**PROBLEM STATEMENT**

Goal is to use the computer vision to detect vehicle on a road. We would use combination of these two datasets:

1. <http://www.gti.ssr.upm.es/data/Vehicle_database.html>
2. <http://www.cvlibs.net/datasets/kitti/>

The decisions have to be made in real time so we would make a software pipeline that is able to detect vehicles in a video from a front-facing camera on a car

The vehicle should be detected as soon as it enters the frame irrespective of the position in the frame

The output should be like this:



**SOLUTION IMPLEMENTED**

**Steps:**

• Perform a Histogram of Oriented Gradients (HOG) feature extraction on a labelled training set of images and train a classifier Linear SVM classifier

• I also apply a color transform and append binned color features, as well as histograms of color, to our HOG feature vector.

• Normalize our features

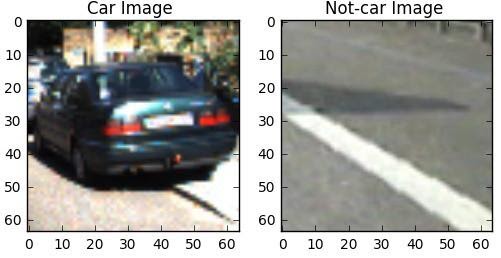
• Implement a sliding-window technique and use your trained classifier to search for vehicles in images.

• Run our 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.

**Reading images from dataset:**

Read images using glob and stored them in 2 separate arrays for vehicle/car and non-vehicle/non-car images. The images have 64 x 64 pixels



The dataset uses vehicle images count = 8792 and non-vehicles = 8968

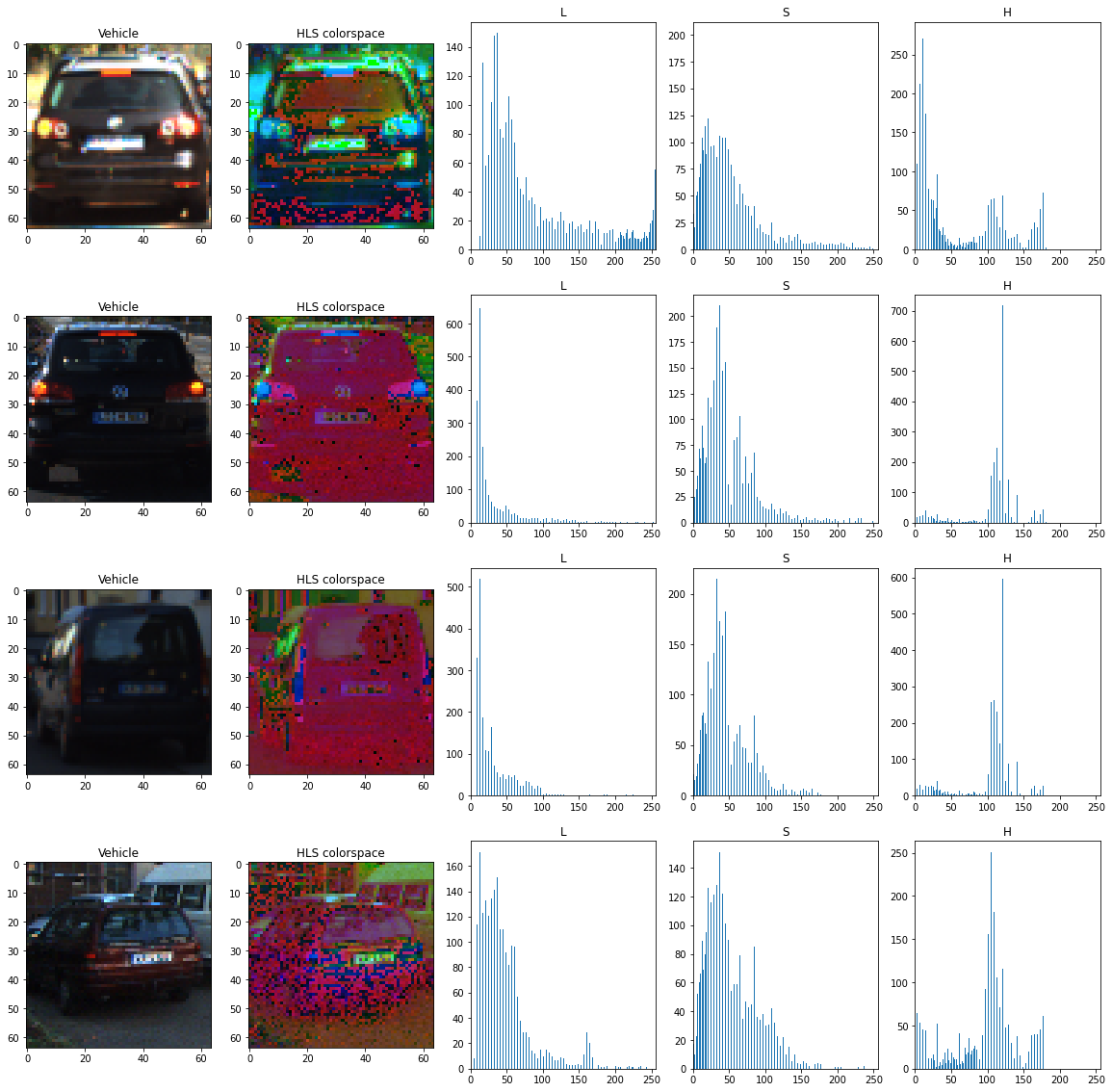
For additional features and to improve efficiency above the HOG features we also used

