Project 5 - Vehicle Detection and Tracking

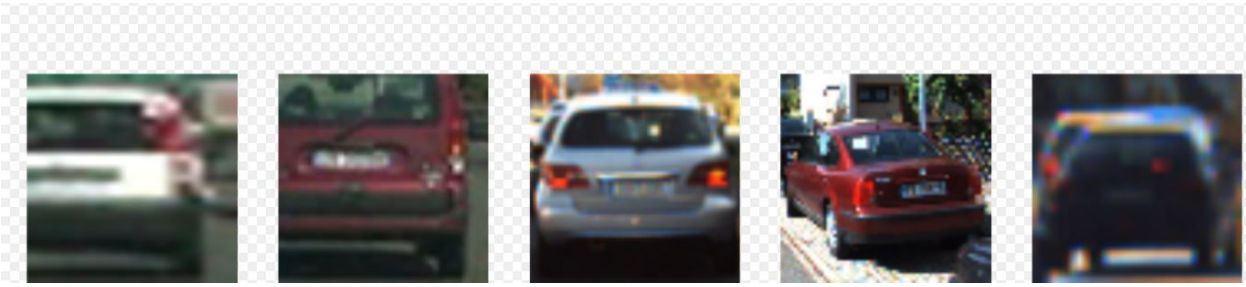
The goals / steps of this project are the following:

- Perform a Histogram of Oriented Gradients (HOG) feature extraction on a labeled training set of images and train a classifier Linear SVM classifier
- Optionally, One 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 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.

