Vehicle Detection (Make Model Recognition)

Qiangyu Zheng (1002144128)   *Yichong Guan (1002730529)*

(Github link : https://github.com/EASONGUAN/DetectAndRecognize.git)

Abstract

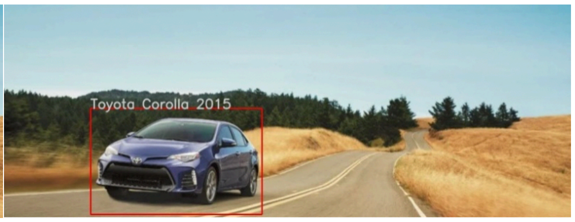
In our project, we detected vehicle and recognized the Make, Model, and Year of the car from an input picture (e.g. BMW M3 coupe 2012). We simply separated the whole project into two steps, the first is finding the position of the vehicle, and the second is detect the model of the vehicle. We combined knowledge from machine learning and visual computing to achieve the final result.

Introduction

The input of our algorithm is an image with vehicle, and the algorithm could detect the position of the vehicle in the image and classify the make, model, and year of the detected car. The output of our algorithm is the original image with the rectangle to cycle the vehicle and the corresponding label to show the information of the vehicle.

Input:

Output:



My responsibility of the project is to the implementation of CarFinder class, which makes a list of window positions, then detect each of window is vehicle or not by using the trained Support Vector Machine [1], and finally detect the position of the vehicle in the entire image. I also implemented an additional experimented method to classifier the model of the vehicle, using SIFT feature matching and Homography transform.

Related Work

The Histogram of Oriented Gradient features (HOG) descriptor [4] has been widely used in many projects. And our experimented method, SIFT feature matching, also uses the HOG descriptor for detection. YOLO [3] is a state-of-the-art, and one of the most famous real-time object detection system, whose criteria is also the Convolutional Neural

Network (CNN) [5], which we used to detect the model of the vehicle. However, we built up a different architecture of neural network, which helps us to fit our own classification model especially for vehicle detection.

Datasets

The KITTI [6] dataset: contains vehicle images and non-vehicle street view images to train SVM for vehicle position detection purpose.

The VMMR [7] and Stanford Vehicle Dataset [8]: vehicle images and information about each car to train CNN for model classification purpose.

KITTI

Type	Train	Valid	Test	Total
Non-Vehicle	7,000	500	100	7,600
Vehicle	7,000	500	100	7,600

VMMR

Number of Classes	Train	Valid	Test	Total
20	3,600	720	200	4,520

Stanford Vehicle Dataset

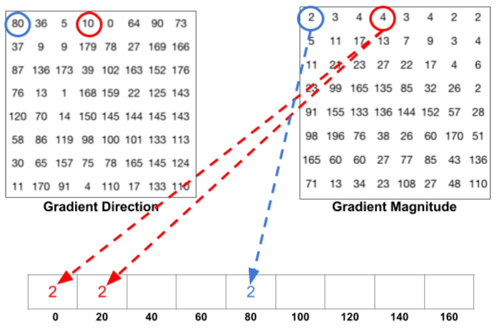
Number of Classes	Train	Valid	Test	Total
10	780	180	100	1,050

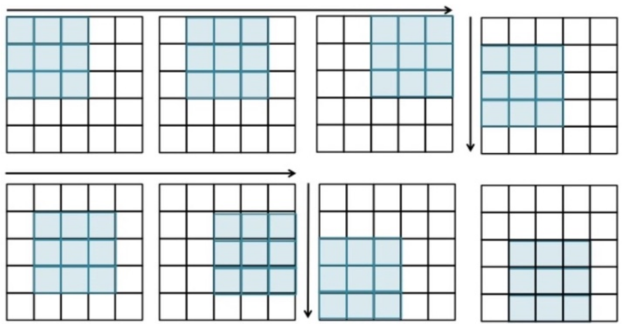
Methods & Results

1. Histogram of Gradient Orientation (HOG):

The first step of the program is to find the coordinate of the vehicle in the entire image. To achieve this goal, we need to find all the windows which are detected as containing vehicle. At the very beginning, we need to calculate the HOG feature descriptor for each image in the training dataset, for the following use in training the SVM.

