Vehicle Detection and Tracking

Feature Extraction:

I used a combination of Spatial and Histogram color features along with HOG features with the following parameters set to get an accuracy of 99.01 using Linear SVC classifier.

Color Space:

I have used 'HLS' color space as it gave the best possible accuracy for the classifier. 'HSV' color space also performed on par.

Histogram of Oriented Gradients (HOG):

Orient = 9
Pixels per cell = 8
Cells per block = 2x2
Channels = 0, 1, 2 (Used all 3 channels as it provided the best results)

Spatial Color Features:

Spatial size = (16, 16) – This is used, instead of (32, 32), to reduce the overall # of features to be trained.

Color Histogram Features:

Range of histogram = (0, 256) - To accommodate all possible color spaces
Number of bins used for histogram = 16 - This is used, instead of (32), to reduce the overall # of features to be trained.

Implementation Details:

Project includes following files to extract features:

- features.py is used to extract the features in desired color space.
    - It provides a function to extract vehicle and non-vehicle features from the data set provided by Udacity.
    - Features extracted from the above function are normalized so as to avoid the case where some features dominating the others.
    - It also provides a function to convert a RGB image to image in desired colored space.
- spatial_bin.py is used to extract spatial features
- color_hist.py is used to extract histogram color features
- hog.py is used to extract HOG features
    - Used skimage's hog feature function to extract the features. However, this function seemed to be slower compared to OpenCV's HOGDescriptor.

Classifier:

I chose LinearSVC provided by sklearn library for identify vehicles in an image. I've also used sklearn library's CalibratedClassifierCV with isotonic regression method, to get a measure of probabilities for the prediction.

Implementation Details:

- classifier.py file contains function 'train_linear_SVC_classifer' that is used for training the classifier.
- Used 7 epochs to train the classifier
- I have tweaked the C parameter in the range of [0.1, 10] and settled with a value of 0.5. This seemed to be a good compromise between accuracy without over-fitting.
- Any prediction of vehicle is discarded if the confidence level is less than 0.95.

Hog Sub-sampling with Heat Maps:

I have used Hog sub-sampling to search an image for vehicles. This method provided to faster than the sliding window approach in the sense that you don't have to extract HOG features for every possible window in the image. HOG sub-sampling will extracts the features once and then uses a subset of them while exploring a window in the image.

Implementation Details:

- classifier.py has a function 'find_vehicles_using_hog_sub_sampling' that takes in an image, scale, trained classifier, and the portion of image (ystart, ystop) and returns an array of windows.
    - 3 sample output images were included in 'sample_output_images' folder with prefix 'hog_sub_sampling'.
- heat_maps.py has useful functions to create heat maps (with a threshold of 2) based on the detected windows and they in-turn are averaged over 5 frames to eliminate false positives.
    - 6 sample output images are included in 'sample_output_images' folder.
    - Used OpenCV's 'connectedComponentsWithStats' to extract the labels for drawing the bounding boxes on the detected vehicles.

References:

1. Some of the code has been adopted from Udacity's self driving car ND's 'Vehicle Detection and Tracking' chapter's lessons.
2. http://jany.st/post/2017-02-11-finding-lane-lines-and-detecting-vehicles-in-a-video-stream.html
3. http://scikit-learn.org/stable/modules/calibration.html