**Histogram of color features :**

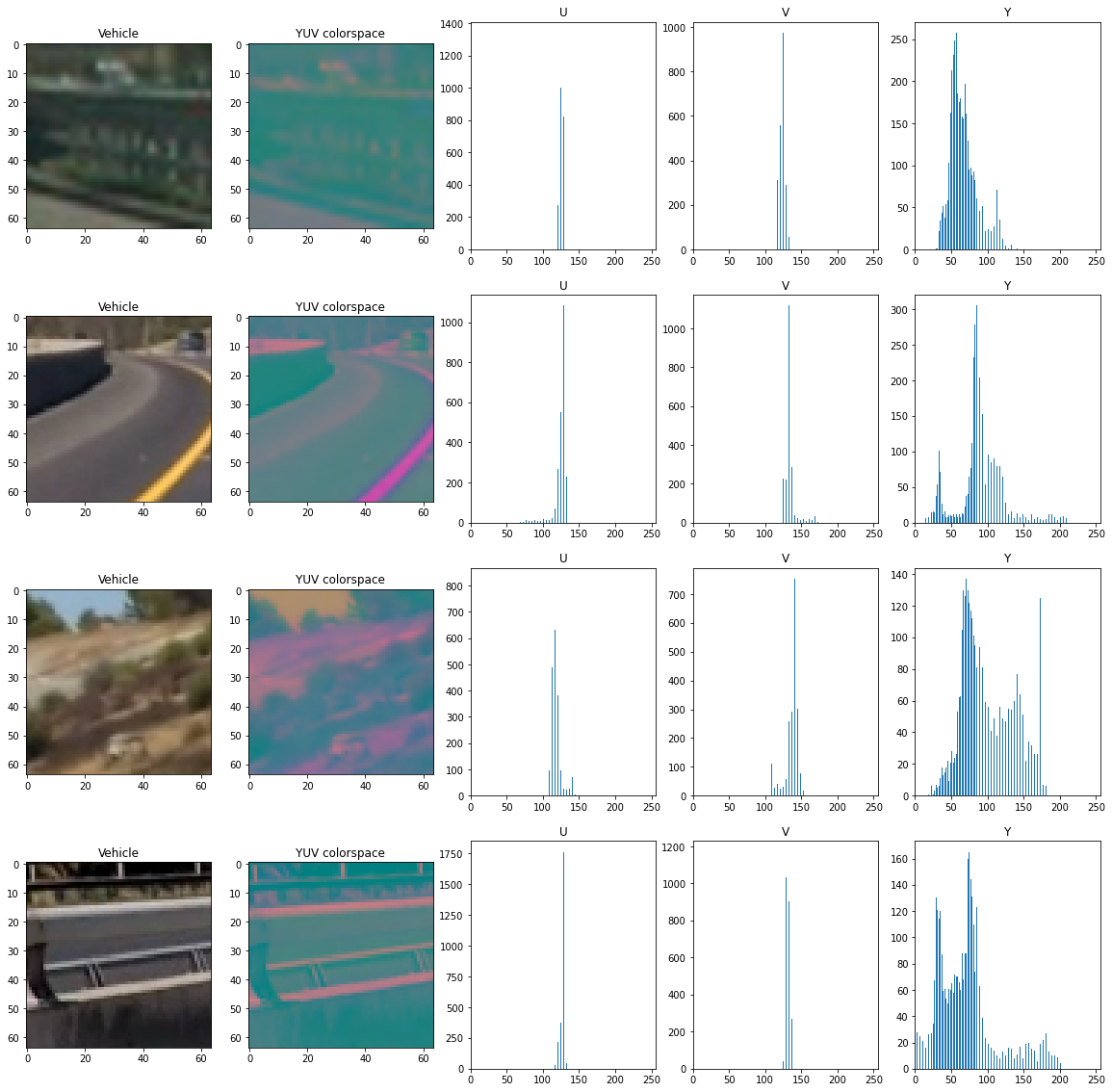
Here I used 3 variable arrays to read the features corresponding to the RGB channels of the images and prepared a feature vector named hist\_features, which was concatenation of the features obtained from 3 channels.