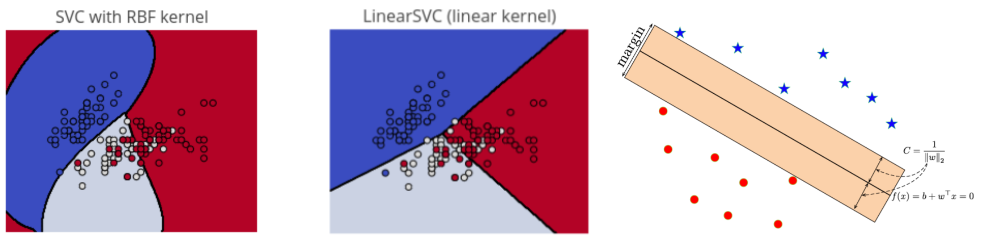
In order to find a more common way to treat all training images, we firstly resize the all input images to the size of 64x64. For the resized image, we separated it in to 64 square-blocks, each has 8x8 pixels. For each pixel, we should consider the gradient magnetite and orientation. And we have total 12 different orientation bins, each represents 15 degrees from 0 to 180. After the normalization, which is to omit the invariant of different lighting conditions, we finally get a 1x2352 vector for each color. Therefore, for the RBG colored image, we get a 1x7056 vector for the entire image.





2. Support Vector Machine (SVM):

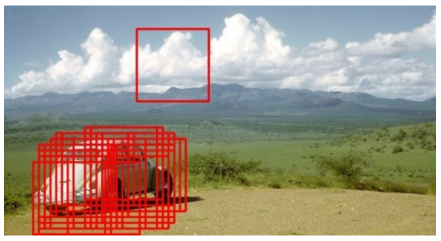
We train the SVM with the HOG features, we calculated before. From the KITTI dataset, we have 7000 vehicle images and 7000 non-vehicle images as the training dataset. We tried two different functions linear and gaussian as the kernel distance functions of the SVM to compare the different performances.



In general, SVM is to find the best separating plane of multiple classifications, which maximize the margin of the classifier. In our project, the SVM is to solve the binary classification, classifications of vehicle or non-vehicle.

After training the training dataset, consider the validation, we randomly choose about the extract number, about 30% training images, of testing images. The test accuracy of the SVM using linear kernel is 96%, and the test accuracy of the SVM using gaussian kernel is 98%. However, we have more than 7000 images for training, our dataset is still not enough, there are still some outliers of some testing images. Especially, outliers mainly reflected in the lack of the ‘negative’ (non-vehicle) data.

