Using Simulations and Domain Randomization for Autonomous Driving

Preprint · July 2020

DOI: 10.13140/RG.2.2.30272.61447

2 authors, including:

Ulrich Göhner
Hochschule für angewandte Wissenschaften Kempten
72 PUBLICATIONS **45** CITATIONS

SEE PROFILE

Technical Reports in Computing Science No. CS-02-2020

Using Simulations and Domain Randomization for Autonomous Driving

Stefan Leiprecht

July 2020

Hochschule Kempten
University of Applied Sciences
Fakultät Informatik

Using Simulations and Domain Randomization for Autonomous Driving

Stefan Leiprecht

(Lecture Notes SS 2020)

***Abstract*—In the last decade autonomous driving has evolved from a science fictional dream to an everyday reality. With the advance of more and more companies bringing their versions of self-driving cars on the street it is just a matter of time before the majority of transportation will be in the hand of computers. But with the deadly car accident involving a self-driving Uber car back in 2018 there is also the question about how reliable autonomous driving really is and how we can validate and test the safety of this new road user[1]. An uprising approach towards creating robust and adaptable neural networks is called domain randomization. This paper explores the possibility of using this method to create training data with driving simulations. It will propose a list of important criteria and factors affecting the selection of a fitting simulation. Furthermore it will present a track generator which is able to create useful tracks and export them to a format which can be used by several common simulations used in the field of autonomous driving research.**

***Index Terms*—autonomous driving, domain randomization, simulation, procedural generation.**

I. Introduction

With the rise of Deep Learning and Neural Networks there is an increasing demand for large amounts of high quality training data. For autonomous driving this data mostly consists of images of the road ahead and the area surrounding the vehicle. Gathering those in the real world is possible though Kalra and Paddock [1] reported it requires hundreds of millions of miles driven worth of training data to verify a compatible grade of reliability. Due to the huge amount of time and money it would need to fulfill this demand a majority of scientists started to use simulations. Simulations are way cheaper, safer and much more scalable than anything including real world driving. The ability to train thousands of cars simultaneously enables researchers to achieve progress much faster. Although this is great there is a known problem called the *reality gap* which describes the discrepancy between simulations and the real world. This includes things such as lacking richness and noise of sensor data or lacking variety of environments which can lead to an insufficient ability to generalize the world enough to be able to react to previously unseen data.

This paper was created as part of a university project with the goal to build up a model car in order to participate in two autonomous driving cups, namely the Carolo-Cup[2] and the VDI ADC[3]. In order to achieve this task, this part of the group looked into possible ways to create sufficient amounts of suitable training data and also a handy way to eventually test the Neural Network. Therefore, this paper explores the usability of simulations to provide a varied data set using the concept of Domain Randomization.

II. Related Work

There is a similar work done by Sulkowski et al. [2] They evaluated three different solutions for autonomous driving research. Namely Microsoft AirSim[4], Carla Simulator[5] and GTA V[6]. They grouped their evaluation criteria into two main sections. The first group contains functionalities to resemble real world features. Specifically those were the quality of the physic simulation, variety of available assets, range of emulated sensors and multiple viewpoints support. The second criterion was the development facilities, including programming languages, operating system support, configuration options as well as license models. Some of those criteria are also of great significance for this work, while some are mostly irrelevant. The specifics of those differences will be discussed in the next section but the main difference which makes this work not redundant compared to Sulkowski et al. works is the scope and the use case of the project this work belongs to. While Sulkowski et al. tried to find a good fit for researching with real sized vehicles in a real traffic environment for this project there are only 1:8 and 1:10 car models, driving on simple round tracks marked by white lines. Therefore, training the model with real world road networks including several different road users such as pedestrians, bicyclists, cars, trucks and motorcycles is irrelevant. Also the environment does not have to reflect detailed urban areas or highways. For those reasons an evaluation of already reviewed as well as new simulations for the specific requirements of model cars is useful. The main difference however is a strong focus on the possibility of Domain Randomization. The idea of this concept will be discussed in the next section. But for now, it is important to understand that even a simulation is limited to the amount of assets and environments which are available for this certain simulation.

