rAIcer: Self Driving AI Simulation Environment

Kirsten Bauman, Ryan Black, George Cooper Jones, Excel Ortega, Zachary Schmalz, Yoshihiro Kobayashi

Arizona State University

University Drive and Mill Avenue, Tempe, AZ 85281, USA

kebauma1@asu.edu, ryan.black@ryblack.com, gcjones1@asu.edu, excel.ortega@gmail.com, zacharyschmalz@gmail.com, ykobaya@asu.edu

ABSTRACT

The goal of this project was to develop a simulated virtual environment for training a self-driving game AI. The simulation can recreate a multitude of different environmental factors including different times of day, weather conditions, etc. The integrated data capture system uses custom depth, segment, and grayscale shaders to minimize file size while optimizing the quality of captured training data. AI models are trained from the data using the TensorFlow deep learning model running on the Nvidia CUDA parallel computing platform. Once an AI model is created, it can deployed and raced against by the user.

Author Keywords

Artificial Intelligence (AI), Neural Network, Self-Driving Car, Convolutional Neural Network (CNN)

INTRODUCTION

Virtual simulations are often used to augment the real life testing of self-driving AI due to the efficiency and low cost of running high quality recreations of realistic events. This can result into the development of a more robust AI system because different environments such as day and night as well as different weather conditions such as dust, rain, and snow can be simulated. Situations that are otherwise dangerous in the real world such as a child 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 run 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 to a huge expense and is often times not feasible.

BACKGROUND

The following is a description of the hardware software/technologies used during the project.

Unity 2017.3

Unity is a popular multi platform game engine known for its ease of use and streamlined workflow. The game makes heavy use of the Simple World unity package that provided the environment assets such as roads, vehicles, and buildings.

Keras 2.1.4

Keras is an open source, high-level neural network API. Written in Python, it runs on top of Tensorflow and is designed for a user friendly approach to building and experimenting with deep neural networks. Keras was used for building the Convolutional Neural Network (CNN) that powers the autonomous driving vehicle.

Python 3.6

Using the Anaconda environment, the CNN modeling (using the Keras framework), training, and driving scripts were written in Python.

Logitech Driving Force GT

While the game supports keyboard and mouse input, the Driving Force GT steering wheel provides more accurate real-world driving data for logging, such as steering angle, acceleration, and braking.

SYSTEM IMPLEMENTATION

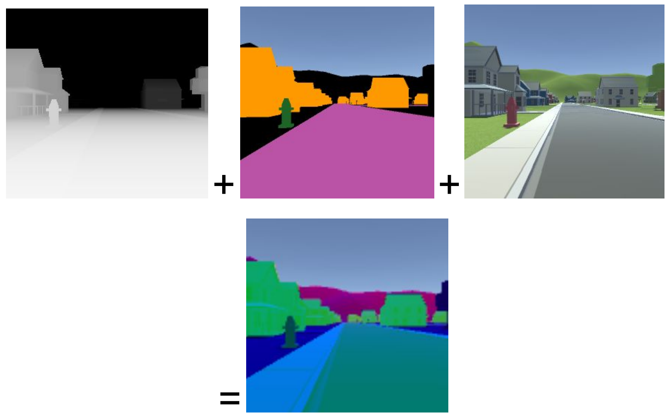
Data Logging

The user generates data while driving around the environment that is logged by the game. The data is arranged as follows: a reference to the image file from the left camera, a reference to the image file from the center camera, a reference to the image file from the right camera, the throttle, reverse, steering angle, and speed. Each of the input is saved into a .csv file per line per frame.

Shader

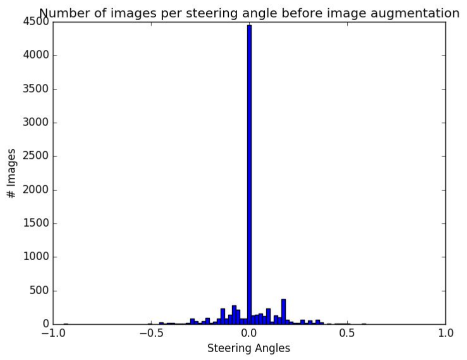
We wanted to develop a data capture system that would maximize the amount of information that could be stored in each frame of data. To accomplish this, the left, center, and right front facing cameras on the vehicle render the scene using three different shader methods. The first method uses a z buffer to create a depth map image of the scene. This gives the AI information regarding the distance of certain objects from the car. The second shader method is a segmentation view of the scene. This recreates the images output by the SegNet neural network by drawing each object a specific solid color based on the type of object it is (ie. building, road, sign, etc.). The final shader is the standard lighting shader which shows a regular view of the scene.

These three shaded images are then downsampled from four channel images (RGBA) into single channel images which are then combined into the red, green, and blue channels of the final output image. Finally, the image is drawn to a 128x128 render target which is exported to a <10kB PNG file every frame.



