A Simulation Method for LIDAR of Autonomous Cars

Hu Su¹,a, Rui Wang²,b, Kaixin Chen³,c, and Yizhan Chen⁴,d

¹School of Electrical Engineering, Southwest Jiaotong University
²School of Electrical Engineering, Southwest Jiaotong University
³School of Electrical Engineering, Southwest Jiaotong University
⁴School of Electrical Engineering, Southwest Jiaotong University
asu@swjtu.edu.cn, b17828183639@163.com, cchrisch@foxmail.com, dchenyizhan@hotmail.com

Abstract. Vehicle-mounted LIDAR simulation module is an important part of autonomous car simulation test platform. According to the working principle of LIDAR, its working process and interface can be simulated in the digital simulation environment. The environment includes LIDAR emulation module, high precision 3D scene model and data interface module. The LIDAR simulation module uses GPU(Graphic Processing Unit) to calculate the depth of high-precision 3D scene and output the results through data interface. On the PC workstation, 64 lines vehicle-mounted LIDAR can be implemented for real-time simulation of large-scale road scene.

1. Introduction
The autonomous car simulation test platform can greatly reduce the road test mileage, test cost and risk by building a preset virtual traffic environment, and can realize the test under complex road conditions or traffic environment, which is of great significance to the autonomous car research and development. At present, the main autonomous car manufacturers, while carrying out road tests, have carried out simulation tests several orders of magnitude higher than the actual road test. Different from the conventional vehicle simulator, the autonomous car simulation platform not only needs to simulate the vehicle dynamic performance, but also needs to simulate various types of sensing layer devices or subsystems that are unique to autonomous car. Among them, the most important is the simulation of vehicle-mounted LIDAR. This paper introduces a simulation method of vehicle-mounted LIDAR. The GPU is used to perform depth calculation on the high-precision traffic environment scene, and the obtained result is output through the simulated LIDAR interface.

2. Related Work
Before it was applied to autonomous cars, LIDAR was mainly used in aviation, aerospace, guidance and other fields. The problem of LIDAR simulation in these areas has been studied. Reference[1] introduces the simulation method of airborne LIDAR, and uses OpenGL to simulate the LIDAR imaging process(Niklas Peinecke et al.,2008). In reference[2], the area method is used to judge the visibility of LIDAR imaging for complex space objects(Wang Ying et al.,2016). In reference[3], the target model and noise model of imaging LIDAR are established and used to guidance LIDAR(Yang Bing et al.,2007). Other applications include simulation of spaceborne or airborne LIDAR for land cover analysis, etc(Wang HongZhang, Valeriano, Angela, BrettBrowning et al.). [4][5][6][7].

In recent years, with the application of LIDAR in autonomous car, the simulation problem of vehicle-mounted LIDAR has been studied. In reference[8], the LIDAR scanning characteristics of roadside vegetation are studied, and a simulation method is proposed(Jean-Emmanuel et al.,2012). In
reference[9], the interaction between the LIDAR and the terrain of UGV (unmanned ground vehicle) is simulated (Seongjo Lee et al., 2015).

3. Algorithm Flow
The basic method for implementing LIDAR simulation in this paper is: establishing a high-precision traffic scene model and using GPU to render the model and get depth data to implement LIDAR simulation. An open source 3D engine named OSG (Open Scene Graph) is employed to load and manage the scene model. In a simulation loop, the position of the virtual LIDAR is updated based on car’s position firstly. According to the position of virtual LIDAR, the relative position relationships between objects in the scene model and LIDAR are calculated by using GPU. Output data are selected based on the LIDAR scan characteristic. The overall flowchart of the algorithm is shown below.

![Flow chart of the algorithm](image)

Figure1. Flow chart of the algorithm

High-precision traffic scene models can be stored in a common 3D file format. Since the scene model is only used for the calculation of depth values, textures, lighting materials, normal vector are not required. To improve computational efficiency, the scene model needs to be stored in a tree structure. According to the characteristics of road scene, nested quad-tree is used to organize the scene model. Considering the detection range of the LIDAR, the scene space corresponding to the last-level leaf nodes is an area of 240 meters in length and width. The structure diagram is as follows. With such a structure, it is possible to quickly eliminate scene model nodes outside the detection range of the LIDAR during cull traversal, and improve computational efficiency.
4. Rendering Algorithm

Before starting the simulation, the traffic scene model file is loaded into the computer memory by the 3D graphics engine OSG. An OSG::Camera object is created to manage the status data of the virtual LIDAR’s position and posture. The LIDAR simulation model is loaded according to the model and parameters of the simulated vehicle-mounted LIDAR. Parameters such as resolution, scanning frequency and output data format of LIDAR need to be considered.

