Vehicle Detection

A) Vehicle Detection Project

The goals / steps of this project are the following:

a) Perform a Histogram of Oriented Gradients (HOG) feature extraction on a labeled training set of images and train a classifier Linear SVM classifier
b) Optionally, you can also apply a color transform and append binned color features, as well as histograms of color, to your HOG feature vector.
c) Note: for those first two steps don't forget to normalize your features and randomize a selection for training and testing.
d) Implement a sliding-window technique and use your trained classifier to search for vehicles in images.
e) Run your pipeline on a video stream (start with the test_video.mp4 and later implement on full project_video.mp4) and create a heat map of recurring detections frame by frame to reject outliers and follow detected vehicles.
f) Estimate a bounding box for vehicles detected.

Image References (examples):
Image-1 : test6.jpg outcome (slide_window, search_window & box_draw)
Image-2 : test5.jpg outcome (slide_window, search_window & box_draw)
Image-3 : test4.jpg outcome (slide_window, search_window & box_draw)
Image-4 : draw_box_test2.jpg outcome of slide_window
Image-5 : slide_window_test5.jpg
Image-6 : find_car_process_test6.jpg (a process outcome)
Image-7 : find_car_process_pipeline_test1.jpg (a process outcome)

Outcome example references:
*Images : ./output_images/test_images/**
*Images : ./output_images/find_cars/**
*video's: ./output_images/processed_video/**

➢ Files Submitted, Code Quality & Concern

Submission includes all required files and can be used to run the simulator in autonomous mode, my project includes the following files:

1. Software solution implemented in native python. Code file submitted :
   - vehicleDetection_main.py : main having implementation of processing algorithms
   - lesson_functions.py : having all user defined method and class including curriculam
   - global_declarations.py : file having all global declared variables
   - laneline.py : function method to integrate lanes line identification with car detection
2. Input files: following source of data has been used in project
   - test_images: as provided with project repository
   - video files: as provided with project repository
     1. test_video.mp4 (not submitted)
     2. project_video.mp4 (not submitted)
   - pickel file: curriculam section varification input data
3. Output files: Submission contain outcome examples placed in a folder "output_images", which further contains three sub folders maintaining submission data as described below;
   - test_images : folder contains images of function outcomes of "vehicleDetection_main.py"
   - find_cars : folder contains images of function outcomes of "vehicleDetection_main.py"
   - processed_video :
     - 2018-02-01 15:07:34.082588project_video_out.mp4 – pipeline execution outcome on project file
     - 2018-02-01 15:07:32.824934test_video_out2.mp4 – pipeline execution outcome on test video
4. VehicleDetection-writeup.pdf describe project summary, submission and outcome
5. python program execution (logs) : terminal log file "executionlog.txt" provides module execution debug prints
6. Additional reference : error_sbin-others.txt gives debug logs which frequently encountered in additional functionality implementation

- **Submission includes functional code & other files**

  Submission has all mentioned source file, output files and data references. Mostly code submitted is functional, output file and verbose print out come is also submitted.

  In implementation of some logics i encountered with a couple of library specific challenged (version, deprecated function/variable/system parameter etc.), details of all such challenges compiled in annexure.

- **Submission code is usable and readable**

  The code is written in python with objectives:
  - architecting a module (vehicle detection) : execution happen through single python file
  - modularity maintained : by defining independent class/method/functions
  - system level paramters : defined to avoid duplicay of code w.r.t
    - parameter initialization
    - global variables
    - input/output paths
    - flag definitions
  - hardcoding : avoided
  - debug print : placed debug print functions
  - readability : put specific remarks/comment for better understanding
  - visualisation : outcome placed in single folder with seperably identifiable file names

  Encountered with a few challenges w.r.t libraries used, details or work arounds and possible mitigation are compiled in annexure. Few point were show stopper wherein i need extended support from udacity experts.

