Vehicle Detection for self driving cars

Usama Sadiq
University of New South Wales
Sydney, Australia
usamasadiq08@gmail.com

***Abstract*—Self Driving car has been a new and upcoming idea\challenge for the computer vision community. This challenge has been tackled with various methods of Deep learning, machine learning method including both supervised and unsupervised learning. The scope of this report is limited to vehicle detection module for the self-driving cars. First, I will discuss various approaches to vehicle detection for self-driving cars and then apply one of these methods keeping in mind the scope and limitations of our project. The whole process from Data augmentation, image preprocessing and training a supervised machine learning model and finally vehicle detection.**

Keywords—Computer Vision, self-driving cars, Vehicle Detection

I. INTRODUCTION AND BACKGROUND

The idea of self-driving cars has been a dream for the computer vision community and, but it presents its own challenges that need to be tackled before this dream can become a reality. For self-driving cars to be part of our transport systems in this modern-day age, the error in not detecting a vehicle on the road can lead to accidents. It is important for us the computer vision community to come up with techniques which can reduce this error. Whereas self-driving cars have their own benefits such as efficient fuel consumptions, prevention of road accidents or comfort and convenience for the community in the future.

The problem of detecting a vehicle for self-driving cars have been studied for a very long time. One of the approaches implemented by a team at Stanford University was to build a probabilistic model for tracking multiple vehicles on the road. Basic idea was to model the vehicles position, geometry and estimate these parameters using a Bayesian filter for each vehicle.[1]

Another approach taken are deep learning approaches which uses the Conventional Neural Network to extract features from an image and feeds it into a fully connected layer. During the training process, the Conventional Neural Networks are able to learn the features of the object and detects it in the image. Famous algorithms based on CNN are R-CNN [2], Faster R-CNN [3] and YOLO [4]. YOLO being the most efficient overall out of the three in terms of time and accuracy [4].

There is alternative approach which is based on feature extraction. After extracting features like (HOG or SIFT) features of images, features are fed to a machine learning algorithm like Support Vector machines, or it can be an ensemble model. Basically, any classifier which can do binary classification. In this approach, the classifier helps us to classify if the image contains a vehicle or not. Classifier and techniques like sliding window can be used to detect vehicles on the road.[5].

It is important in this supervised machine learning approach to have many training and test examples of images where dataset contains vehicles of different shape, color, and size so that the classifier can learn to classify all type of vehicles better. In addition, Dataset should also contain lots of images which can force our classifier to learn to tell the difference between vehicles and all other objects.

II. METHOD AND IMPLEMENTATION

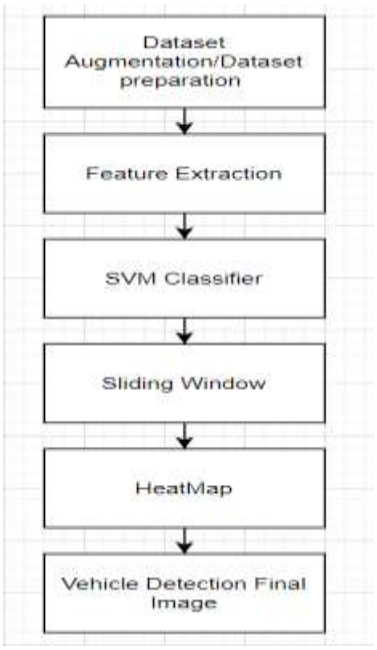

Fig 1. Flowchart of Vehicle Detection Pipeline

A. Scope and Limitations

In my project, I am required to detect the vehicles in image by just using a single frame. So, I will not be able to use the Frame Differencing technique or methods which use Background subtraction methods since I am limited in that aspect. The second limitation I have for our project is that I cannot use Deep learning techniques as part of the project specifications. This leaves me only with a Feature based approach which I am going to adopt here. Additionally, for this project, it is required to only use the Dataset provided to us.

B. Dataset Augmentation and Dataset Preparation

The data set contains images which were extracted from a video of 20 frames for 2 sec clips. Annotations files are also available which have information about the top, right, left, and bottom coordinates of the bounding box for the vehicles.

First Step in my implementation is to prepare the dataset of images for my model. The dataset will include images for positive class. i.e., only images of vehicles and negative class

which will be any portion of the image containing any objects other than vehicles.

I have used the annotations file to extract the images of vehicles. I have only used the 40th frame of the clip. After extracting the images, images are resized to 60 * 60.

The next challenge in building the dataset was to get images for my classifier for the negative class. Since the dataset contains whole images taken from a camera mounted on the roof of the vehicle. I used the technique of Data Augmentation to extract the images for the negative class. For this purpose, 40th Frame of the clips were randomly selected and by using Super Annotate software, parts of images where annotated which do not contain vehicles by manually annotating the images.

