Self Driving Simulation for Autonomous Formula Technion Project

Submitted by: Tom Hirshberg, Dean Zadok

Advisors: Boaz Sternfeld, Dr. Kira Radinsky

Center for Intelligent Systems Laboratory, CS, Technion

January 2018

Background

The formula student association consist of approximately 600 universities from all around the world. Each university is required, under strict regulations, to design and build a formula race car in one year. At the end of the year, the team can participate in a various competitions around the world.

In the last year, a new competition has been raised, in which the students are required to take one of their formula cars from previous competitions, and make it driverless. So, the car will have the ability to finish the race track without driver and fully autonomous.

The selected formula car for the driverless competition.

Left - The real car. Right - The graphic model.

Abstract

Establishing the driverless formula students car is a big project. Creating an algorithm to perform a smooth and fast self driving is a part of it.

Developing and training AI based algorithms takes a lot of driving hours. Therefore, we need a simulation to be able to train this kind of algorithms.

We took a simulation project which was built for usual autonomous driving, and made it ideal for Technion Formula Student Team.

Our goal was to design the simulation as real as we could.

The driving mission

The race track lines are made of traffic cones. This is to ensure that the car will stay steady while maneuvering between tough turns.

The highest score is granted to the team which will finish with the best time.

Driving in Formula Student

The simulation

Our simulation is based on an open source project called AirSim, made by Microsoft.

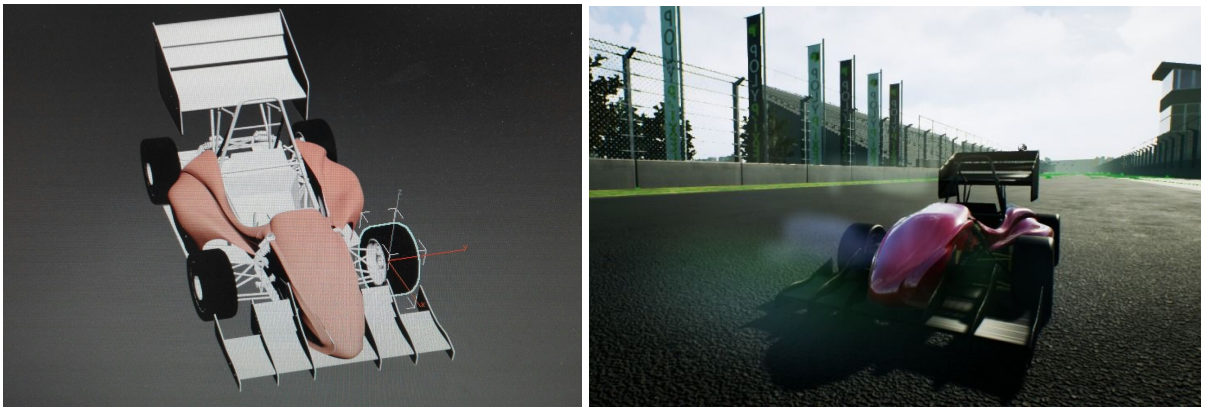
The project is built on the Unreal Engine 4, a professional graphics engine that mostly serve video games developers.

By implementing race track environment, we were able to turn AirSim into an ideal simulation for Technion Formula Student Team.

The graphic model of the car

The simulation is built to serve the mission of training a self driving algorithm using different deep learning methods.

In order to simulate the physics and dynamics of the formula car, we inserted the graphic model of the car into the simulation, and imitated the real dynamics.



Left - the model in 3DS Max. Right - the model in UE4

We added the complete graphic model in several steps:

1. Adjusting the model:

   We used the exact model which the real car components were built from. The model was built in SolidWorks and converted to a Rhino project.

   To make it more simple to work with, we converted the project to a 3DS Max 2018 project.

   In 3DS Max, we merged components into layers: car body, and four wheels.

   We adjusted the directions of each layer, so that every component was straightened to the x-direction in world and local space.

   We also adjusted the size of the components, to make the import to UE4 more simple and straight.