(B) Functionalies overview

Vehicle detection module developed in python native language. I have used spyder editor to write code functions and architected considering above objectives (mentioned in above section). Software module has three key files to implement all functionalities :

1. vehicleDetection_main.py : main having implementation of processing algorithm
2. lesson_functions.py : having all user defined method and class including curriculam
3. global_declarations.py : file having all global declared variables

Code files have comments to describe high level functionality and variable definitions used in implementing them. Let us summarise objective of functions implemented in above files, execution and outcome for better understanding.

global_declarations.py : file has all variables decleration used in performing a Histogram of Oriented Gradients (HOG) feature extraction of image and train a SVM linear classifier.

lesson_functions.py : this file has definition of all user defined methods and initialization functions. Here is outliner summary all such functions -

- init_param(): initializes and return all HOG and SVM linear classifier global variables
- get_hog_f eatures() : called with a set of input parameters to extract HOG features of images
- bin_spatial() : to create feature vector of image
- color_hist() : called with specific inputs to extract color histogram of image
- extract_features() : functions take a set of images to extract features and return. Function returns a concatenated and appended feature lists of images w.r.t hog, color histogram and bin spatial
- slide_window() : function processes following on image;
  - Define a function that takes an image,
  - start and stop positions in both x and y,
  - window size (x and y dimensions),
  - and overlap fraction (for both x and y)
  - return the list of windows
- draw_boxes() : a function to draw bounding boxes
- draw_labeled_bboxes_n() : processes following
  - Define a function you will pass an image
  - and the list of windows to be searched (output of slide_windows())
- search_windows() : to predict a window in windows
- single_img_features() : a function to extract features of single image

- convert_color() : a function to convert color formats
- Section-B functions are for
  - add_heat() : calculate heatmap
  - apply_threshold() : apply a heatmap thresold
  - filt(): to smoothen the boxes
  - len_points(): calculate distance between 2 point
  - track_to_box(): create box co-ordinates out of its centre and span
  - draw_labeled_bboxes() : to draw label box
- Section-C : has different pipelines to find cars i.e inspired from various sources – curriculam, Udacity students work and other references. Multiple are tried however in many i got stuck in OpenCV library error or skimage predict feature vectors mismatch. Explaining one which sucessfully run however seems location updation from frame to frame is not giving desired objectives.
  - find_cars() : identify match making boxes (cars) in image
  - find_car_process(image) : building a pipeline to process video frames
  - class Vehicle_Detect()
    - a class to store data from video
    - maintain history of rectangles previos n frame
    - through out condition based rectangles
  - find_car_process_video(image) : building a pipeline to process video frames maintaining previous images rectangles and applying additional logics.

vehicleDetection_main.py : executes above defined functions, methods in desired manner to realize project objectives and visualize them for reference. This file at very high level achieves following functionalities:

- Setup environment of “Vehicle Detection” module : first section setups an environment for this module through following steps.
  - Import essential image processing libraries : cv2, numpy, matplot, globe, label, datetime, os and pickle
  - Import classifier libraries and functions : LinearSVC, StandardScaler, hog and label
  - User defined function/module/packages : Initialize HOG and classifier variables, function defines in lesson_functions.py
  - misc utilities : video processing, system level variable declarations
  - setting up paths : global paths for INPUT and OUTPUT
- reading data from various location : car and non car data reading limited or full data for classifier features
- Classifier Section Setup :
  - extract features of images : extract features for all car and non cars as described in above functions
  - initialize SVC and StandardScaler
  - also prepare random data for train and test to fit, transform extracted data to fit in classifier
- **Result of above steps : available in log file**
- classifier test on all test images
- slide window test on sampled test images
- apply find_car_process() pipeline on individual images and video
- apply find_car_process_video() pipeline with previous rectangle store on video’s

➢ vehicleDetection_main.py step by step execution and outcome visualisation (examples)

Let us execute vehicleDetection_main.py through python. Commond used :

```
python  vehicleDetection_main.py >> executionlog.txt
```

executionlog.txt – provides full logs for execution of program (available in submission). Here i have depicted a few sections of it.