Visualizing random histogram features from dataset:

Vehicle Images:



Non-vehicle images:



**Spatial featues :**

Depending upon the type of the features that is the colorspace used we would create a feature vector and return it using cv2.cvt() function.

**USING HOG features :**

The HOG extractor is the heart of the method described here. It is a way to extract **meaningful features** of a image. It captures the “general aspect” of cars, not the “specific details” of it. It is the same as we, humans, do: in a first glance, we locate the car, not the make, the plate, the wheel, or other small detail.

HOG stands for “Histogram of Oriented Gradients”. Basically, it divides an image in several pieces. For each piece, it calculates the gradient of variation in a given number of orientations.

Choice of HOG parameters:

These choices would decide the efficiency. I tried various combinations of parameters and measured the accuracy of each combination. Finally below combination gives me the best result with accuracy around 95.1%

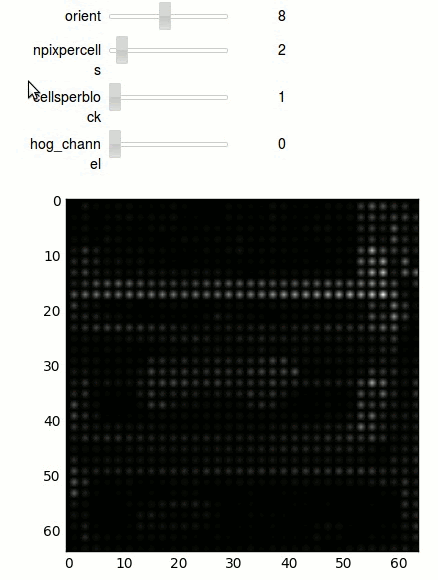
**color\_space** = 'YUV'  # Can be RGB, HSV, LUV, HLS, YUV, YCrCb