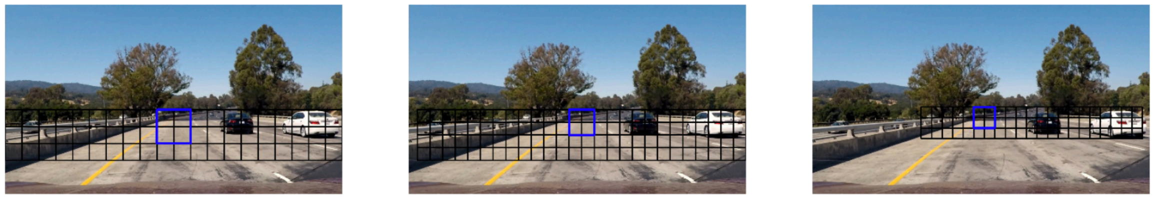
Outlier



3. Sliding Window for Vehicle Detection

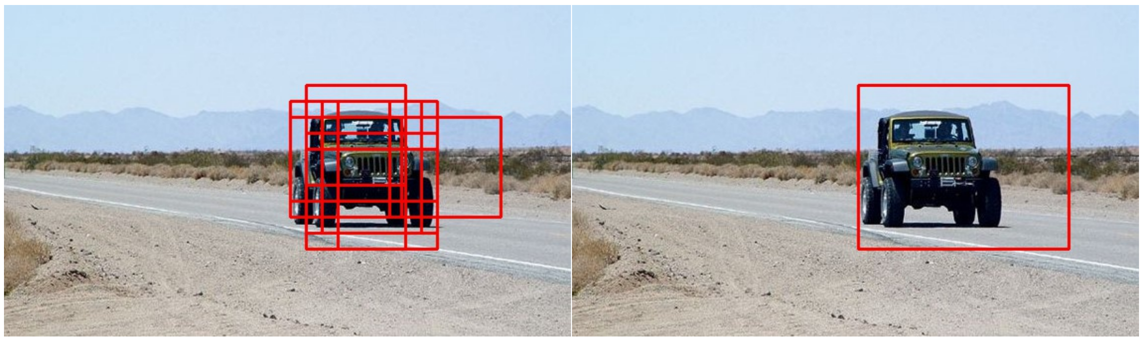
After we get the trained SVM classifier, our program would loop over the entire images, sliding around and making a list of window positions with corresponding size of the image size. For each window, we need to resize the windows into 64x64 size and calculate the HOG features as the same as before. Then with the HOG features and the SVM classifier, to detect the window contains vehicle or not. Continually storing the coordinates positive windows, those windows contain vehicle. Finally, grouping the valid windows, in other words, sort the interconnected windows into a group, find the whole window of the entire car.

List of window positions

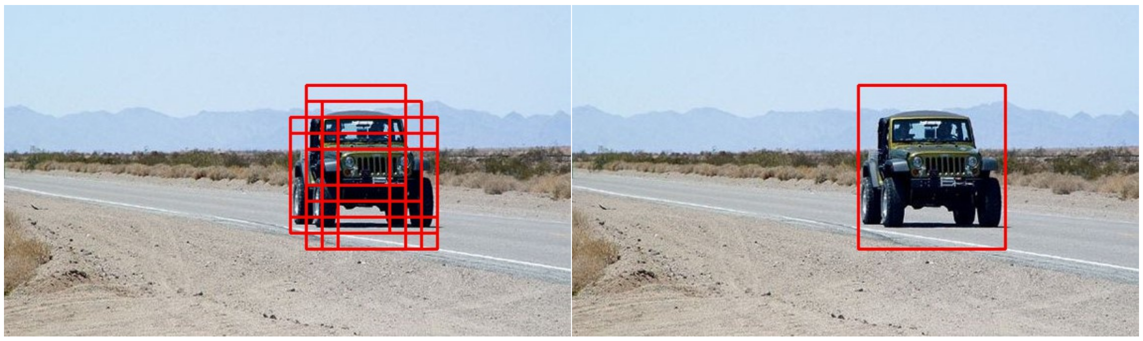


Valid windows:

Detected windows using Linear Kernel:



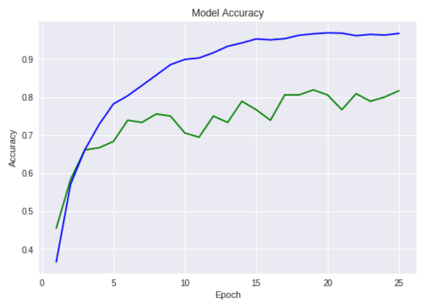
Detected windows using Radial Kernel:



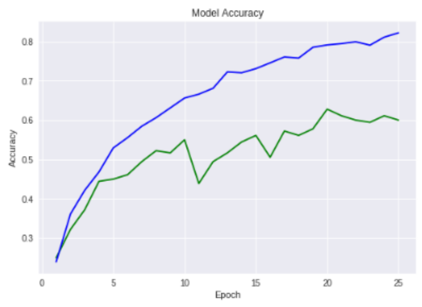
4. Convolutional Neural Network (CNN)

My partner implemented the Convolutional Neural Network, which is also the final step of our algorithm. In order to classify the model and year of the detected vehicle, we need a specific neural network. There are several layers in our CNN, including Convolutional Layers, Pooling Layers, Rectified Linear Unit Layers and Fully Connected Layers.