Setup environment of “Vehicle Detection” module: data reading & classifier train/test

```
(carnd-term1) rajeev (master) CarND-Vehicle-Detection $ python test_main.py
Corners were found on 17 out of 20 it is 85.0 % of calibration images
Initialized Paramaters:
color_space : YUV
spatial_size : (16, 16)

first set of vehicles read (GTI) :  2826
Second set of vehicles read (KTTI) & append :  8792
```

First set of non vehicles read (GTI) : 3900
Second set of non vehicles read (Extras) & append : 8968

length of cars images : 500
length of non car images : 500
Car samples (features) : 500
Notcar samples (feature)) : 500

Using: 11 orientations 8 pixels per cell and 2 cells per block
Feature vector length: 7332

3.27 Seconds to train SVC...
Test Accuracy of SVC = 0.935

Apply SVC classifier of test_imges
Multiple test images read through glob and apply slide_window, search_window and draw_box function to process images individually through a for loop. Output get stored in specific path for visualisation

draw box image processed (glob & test input) : ./output_images/test_images/test5.jpg
draw box image processed (glob & test input) : ./output_images/test_images/test6.jpg
draw box image processed (glob & test input) : ./output_images/test_images/test2.jpg
draw box image processed (glob & test input) : ./output_images/test_images/test1.jpg
draw box image processed (glob & test input) : ./output_images/test_images/test4.jpg
draw box image processed (glob & test input) : ./output_images/test_images/test3.jpg
12.14 Seconds to process test images

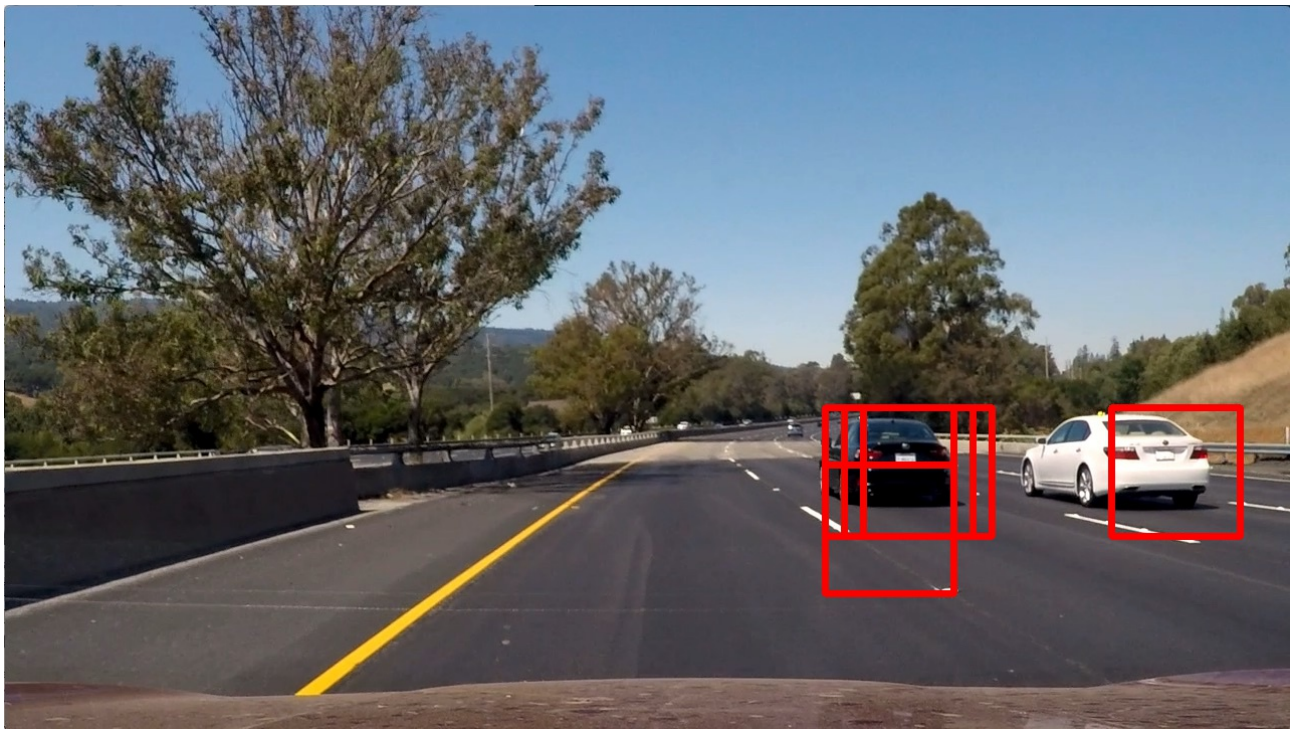

Image-1 : test6.jpg outcome (slide_window, search_window & box_draw)

Image-2 : test5.jpg outcome (slide_window, search_window & box_draw)

Image-3 : test4.jpg outcome (slide_window, search_window & box_draw)

- **Draw box and slide window function (on test image – 2 & 5)**

  This section of code for **vehicleDetection_main.py** (line's : 207 to 261) applies slide_window & draw_box functions on two test images.

