TUM School of Computation, Information and Technology
Technical University of Munich

Semester Thesis

Autonomous driving simulator based on Unity3D

Autonomous Fahren Simulator Basiert Auf Unity3D

Supervisor Prof. Dr.-Ing. habil. Alois C. Knoll

Advisor Zhou Liguo, M.Sc.

Author Li Haichuan

Date March 10, 2023 in Munich

Abstract

Self-driving cars have the potential to revolutionize transportation, but they require extensive testing to ensure their safety and effectiveness. Real-world testing can be time-consuming and costly, and it also raises safety and regulatory concerns. Moreover, testing under extreme weather conditions and complex traffic scenarios is challenging to replicate in the real world. In response to these challenges, the industry has turned to automated driving simulation testing, which has become a widely adopted method for testing autonomous driving algorithms. Statistics show that the vast majority of autonomous driving algorithm tests, around 90%, are conducted through simulation platforms, while only 1% of tests are performed on actual roads. This trend underscores the importance of simulation testing as a vital tool for the development and optimization of self-driving cars.

Zusammenfassung

Selbstfahrende Autos haben das Potenzial, den Transport zu revolutionieren, aber sie erfordern umfangreiche Tests, um ihre Sicherheit und Effektivität zu gewährleisten. Real-World-Tests können zeitaufwändig und kostspielig sein und werfen auch Sicherheits- und Regulierungsbedenken auf. Darüber hinaus ist es schwierig, Tests unter extremen Wetterbedingungen und komplexen Verkehrsszenarien in der realen Welt zu replizieren. Als Reaktion auf diese Herausforderungen hat sich die Industrie dem automatisierten Fahrsimulationstest zugewandt, das zu einer weit verbreiteten Methode zum Testen autonomer Fahralgorithmen geworden ist. Statistiken zeigen, dass die überwiegende Mehrheit der Tests von autonomen Fahralgorithmen, etwa 90%, über Simulationsplattformen durchgeführt werden, während nur 1% der Tests auf realen Straßen durchgeführt werden. Dieser Trend unterstreicht die Bedeutung von Simulationstests als wichtiges Werkzeug für die Entwicklung und Optimierung selbstfahrender Autos.

Contents

Chapter 1

Overview Of Design

We offer a user-friendly interface, this system can accommodate various development needs and help researchers and engineers to efficiently design, test, and validate their autonomous driving algorithms.

To achieve high-quality environment perception simulation, this system integrates various types of sensors, such as lidar, radar, and camera, to accurately capture the surrounding environment of the virtual vehicle. The simulation platform also takes into account various factors that can affect the perception of the autonomous vehicle, such as lighting conditions, weather conditions, and road surface conditions, to create a realistic and challenging environment for testing.

By leveraging the Neurorobotics Platform's powerful extension capabilities, this system can seamlessly incorporate a range of artificial intelligence algorithms, such as deep neural networks and spiking neural networks, into the simulation environment. This allows researchers and engineers to test and validate their autonomous driving algorithms with a higher level of accuracy and realism.

Moreover, the system's efficient communication mechanism enables real-time information exchange between multiple targets, such as the virtual vehicle, traffic signals, and other road users, to simulate complex traffic scenarios and evaluate the performance of the autonomous driving algorithms under various conditions.

In summary, this system offers a comprehensive and flexible solution for developing and testing autonomous driving algorithms, with its advanced environment perception simulation, versatile artificial intelligence integration, and efficient communication mechanism. Its user-friendly interface and support for multiple programming languages make it an ideal platform for researchers and engineers to explore the potential of self-driving technology and drive innovation in the field and simplifying interface calls, the system has great advantages in ease of use and expansibility, and provides the basis and possibility for real simulation of real environments.

Chapter 2

Research Work

Collision avoidance is a critical component of autonomous driving systems, as it helps to ensure the safety of the vehicle and its passengers, as well as other road users. Here are some details on how collision avoidance works in autonomous driving:

Perception: Collision avoidance systems rely on sensors such as LiDAR, radar, and cameras to detect obstacles and other vehicles in the surrounding environment. The sensors collect data on the position, speed, and direction of objects, and this information is used to create a 3D map of the environment.

Decision-making: Once the sensor data has been collected and processed, the autonomous driving system needs to make decisions about how to avoid collisions. This involves analyzing the trajectory of the vehicle and other objects in the environment, and determining the best course of action to avoid a collision.

Control: Finally, the autonomous driving system needs to take action to avoid the collision. This can include slowing down or stopping the vehicle, changing lanes, or steering away from the obstacle.

