Autonomous Vehicle Simulation Project

Sangho Lee¹, Janghee Cho² and Da Young Ju¹
¹Yonsei University
²Digipen Institute of Technology
sangholee@yonsei.ac.kr, roycho1111@gmail.com, dyju@yonsei.ac.kr

Abstract

As the vehicle technology grows rapidly in these days, the factors of practicable tests are increasing. However, it can be exposed to critical conditions to perform field tests with unverified implementations, since when error occurs while driving, directly leads to an unpredictable accident. For this reason, there are various simulators in the market to ensure the safety and provide various kinds of results, yet they are only targeting high value industries with their remarkable expense, which makes private enterprises challenging to purchase. Therefore, in this paper, autonomous vehicle simulation software based on an open-source project is proposed to provide an extensible platform with public license to perform tests in the virtual world.

Keywords: vehicle simulation, automotive vehicle, carsim, simulation software

1. Introduction

Recently, there have been significant improvements in vehicle technologies. Along with a development, various researches are still in progress to overcome existing obstacles. Autonomous vehicles are being tested under the real road conditions, and similar ongoing challenges such as DARPA [1] encourage development of advanced technologies. In detail, performing numerous experiments is crucial to adapt a new technology into the real world after buildup process. Throughout the field test, it can be analyzed to see the difference whether the method shows enhancement or not, compared to existing functions.

However, field test can be exposed to unexpected conducts while testing with unstable implementations. Once an error occurs in the program, it is critical since an error while driving directly turns out to unpredictable situations, such as serious accident and injury of the driver. For this reason, most experimental vehicles equip an emergency button to avoid
dangerous situations, as shown in Figure 1. However, passive controller like an emergency button is not helpful once a vehicle abruptly begins unexpected actions. Thus, a software simulation with real-world scale is recommended to get into an experiment with the vehicle. Nevertheless, software vehicle simulators such as Carsim [9] and Dynacar [10] are too expensive to acquire a license. Moreover, extensibility of the simulator is limited, so they only support the driving simulation with the vision and steering device. In this paper, we present the vehicle simulation software with maximization of flexibility using plug-in models based on open-source project. To propose our research, we compare our research with previous works. Then, essential components to simulate an autonomous vehicle are introduced. A structure of the autonomous vehicle simulator is presented with an actual project sequentially. As a result, further research plan with limitation and conclusion is claimed.

2. Previous Works

The needs for the vehicle simulation software have been constantly existed. On behalf of this, a number of companies have tried to develop software with many studies. As a research of vehicle simulation software, i3Drive [2] has been introduced several years ago. In this paper, the author proposed interactive simulation software of driving dynamics. However, input device does not support any other types than control device. Because an autonomous vehicle requires LIDAR and GPS systems [3], proposed research creates artificial LIDAR and GPS signals as essential components. Additionally, developers can add or modify original modules to their needs since our method is based on open-source project.

3. Essential Components of Autonomous Vehicles

Figure 2. Required Equipments of Autonomous Vehicle. Left is Differential GPS(DGPS), and the Right is LIDAR

There are required equipments to operate autonomous vehicles, such as LIDAR and GPS in Figure 2. Without essential devices, it is impossible to read a current state and make a sufficient decision in real-time. Existence of these components is crucial to an autonomous vehicle. In our research, we generate an artificial LIDAR signal based on the virtual world, along with artificial GPS signal which traces the drive path of the vehicle.

There are numerous types of devices required for the case of special experiments. In this case, a software simulator must support these devices, and it is almost impossible to analyze an entire simulator for adding specific components. We opened components as modules to developers to implement artificial components as they need based on an opened interface, and
attached to the core engine of the simulator. Module system [7] reduces efforts to understand whole structure of the software since developers only need to know how to link modules to the system.

4. Autonomous Vehicle Simulation Software

The autonomous vehicle simulation software consists of two parts: configuration manager and plug-in simulator. A relationship between configuration manager and plug-in simulator is described in Figure 3.

![Figure 3. A Relationship between Configuration Manager and Plug-in Simulator](image)

4.1. Configuration Manager

A proposed configuration manager specifies the properties, attributes, and the simulators to be used in third party simulator. Properties include various types of an action which the vehicle can take. For example, a breaking system is one of the properties since it is an action which the vehicle can choose to decrease the speed. Attributes compose the specific model of the property. For instance, linear and non-linear model of the breaking system are intended as attributes belong to the breaking system property, since they do not affect to an operation of the vehicle. However they provide different methods to achieve as well. Once configuration of the properties and attributes are done, the simulation section will actually transmit configurations to the plug-in simulator [8]. A configuration manager exports required properties and attributes to execute a simulation. Then, the plug-in application imports and loads a simulator depends on the characteristics of the plug-in application. Considering extensibility, each property in the configuration manager exists as library. In detail, one .xml header [6] with .dll library [5] is linked to one property, as described in Figure 4. Because attributes are a subset of the property, a library includes every attributes presented in the manager. Plug-in simulator is executed after library export process is completed. Preserving standard format to communicate between the manager and simulators is important for the developer of plug-in simulators. Thus, the file format and parameters are consistent when the configuration manager calls a simulator. After plug-in simulator is launched and loaded successfully, it provides a specific experimental environment to simulate the vehicle.
4.2. Plug-in Simulator