  First implementation (line 209 to 222) of slide window for car detect, in a image which doesn't have car's can identify fake objects. Slide window parameter tunning can also impact outcome which visualized in next 2 images.

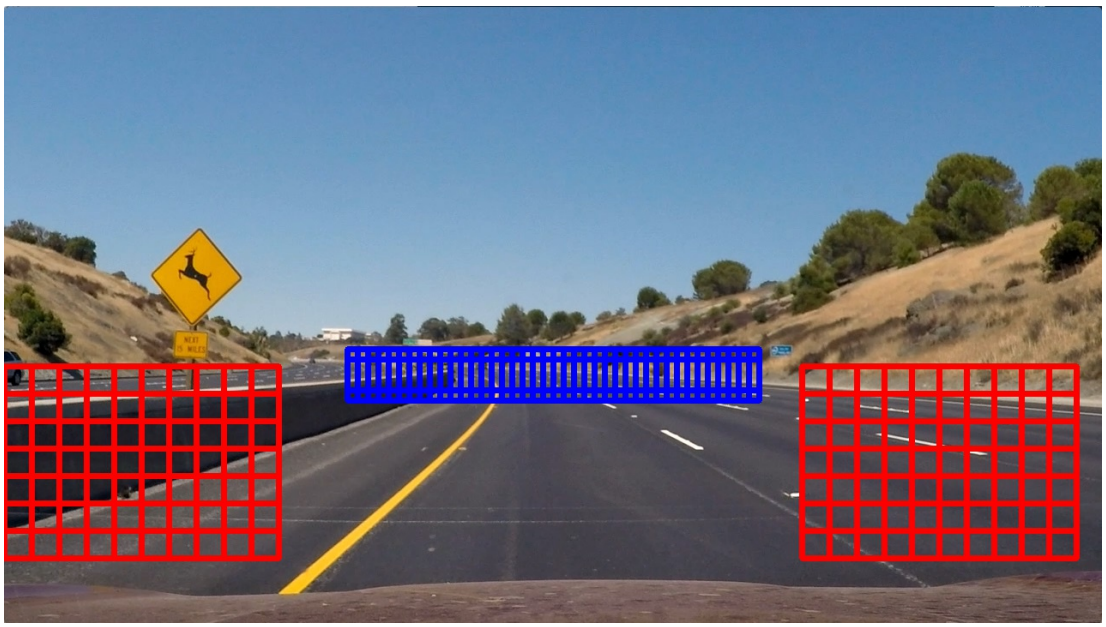

Image-4 : draw_box_test2.jpg outcome of slide_window

Slide_window function tested on test6.jpg image for specific values of input parameters (track, w_size) to capture cars and visualize. Track variable is a representation of x_start_stop and y_start_stop inputs in association with w_size. Function outcome captures cars, draw boxes and stores in a file for visualisation.