There are several techniques and algorithms that can be used to implement collision avoidance in autonomous driving systems. These include:

Predictive modeling: By predicting the behavior of other road users, the autonomous driving system can anticipate potential collisions and take evasive action.

Decision trees: These are algorithms that use a series of binary decisions to determine the best course of action in different scenarios.

Reinforcement learning: This involves training the autonomous driving system through trial and error, so that it learns the best strategies for avoiding collisions in different situations.

2.1 Trajectory Prediction

Trajectory prediction is a key component of collision avoidance in autonomous driving. It involves predicting the future path of other vehicles and objects on the road to determine whether they are on a collision course with the autonomous vehicle, and taking appropriate actions to avoid a collision.

There are different approaches to trajectory prediction, but one common method is to use machine learning algorithms to analyze data from sensors such as LiDAR, radar, and cameras. By analyzing patterns in the movement of other vehicles and objects on the road, these algorithms can predict their future trajectory with a high degree of accuracy.

Once the trajectory of other vehicles and objects has been predicted, the autonomous driving system can use this information to plan its own trajectory and make decisions about how to avoid a collision. This can include adjusting the vehicle's speed, changing lanes, or coming to a stop if necessary.

In addition to machine learning, there are other techniques that can be used for trajectory prediction, such as mathematical models and rule-based systems. However, machine learning is often preferred because it can adapt to changing driving conditions and is more accurate than other methods.

Overall, trajectory prediction is a critical component of collision avoidance in autonomous driving, and it plays a key role in ensuring the safety of passengers and other road users. By accurately predicting the trajectories of other vehicles and objects on the road, autonomous driving systems can make informed decisions about how to avoid collisions and operate safely in a variety of driving scenarios.

2.2 Our Method Based On Trajectory Prediction

Trajectory prediction is a crucial component of autonomous driving, especially for collision avoidance. Without accurate trajectory prediction, autonomous vehicles (AVs) would not be able to plan their future path and avoid potential collisions with other objects in the environment. To achieve this, the goal is to predict the future trajectory of the vehicle based on its current and past states, including its position on the X and Y axes and its yaw angle.

To obtain the necessary data for trajectory prediction, a model and dataset are required. The model must be able to accurately capture the relevant features of the environment, such as the position and velocity of other objects in the AV's vicinity. The dataset must be large enough to provide a diverse range of scenarios and driving conditions, ensuring that the model is robust and can handle a variety of real-world situations.

Once the model and dataset are in place, it becomes possible to calculate the velocity and acceleration that the AV needs for safe and efficient driving. By combining information about the vehicle's position and velocity, it is possible to calculate the acceleration required to control the motor force, just like human driving via an accelerograph. Additionally, the yaw angle can be used to adjust the driving wheel, providing greater control and stability during turns and maneuvers.

Of course, these processes are only the basic requirements for autonomous driving, and there are many other factors to consider, such as traffic laws, road conditions, and pedestrian safety. However, by accurately predicting the trajectory of the vehicle, AVs can navigate these challenges with ease and provide a safe, efficient, and comfortable driving experience for passengers. With continued research and development in this area, the future of autonomous driving looks bright.

In this project, we aim to solve this problem by proposing an algorithm that takes into account the advantages of convolutional neural networks (CNNs) in capturing regional features while establishing feature correlation between regions using variants of attention. We introduce a novel attention mechanism that enhances the correlation between features and allows the model to focus on the most relevant regions of the image.

Our proposed model achieves better performance in the test set of L5Kit compared to the other vision models. The average number of collisions is 19.4 per 10000 frames of driving distance, which greatly improves the success rate of collision avoidance. The results demonstrate the effectiveness of our approach in addressing the limitations of existing vision models and highlight the potential of incorporating attention mechanisms in future autonomous driving research.

2.3 Result

The tables of test results which are processed by four different network structures are shown in Tab. 2.1. Compared with other transformer-based and convolution-based counterparts, our model achieved better accuracy and faster processing speed. In particular, our model achieves 2.6 times on front collision which is 13.6 times less than RepVGG, 5.8 times less than ViT, and 12.6 times less than ResNet50, indicating the benefit of SSN block for capturing both local and global information. We can see that SSN consistently outperforms other models by a large margin.

Table 2.1: Different collision times of four models per 10000 frames.