After this step, I was able to export these images and annotations files using the Super Annotate software. As mentioned earlier, annotations files return us the coordinates of the bounding boxes which can easily be used to extract the images. The size of the negative images is the same as the positive class i.e., 60 * 60. All the processing of extracting the images for both classes is done by a python script which extracts these images and saves these images for later use.

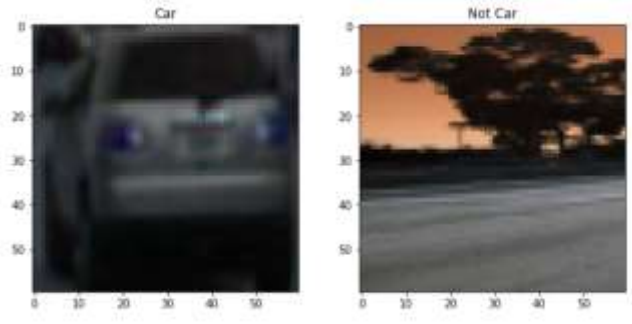

Fig 2. Showing the postive class and negative class.

C. Feature Extraction - HOG

Next Step in my implementation is to extract the features of the positive and negative class. So that these features can be fed to the classifier. Any well-known features can be used in this step for example Histogram of Oriented Gradients (HOG) or Scale-invariant feature transform like features among many others. I am using HOG Features instead of SIFT features because the performance of SIFT features-based classifier is slow and less accurate as compared to HOG features-based classifier [5].

D. Input Matrix for Classifer

I extracted the Hog features of positive and negative class images. Result of this extraction procedure produces a Numpy array of shape like this (No. of images * Hog Features of the image). HOG features of the classes are combined along the axis = 0 to get the input matrix for the classifier. The input matrix is normalized using python Standard Scaler which normalized the input matrix by removing mean and scaling it to unit variance [6].

E. Output Matrix for Classifer

Since this classification problem is a binary classification problem, the output matrix is of shape (No. of images * 1). This contains only labels with 1 and 0 where 1 being the vehicle present in the image and vice versa.

F. Support Vector Machine Model

Now, I will use my input and output matrix to train any classifier. I chose support vector machines as it is known to perform better on classification and pattern recognition problems [7]. Before feeding the data to the svm model, I will also split the train dataset into two sets. One for training and the other for testing the performance.

G. Sliding Window Approach:

Now that I have a classifier that can classify between both the classes. I can use the sliding window algorithm to first generate a windows list of different window sizes keeping in mind the size of positive and negative images [8]. This list of windows will be used to search for the vehicles in these windows. I can also define if I want to limit the region for searching the vehicles for both x and y axis. Additionally, I will be defining a parameter for overlap of windows for both the axis. Overlap parameter here means that we will iterate the image with stepSize < 1. Next step is to iterate through this windows list, and I will extract the window from the original whole image. I will resize the window to 60 * 60. Extract hog features for this resized window, normalize using the same instance of standard scaler. I will pass this normalized window size input to the classifier to make a prediction if a vehicle is present in the window or not.

H. Heat Map Approach

After the prediction, I can draw the bounding boxes around the detected vehicles. By just following this approach, the result showed to many false positives. Heatmap approach led to less false positives.

In heatmap approach, I just add 1 to all the pixels in the bounding boxes. In this way, we highlight all the pixels of the bounding boxes and then apply a basic threshold to turn pixels to zero which are under that threshold [8].

III. EXPERIMENTS

A. Watershed Algorithm And Mean Shift Algorithm

Other than the feature-based and classifier methods, I also tried to implement the watershed and mean shift algorithm to detect the vehicles in the image. But the image after thresholding the image into a binary image, I was not able to clearly separate the background and foreground. Both the techniques did not give me good results and produced a very distorted image.

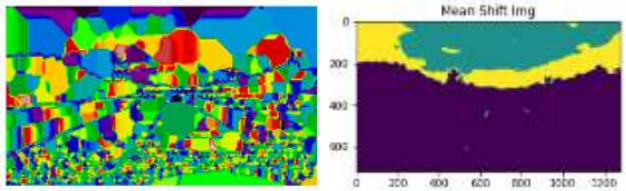

Fig. 3 Results of Watershed and Mean shift Algorithms

B. HOG Features

I tried two ways to extract the HOG features of the images. First, I tried to pass the whole image to the function for extracting the HOG features, but I found that better results

were produced when I extracted HOG features for color channels separately. One of the reasons for this improvement can be that more information is extracted when I followed the latter approach. Furthermore, different values for orientation, pixels per cells and cells per block were tested but the final values are [Orientations = 9, Pixel per cell = 16, cells per block = 1].

