Vehicle Detection with HOG and Linear SVM

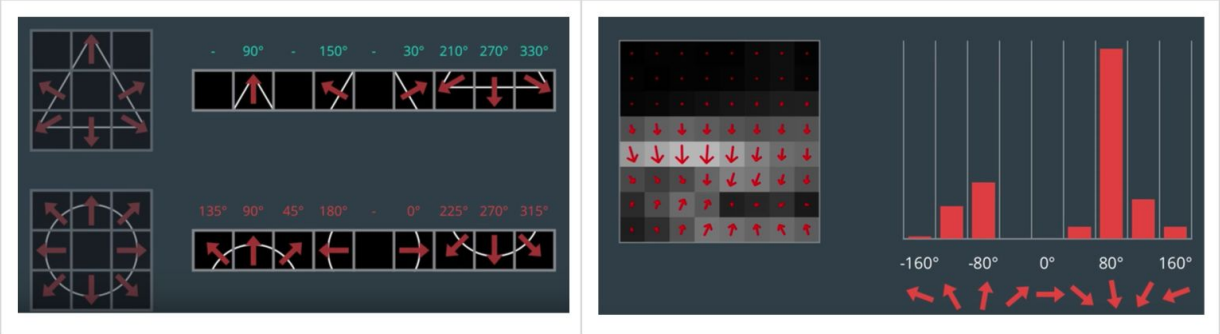
For this project, I created a vehicle detection and tracking pipeline with OpenCV, SKLearn, histogram of oriented gradients (HOG), and support vector machines (SVM). I then optimized and evaluated the model on video data from an automotive camera taken during highway driving.

To accomplish this, I have done the following:

- Performed a Histogram of Oriented Gradients (HOG) feature extraction on a labeled training set of images
- Trained a classifier with the set of vehicle and non-vehicle images
- Implemented a sliding window to get subregions of a single video frame
- Used my trained classifier to detect vehicles on subregions
- Created a heat map of recurring detection frame by frame to follow and detect vehicles

HISTOGRAM OF ORIENTED GRADIENTS

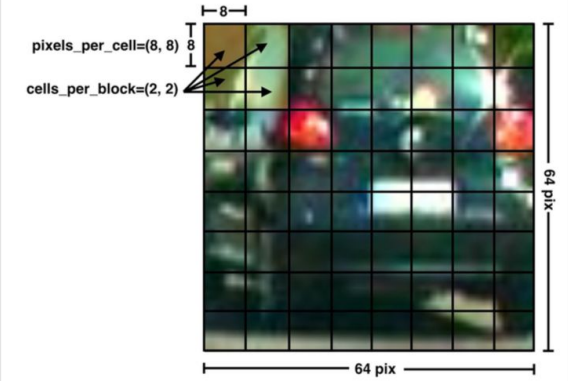
A class of objects such as a vehicle vary so much in color. Structural cues like shape give a more robust representation. Gradients of specific directions captures some notion of shape. To allow for some variability in shape, we'll use features known as Histogram of Oriented Gradients (HOG).

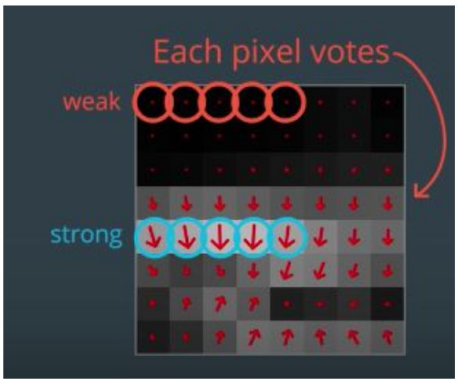


Taken from one of Udacity's lecture videos

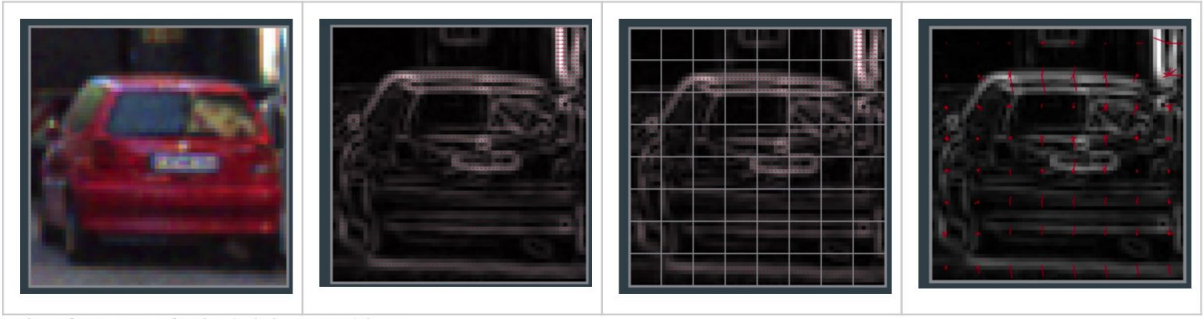
The idea of HOG is instead of using each individual gradient direction of each individual pixel of an image, we group the pixels into small cells. For each cell, we compute all the gradient directions and group them into a number of

