P5: Vehicle Detection & Tracking

The goals of this project were the following (as taken from UDACITY template writeup)

- Perform a Histogram of Oriented Gradients (HOG) feature extraction on a labeled training set of images and train a classifier Linear SVM classifier
- Apply a color transform and append binned color features, as well as histograms of color, to HOG feature vector.
- Implement a sliding-window technique and use your trained classifier to search for vehicles in images.
- 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.
- Estimate a bounding box for vehicles detected.

Feature Generation

For detecting and tracking vehicle, a combination of feature generation methods was applied. For classification based on color, a combination of spatial binning of colors and color histogram proved effective. In case of spatial binning of color, spatial binning dimension was a hyper parameter, which was tuned to get best results. Although a dimension of (32, 32) was making predictions more accurately, it significantly increased the feature vector length and the time consumed to generate video was very high. Hence the dimension was fixed at (16, 16) to reduce the computation time. The value of histogram bins was chosen as 32. Histogram bins didn't add much to the feature length. However, no significant improvement was observed by increasing the histogram bins beyond 32; hence it was fixed at that value.

For gradient based classification, HOG (Histogram of Oriented Gradients) was used. This was the trickiest of the lot as there were a number of hyper parameter to play around with. For choosing the parameters, keeping the feature length reasonable was kept a priority. With this in mind, using 8 orientations, 8 pixels per cell & 2 cells per block proved to be the fastest while giving reasonable results. Colors channel HSV with S channel and YCrCb with all channels gave highest test accuracy with good speed while video processing. Features were scaled to zero mean and unit variance before training the classifier.

Example of using the YCrCb color space with mentioned parameters are presented below:

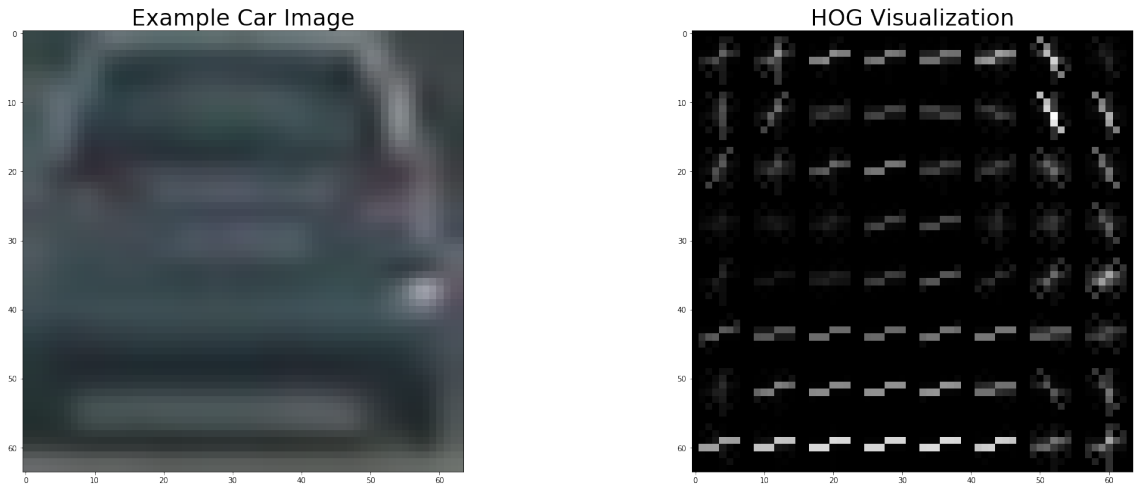

fig 1: comparision of original car image v/s its HOG visualization

Training

For training the model, a set of training and test data was prepared. As the image data was extracted from a video, many of the images were similar to each other. Hence, randomly shuffling the data was not enough to enough dissimilarities in the train and test sets. To mitigate this, test set was taken randomly only from the 'KITTI extracted' folder, which had a random set of images rather than taken from a video. For the non vehicle category, randomly choosing from the entre data set was good enough as the images were pretty random on it's own. Finally, the entire data was split up in the ratio 82.5:17.5 between train and test sets to get around 7200 for the training set and 1500 for the test set of each category. Since, the data was balanced between the two classes, there was no need to generate additional data to any particular class.

For this problem, I started off with a linear SVM classifier as it was the simplest option to begin with. The classifier took under 10 s to train the model, and produced around 99% accuracy each time. The size of the data set used for the training for fairly small and hence the high accuracy. The model was then tested on the actual snippets from the video to further tune the model.

Sliding Window Search (including Video implementation)

After training the model, it was tested on the snippets taken from the video. For this, sliding window technique was employed. It was very clear looking at the video that one window size would not be sufficient to search for cars of all sizes. Hence, window sizes of (64, 64), (96, 96), (144, 144) & (196, 196) were used. The area of search for each window size was restricted to certain areas of the picture where the probability of finding a car within that box was high. E.g. the smaller boxes were restricted to the middle region of the image, while the bigger boxes searched more towards the bottom portion of the image. This localized searching technique reduced the computation time to search cars in the image.

fig2: Figure showing the original image vs the output of sliding window search

Sliding Search technique was not perfect and resulted in many false positives. To remove them, heat maps were generated from overlapping images. These images gave a fair idea about false positives.

To further refine the images for false positives in videos the heat map from several frames were combined in the following manner:

- For every frame, sliding search technique gives out positive frame locations (called hot windows).
- Heat map is generated based on 'hot windows' and a threshold of 4.
- Bounding boxes is generated based on that heat map.
- This bounding box is appended to the 'prev' class which keeps track of previous five frames and previous 8 heat maps.
- In the next frame, the previous bounding box along with current 'hot window frames' are used to generate current heat map.

- This ensures that heat maps are generated only for those region which maintain a threshold of 4 over two consecutive frames.
- These heat maps are appended to ‘prev’ class.
- After 8 such heat maps are generated, box is drawn around pixels which were active in all 8 heat maps (found using np.bitwise_and function)

fig 3: outputs from sliding window technique and heat map thresholding

Conclusion

The final output video is can be seen at the very end of the Jupyter Notebook attached in the submission or viewed directly by opening the mp4 file ‘project_output’. The final output was good and was able to detect the cars fairly accurately. There was one positive, but non that lasted too long. There are many scope of improvement for this project:

- Training of the classifier should be improved by using a much larger dataset. Increasing the feature length too much is not a good option since it comes with a cost of higher computation time even during detection. Improving the data set will certainly help to reduce the number of false positives in the project.
- A track of the centroid should be kept for the vehicle so that the next location of the car can be predicted. If the approximate location of the car is already known, then localized searches can be made to improve the prediction speed.
- Choice of window size can be made dynamic by choosing the size generated while hot labelling the image in the previous frame.