Plug-in simulator is an independent application which is developed as an isolated project. The data is exported from the configuration manager. In order to equip libraries to the simulator, the header is loaded first to recognize the libraries to read. Based on the header, the libraries are attached to the simulator. Once the simulator is executed, it loads configuration data to construct an environment. Then, input modules are attached to an application. An input module generates its unique message with specific signals, and transmits to the main simulator. For example, the module for autonomous driving algorithm must receive signals from input modules since it is attached to the main simulator to enable auto-driving. The input modules are exported and imported as .dll to maximize flexibility. Because there are shared data among the modules, the memory manager exists. Since it is located as a singleton object, the modules can access the data easily with the nickname, as described in Figure 4. Once loading stage is done, the simulator renders the world in 1st person perspective. Simulator of an autonomous vehicle begins driving after the loading procedure is completed. Input devices, such as keyboard and mouse are discarded once an auto-driving algorithm is attached. In this case, control signals for an autonomous vehicle, like an angle of the steering and pressure of an acceleration pedal is returned to the simulator. Finally, the simulator renders the vehicle with an environment using positions in the world space [4].
5. Development

To implement the research proposed in this paper, we implemented the prototype project called OpenCarSim. OpenCarSim contains two projects – OpenCarSimManager (Configuration Manager), and OpenCarSimRenderer (Plug-in Simulator). As described in 4.1, OpenCarSimManager is the root application with a child application, OpenCarSimRenderer.

5.1. License Policy

Since the project accepts open source model, the whole implementation is stored and updated through the public SVN[11] repository as shown in Figure 6.
We employ General Public License (GPL) for the policy of the project. In accordance to the GPL policy, anyone who re-creates the software package to simulate an autonomous vehicle based on an implementation of OpenCarSim must release their source codes with GPL as well. An open source license policy model leads rapid propagation of the project with great improvements. For example, cocos2d-x[12] is an open source 2D graphics rendering library. Since the release of the project, it becomes one of the most popular 2D rendering frameworks in the world. Likewise, OpenCarSim expects to be widely used with improvement.

5.2. OpenCarSimManager

OpenCarSimManager is a configuration manager of OpenCarSim. To implement this, we used C# with internal XMLParser. As mentioned in 4.1, there are properties, attributes, and simulators in user interface. As shown in Figure 7, they can be added by clicking the button “Add Property module”. Once configuration process is successfully done, a plug-in simulator can be loaded. On behalf of the project, we created OpenCarSimRenderer to render a simulation result visually in real-time.

![Figure 7. A Screenshot of OpenCarSimManager](image)

5.3. OpenCarSimRenderer

For the presentation of the prototype, a visualized simulator, OpenCarSimRender is developed as a child application of OpenCarSimManager. Since OpenCarSimRenderer is a child application, it cannot be executed as stand-alone package. Thus, OpenCarSimManager must create a child process with a configuration file of OpenCarSimRenderer to transmit required data to the renderer.

OpenCarSimRenderer is formulated with two layers. In Figure 3, middle layer is the main renderer, and the bottom is input modules. In this project, we have artificial LIDAR and GPS module, as provided in Figure 8. To implement the main world, a plane with the road texture is installed. Then, the vehicle is located onto the road. The camera is installed and follows the vehicle. For the background, the constant background image is displayed after projection process is done [4]. Due to the difficulties to generate an artificial artifact for LIDAR,
wireframe is rendered instead of displaying a real N-channel artificial LIDAR. GPS location is determined by the position of the vehicle object in the world space. To visually see a GPS plot mapped with the geometry, a top-view camera is installed, as shown in Figure 8. Once the camera focuses on the bottom, it appears as an artificial GPS coordination map.

![Screenshot of the Prototype Simulator Result. OpenCarSimRenderer(bottom right) with Artificial LIDAR(top left) and GPS(top right) Modules](image)

Since an auto-driving module is attached, the vehicle is driven automatically. In detail, the module will send signals including actions how to operate the vehicle. For example, the vehicle will press brake once a received signal is to-brake. If this module is not specified from the OpenCarSimManager, the main controller will be a keyboard.
6. Limitations and Conclusion

This paper presents a step to simulate an autonomous vehicle control algorithm in the virtual space with the real world scale. So far, we did not perform comparisons of the simulator and the real world since the plotting method is not developed. Additionally, an accurate physics module for propulsion, braking, and steering systems is needed to be developed by professionals in advance. We are continuing to research and develop OpenCarSim project. We focus on improving flexibility of the modules in the configuration manager as well as plug-in simulators. After the development of this project, we are planning to support various platforms for an integration of simulated algorithms to the vehicle to extend our flexibility. Finally, anyone who wants to share our passions of the simulator may participate further as long as this project is an open-source based project with GPL license policy.

Acknowledgements

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the “IT Consilience Creative Program” (NIPA-2013-H0203-13-1002) supervised by the NIPA (National IT Industry Promotion Agency).

References

Authors

**Sangho Lee**
2012: School of Integrated School at Yonsei University
2007-2012: BS, Computer Science at Digipen Institute of Technology

**Janghee Cho**
2009: BS, Computer Science at Digipen Institute of Technology

**Da Young Ju**
2012: Assistant Professor, Yonsei University, Korea
2002-2011: PhD, Media Technology, Sogang University, Korea
2007-2008: MA, Digital Arts, University of the Arts London, UK
2000-2002: MS, Media Technology, Sogang University, Korea
1995-2000: BFA, Painting & Design, Hongik University, Korea