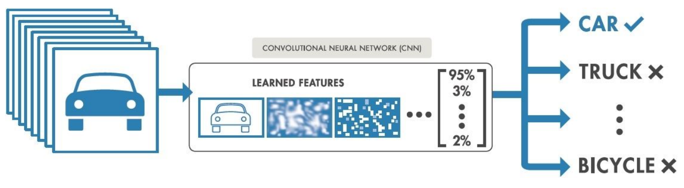
Model Generation and Training

With the data and images generated, it is time to start generating the AI model and training. If the dataset of steering angles is loaded into a histogram, we can see that the steering angles have a strong bias towards driving straight, i.e. steering angle ~ 0. If this bias is not accounted for during the training, it will be reflected in the final model, yielding poor results.

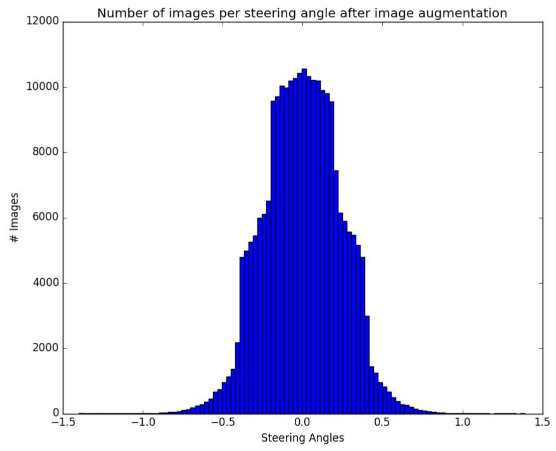


To account for this bias, the current literature on image deep learning and industry standard is to perform a series of image augmentations and manipulations. Some image augmentation functions include zoom, rotation, horizontal and vertical shift, horizontal flip, channel shift, crop and resize, and brightness shifting, all of which Keras provides in its API. These image augmentations allow the model an opportunity to learn how to recover from slight deviations from the norm behavior. The original, unedited images are never sent to the neural network. As was learned the hard way, 3 cameras on the vehicle, front, center, and right, are absolutely necessary for model generation. The side cameras provide recovery paths for when the vehicle is slightly off the optimal driving path, otherwise the model does not know to stay on the optimal path and will crash.

With the image dataset augmented, we then begin building the Convolutional Neural Network model with Keras. CNN's are especially good at performing object recognition. The basic idea of a CNN is that an image is inputted into a trained model designed to recognize image patterns and elements, and give an output based on the image, such as an image caption, or in this case, a steering angle for the autonomous vehicle.

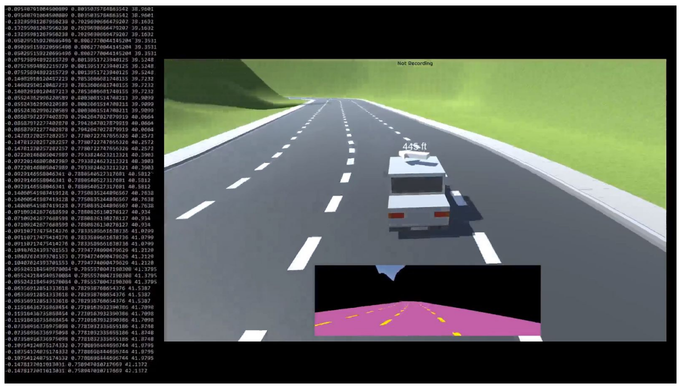


The CNN architecture used in this project is based on the architecture described in NVIDIA's groundbreaking paper on a similar topic, *End to End Learning for Self-Driving Cars*, and Keras allows implementation of this structure with just a few lines of code. The models created for this project are mainly based on users driving laps around a track using the GT steering wheel; each model was trained on approximately 30,000 images (10,000 from each camera), and the entire training process takes approximately 3-5 hours to complete depending on the number of images, and the number of "epochs" or model instances. At the end of the training, it can be seen how the forward driving bias is accounted for and normalized in the model.