**orient** = 9       # HOG orientations

**pix\_per\_cell** = 8   # HOG pixels per cell

**cell\_per\_block** = 2  # HOG cells per block

**hog\_channel** = 0         # Can be 0, 1, 2, or "ALL



*shows the effect of the number of pixels per cell.*

**spatial\_size** = (16, 16) # Spatial binning dimensions

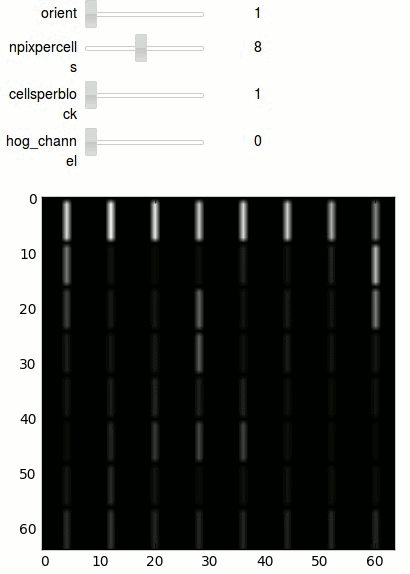
**hist\_bins** = 16    # Number of histogram bins

**spatial\_feat** = True # Spatial features on or off

**hist\_feat** = True # Histogram features on or off

**hog\_feat** = True # HOG features on or off

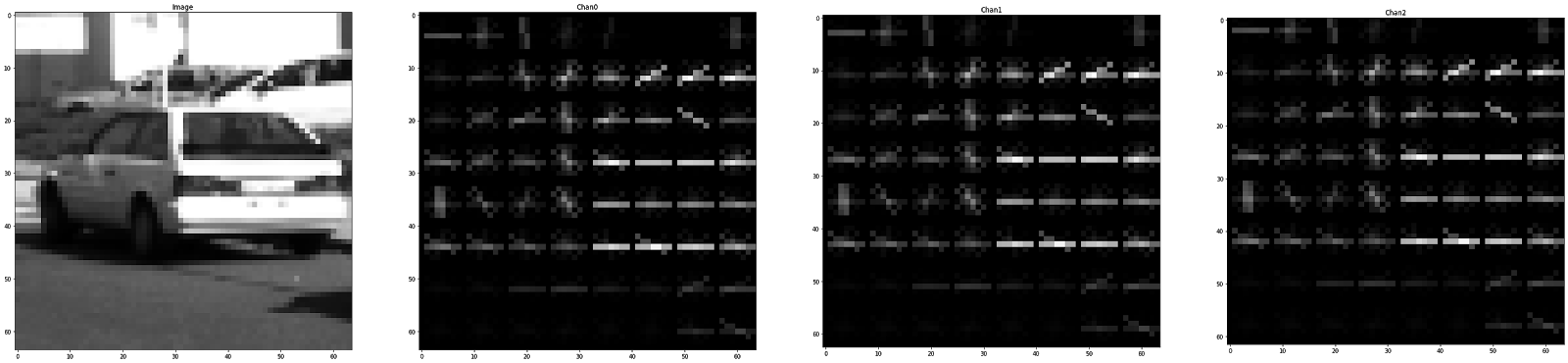
**y\_start\_stop** = [400, 650] # Min and max in y to search in slide\_window()

 *shows the effect of the number of pixels per cell.*

The final step of the algorithm is to take a histogram of directions and orientations, make a block regularization and return an single dimension array of data to be fed in a classifier

The less the number of pixels per cells (and other parameters), more general the data, and the more, more specific. By playing with the parameters, I found that orientations above 7, and 8 pixels per cell are enough to identify a car.

The HOG algorithm is robust for small variations and different angles. But, on the other way, it can detect also some image that has the same general aspect of the car, but it not a car at all — the so called “False positives”.



*Example of HOG features identified in a image using HOG*

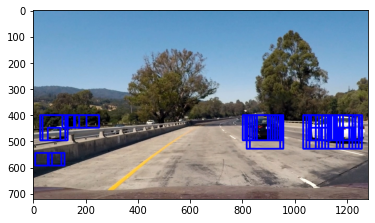
**Training the SVM**

I then trained a linear SVM using hog features and spatial features. Used 80% example for training and 20% for testing. Normalizing ensures that a classifier's behavior isn't dominated by just a subset of the features, and that the training process is as efficient as possible. That is why, feature list was normalized by the StandardScaler() method from sklearn.

