Project 5 Report – Vehicle Detection and Tracking

Rafael Barreto

INTRODUCTION

In this project each frame from the project video was processed for Vehicle Detection and Tracking. The classification task was made suing Support Vector Machines. HOG and Color features were extracted and used in the classifier. Some techniques to remove all false positives were applied with success.

LOAD THE IMAGES AND MAKING A DATASET

All images were loaded and placed in two lists: one list of car images and other list of non- car images. The list of car has 7563 images and the list of non-car has 8968, totalizing 16531 images.

This dataset has a large number of images and several of them are similar. Therefore to avoid problems of overfitting and lack of memory when processing all images only 4000 of each label were used. Small sizes could be used, as 2000 or 1000 images, and similar results during the training the classifier could be achieved.

EXTRACT THE FEATURES USED FOR CLASSIFICATION

During the project, all three features were analyzed: Histogram of Oriented Gradients (HOG) classify and Color classify(binned and histogram color features). After several attempts, the better results came when all the three features are concatenate at the same vector. Furthermore, the results varied significantly, depending on the color space used. I got more stable bounding boxes and less false positives with the color space “YCrCb”, even making choices mainly based on results published by researchers (for instance [1]), all channels of this color space generated better results.

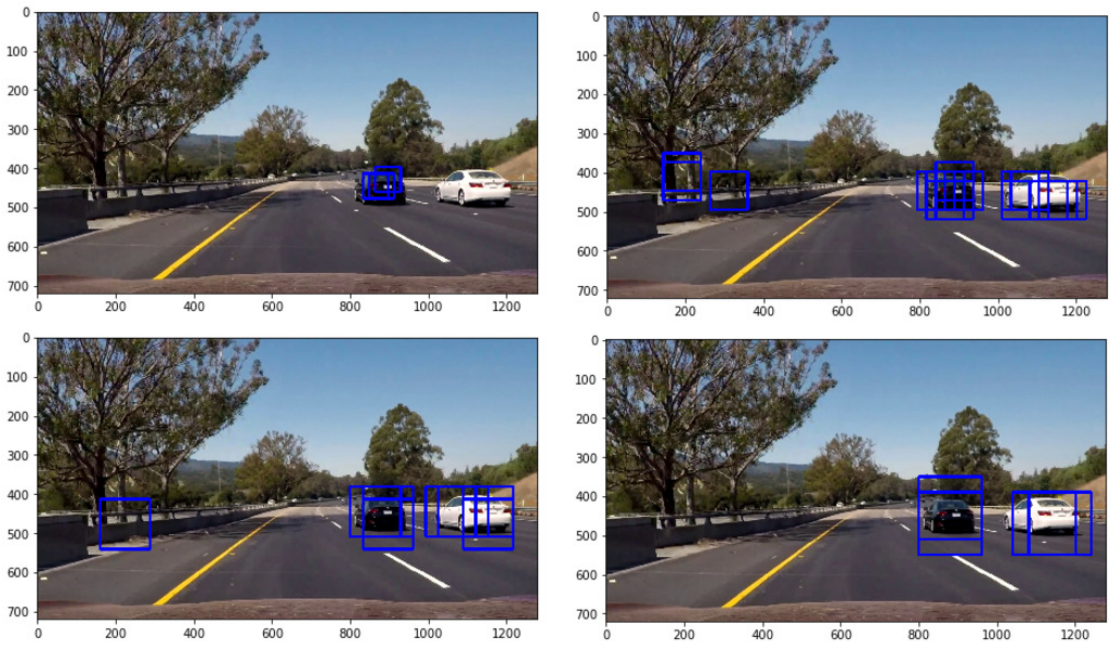
BUILD AND TRAIN THE CLASSIFIER

For binary classification as this project Support Vector Machines are very popular choice and I achieved always more than 95% of test accuracy. So I decided keep this suggestion from Udacity. The final tuned parameters after several attemps were:

```
### Parameters
color_space = 'YCrCb' # Can be RGB, HSV, LUV, HLS, YUV, YCrCb
orient = 9  # HOG orientations
pix_per_cell = 8 # HOG pixels per cell
cell_per_block = 2 # HOG cells per block
hog_channel = 'ALL' # Can be 0, 1, 2, or "ALL"
spatial_size = (32, 32) # Spatial binning dimensions
hist_bins = 32    # Number of histogram bins
spatial_feat = True # Spatial features on or off
hist_feat = True # Histogram features on or off
hog_feat = True # HOG features on or off
y_start_stop = [None, None] # Min and max in y to search in slide_window()
```