Model Networking

With the model generated, that model is then used to predict steering angles for an autonomous car. The model is loaded onto a python server that establishes a socket that networking scripts in Unity can connect to on any IP address. With the socket established, Unity sends an image stream from the center camera of the car as well as telemetry data such as current steering angle, throttle, and speed. Using this data, Keras inputs this data to the model and returns a steering angle and throttle value to the autonomous car control scripts in Unity.



```
-0.09540791064500809 0.8035035784863542 38.9601
-0.09540791064500809 0.8035035784863542 38.9601
-0.13295981287956238 0.7929690666479207 39.1632
-0.13295981287956238 0.7929690666479207 39.1632
-0.13295981287956238 0.7929690666479207 39.1632
-0.050295159220695496 0.8062770044145204 39.3531
-0.050295159220695496 0.8062770044145204 39.3531
-0.050295159220695496 0.8062770044145204 39.3531
-0.07575694892215729 0.8013951723940125 39.5248
-0.07575694892215729 0.8013951723940125 39.5248
-0.07575694892215729 0.8013951723940125 39.5248
-0.14082910120487213 0.7853606681748155 39.7232
-0.14082910120487213 0.7853606681748155 39.7232
-0.14082910120487213 0.7853606681748155 39.7232
-0.05524362996220589 0.8003061514708211 39.9099
-0.05524362996220589 0.8003061514708211 39.9099
-0.05524362996220589 0.8003061514708211 39.9099
-0.08687972277402878 0.7942647027879919 40.0664
-0.08687972277402878 0.7942647027879919 40.0664
-0.08687972277402878 0.7942647027879919 40.0664
-0.14781270257282257 0.7780722747656326 40.2572
-0.14781270257282257 0.7780722747656326 40.2572
-0.14781270257282257 0.7780722747656326 40.2572
-0.07220146805047989 0.7933824623212321 40.3903
-0.07220146805047989 0.7933824623212321 40.3903
-0.07220146805047989 0.7933824623212321 40.3903
-0.0929146558046341 0.7880540527317681 40.5812
-0.0929146558046341 0.7880540527317681 40.5812
-0.0929146558046341 0.7880540527317681 40.5812
-0.14060541987419128 0.7750835244896567 40.7638
-0.14060541987419128 0.7750835244896567 40.7638
-0.14060541987419128 0.7750835244896567 40.7638
-0.07109242677688599 0.7880826130278127 40.934
-0.07109242677688599 0.7880826130278127 40.934
-0.07109242677688599 0.7880826130278127 40.934
-0.09110717475414276 0.7833589661638736 41.0799
-0.09110717475414276 0.7833589661638736 41.0799
-0.09110717475414276 0.7833589661638736 41.0799
-0.10407624393701553 0.7794774090479629 41.2128
-0.10407624393701553 0.7794774090479629 41.2128
-0.10407624393701553 0.7794774090479629 41.2128
-0.055242184549570084 0.7855578047190308 41.3795
-0.055242184549570084 0.7855578047190308 41.3795
-0.055242184549570084 0.7855578047190308 41.3795
-0.06356912851333618 0.782938768654376 41.5387
-0.06356912851333618 0.782938768654376 41.5387
-0.06356912851333618 0.782938768654376 41.5387
-0.11918436735868454 0.7710163932390386 41.7098
-0.11918436735868454 0.7710163932390386 41.7098
-0.11918436735868454 0.7710163932390386 41.7098
-0.07358956336975098 0.7781032335655186 41.8748
-0.07358956336975098 0.7781032335655186 41.8748
-0.07358956336975098 0.7781032335655186 41.8748
-0.10754124075174332 0.7788696444696744 41.9795
-0.10754124075174332 0.7788696444696744 41.9795
-0.10754124075174332 0.7788696444696744 41.9795
-0.14781772011613031 0.758947010717669 42.1372
-0.14781772011613031 0.758947010717669 42.1372
```

CASE STUDY

Udacity, an online educational website offering courses on computer science and mathematics courses. In 2017, Udacity launched an open source project for building a self driving car. Their process of data gathering, model generation and training, and networking is essentially the same as has been described in this paper, albeit a lot more streamlined and polished. In fact, this project has been a valuable resource for replicating the AI systems and CNN generation and training process.

CONCLUSION and FUTURE WORK

This project is a successful implementation of an efficient simulation environment that allows for streamlined training and testing of self-driving AI models. The game's internal data logging makes recording training data as simple as playing the game and once the data is created, it can be sent to the AI server for training. Testing confirmed that the combined shader method used for data capture in this system greatly improved the AI's ability to interpret a scene over a standard image capture method. Data optimizations also kept training data small and manageable without reducing image quality.

This project is not only a proof of concept for a self driving car simulation, but also a demonstration of an agile AI that can be deployed to learn how to play almost any type of game. Further development of this project could extend beyond driving and demonstrate secondary applications of the AI's game learning capability. Additionally, more realistic data collection methods such as simulated lidar and a live SegNet AI implementation could improve the realism of training data collected.

REFERENCES

1. Badrinarayanan, V., Kendall, A. and Cipolla, R. (2017). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*,
2. [online] 39(12), pp.2481-2495. Available at: http://mi.eng.cam.ac.uk/projects/segnet/#publication.

3. Raval, S. (2017). *How to Simulate a Self-Driving Car*. [video] Available at: https://www.youtube.com/watch?v=EaY5QiZwSP4 [Accessed 24 Feb. 2018].
4. NVIDIA (2016). *End to End Learning for Self Driving Cars.* Available at: https://devblogs.nvidia.com/deep-learning-self-driving-cars/ [Accessed January, 2018]
5. Udacity (2017) *An Open source Self Driving Car*. Available at https://github.com/udacity/self-driving-car