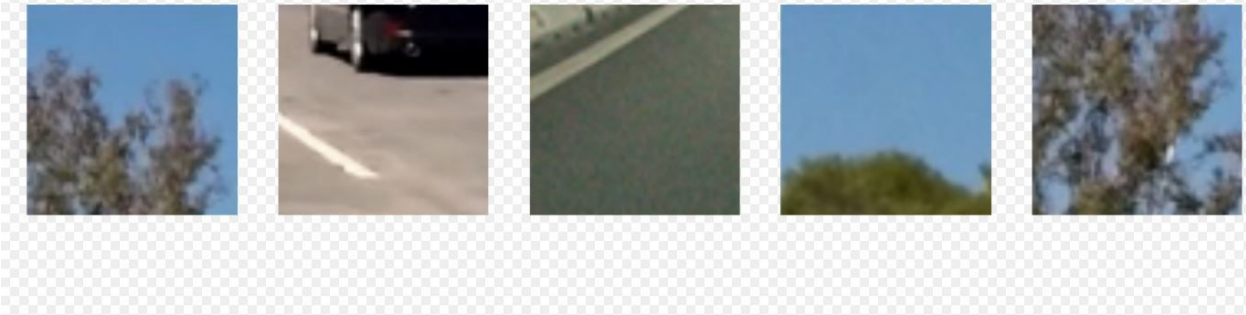
Loading DataSets and Visualizing

The sample images of **car and Non-car** data set is following

Cars



NonCars



The List of functions utilized in the project are -

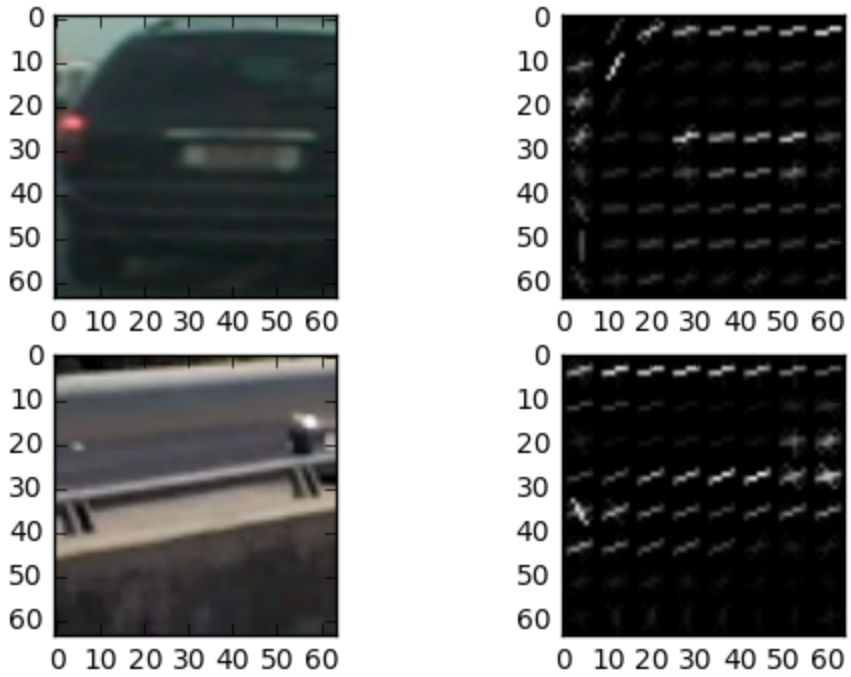
1. Get_hog_features - *return HOG features and visualization*
2. Bin_spatial - *compute binned color features and return features vector*
3. Color_hist - *Return the individual histograms, bin_centers and feature vector*
4. Single_img_features - *Returns concatenated features of a single image*
5. Extract_features - *extract features from a list of images*
6. Search_windows - *Return windows for positive detections*
7. Slide_window - *Returns list of windows*
8. Draw_boxes - *function to draw bounding boxes*
9. get_s_from_hls

HOG Features

HOG Parameters -

- color_space = '**RGB**' # Can be RGB, HSV, LUV, HLS, YUV, **YCrCb**
- orient = 8 # HOG orientations
- pix_per_cell = 16 # HOG pixels per cell
- cell_per_block = 1 # HOG cells per block
- hog_channel = "ALL" # Can be 0, 1, 2, or "ALL"
- spatial_size = (16, 16) # Spatial binning dimensions
- hist_bins = 16   # Number of histogram bins
- spatial_feat = False # Spatial features on or off
- hist_feat = False # Histogram features on or off
- hog_feat = True # HOG features on or off

Example of hog images is in below plot for ***car*** and ***Non-car*** image is below



- The HOG parameters were finalized after trying several combinations in the quizzes section by keeping some of them constant and changing the remaining.

- Choose *SVC* as classifier
- Trained the classifier with hog_data features collected from feature extraction.
- *LinearSVC* gave an accuracy of 95% and SVC gave an accuracy of 99%.

```
car_features = extract_features(comb_cars, color_space=color_space,
                        spatial_size=spatial_size, hist_bins=hist_bins,
                        orient=orient, pix_per_cell=pix_per_cell,
                        cell_per_block=cell_per_block,
                        hog_channel=hog_channel, spatial_feat=spatial_feat,
                        hist_feat=hist_feat, hog_feat=hog_feat)

notcar_features = extract_features(comb_noncars, color_space=color_space,
                        spatial_size=spatial_size, hist_bins=hist_bins,
                        orient=orient, pix_per_cell=pix_per_cell,
                        cell_per_block=cell_per_block,
                        hog_channel=hog_channel, spatial_feat=spatial_feat,
                        hist_feat=hist_feat, hog_feat=hog_feat)
```

Sliding Window Search

Initially tried various sizes of windows on the test image and also restricted the area to cars visibility and ignored sky and complete ground

128 x 128 windows

96 x 96

80 x 80

Here is the snippet of the code used to compute hot boxes and applying heat map and then averga boxes eliminating unnecessary portion like false positives and the repeaters.

```
image = image_orig.astype(np.float32)/255

# hot boxes
hot_boxes, image_with_hot_boxes = get_hot_boxes (image)
# heat map
heat_map = get_heat_map (image, hot_boxes)

# average boxes
avg_boxes = calc_average_boxes (hot_boxes, 2)
image_with_boxes = draw_boxes(image, avg_boxes, color=(0, 0, 1), thick=4)
```

Sliding Windows with heat maps

Below plotted are all test images with their hot boxes , heat maps and after averaging boxes. You can clearly observe above snippet implementation on the test images in the images below.


