rAlcer

Team Members

Kirsten Bauman, Ryan Black, George Jones, Excel Ortega, and Zachary Schmalz

Spring 2018 Gaming Capstone - AI Driving, Yoshihiro Kobayashi

About the Project

For our project, we aimed to create a virtual environment where a simulated self-driving car can drive through. This can result into the development of a more robust system because different environments during the day and/or night can be simulated with different weather conditions such as dust, rain, and snow on the road. Situations that are otherwise dangerous in the real world such as a kid suddenly walking across a road, other cars driving recklessly, and car parts malfunctioning can be simulated as well.

In addition, many iterations of the simulation can be ran and huge amounts of data can be generated and analyzed with no additional cost. This is important because creating these scenarios in the real world can accumulate into a huge expense and can sometimes be not feasible.

Game Modes

- **Package Delivery** mode serves as a tutorial to get players acquainted with the control system and gameplay of the simulator. The task is to deliver packages to residences.
- **Racing** mode is used to train and test AI driving models. The player is able to record racing data, train the AI using the data, and then race their generated AI.

Objectives

- To train a simulated self-driving car on a virtual environment and develop an AI model.
- To create a virtual vehicle driven by the AI model on a virtual environment to race against.

Level Design

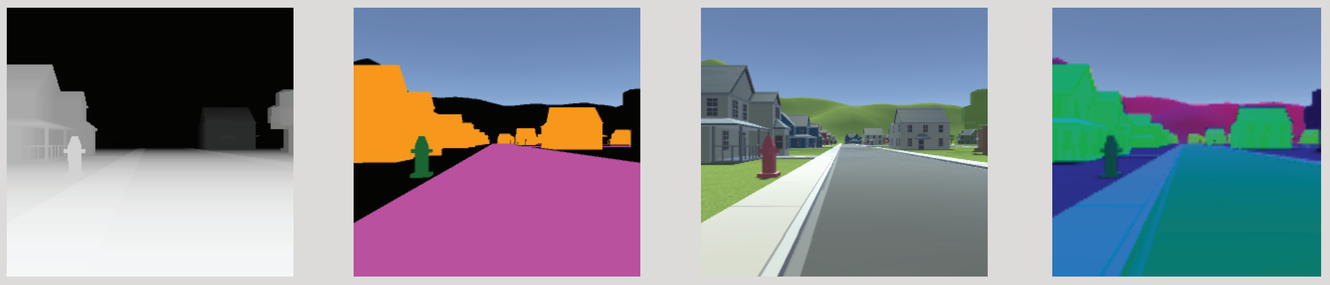
Package Delivery
The first map is a large neighborhood with a daytime setting. This map is easier to navigate than the second delivery map. The second map is a city at night. There are many stop lights and differing traffic interactions in this map. Both maps could eventually be used to train the AI for more complex situations.

Racing
We also created three racing loops to train and test the first AI models. The first racing loop is a simple Nascar style track with four lanes and only left turns. The second map is slightly more complicated with varying straights and curves, and a right turn. The final racing map follows a complicated pattern and the course is littered with obstacles that the AI can be trained to avoid.

AI Implementation

Data Logging & Shaders

The data given to the AI is generated by a user driving around an environment and saving their input in a comma-separated file (csv) file. This input consists of images from three camera views in the left, center and right of the car, the throttle, reverse, steering, and speed. Each image is a combination of 3 shaders mapped to the red channel (depth shader), green channel (segment shader) and the blue channel (grayscale).



Training

Once the image data and driving logs have been created, that data is then used to build and train a convolutional neural network that performs the automatic driving. A convolutional neural network (CNN) is a black box that receives input and spits out an output. In our case, we input the generated camera images and the driving log data, the black box performs image and data augmentations such as cropping, zoom, rotation, axis flipping, brightness, saturation, and outputs a steering angle prediction for the car.

The CNN model generation code is written in Python using Keras, a high-level neural network API running on top of Tensorflow, a low-level computational framework for building machine learning models. Using Keras, we can generate models from the recorded data and then use those models to predict steering angles for the autonomous car.

Server Side

In our project, a python server running the trained keras model is established via a socket that Unity can connect to. A live camera feed and driving log attached to the autonomous car is sent to the CNN that predicts what the current steering angle should be.

GAS Studio

Ira A. Fulton Schools of Engineering
Arizona State University