2. Adding the model to UE4:

   To simulate the dynamics of the car, we made a hierarchy on the layers:

   The main part was the car body, and the four wheels were sons.

   We imported the model as a skeletal mesh to UE4.

3. Collision and mass:

   We adjusted the mass of each layer, and added collision as follows:

   Car body - single convex hull.

   Wheels - sphere.

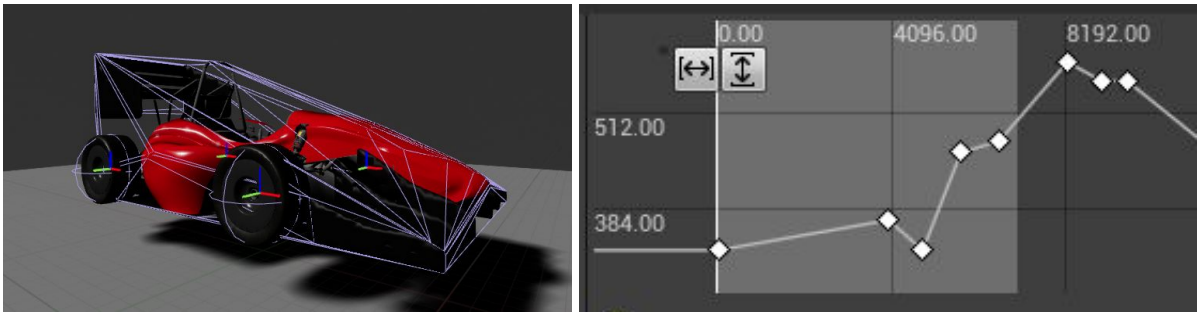
4. Materials and textures:

   Making a lot of conversions between programs can damage materials and textures. We decided to change the materials and textures after importing the model to UE4, so that we could adjust the RGB colors and also the light reflection.

The dynamic model of the car

By default, the car's dynamic model based on a usual private car, designed by nVidia and known as "nVidia PhysX Vehicle".

This dynamic model is based on a vehicle which is far more heavier and taller than our vehicle. Also, the components themselves are different than our components.



Left - car physics model. Right - torque graph

Therefore, in accordance to change our graphic model, we needed to change parameters related to the behaviour of the vehicle:

- Engine parameters - torque function, max rpm, manifolds rates and more.
- Gear parameters - ratios, switch time, latency and more.
- Dynamic parameters - differential setup, center of mass vector, suspension properties and more.
- Wheels parameters - dimensions, weights and tire friction parameters.
- Body weight and dimensions.

The results of changing these parameters expressed not only in a more realistic drive, but also in better grip, steady acceleration and faster response.

The traffic cones model

In order to create a formula student race track, we had to use traffic cones to mark the lines of the track.

The Technion Formula Student Team is heading to a competition, in which the track is built as follows:

The right line is marked by yellow traffic cones, and the left line is marked by blue traffic cones. The start line is marked by two orange traffic cones.

We created and designed the cones due to the competition regulations, which includes the dimensions, weights and colors.

We used 3DS Max to design our traffic cone model:

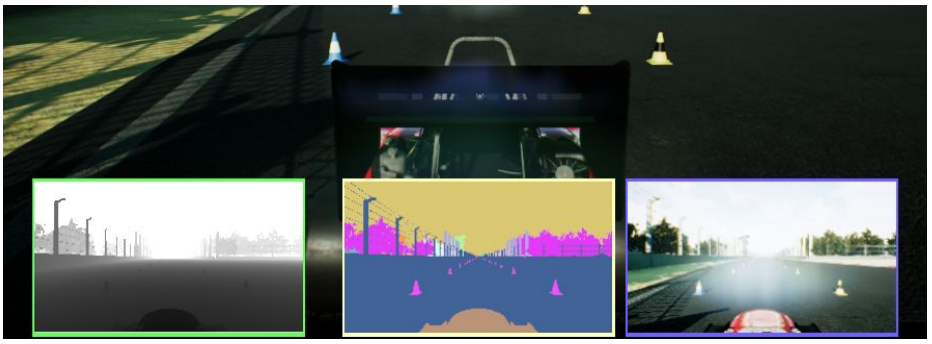
Left - Formula Student traffic cones in UE4. Right - modeling in 3DS Max.