For training the CNN, my partner tried several ways, the first one is to train from scratch, and to use the architecture of VGGNet. The second way is to fine-tune a pre-trained CNN and combined ours to try to get a better result using a smaller number of images. We used the VMMR and the Stanford dataset for training the CNN, which contains totally more than 120,000 images, and we finally decided to choose the vehicle models in the dataset with sufficient number of images to train the CNN and only consider these types of vehicles to reduce the training time.

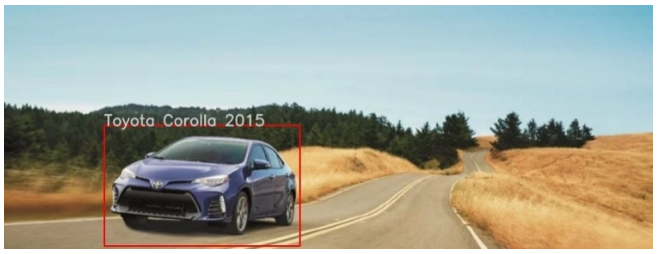


Validation and Test Accuracy of fine-tuned VGG16



Validation and Test Accuracy of train-from-scratch

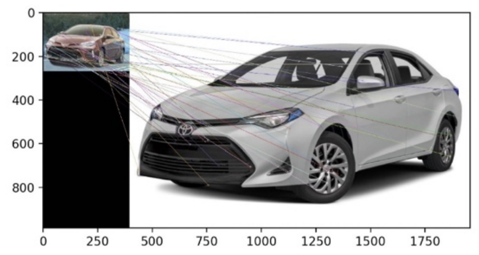
Final result



5. Experimented Classification Method: SIFT Feature Matching and Homography

I implemented SIFT feature matching to try a different vehicle classification method, which we learned from the this course. Firstly, computing all the SIFT key points for the image and standard car model images. Then, through comparing to the 'ratio' to find all the matching point.

After we get all the matching points, using at least 4 pairs of matching points to calculate the homography transformation matrix. During this computation, we perform RANCSAC algorithm to find the number of real matching points, which are inliers. Count the number of inliers as the mark for this type of vehicle, which gets the maximum mark is the final model of the input image.



Challenges & Conclusion:

During the project, there are two main challenges of my implementation, the first one is the implementation of 'CarFinder' class, how to find a general step and window size for all the image is hard. Finally, I decided that the windows could be related to the size of the entire image. The second is to how to find the interconnected fragmentation as the whole detected vehicle. In order to solve this, I wrote a helper function in 'util.py' to find the combined rectangle with all coordinates of the valid windows.

By this project, I have learnt how to combine the knowledge from computer vision and the knowledge from machine learning. I also learnt how to use the knowledge from the other course (CSC411) and apply the CNN into our project. A good project is to cover many aspects of the computer science. And I also learnt how to cooperate with partner, how to divide the whole work in to two parts, and how to read the code of your partner. This whole project is huge, therefore, we cannot finish it by just one of us.

References

[1] "1.4. Support Vector Machines — Scikit-Learn 0.20.1 Documentation". Scikit-Learn.Org, 2018, https://scikit-learn.org/stable/modules/svm.html. Accessed 2 Dec 2018.

[2] Redmon, Joseph. "YOLO: Real-Time Object Detection". Pjreddie.Com, 2018, https://pjreddie.com/darknet/yolo/. Accessed 2 Dec 2018.

[3] "Histogram Of Oriented Gradients — Skimage V0.15.Dev0 Docs". Scikit-Image.Org, 2018, http://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html. Accessed 2 Dec 2018.

[4] "CNN Architectures: Lenet, Alexnet, VGG, Googlenet, Resnet And More ….". Medium, 2018, https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5. Accessed 2 Dec 2018.

[5] Ai.Stanford.Edu, 2018, https://ai.stanford.edu/~jkrause/cars/car_dataset.html. Accessed 2 Dec 2018.

[6] "Vehicle Make And Model Recognition Dataset (Vmmrdb)". Vmmrdb.Cecsresearch.Org, 2018, http://vmmrdb.cecsresearch.org/. Accessed 2 Dec 2018.

[7] "The KITTI Vision Benchmark Suite". Cvlibs.Net, 2018, http://www.cvlibs.net/datasets/kitti/. Accessed 2 Dec 2018.