orientation bins. We sum up the gradient magnitude in each sample. So stronger gradients contribute more weight to their bins, and effects of small random orientations due to noise is reduced. This histogram gives us a picture of the dominant orientation of that cell. Doing this for all cells gives us a representation of the structure of the image. The HOG features keep the representation of an object distinct but also allow for some variations in shape.





Taken from one of Udacity's lecture videos

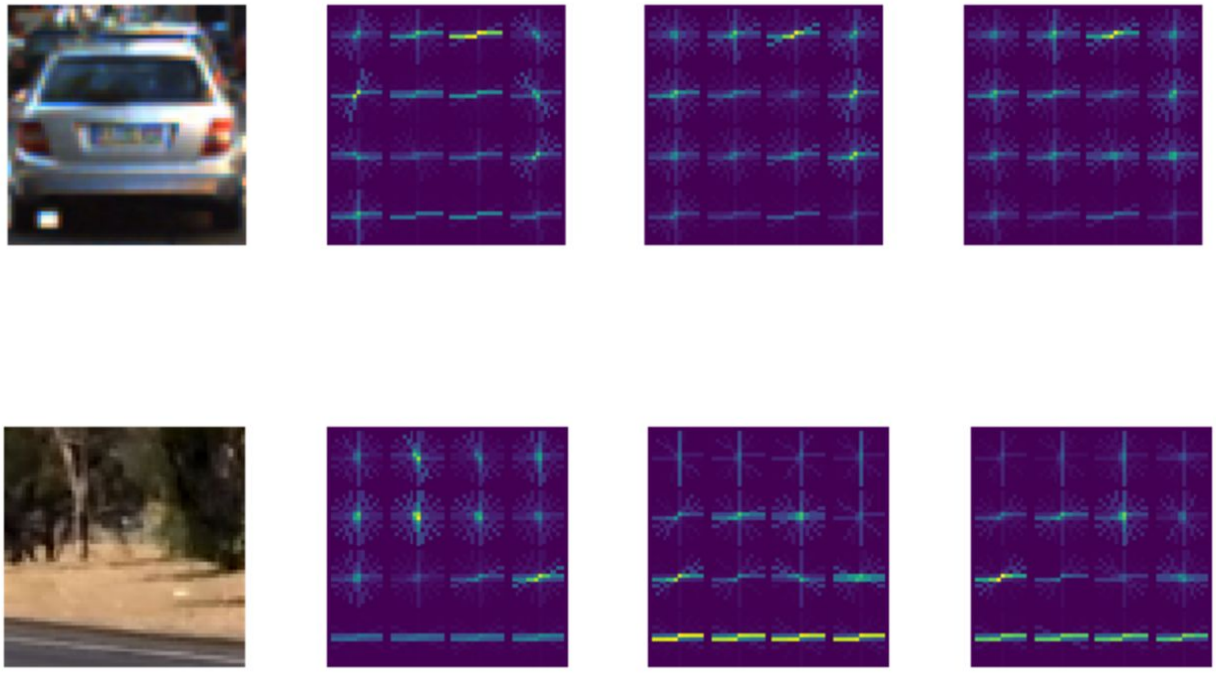


Taken from one of Udacity's lecture videos

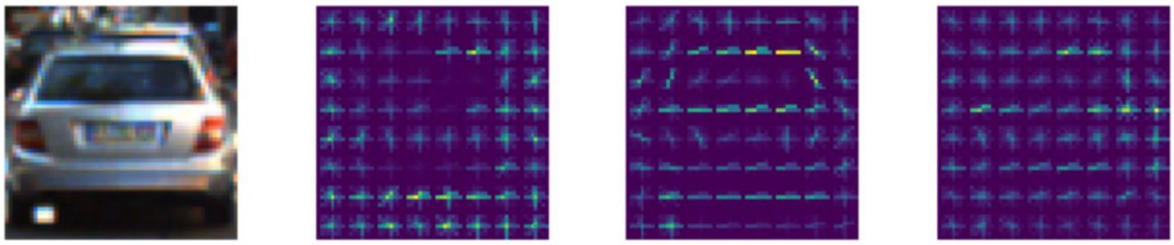
FEATURE EXTRACTION

We can specify the number of `orientations`, `pixels_per_cell` , and `cells_per_block` for computing the HOG features of a single channel of an image. The number of `orientations` is the number of orientation bins that the gradients of the pixels of each cell will be split up in the histogram. The `pixels_per_cells` is the number of pixels of each row and column per cell over each gradient the histogram is computed. The `cells_per_block` specifies the local area over which the histogram counts