C. YCrCb Colorspace

As we know that there are a lot of color spaces available and different color space are useful for different problems. First, I tried RGB and HSV channels, but the final and best results are achieved using the YCrCb color space.

D. Sliding Windows

If I set no range for the window size, x axis, y axis and overlap for both the axis of the rectangular coordinate system. The resultant image has to many false positive. To improve the results, a limit needs to be set on the y axis to only consider the area where I expect the vehicles to be present in the image. i.e., 341 to 401. I also needed to adjust the overlap area in the x and y axis. Final values used for windows sizes [(86,59)] and overlap are and (0.81,0.81) respectively.

E. HeatMap Threshold

Final value used for heatmap threshold is 2.

IV. RESULTS AND DISCUSSIONS

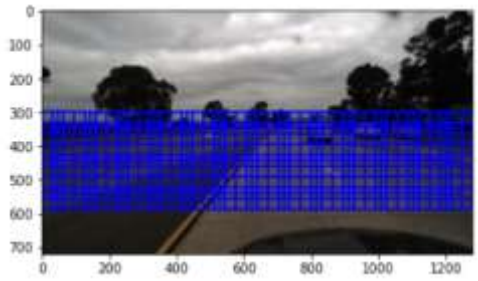

Fig. 4 Sliding Windows before classification

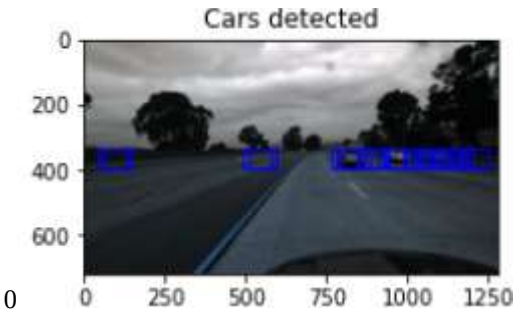

Fig. 5 Showing the cars detected.

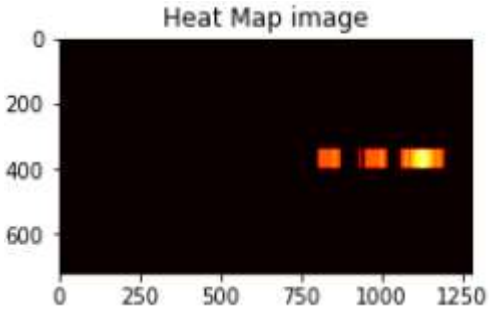

Fig. 6 Heatmap of the image after applying threshold value of 2.

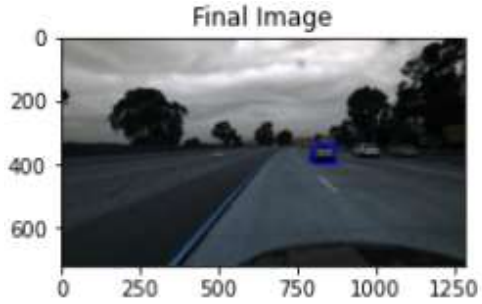

Fig. 7 showing the final Image of the detected vehicles.

The results are shown using only the train and test data. I have not used supplementary data to train and test my classifier due to time constraint. As the supplementary data has annotations for a greater number of vehicles, this approach should be able to detect other vehicles in the image as well. The accuracy for the svm classifier is 97.36%. In future, I can also use other classifiers instead of svm to check if it improves the performance overall.

REFERENCES

[1] Anna Petrovskaya and Sebastian Thrun, “Model Based Vehicle Tracking for Autonomous Driving in Urban Environmnets”.

[2] D. J. T. M. J. Girshick R., “Rich feature hierarchies for accurate object detect and semantic segmentation,” in Computer Vision and Pattern Recognition.

[3] G. R, “Fast R-CNN,” 2015.

[4] J. Redmon, S. Divvala, R. Girshick, Ali Farhadi, “You Only Look Once”.

[5] S. G. D. S. Filho, R. Z. Freire and L.D.S.Coelho, “Feature Extraction for on road Vehicle Detection Based on Support Vector Machine”

[6] Standard Scaler, Scikit preprocessing available at https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html (online)

[7] S. Kim, S.Lee, K. Min and K. Cho, “Design of unified support vector machine circuit for pedestrains and car detection”, 2012

[8] Wael A. Farag, “A lightweight vehicle dectection and tracking technique for advanced driving assistance systems”, 2020

[9] Sliding window approach is available at https://en.wikipedia.org/wiki/Sliding_window_protocol (online)