here. In function `find_cars`, which is not used in the pipeline, the so called HOG Sub-sampling technique is used. In this we use scales and for each scale we pre-calculate the HOG-features for the whole image and so we can save significant amount of CPU time. You can read more about this technique in this lesson.
Without this technique we simply cut out subimages, we resize them to 64x64 and get the features which can be checked with the classifier. So here, we only need a window size without any scale. In function `find_carsII(...)` this second approach is used, but for some reason there is a scale parameter, too. The determination of the windows on the images are calculated based on the window size (variable `xy_window`) and overlap ratio, so far so good. But the feature extraction uses the `scale` parameter. The problem is that the two parameters are not consistent: in each call the `xy_window=(64,64)`, while the `scale` parameter varies from 1.0 to 2.0. To fix the issue, one of these parameter can be left out (e.g. the `scale`), and the other one should be used consistently. After this change, also, the calls might have to be corrected.

(The other solution with `find_cars(...)` gives better CPU usage, therefore you might want to consider using that one instead of the current one.)

There are frames in which the white car is clearly visible and yet it is not detected (false negative detections). One reason for this can be that for certain window sizes the overlap ratio is not big enough, try to increase the overlap ratio.

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.)

To eliminate false positives you created a heatmap from the pixels of classified images. The intuition behind is that those pixels belong to cars where there is many positive classification which means hot pixels in the heatmap. Well done.

However it seems that there are many false positive detections in the video, so further steps are needed. Here are some ideas:

- To get even better classification results you might want to fine tune the `C` parameter of `LinearSVC` classifier which controls between the training accuracy and generalization. As the test and training images came from the same source, high test accuracy might imply overfitting. Agains overfitting you need more generalization, meaning C<1.0 values. You can try 0.01 or even 0.0001.
- Also you can threshold the result of the method `decision_function(...)` of `LinearSVC` to avoid false positives.
- If there are still some false positives, you can try negative mining: cropping out non-car images from the project video, adding them to the training set and finally retraining the classifier.

Video Implementation

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.

The sliding window search with SVC classifier has been used correctly to identify the vehicles. Great job.

However there are still many false positive and negative detections in the video and the bounding boxes of the detected vehicles are not stable, sometimes splitting. To overcome these issues you might consider applying the ideas given in the comments of the previous and next rubric points.

Another issue in the pipeline is that the heatmap-thresholding happens inside the loop of different window sizes. I suggest using the thresholding after the loop, so that the thresholding works on "more information" and this can result better detections.

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.

Here the idea is utilizing the fact that in the subsequent frames the cars are located at or near the same positions, while false positives are present only for 1-2 frames. There is no such optimization in the source code.

The simplest way to implement this is using multi-frame accumulated heatmap: just store the heatmap of the last **N** frames (**N** can be 5 or 8) and do the same thresholding and labelling on the sum (or average) of these heatmaps.

As a side effect this techniqe results much more stable bounding boxes as well.

To store the heatmaps I suggest using `collections.deque`, in this way you do not need to delete the oldest heatmap:

```
from collections import deque
history = deque(maxlen = 8)
...
history.append(current_heat_map)
...
```

Discussion

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.

You provided a reflection on your own work. Well done.

RESUBMIT

DOWNLOAD PROJECT

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH

Student FAQ