III. Domain Randomization

Generally, the goal of Domain Randomization is to overcome the *reality gap*. Usually there is a discrepancy between the real world and a physics simulation. Physics simulations conventionally abstract their calculations in some way to

[1] https://www.nytimes.com/interactive/2018/03/20/us/self-driving-uber-pedestrian-killed.html

[2] https://wiki.ifr.ing.tu-bs.de/carolocup/en/carolo-cup

[3] https://www.vdi-adc.de/

[4] https://microsoft.github.io/AirSim/

[5] http://carla.org/

[6] https://www.rockstargames.com/V

enable at least real time computation. Therefore, physical properties like friction or density of materials must be set manually. The process of tuning these parameters to fit the real world is time consuming and often error-prone. Also, simulations suffer of low-fidelity simulated sensors. The created data lack the richness and noise real world sensors offer. A possible way to address this problem is proposed by Tobin et al. [3]. They have shown successfully that it is possible to train a model with domain randomized data up to a point where it can generalize real world data so that there was no need for further real world training. Another advantage of relying on simulated data is an easy way to label data without any additional effort once there is a system to render semantic segmented images or labeled data in general within the simulation.

Like mentioned in the previous section there is a problem with driving simulations as well. Most of the available simulations use a set of hand-build environments which are populated by a variety of also hand-built road users. Therefore there is a chance to reach a bottleneck concerning the variety of the data set. In this case it is the opposite problem to the real world where a practical endless amount of diverse roads and environments with fast changing elements already exist. Even if it is possible to collect data of a million road miles there is a chance of getting some kind of bias. For example, if the simulation only has roundabouts with 4 entries, the car might behave in an unexpected way if it encounters roundabouts with more or less entries. Also, there is a risk of only recognizing very specific pedestrians, if the simulations only has white male adults with cargo shorts. This way the agent might not learn to identify children or disabled people in a wheelchair [4].

For autonomous driving there is a series of possible factors affecting the robustness of a ML agent:

- The **Track** itself is quiet important. The agent has to learn to drive any kind of route even if it has not seen it before.
- **Environment, like trees, buildings and pedestrians**: Everything around the track can be quite diverse. The agent needs to correctly separate the road from the surroundings regardless of shapes, colors or position.
- **Obstacles** are also very important. It is necessary to identify obstacles on the road. No matter if it is another car or a person walking across, the car must react reliably.
- **Physical properties** like grip of the wheels, brake force, weight of the vehicle or composition of the underground.
- The **View:** Usually diversity depends on the weather, daytime or pollution. Also cracks in the camera or lens effects can affect image data.
- At least the **Vehicle** itself can differ in its appearance and properties. Different wheel diameter and steering angles heavily influence the driving behavior of each and every car. Also malfunctions or defect parts can change how the car responds to input data.

IV. Evaluation Criteria

Training a vehicle to follow a marked lane or drive within an enclosed track is quite different compared to real world driving. There is no need to learn traffic rules or complex road patterns like crossings or roundabouts. Also, it is not necessary to train difficult maneuvers like overtakes or dealing with blocked roads or how to behave encountering accident sites. Nevertheless, it is recommended to approach some of those issues in a fitting adaption. So even a toy car can be taught to avoid obstacles blocking its way or to avoid other moving objects like other cars. Also, it is possible to train the car to identify signs and reacting accordingly to their meaning for example breaking at stop signs or adjusting their maximum speed according to speed limits. For this project there were the following criteria determined:

- Offer Domain Randomization
- Open AI Gym Support [5]
- Open Source and Free to Use
- Adjustable
- In-the-loop training and evaluation

V. Existing Simulations

The following chapter lists a number of existing open source and free to use simulation. There are many more simulation systems which would fit the requirements to be evaluated but for this work the focus was on the more popular and extensive representatives.

A. Carla

The first simulation system that was considered is Carla [6] developed by the Computer Vision Center of the Universitat Autonoma de Barcelona which can be seen in figure 1. It runs with the Unreal Engine 4 and can be used under the MIT License. It is free to use and completely open source. Unfortunately, it does not offer a handy way to realize Domain Randomization. Although it offers a variety of different ready to use maps including environmental assets and various road users such as different cars, cyclists and pedestrians. Also, it is possible to integrate additional vehicles which is very important. This should be mentioned though the integration is a little complicated and inconvenient. Also creating and loading maps can be quiet challenging. The easiest way to create and modify maps is by using the tool RoadRunner by VectorZero [7]. Also, it is possible to import a road network with an OpenDRIVE file. Even if it does not support Open AI Gym there a several wrappers to connect the simulation with the library. The ability to modify the simulation easily is lacking. Development is poorly documented which makes it quite hard to implement a customized system to randomize anything automatically. Also the ability to create content and train it within Carla is not supported. Also the driving physics are rudimentary and offer little parameters to adjust the system in order to fit the real world.

