Vehicle Detection and Tracking

The objective of this project is to create a pipeline (model) to draw bounding boxes around cars in a video. This video was taken from a camera mounted on the front of a car.

Project Outline

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, you 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 (create a heat map of recurring detections frame by frame to reject outliers and follow detected vehicles and Estimate a bounding box for vehicles detected)
* Find False Detection.

Sliding window search Functions

We tried different sliding window size with different overlap. If we make the window size small and overlap less we miss some cars undetected. At the end we found window size 128X128 with 90% overlap gave us better result than my previous submission (64X64 with 50% overlap). Please also refer to the code file for more detail on this.

HOG Parameters

The HOG visualization is not actually the feature vector, but rather, a representation that shows the dominant gradient direction within each cell with brightness corresponding to the strength of gradients in that cell.

The HOG features for all cells in each block are computed at each block position and the block steps across and down through the image cell by cell. So, the actual number of features in your final feature vector will be the total number of block positions multiplied by the number of cells per block. We tried different block size and these are the optimized size we get.

We tried different color_space. With RGB we have more False Detection. Same goes with HSb. We tried different color and seems like YCrCb is giving us the best detection with less false detection.

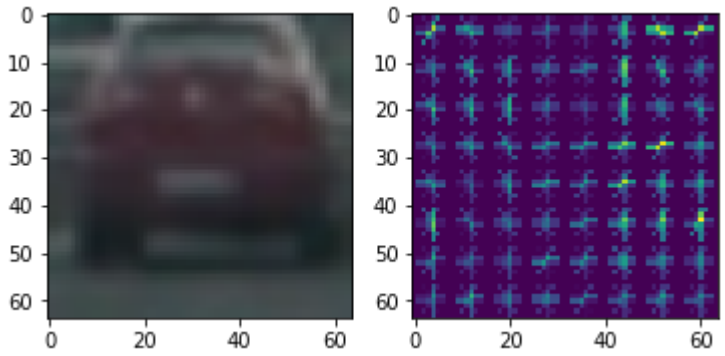
Heat Map

```
Heat map of recurring detections frame by frame to reject outliers and
follow detected vehicles and Estimate a bounding box for vehicles
detected). We also added small binary bird-eye projection on top left
screen for “Heat Map”
```

Test Image Features Functions

In [10]:
```
test_img = mpimg.imread(cars[42])
test_feature, test_hog_image  = single_img_features(test_img, color_space=col
                        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, vis = True)


plt.subplot(121)
plt.imshow(test_img)
plt.subplot(122)
plt.imshow(test_hog_image)
print("cars 42")
```

```
cars 42
```



HOG visualisation is a repr that shows the dominant gradient direction within each cell with brightness corresponding to the strength of gradients in that cell.

In [13]:
```
from sklearn.utils import shuffle
scaled_X, y = shuffle(scaled_X, y, random_state=42) # siraj 10-9-2017
# Split up data into randomized training and test sets
X_train, X_test, y_train, y_test = train_test_split(scaled_X, y, test_size=0.

print('Using:',orient,'orientations',pix_per_cell,'pixels per cell and', cell
print('Feature vector length:', len(X_train[0]))
```

```
Using: 5 orientations 8 pixels per cell and 3 cells per block
Feature vector length: 5718
```

Task 6 : Test video pipeline on a video stream

In [16]:

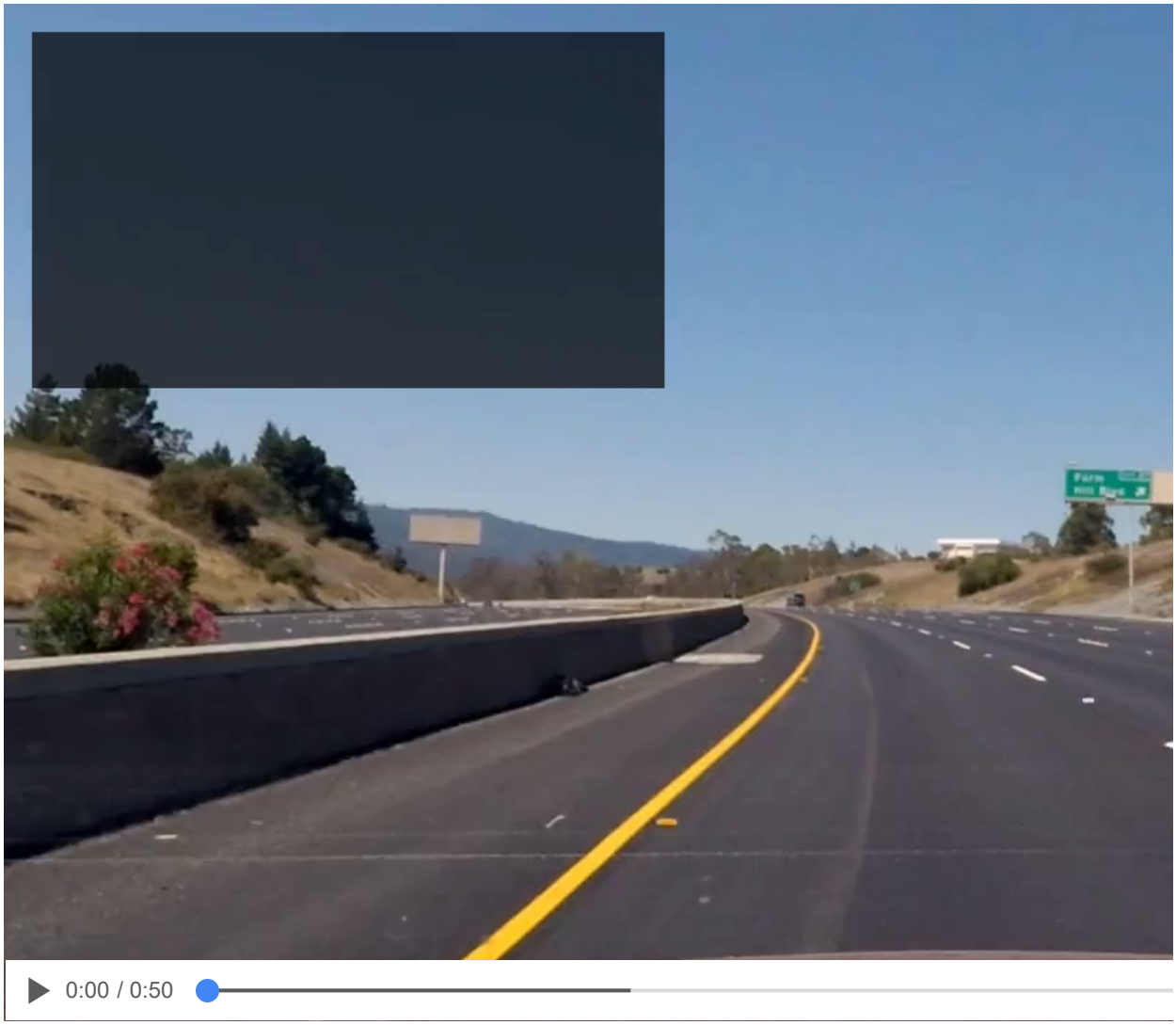
```
bboxes_list = []

video = VideoFileClip("project_video.mp4")#.subclip(5,15)
project_clip = video.fl_image(process_image) #NOTE: this function expects col
video_output = "output_v5.mp4"
%time project_clip.write_videofile(video_output, audio=False)

print("Bounding boxes: ", len(bboxes_list))
```

```
[MoviePy] >>>> Building video output_v5.mp4
[MoviePy] Writing video output_v5.mp4

100%|████████████████████████████████████████████████████████████████████
████| 1260/1261 [19:35<00:00,  1.08it/s]

[MoviePy] Done.
[MoviePy] >>>> Video ready: output_v5.mp4

Wall time: 19min 36s
Bounding boxes:  1261
```

In [17]:

```
HTML("""
<video width="960" height="540" controls>
  <source src="{0}">
</video>
""".format(video_output))
```

Out[17]:



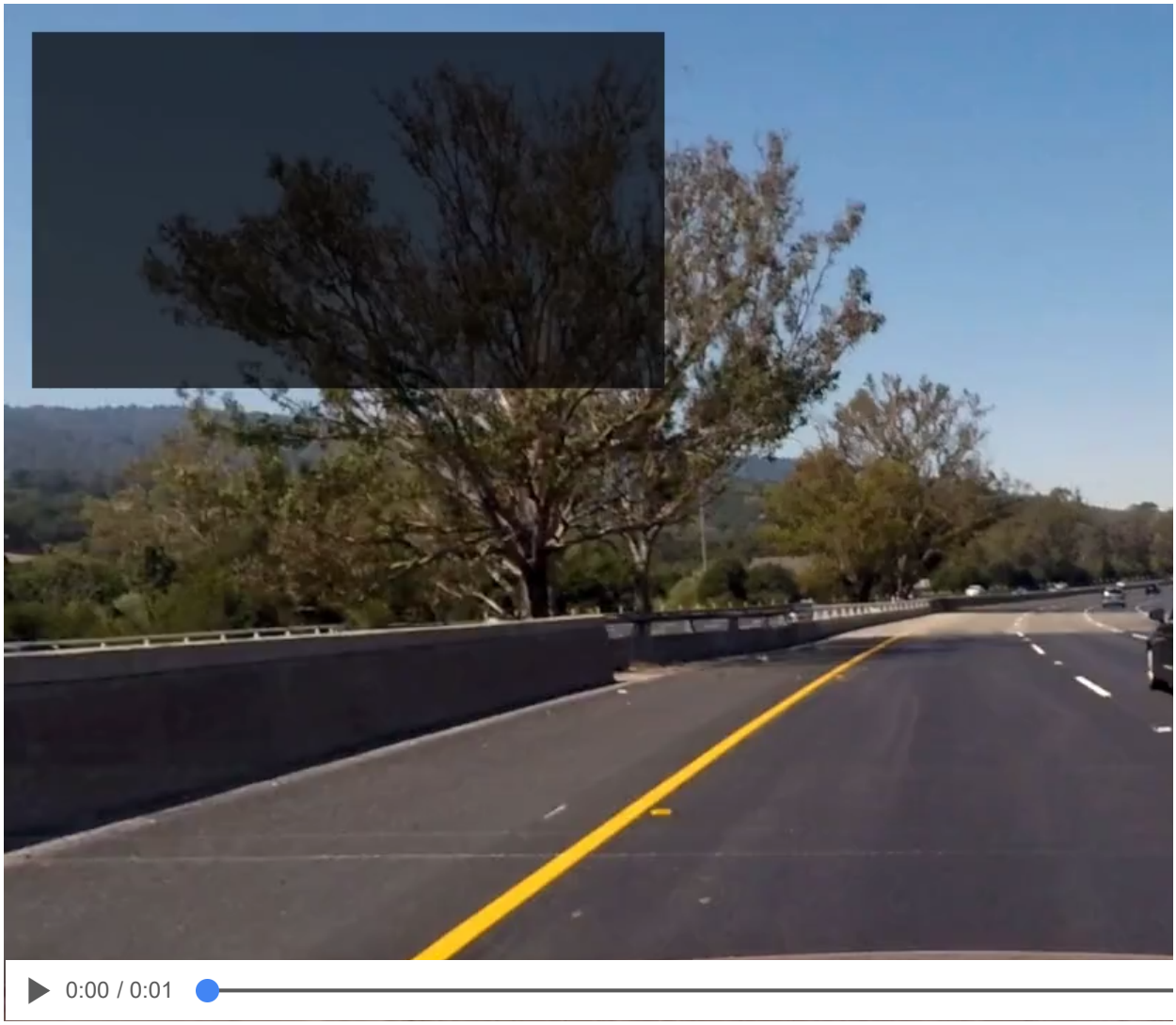
In [18]:

```
bboxes_list = []

video = VideoFileClip("test_video.mp4")
project_clip = video.fl_image(process_image) #NOTE: this function expects col
video_output = "output_2_v5.mp4"
%time project_clip.write_videofile(video_output, audio=False)

print("Bounding boxes: ", len(bboxes_list))
```

```
[MoviePy] >>>> Building video output_2_v5.mp4
[MoviePy] Writing video output_2_v5.mp4
 97%|████████████████████████████████████████████████████████████████████
██████  | 38/39 [00:35<00:00,  1.05it/s]

[MoviePy] Done.
[MoviePy] >>>> Video ready: output_2_v5.mp4

Wall time: 36.5 s
Bounding boxes:  39
```

In [19]:
```
HTML("""
<video width="960" height="540" controls>
  <source src="{0}">
</video>
""".format(video_output))
```

Out[19]:



Discussion

1. The pipeline still misses out cars often when there are shadows.
2. Adding more features (even if they sound like they might help) can make a model do worse.
3. Heat map helped of recurring detections frame by frame to reject outliers and follow detected vehicles and Estimate a bounding box for vehicles detected.

Acknowledgement

1. Special thanks to my friend Siraj.
2. Special thanks to Nicolas at https://github.com/NikolasEnt/Vehicle-Detection-and-Tracking

```
1 # Future Plan
```

```
1 I have done some negative mining by cropping out non-car images from the
  project video, adding them to the training set and finally retraining the
  classifier. I would like to continue doing it to make it more robust.
```