Method	Front	Side	Rear
L5Kit (ResNet-50)	15.2	20.7	8.3
RepVGG	16.2	11.7	10.6
Vit	8.4	**7.7**	9.2
SSN (Ours)	**2.6**	13.3	**3.5**

2.4 Feature Work

To integrate our algorithm model with the Unity virtual world, we need to establish communication between them. We accomplish this by using Unity's built-in API and message-passing techniques, a first-person view, and bird eye view figure shown in Fig. 2.1 . The communication protocol is established such that the algorithm sends the input information to the virtual world, and the virtual world returns the output information to the algorithm.

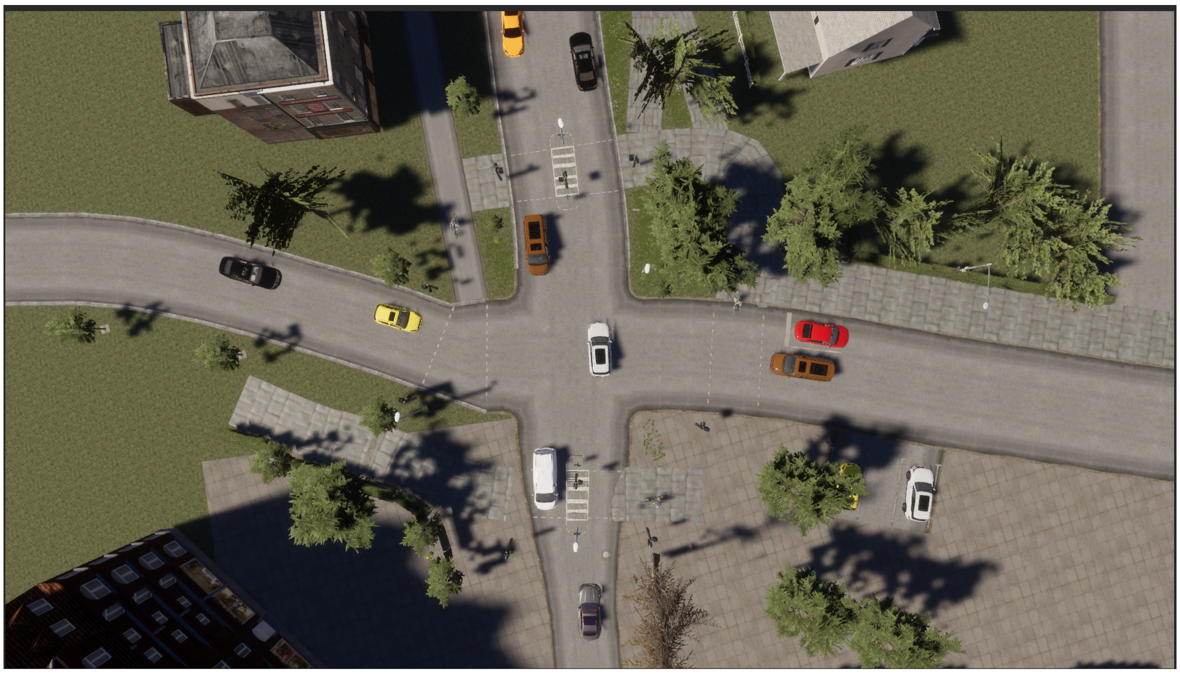
To ensure accurate physical modeling and simulation, we need to perform coordinate transformation between the image coordinates and the world coordinates. The process involves mapping the image coordinates to the virtual world coordinates using the camera parameters and the intrinsic and extrinsic calibration matrices. The resulting coordinates are then transformed to the physical world coordinates, where the physical model can calculate the necessary forces and physical parameters for the autonomous driving system.

This transformation can be mathematically represented as:

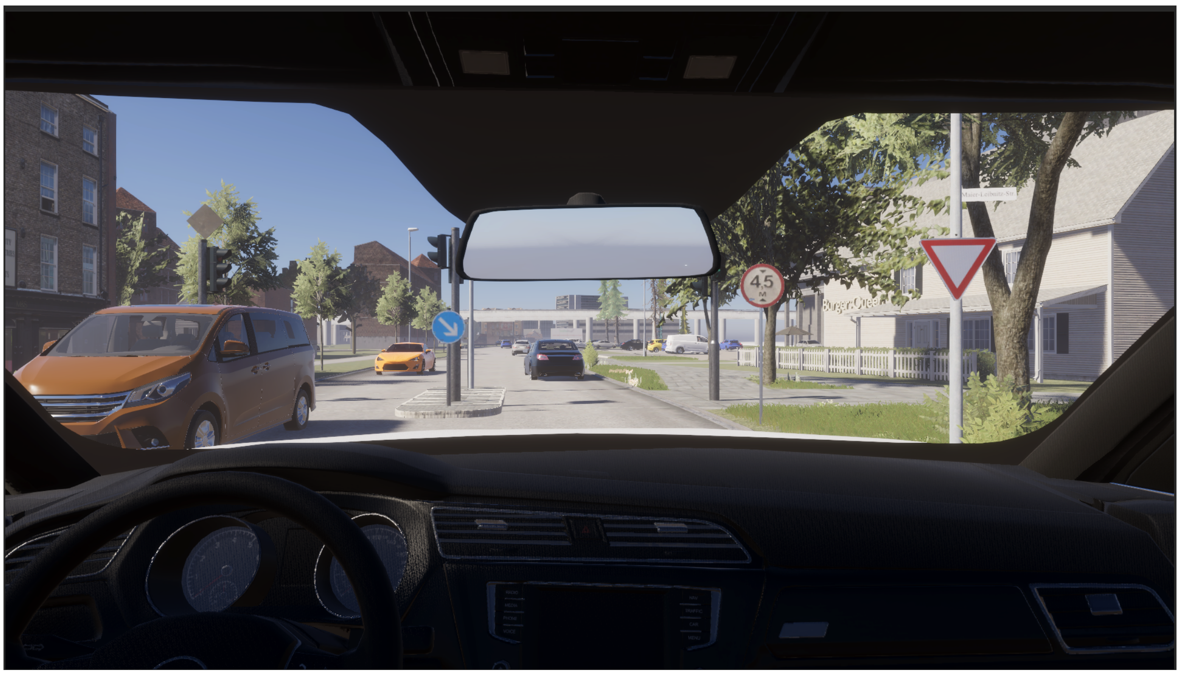
$$\begin{bmatrix} X_{world} \\ Y_{world} \\ Z_{world} \end{bmatrix} = R \begin{bmatrix} X_{camera} \\ Y_{camera} \\ Z_{camera} \end{bmatrix} + T \tag{2.1}$$

where **R** is the rotation matrix, **T** is the translation vector, and $[X_{camera}, Y_{camera}, Z_{camera}]$ are the coordinates in the camera frame.

To obtain accurate values for **R** and **T**, we need to perform calibration using various methods such as checkerboard pattern or laser scanning techniques [**zhang2018evaluation**]. The resulting calibration parameters can then be used to transform the image coordinates to world coordinates for accurate physical modeling and simulation



(a) Bird's eye view of virtual city



(b) First person view of virtual city

Figure 2.1: Autonomous driving vehicle in Unity virtual city

Chapter 3

Build Up Autonomous Driving system

Creating a virtual environment: Unity3D provides a powerful environment for creating virtual worlds. To develop an autonomous driving system, the first step is to create a virtual environment that simulates real-world driving scenarios. This can include roads, traffic signals, pedestrians, and other vehicles.

Adding sensors: Autonomous driving systems rely on sensors to perceive the environment around them. In Unity3D, sensors such as LiDAR, radar, and cameras can be added to the virtual vehicle to simulate the perception capabilities of a real autonomous vehicle.

Building the driving algorithm: Once the virtual environment and sensors are in place, the next step is to develop the driving algorithm. This algorithm should be able to take input from the sensors, make decisions based on that input, and control the virtual vehicle accordingly.

Testing and validation: With the driving algorithm in place, the virtual environment and the sensors can be used to test and validate the performance of the algorithm. Different scenarios can be created to simulate different driving conditions, and the algorithm can be fine-tuned and optimized based on the results.

Integration with real-world hardware: Once the algorithm has been tested and validated in the virtual environment, it can be integrated with real-world hardware. For example, the algorithm can be tested on a physical autonomous vehicle to ensure that it performs as expected in the real world.

Appendix A

Install Unity3D

Go to the Unity3D website and click on the "Get Started" button. Select the Unity version you want to download from the list of available versions.

Select the platform you want to install Unity3D on. Unity3D supports Windows, Mac, and Linux.

Select the components you want to install. Unity3D offers different components, such as documentation, standard assets, and examples. You can select or deselect components based on your needs.

Choose the installation location where you want to install Unity3D. You can choose the default location or select a different one. Click the "Download" button to start downloading Unity3D.

Once the download is complete, open the installer and select the language you want to use for the installation.

Accept the terms and conditions and click on the "Next" button. Choose the installation options you want, such as creating a desktop shortcut or adding Unity3D to the system path.

Click on the "Install" button to start the installation process. Wait for the installation to complete. This may take several minutes depending on the components you selected and your system configuration.

Once the installation is complete, click on the "Finish" button to launch Unity3D.