B. LGSVL

The second one is the LGSVL Simulator(LG Silicon Valley Lab)[7] by LG Electronics [8] seen in figure 2. The simulation runs with the Unity Engine. It is free and open source and

[7]https://www.lgsvlsimulator.com/

Figure 1. Carla running with connected Client

Figure 2. LGSVL Simulation

can be used under the Simulator Software License Agreement. Like Carla it supports the creation of maps with RoadRunner, but there are also several maps already available which are ready to use. It also offers a variety of cars and pedestrians. Again it is possible to integrate specific vehicles. Also, it offers native Open AI Gym support. Unfortunately, it also lacks the flexibility just as Carla. There is no existing system in place to randomize anything. Also, there is no handy way to run the simulation while continuously loading new content from outside the simulation system.

C. AirSim

The third one is AirSim [9] by Microsoft seen in figure 3. AirSim runs on the Unreal Engine 4 as well on the Unity Engine. Being able to develop on both engines can be beneficial as long as both stay synchronized featurewise. Like the previous two systems it is open source and free to use under the MIT License. Originally it was developed as a drone simulation but it supports also ground vehicles since 2017. Unlike Carla and LGSVL it already provides some kind of domain randomization in the form of texture swapping. How Tobin et al. [3] showed, it is possible to achieve promising results even with simple plain colors. Due to its origin in the drone development it is possible to integrate almost any kind of vehicle into the simulation. Out of the box it offers a variety of maps including a large area called *Windridge City* which offers a bigger area than most other simulations. Also it has a variety of ready to use vehicles.

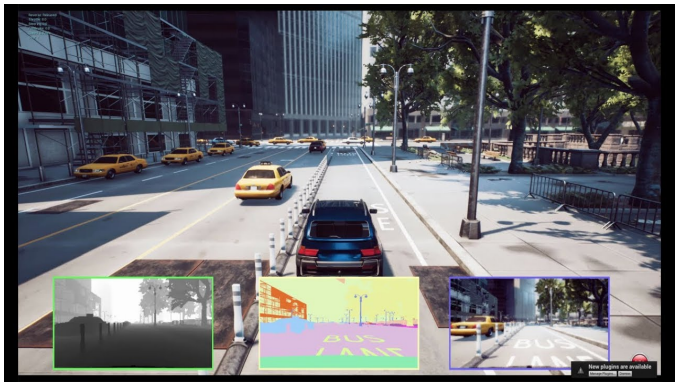

Figure 3. AirSim by Microsoft

VI. Evaluation

Unfortunately none of the tested simulation systems where able to fit all the criteria. Carla mostly lacks in flexibility especially regarding the build process and running the simulation during the development process. LGSVL is a little more approachable than Carla, although again building and running the simulation with own content repeatedly led to errors and problems. AirSim might be the most flexible one in regards to adjusting the simulation system. Nevertheless, all of them lacked the ability to implement Domain Randomization in an easy and useful way. Therefore, it was necessary to find a suitable solution to supplement the existing features. Due to the lack of easy convertibility an extern tool was created which is supposed to replace at least some missing features.

VII. Track Generator

Due to the lacking ability of all simulations to generate randomized tracks a new track generator was developed to solve this problem. The result was a fast and simple iterative algorithm that creates tracks fitting the requirements. It runs in the Unity Engine and creates round tracks with a variable amount of control points which vary in relative position to the middle point. It is easy to use and delivers appropriate results. Also the performance of this tool works in the range of microseconds. Therefore it can create large amounts of unique tracks in an extremely short time which makes it capable to be used within a pipeline which runs hundreds of trainings simultaneously in a short amount of time.

In figure 4 the result of this generator can be seen.

For further information read 'Procedural race track generation for domain randomization' by Behrens [10].

VIII. OpenDRIVE Exporter

To use the generated tracks with existing simulations there was the need to export it to a format that can be read by the simulation. Such a format is the OpenDrive Format xodr. Xodr is an open data format to describe road networks. Its goal is to standardize the data transfer between different simulations in order to use created content with various simulations regardless of their internal structure. This format was picked because Carla and LGSVL already support the import of OpenDrive files in some way. To test the export and import of a track