Thanks to Unreal physics engine, we were able to imitate the realistic physics of the cones, so the cone will perform a realistic hit result in case of collision with the car.

Sensors and API using AirSim

In order to allow us to record and train self driving, AirSim simulates the following devices:

- Cameras - first person view, depth and segmentation cameras.
- Sensors - accelerometer, Gyro and GPS.
- Other statistics - speed, acceleration and engine RPM.



Camera views

In addition to these abilities, the simulation allows us to work with AI algorithms, using dedicated Python API, so we can train and test our algorithms on the simulation itself.

Track creation tool

When using learning algorithms that are based on visual recognition, it is important to use various tracks on the training part, in order to reduce overfitting.
We built a track creation tool in order to build tracks in a more simple, easy and fast way.

Building track with the track creation tool

The tool helps to build a track for the competition that the Formula Student Technion Team is heading to (track as we mentioned above).
The tool is made with a blueprint of a spline.
By only clicking on a point on the road, we can edit the track as follows:

The points set the right line of the track (yellow cones). The tool builds automatically parallel left line (blue cones) in a constant distance. Moreover, the tool places automatically each cone in each line in a constant distance from the previous one. The constant distances can be change even after placing the track.

Algorithms

The main purpose of this project is to build an environment to develop self driving algorithm on it. The more realistic the environment is, the more precise the algorithm can be after transferring it to a real car.

Testing a trained model using imitation learning

We are focusing on two different types of learning methods:

<u>Reinforcement Learning (Deep Q Learning):</u>

The reward function is using the environment to compute the reward values.

For example, we can use cones locations to compute the center of the track at a given timestamp, or detect collision with another obstacle and also compute the velocity of the car.

<u>Supervised Imitation Learning:</u>

In this approach, we are trying to train our algorithm to make driving decisions like a human driver.

The simulation allows us to record driving datasets from human drivers.

Then, we can take the dataset, train an algorithm with it and test the accuracy of the algorithm.

Future work

Weather

When using algorithms that are based on visual recognition, various weather conditions are important for the reliability of the algorithm.

One way to do this, which is what we're doing so far, is to synthesize the image when training the algorithms. For example, changing the colors relatively or removing unnecessary parts of the image.

Since landscape colors can confuse our algorithm prediction, we are planning to change cloud structure and density.

Furthermore, rain is also a possible condition. Not only that we need to simulate captured rainy images, we need also to simulate wet surfaces.

Lidar using AirSim

Capture distance from obstacles is an important mission in the autonomous driving world. depth cameras can do the job, but not precisely as Lidar sensors.

Lidar sensors are still under development in AirSim.

Once it will be done, we will work on simulating the exact Lidar sensor that we have on our car. This is to train an algorithm using Lidar data, and test it on the real vehicle.

Virtual Reality (Optional)

When recording human driving, wider view range can improve driving capabilities. This can improve the quality of the dataset for the imitation learning.

Bibliography

Unreal Engine 4 documentation - https://docs.unrealengine.com/latest/INT/

UE4 AnswerHub forum - https://answers.unrealengine.com/index.html

3DS Max support forum - https://forums.autodesk.com/t5/3ds-max-forum/bd-p/area-b200

3DS MAX Autodesk support - https://knowledge.autodesk.com/support/3ds-max

AirSim project git - https://github.com/Microsoft/AirSim

Nvidia PhysX vehicle documentation - http://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/apireference/files/group__vehicle.html

Facebook groups:

UE4 - https://www.facebook.com/groups/ue4devs/

UE4 Israel - https://www.facebook.com/groups/1225832467530667/

AirSim - https://www.facebook.com/groups/141498329323928/

Youtube (tutorials) - https://www.youtube.com/