**Classifier in the image frame**

 I made a function that would take positions of x and y as  input  that would be window sizes.I would then compute the region of the window to be stretched according to the given postion parameters.This function would return the list of co ordinates of window that would be used for searching .I also made a function called search window ,which would test in a givenwindow that whether the image inside the window is a car or non car. If found car we would append the window to a list .This list would help us calculate the number of vehicles in a window

I used windows of  3 pixel sizes (96,96) , (48,48) ,(128,128). Called slide\_window function on each one of them .From each one I would get a list of possible window positions and using this I  would perform search on windows



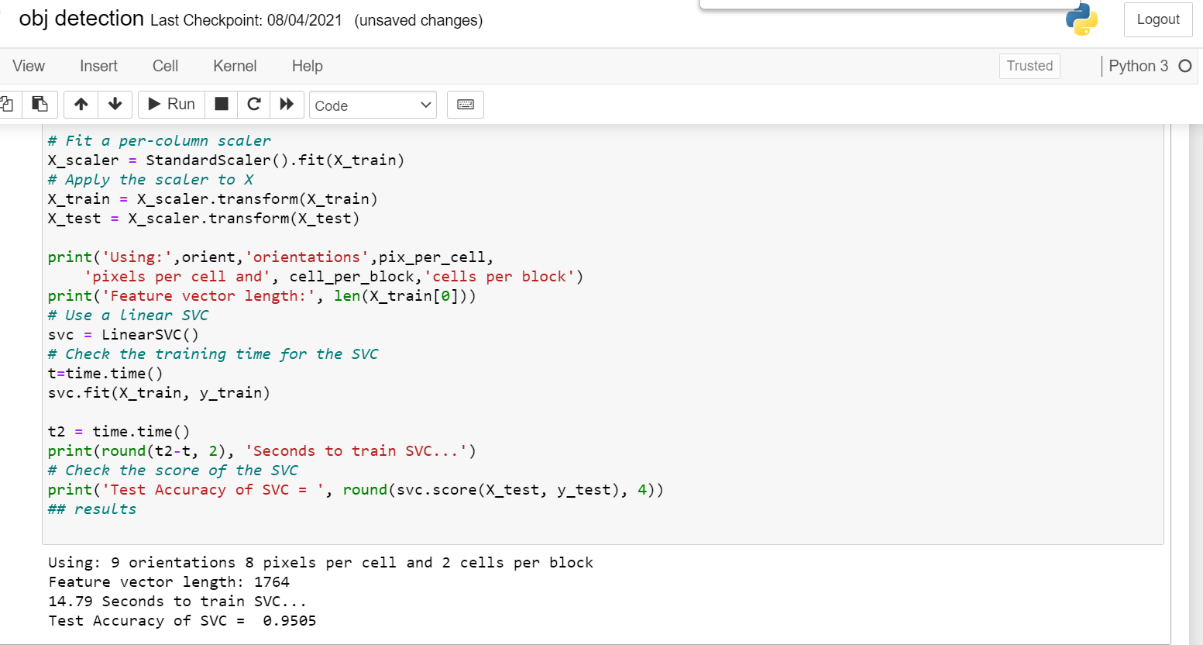
*Searching for proper windows*

I used a function to draw heatmap and in such a way  that all the windows that are below a certain level of threshold would be removed and darkened thus highlighting the car part