HOG SUB-SAMPLING WINDOW SEARCH

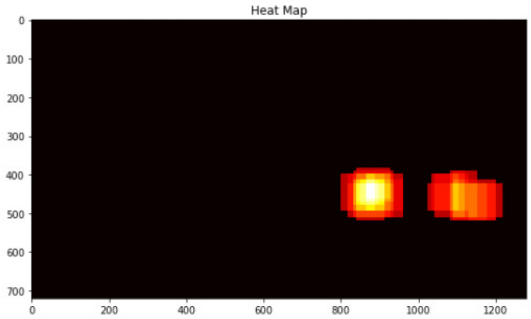
After tried various combinations, four sizes of windows were considered enough for detection from different distances. After sliding some windows some false positives appear and it will necessary remove them later. The scales chosen were: 1.0, 1.5, 2.0 and 2.5.



FILTER OUT THE FALSE POSITIVES AND AVERAGE HEATMAP

Filtering the false positives detections was the most difficult part of the project. Increasing the threshold level filter damages the true positives detections too. Besides this problem, even consecutive frames have detections with considerable changes, especially in false positives. However, it was observed that the true positives tend to be more constant over several consecutive frames, so it was performed and average heatmap using single heat features in the last ten frames.

The average heatmap solves other two problems: the occasional detection of cars from the opposite track and the high instability of the bounding boxes sizes along the consecutive frames.



CALCULATE THE DISTANCE BETWEEN CARS

The distance between the car and the cars detected were made using the theory of the euclidean distance between two points in the Cartesian Plane System. This calculus is made after an approximate conversion between pixel and meters, but the result is not always accurate.

The two points considered were the center of the bounding boxes and the half bottom of the image. Below is shown the function that computes the distance:

```python
### CURVATURE MEASURE

# This function receive the centers of the rectangles drawed
def distance_measure(bbox):

    # Define conversions in x and y from pixels space to meters
    ym_per_pix = 30.0/720 # meters per pixel in y dimension
    xm_per_pix = 3.7/700 # meters per pixel in x dimension

    #bbox = ((x1, y1), (x2, y2))
    x1 = (bbox[0][0])*1.0
    y1 = (bbox[0][1])*1.0
    x2 = (bbox[1][0])*1.0
    y2 = (bbox[1][1])*1.0

    center = (((x1+x2)/2 , (y1+y2)/2))
    center_m = (center[0]*xm_per_pix , center[1]*ym_per_pix)

    # (720, 1280, 3) Image shape

    car_front = (((1280./2)*xm_per_pix), ((720.)*ym_per_pix))

    distance = np.sqrt((car_front[0] - center_m[0])**2 + (car_front[1] - center_m[1])**2)

    return distance, center
```

RUN THE PIPELINE ON THE VIDEO

The pipeline was applied in two videos. At first an short length test video and later applied on the final output video of the Project 4 – Advanced Finding Lines. The final result of each frame is shown below:

CONCLUSION

The pipeline achieved reasonable results along after all steps. It was capable detection and tracking the cars positions and measure the distance, but the accuracy is median because of the bounding boxes not perfect match with the cars.

The average heatmap allows keep the threshold not too high and eliminate all false positives during the video. Besides this, it allows bounding boxes with a more stable shape.

REFERENCES

[1] *Feature extraction for Vehicle Detection using HOG+:*

https://medium.com/@mohankarthik/feature-extraction-for-vehicle-detection-using-hog-d99354a84d10