Younghoon Choi

COM 499

12/20/2022

Self-Driving Simulation with Computer Vision and Carla

As many automobile companies have released self-driving vehicles in the market, the research for autonomous driving has grown exponentially grown over the decade. Even though those self-driving cars already have a big impact on our lives, there are areas where the vehicles struggle, in other words, do not perform well which can lead to a serious issue since it is responsible for human life.

For my senior research at Connecticut College, I wanted to explore the computer vision side of self-driving vehicles using Carla, CAR Learning to Act. Carla is an open-source autonomous driving simulator powered by Unreal Engine which can be found in high-graphic gaming in today’s world. It was developed to support the training and validation of the autonomous driving system. This makes Carla efficient in terms of time, cost, and safety by testing algorithms in the simulator rather than testing on a real-life car. This process helps eliminate the errors of a self-driving system which can end up in an accident. Carla also provides various environmental conditions such as weather, map, etc. which allows users to explore all the different situations a vehicle might face in real life.

Carla has features that allow vehicles in simulation to work similarly to real cars. Features such as cameras, control, and sensors are the main features I will be using in this research. The camera in Carla is free of positioning meaning multiple cameras can be attached anywhere in the car. For example, a dashcam camera and a rear-view camera can work synchronously. Control of the vehicle has three functions, throttle, steer and break where each function can be given any numeric value to make changes in its control. Lastly, sensors in Carla provide ground truth such as collisions with objects or lanes and location. This feature was only used to validate my predictions and was not used to control the vehicle.

The goal of this research is to achieve autonomous driving through simulation including computer vision, vehicle control, and routing. Using Carla, I was able to collect my own dataset directly from the simulator, train, and test different learning methods, and compare results from different conditions of the simulation. The dataset I collected included a total of 904 images including 446 images of 30 MPH, 282 images of 60 MPH, and 176 images of 90 MPH. These images were captured in four different weather conditions Mid Rainy Noon, Cloudy Evening, Clear Sunset, and Clear Noon. Each of these conditions has a unique look to it as I wanted to compare how and what the algorithms struggle with. I located the camera on the top and center of the windshield. This dashcam view allows the full potential of computer vision as it let the camera look at the world clearly. It is important for the camera to be reliable when it comes to looking at world objects such as roads, lanes, signs, and lights to make consistent predictions.

To keep the vehicle in the center of the road, lane detection was implemented. Hough transform was used to find the lines by looking at the lanes on each side of the road which is a feature extraction technique in computer vision. It is a voting procedure to find imperfect instances of objects within a certain class of shapes. Once the two lines meaning lanes are detected, I find the center point of the two which is the center point of the road. I compare the center point of the vehicle to the center point of the road detected then steer as needed.

Traffic light detection was used for vehicles to react to the different colors of lights, green and red. To detect the circles in the traffic lights, I once again used the Hough transform which is a placeholder for now as other objects may have similar look to it. For example, red lights on a back of a vehicle or even a fire hydrant can be detected as traffic lights. Once the circle is detected, I check its HSV, Hue Saturation Value, to determine whether it is green or red and control the vehicle. If red, I give values to break. If green, I give values to throttle to get it moving.

Speed sign classification was implemented to change the values given to the vehicle’s throttle. This process has three steps, Hough circle detection, template matching, and SVM Grid Search. Like traffic light detection, I first look for any circles in each frame. Once a circle is detected, I crop out the frame image to the exact size of the speed sign by the diameter of the circle. This allows the process to focus on the circle itself. Then I take the cropped-out image then compare the templates of 30 MPH, 60 MPH, and 90 MPH to find the correlation coefficient, a pixel-by-pixel comparison between the template and the region. If the correlation is greater than the threshold and decides whether the circle detected is even a speed sign. Once I decide that it is a speed sign, I use Support Vector Machine to classify the sign between 30 MPH, 60 MPH, or 90 MPH. SVM is a linear model for classification and regression problems as it is suitable for practical image classification. It solves both linear and non-linear problems. The idea is to create a line or a hyperplane that separates the data into classes.

As mentioned earlier, vehicles in Carla have three functions for their control. The vehicle is in closed-loop control meaning it works in real-time operation in simulation time. The values of each function, steer, throttle, the brake is given based on my predictions from lane detection, traffic light detection, and speed sign classification. For example, the controller tries to match the center point of the dashcam to the center point of the lane. It provides a sharper steer if the distance between the two center points is greater than a threshold. For speed sign detection, I give a smaller value to the throttle if 30 MPH is detected and higher if others are detected.

The results for the traffic light were 100% as it was just looking at its HSV value. The problem was the detection part as light itself was not detected from time to time as other circle objects got a higher vote over lights in the Hough transform meaning the circle is neither red nor green. On the other hand, speed sign detection had a total accuracy of 92.46%, 139 out of 150. 30 MPH had 90%, 45 out of 50, as it had the lowest accuracy of all classes in certain weather conditions like Mid Rainy Noon and Clear Sunsets where captured images often had blurry due to rain or intense lighting. 60 MPH had 96%, 48 out of 50. 90 MPH had 92%, 46 out of 50.

Future work involves:

1. Exploring YOLO detection

* As mentioned earlier, I would like to implement object detection for traffic light detection

1. Create my own customized map

* Having my own map would save me time as I can create a route that can test all my algorithms with consistency

1. Explore other learning methods such as deep learning methods

* Develop deep learning to compare to the SVM model I have currently

1. Routing system for interactions with no lanes

* A navigation system that can guide a vehicle from point A to point B

1. Evaluate how well comfort travel time and safety work

* Have a new vehicle control system. Instead of giving values to throttle each frame, look at the current speed and adjust my throttle.

Reference

“Carla Settings.” *CARLA Settings - CARLA Simulator*, https://carla.readthedocs.io/en/stable/carla\_settings/.

Am. “120 Dog Breeds - Classification.” *Kaggle*, 20 Apr. 2022, https://www.kaggle.com/datasets/66c1d9bf5dc19d7b625c8dc2ab926fdfba7a66b7ccaac60cf70f8fa480f086ae.

“3.3. Scikit-Image: Image Processing¶.” *3.3. Scikit-Image: Image Processing - Scipy Lecture Notes*, http://scipy-lectures.org/packages/scikit-image/.

“1.4. Support Vector Machines.” *Scikit*, https://scikit-learn.org/stable/modules/svm.html.

“Sklearn.model\_selection.GRIDSEARCHCV.” *Scikit*, https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.GridSearchCV.html.

“Generate Maps with OpenStreetMap.” *Generate Maps with OpenStreetMap - CARLA Simulator*, https://carla.readthedocs.io/en/latest/tuto\_G\_openstreetmap/.

“[CAVH] Carla Simulation Project.” *CAVH Research Group*, https://cavh.cee.wisc.edu/carla-simulation-project/.

angelkim88. “Angelkim88/Carla-lane\_detection.” *GitHub*, https://github.com/angelkim88/CARLA-Lane\_Detection.