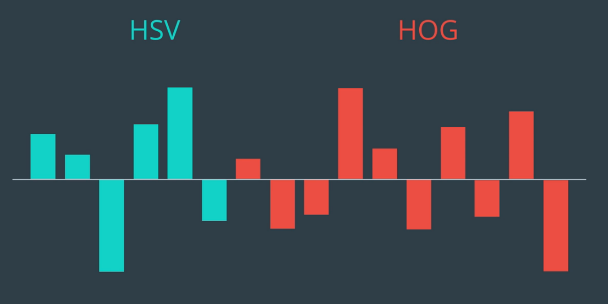


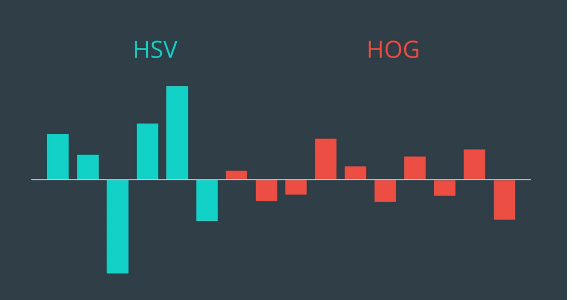
**RESULTS**

Just using the SVM classifier along with HOG features we were able to get 1764 features and the efficiency obtained was 95.05 %

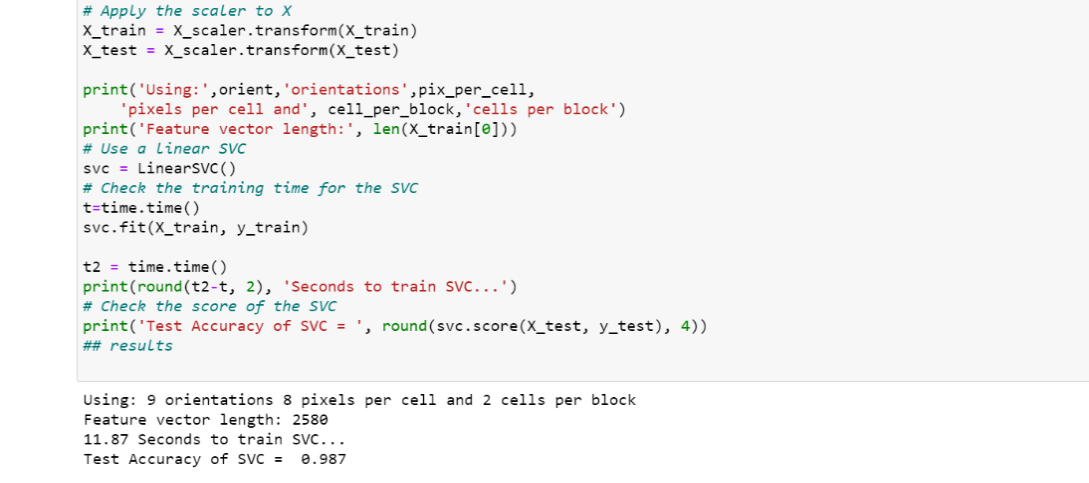


To improve this I further increased the features .Features were collected using the color based and spatial information of the images These can be concatenated along with the given HOG features and then normalized.

   *concatenating features*

  feature vector obtained after normalization

The new feature vector was of 2580 size and efficiency obtained by integrating these features was around 98.7 percent



**LEARNINGS**

I read a number of research papers which gave me idea about the complexity of the various problems to be solved in self driving cars and also the advancements that are being done in the field.

During the course of this project I also learnt about how to preprocess and deal with the image data .How to extend the simple classification problem in a way so that we would be able perform the same on a image where position of the object and orientation can be not known prior to the classification and doing so in real time.I learnt how I can make the classification pipeline for a video stream .

I also learnt about various methods to extract features from the images like HOG,spatial , and color based features, SVM classifier in much more details .

**Future work**

We can use deep network to perform the detection,  replacing the HOG+SVM pipeline. For this task employed the recently proposed SSD deep network for detection. This can have some huge advantages:

* the network performs detection and classification in a single pass
* there is no more need to tune and validate hundreds of parameters related to the phase of feature extractions the network outputs a confidence level along with the coordinates
* of the bounding box, so we can decide the tradeoff precision and recall just by tuning the confidence level we want (*less false positive*)

And in the same domain we can work on problems like lane detections and also combine the two projects that is detecting the lanes as well as vehicle simultaneously.