In a simulation loop, current posture of simulated autonomous car’s will be got firstly. These parameters include XYZ 3D coordinates of the virtual vehicle centroid, velocity, acceleration, pitch angle, yaw angle, roll angle, angular velocity and angular acceleration and so on. Based on these data and the preset installation parameters of the vehicle-mounted LIDAR, the posture of the simulated LIDAR in the global coordinate system of the test scene is calculated.

Figure 2. Structure of scene model

Figure 3. Coordinate transformation of LIDAR
As shown above, the coordinates of the LiDAR in the world coordinate system \((X_L, Y_L, Z_L)\) are as follows:

\[
\begin{bmatrix}
X_L \\
Y_L \\
Z_L
\end{bmatrix} = \begin{bmatrix} X_V \\
Y_V \\
Z_V \end{bmatrix} + \begin{bmatrix} X_{\text{Offset}} \\
Y_{\text{Offset}} \\
Z_{\text{Offset}} \end{bmatrix}
\] (1)

\((X_L, Y_L, Z_L)\) is the coordinates of the LiDAR in the world coordinate system;

\((X_V, Y_V, Z_V)\) is the coordinates of the car's centroid in the world coordinate system;

\((X_{\text{Offset}}, Y_{\text{Offset}}, Z_{\text{Offset}})\) is the coordinates of the LiDAR in the vehicle coordinate system in which the vehicle centroid is the origin.

The parameters of the OSG::Camera object are set according to the parameters of the simulated LiDAR. After that, the relative positional relationship between objects in test scene and the simulated LiDAR can be calculated. This calculation is done by the GPU, so Shader programming is required. In OSG, the world space coordinates of a vertex is transformed by the formula 2 in the Vertex Shader to obtain the clip coordinates (also called homogeneous coordinates) which will be used for perspective division. The calculation method is as follows.

\[
V_{\text{Clip}} = V_{\text{World}} \cdot M_{\text{View}} \cdot M_{\text{Projection}} = V_{\text{Local}} \cdot M_{\text{Model}} \cdot M_{\text{View}} \cdot M_{\text{Projection}}
\] (2)

\(V_{\text{Clip}}\) is the calculated clipping coordinates;

\(V_{\text{World}}\) is the coordinates of the world space;

\(V_{\text{Local}}\) is the coordinates of the model vertex in the model local space;

\(M_{\text{Model}}\) is the transformation matrix of the model in world space;

\(M_{\text{View}}\) is the transformation matrix of the camera in world space;

\(M_{\text{Projection}}\) is the projection matrix of the perspective projection.

After perspective division, the clip coordinates are mapped into the screen space. Objects which are invisible to the simulated LiDAR are culled and visual objects are mapped to the screen space. Therefore, the position in the world coordinate system corresponding to each pixel in the screen space is the point that the LiDAR may collect in each scan. However, due to the limited resolution of the LiDAR scanning, these pixels need to be effectively screened. The specific method is: performing inverse transformation on the coordinates in the clipping space according to Equation 3 to obtain world space coordinates corresponding to the pixels.

\[
V_{\text{World}} = V_{\text{local}} \cdot M_{\text{Model}} = V_{\text{Clip}} \cdot M_{\text{Projection}}^{-1} \cdot M_{\text{View}}^{-1}
\] (3)

Next, each pixel needs to be screened in the Fragment Shader, and the eligible points will be output. Let the number of lines of the LiDAR be \(n\), the vertical scanning range FOV be \(\alpha\), the maximum scan range be \(R_{\text{max}}\), and the vertical resolution be \(\gamma\), angle between the conical surface and the horizontal plane be \(\theta_n\) (when the conical surface is above the horizontal plane, \(\theta_n < 0\), \(n = 1, 2, 3...\)). Each laser of the LiDAR will scan along a conical surface whose conical point is the location of the LiDAR, the length of the generatrix is \(R_{\text{max}}\), apex angle is \(90° - |\theta_n|\). Thus, as shown in Fig. 4., the coordinate of the LiDAR position is \((x_0, y_0, z_0)\), LiDAR scanning deviation is \(\sigma\), when the point at coordinates \((x, y, z)\) satisfies the formula 4, the point can be scanned by the LiDAR. The coordinates of points that meet the condition will be recorded and output as the final result of the LiDAR scan.

\[
(x - x_0)^2 + (y - y_0)^2 - \frac{(z - z_0)^2}{\tan^2 \theta_n} < \sigma
\] (4)

The Fragment Shader performs parallel operations for each pixel of the screen, which is highly computationally efficient.
All calculated scan points form the simulated point cloud data. These point cloud data are stored in the texture object of the OSG. These data need to be written into the frame buffer and transferred to main memory.