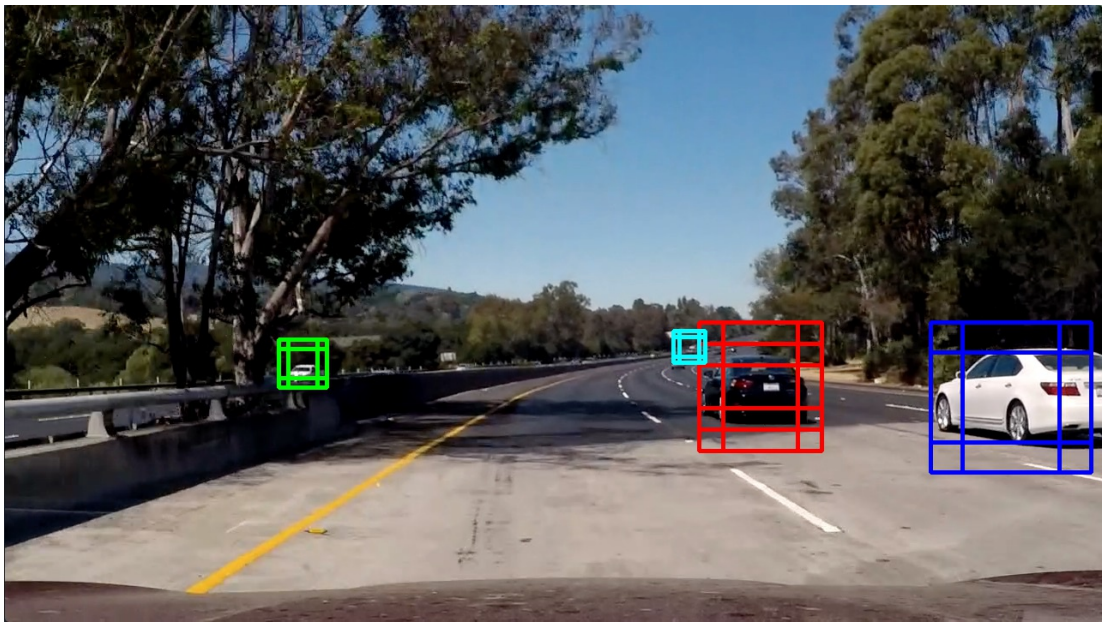

Image-5 : slide_window_test5.jpg

- **Define process – find_car_process() & test**

This is a processes pipeline (line 700 to 739) defined (in file - lesson_functions.py) to identify cars in images. Process pipeline tested on test6.jpg in file vehicleDetection_main.py (line's : 261 to 270) and outcome stored in a file (find_car_process_test6.jpg)

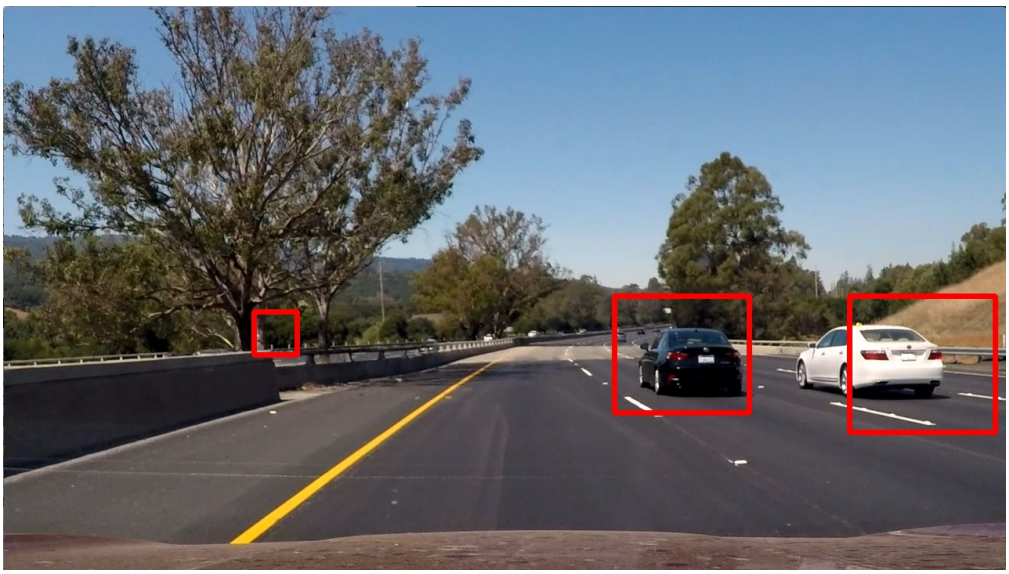

Image-6 : find_car_process_test6.jpg (a process outcome)

Process pipeline tested (code lines – 287 to 297 of vehicleDetection_main.py) on all test images through glob and outcomes stored in folder "/output_images/test_images".