in a given cell will be normalized. Having this parameter is said to generally lead to a more robust feature set. We can also use the normalization scheme called `transform_sqrt` which is said to

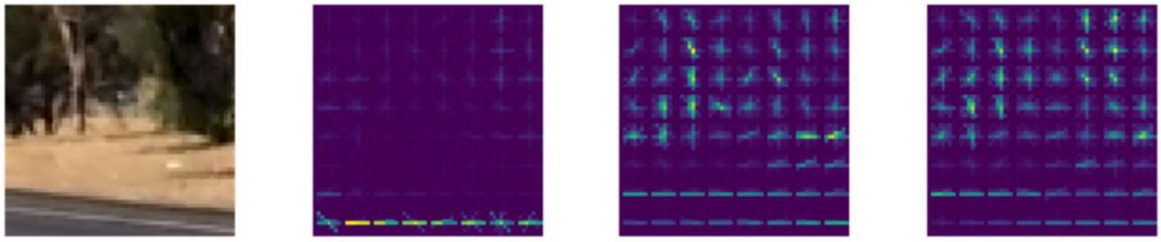
help reduce the effects of shadows and illumination variations.



Vehicle: visualization of the HOG features for Hue, Saturation, and Lightness respectively



Non-Vehicle: visualization of the HOG features for Hue, Lightness, and Saturation respectively



```
YUV Feature Extraction Time Taken: 471.28
```

```
HLS Feature Extraction Time Taken: 1781.44
```

CLASSIFIER TRAINING

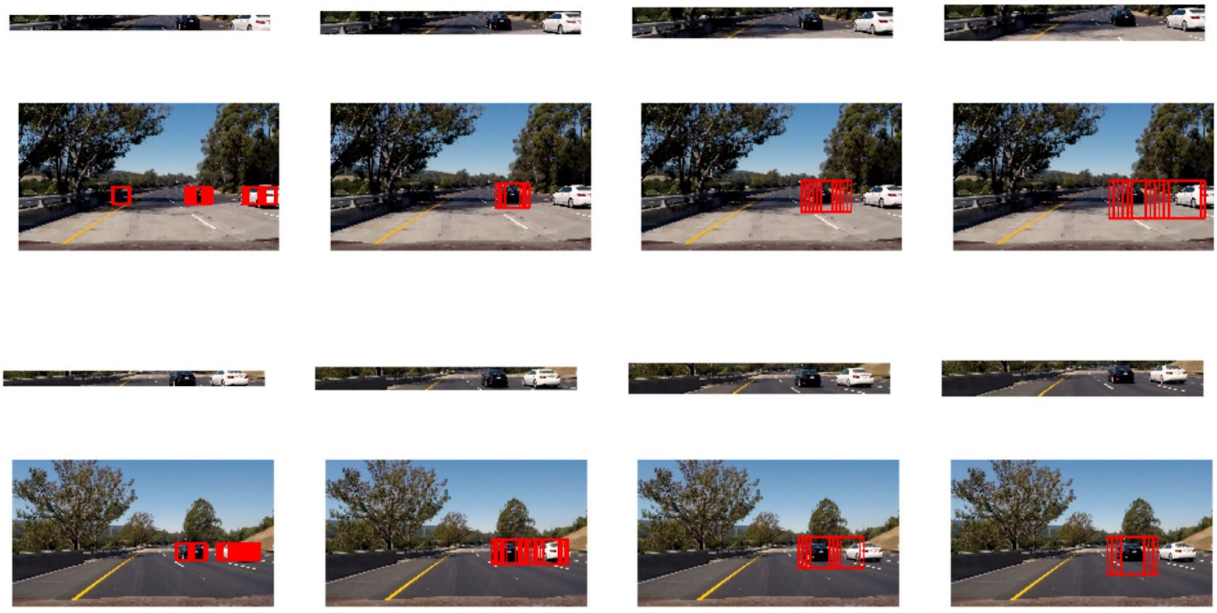
The classifier algorithm I used is called a **Linear Support Vector Machine**. I have used a total of **8,792** samples of vehicle images and **8,968** samples of non-images. This data set is preselected by Udacity with images from the GTI vehicle image database and the KITTI vision benchmark

suite. As a safety measure, we use a `scaler` to transform the raw features before feeding them to our classifier for training or predicting, reducing the chance of our classifier to behave badly.