hot boxes
heat map
average boxes

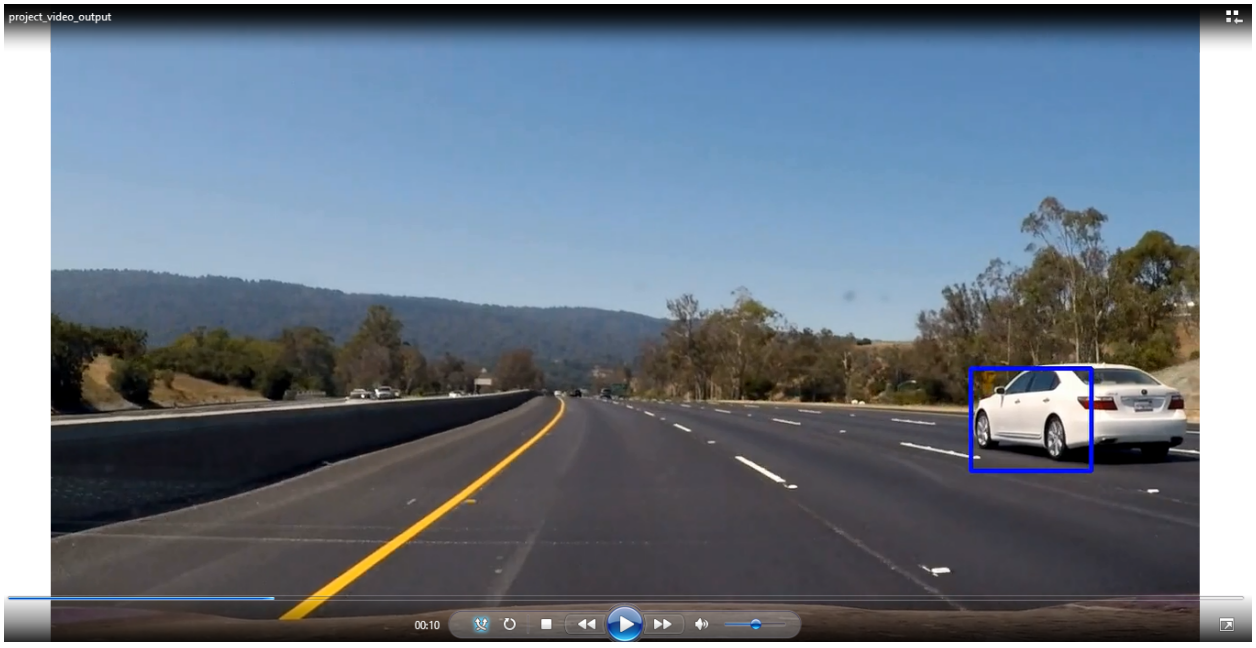
Video Pipeline

Here is the snippet showing how do i process an image applying all the necessary steps like calculating hot boxes and then applying heat map. This process image function is used on every frame of a video and accumulating hot boxes over last 10 images.

```
def process_image (image1):

    image1 = np.copy (image1)
    image = image1.astype(np.float32)/255

    # accumulating hot boxes over 10 last frames
    hot_boxes, image_with_hot_boxes = get_hot_boxes (image)
    last_hot_boxes.put_hot_boxes (hot_boxes)
    hot_boxes = last_hot_boxes.get_hot_boxes ()

    # calculating average boxes and use strong ones
    # need to tune strength on particular classifer
    avg_boxes = calc_average_boxes (hot_boxes, 20)
    image_with_boxes = draw_boxes(image, avg_boxes, color=(0, 0, 1), thick=4)

    return image_with_boxes * 255
```

Here are some outputs from the video after running through the pipeline

1) One car being identified by the box



2) 2 cars being blocked in hot boxes in the video.

Here is the link to my project video output - [Project_Video_Output]

Discussion / challenges

1. The main components of this project are :-
    a. Features extraction
    b. Training a classifier with extracted features
    c. Finding the hot boxes using sliding windows and heat maps.

2. Run the above steps on all the frames in a video and keep track of last 10 frames to eliminate duplications.

Challenges -

- To increase the accuracy of prediction, i should include more non car images into the data set and train the classifier. My further exploration would be increasing the dataset by extracting data sets such as Autti dataset. I can augment the data from those datasets and then extend the non car images so that false positives would become zero.
- Another exploration would be linking this algorithm with advanced lane identification algorithm.