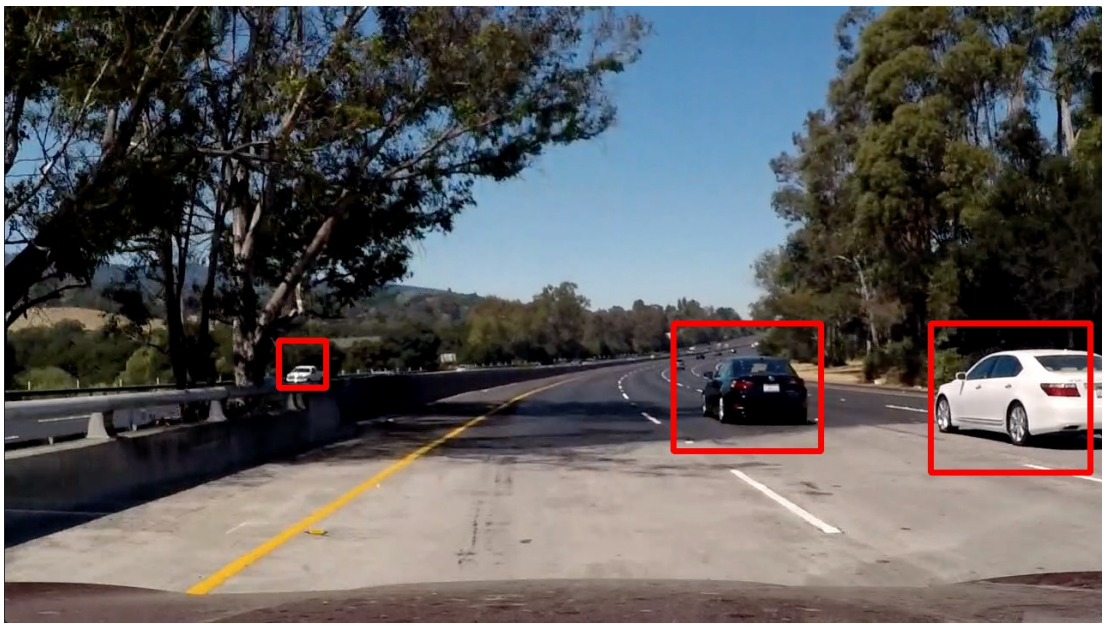

Image-7 : find_car_process_pipeline_test1.jpg (a process outcome)

➢ Define Process Pipeline & Test

find_car_process() pipeline test with video (test_video.mp4) in file vehicleDetection_main.py (lines 299 to 306), also visualized outcome stored in path : /output_images/processed_video/

Outcome file : 2018-02-01 15:02:55.682760test_video_out.mp4

Pipeline further improvised by introducing a class and method to maintain state of previous identified box (cars) and adjust cusrrent position. New pipeline is find_car_process_video () defined in lesson_functions.py (line 742 to 797). find_car_process_video () pipeline test with video (test_video.mp4) in file vehicleDetection_main.py (lines 299 to 306), also visualized outcome stored in path : /output_images/processed_video/

outcome file : 2018-02-01 15:07:32.824934test_video_out2.mp4

➢ Process - Project Video

find_car_process_video () pipeline test with video (project_video.mp4) in file vehicleDetection_main.py (lines 299 to 306), also visualized outcome stored in path : /output_images/processed_video/

outcome file : 2018-02-01 15:07:34.082588project_video_out.mp4

Summary:

lesson_functions.py contains all function definitions and algorithm implementation. vehicleDetection_main.py provides vehicle detection module functionalities execution framework. Module has various additional functions learned in curriculam. Most of the objectives mentioned in project objects get implemented, also executed in silo's however while i was tried to run a matured pipeline (updating vehicle location in video) than encountered with libraries related issues/challenges (annexure & errorlog.txt file provides there detail – openCv, memory or SVC classifier specific).

The video outcome is available with submission for review.

- All execution done Ubuntu 17.10, GPU (GeForce GTX 1060)/CPU grade machine and compute execution being fairly fast.
- /output_images/processed_video/ folder contains all processed video's

Project has several additional functions, pipelines and functionalities which has been tried as optional material. Intention being to implement a full operational vehicle detection module. However almost all advance logic stuck in libraries issues (OpenCv transform, resize or scalar predict) or errors whose solution is not identified. Apart of these many other concerns encountered which were resolved through a work around found in open community as compiled in next section.