- `HLS Features Classifier Accuracy: 0.9878`
- `YUV Features Classifier Accuracy: 0.9834`

SLIDING WINDOWS

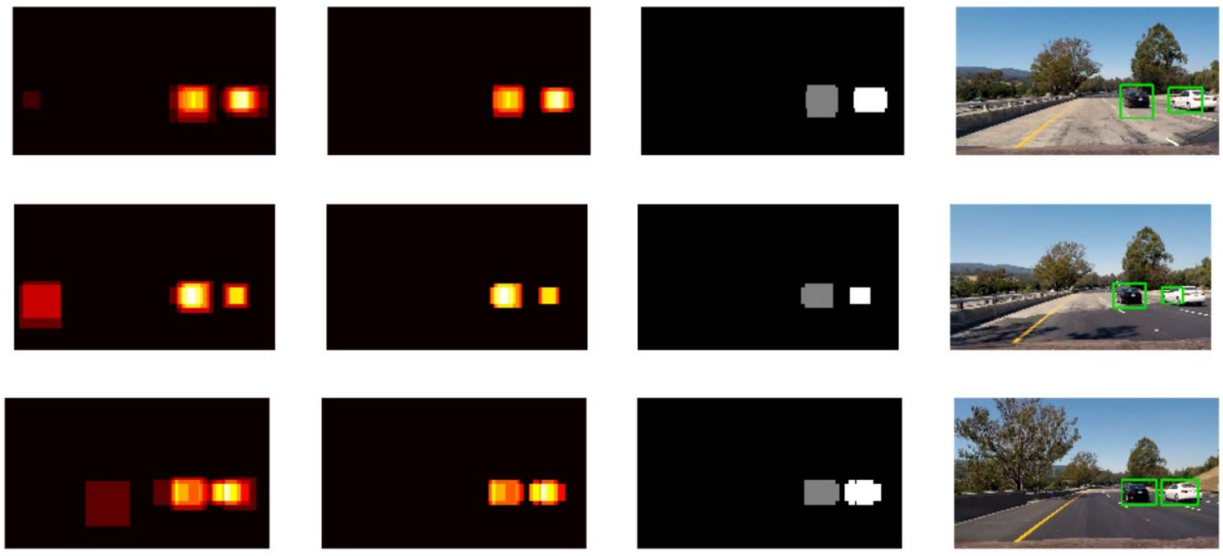
We can get a subregion of video frame and run that classifier there to see if that patch contains a vehicle. Firstly, we have to consider that getting the HOG features is extremely time consuming. Instead of getting the HOG features for each patch region with have many overlaps with each other each time, we extract HOG features of the whole frame at the beginning, then we pull out the features for each frame's subregion as we need them.



The strip where the search was performed is decided by eyeing the vehicles on the video. Notice that we should only search below the horizon as we don’t expect to see vehicles in the sky.

HEATMAPS

Given a few consecutive frames, notice that there are overlapping detections and false positive detections are spaced out. We can build a *heat map* to combine overlapping detections and remove false positives. To make a heat map we start with a blank grid and *"add heat*" for all pixels within windows where positive detections are reported by the classifier. The *"hotter"* the parts, the more likely it is a true positive, and we can impose a threshold to reject areas affected by the false positives. If we integrate a heat map over several consecutive frames of video, areas with multiple detections get *"hot"* while transient false positives stay *"cool"*.



SUMMARY

For this project, I wrote a software pipeline to detect and track vehicles from a video of a highway. To do this, I have extracted HOG features of previously collected data and fed them to a Linear Support Vector Machine classifier algorithm. I used a sliding window technique to check if subregions of a frame contain vehicles. Then I used heat maps over multiple consecutive frames to weed out transient false positives and gain confidence over multiple detection on the same location.

HOG features of images in HLS and YUV color formats are good features to be used for classifying vehicles. However, extracting 1,188 YUV HOG features is extremely faster than extracting 7,056 HLS HOG features so better use YUV (with 16 x 16 pixels per cell and 11 orientations) over HLS (with 8 x 8 pixels per cell and 12 orientations).