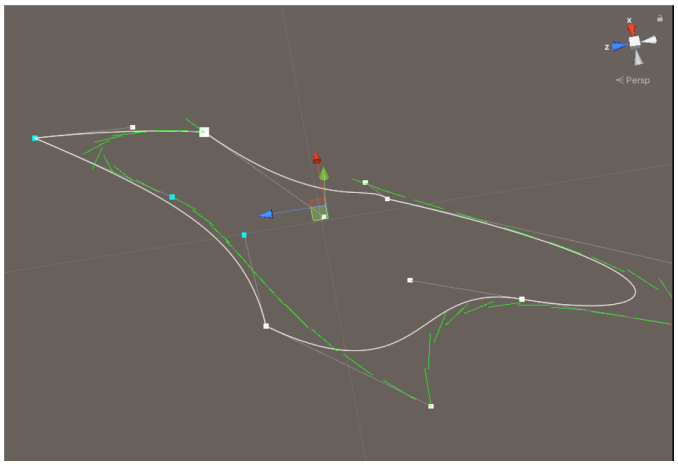

Figure 4. Generated track in Unity

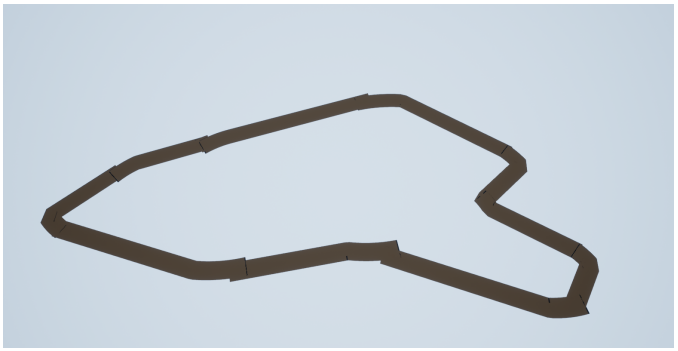

Figure 5. Imported track in Carla

several test tracks were imported to Carla. This can be seen in figure 5.

A big advantage is the possibility to import OpenDrive data not only in various different simulations but also into customized simulations. Plus, it is possible to switch between simulations according the task which fits best to all needs while always using the same prebuilt road networks.

IX. Results

The created and imported tracks still have some issues concerning the procedural mesh generation of Carla which led to overlapping meshes and blocking walls. Even though most of the problems could be fixed there is still room for improvements. Also a virtual version of the model car that is supposed to be trained in the simulation is still missing. But with the existing cars in Carla it is already possible to record training data to develop and validate Lane Detection Algorithms.

X. Conclusion

As shown in this paper there is no perfect solution for a specific problem. All of the considered simulations were lacking some important features. The presented solution hopefully enables future research to deal with those shortcomings. Nevertheless the developed Track Generator is a good tool which can be used and extended to be even more useful.

Also, random domain generation might become a bigger factor for classical autonomous driving research and therefore there might be more options to be included to existing simulations.

For the future there also should be considered a different approach for simulations. A sometimes valid and often more adequate approach can be to create a specific simulation. Even if the work might seem unreasonable in the beginning, being able to fit a simulation exactly to specific demands can pay off in the long run. Specially if the requirements differ in such manner to the usual research.

Acknowledgment

The author would like to thank Prof. Dr. Göhner, Bonifaz Stuhr and Johann Haselberger for creating and supervising this project.

References

[1] N. Kalra and S. M. Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? Santa Monica, CA: RAND Corporation, 2016.
[2] T. Sulkowski; P. Bugiel and J. Izydorczyk, 'In Search of the Ultimate Autonomous Driving Simulator', 2018.
[3] J. Tobin, R Fong, A. Ray, J. Schneider, W. Zaremba and P. Abbeel. Domain Randomization for Transferring Deep Neural Networks form Simulations to the Real World, 2017.
[4] S. Pouyanfar, M. Saleem, N. George, S. Chen. ROADS: Randomization for Obstacle Avoidance and Driving in Simulation, 2019.
[5] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman and J. Tang and W. Zaremba. OpenAI Gym, 2016.
[6] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. Proceedings of the 1st Annual Conference on Robot Learning, pages 116, 2017.
[7] Carla Simulator. How to make a new map with RoadRunner, 2020
[8] G.Rong1, B. Shin1, H. Tabatabaee1, Q. Lu1, S. Lemke1, E. Boise1, G. Uhm1, M. Gerow1, S. Mehta1, E. Agafonov1, T. Kim1, E. Sterner1, K. Ushiroda1, M. Reyes1, D. Zelenkovsky1 and S. Kim1. LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving, 2020.
[9] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim, High-fidelity visual and physical simulation for autonomous vehicles. Field and Service Robotics, 2017.
[10] F. Behrens. Procedural race track generation for domain randomization, 2020.