Annexure – Challenges & resolution (or work around)

Challenge-1 : warning w.r.t skimage_deprecation & openCV error throwing in resize

/home/rajeev/anaconda3/envs/carnd-term1/lib/python3.5/site-packages/skimage/feature/_hog.py:119: skimage_deprecation: Default value of `block_norm`=='L1` is deprecated and will be changed to `L2-Hys` in v0.15
'be changed to `L2-Hys` in v0.15', skimage_deprecation)

OpenCV Error: Assertion failed (dsize.area() > 0 || (inv_scale_x > 0 && inv_scale_y > 0)) in resize, file /home/travis/miniconda/conda-bld/conda_1486587071158/work/opencv-3.1.0/modules/imgproc/src/imgwarp.cpp, line 3230
Traceback (most recent call last):

```
  File "test_main.py", line 351, in <module>
    car_boxes = find_cars(image, ystart, ystop, scale, colorspace, hog_channel, svc, None, orient, pix_per_cell, cell_per_block, None, None)
  File "/home/rajeev/Self Driving Car/Udacity/Udacity_full_workingProject/Term1-CarsIdentification/CarND-Vehicle-Detection/lesson_functions.py", line 618, in find_cars
    spatial_features = bin_spatial(subimg, size=spatial_size)
  File "/home/rajeev/Self Driving Car/Udacity/Udacity_full_workingProject/Term1-CarsIdentification/CarND-Vehicle-\
Detection/lesson_functions.py", line 56, in bin_spatial
    features = cv2.resize(img, size).ravel()
cv2.error: /home/travis/miniconda/conda-bld/conda_1486587071158/work/opencv-3.1.0/modules/imgproc/src/imgwarp.cpp:3230: error: (-215) dsize.area() > 0 || (inv_scale_x > 0 && inv_scale_y > 0) in function resize
```

Resoulution : work arounds explored through open community and tried with different function
Step-1 : vi /home/travis/miniconda/conda-bld/conda_1486587071158/work/opencv-3.1.0/modules/imgproc/src/imgwarp.cpp
Step-2 : comment line : CV_Assert(ssize.area() > 0);
CV_Assert(dsize.area() > 0);

Challenge-2 : skimage site packeges throughing warning w.r.t to normalisation level and misbehaviour of algorithm.
/home/rajeev/anaconda3/envs/carnd-term1/lib/python3.5/site-packages/skimage/feature/_hog.py:119: skimage_deprecation: Default value of `block_norm`=='L1` is deprecated and will be changed to `L2-Hys` in v0.15 'be changed to `L2-Hys` in v0.15', skimage_deprecation)

Solution : Changing the recommended value for trial : line-26 _hog.py function input value changed to 'L2-Hys'

Challenge-3 : data.py function returning error in Tranform function (xxxx)
Solution : X_scaler = StandardScaler(copy=True, with_mean=False, with_std=False).fit(X) # Fit a per-column scaler

Initialized with changing default values of flags as : *copy=True, with_mean=False, with_std=False*

Challenge-4 : Highlight points w.r.t changes made in file and functions
Init () function

- 3 pixel block didn’t worked
- repurcussion of change of flag – SVC accuracy level dropped
- Hog features tried – All channel and 0 one

Challenge-4 : could not able to find out a way to represent HOT through plt.imshow(heatmap_img, cmap='hot')
Solution : not able to find out

Challenge-4 :

(a) Sampled log piece of one error commonly encountered

```
################
draw outcome :  ./output_images/find_cars/test1.jpg
...
(6468,) hog feature shape
Traceback (most recent call last):
  File "test_main.py", line 410, in <module>
    spatial_size, hist_bins, hist_range, show_all_rectangles=False))
  File "/home/rajeev/Self Driving Car/Udacity/Udacity_full_workingProject/Term1-CarsIdentification/CarND-Vehicle-
Detection/lesson_functions.py", line 653, in find_cars
    test_prediction = svc.predict(test_features)
  File "/home/rajeev/anaconda3/envs/carnd-term1/lib/python3.5/site-packages/sklearn/linear_model/base.py", line 324, in predict
    scores = self.decision_function(X)
  File "/home/rajeev/anaconda3/envs/carnd-term1/lib/python3.5/site-packages/sklearn/linear_model/base.py", line 305, in
decision_function
    % (X.shape[1], n_features))
ValueError: X has 6468 features per sample; expecting 7332
##################
```

(b) other issue was related to resize() wherein int limit became an issue
Solution : not able to find out