The obtained virtual point cloud data is the coordinate and laser ID data. The output format of the position information is as follows:

### Table 1. Output data format

| Name     | X    | Y    | Z    | ID   |
|----------|------|------|------|------|
| Definition | x-coordinate | y-coordinate | z-coordinate | Laser ID |

5. Experimental Results

In order to experimentally verify the above method, a high-precision road scene model for testing was established. The road scene model has a range of approximately 6.5 $km \times 3.2 km$. The scene contains 45,917,504 vertices, 15,314,146 elements. And the number of final group nodes is 362. Main hardware parameters of test platform are: CPU is Intel I7 3.4GHz, graphics card is GTX1080 8GB, main memory is DDR4 16GB. The simulation object is Velodyne's mechanical 64-line LIDAR which model is HDL-64E S3. This LIDAR has an up angle of $2^\circ$ and a down angle of $25.8^\circ$. The measurement range is 0.9 to 120 meters. When the LIDAR’s scanning frame rate is 5 $Hz$, the corresponding angular resolution is 0.0864$^\circ$, which means that it needs to emit 4167 lasers per scan. Figure 5 gives the simulation results.

The measured distance is rendered in pseudo-color. Figure 5-a & 5-b show the snapshot of front view and its corresponding simulated point cloud image. Figure 5-c shows the simulation result of one time scan. Figure 5-d shows the point cloud reconstruction scene by multiple times scans. Figure 6 gives the time consumption curve of LIDAR simulation.
As can be seen from the above figure, the average Cull time in one loop during the simulation calculation is about 1.43 ms, the average Draw time is about 14.89 ms, the GPU calculation time is about 13.55 ms. The overall time consumption of simulating one time scan is about 28-35 ms. The scanning frequency of the Velodyne 64-line LIDAR is 5 Hz. It can be seen that this method can meet the real-time requirements of simulation of this type of LIDAR.
Table 2. Calculation Error

| Actual point coordinates (X,Y,Z) | Simulated point coordinates (X,Y,Z) | Absolute calculation error (X,Y,Z) |
|----------------------------------|-----------------------------------|-----------------------------------|
| (40.000000, 0.1500000, -0.1500000) | (40.000000, 0.1499990, -0.1500000) | (0.0000000, 0.0000010, 0.0000000) |
| (40.000000, 0.0500000, -0.1500000) | (40.000000, 0.0500000, -0.1500000) | (0.0000000, 0.0000000, 0.0000000) |
| (40.000000, -0.0500000, -0.1500000) | (40.000000, -0.0500000, -0.1500000) | (0.0000000, 0.0000000, 0.0000000) |
| (40.000000, -0.1500000, -0.1500000) | (40.000000, -0.1500000, -0.1500000) | (0.0000000, 0.0000000, 0.0000000) |
| (40.000000, -0.2500000, -0.1500000) | (40.000000, -0.2500000, -0.1500000) | (0.0000000, 0.0000000, 0.0000000) |
| (40.000000, 0.1500000, -0.0500000) | (40.000000, 0.1500000, -0.0499999) | (0.0000000, 0.0000001, 0.0000000) |
| (40.000000, 0.0500000, -0.0500000) | (40.000000, 0.0499992, -0.0499999) | (0.0000000, 0.0000001, 0.0000000) |
| (40.000000, -0.0500000, -0.0500000) | (40.000000, -0.0499999, -0.0499999) | (0.0000000, 0.0000000, 0.0000000) |
| (40.000000, -0.1500000, -0.0500000) | (40.000000, -0.1500000, -0.0499999) | (0.0000000, 0.0000000, 0.0000000) |
| (40.000000, -0.2500000, -0.0500000) | (40.000000, -0.2500000, -0.0499999) | (0.0000000, 0.0000000, 0.0000000) |

Table 2 gives the calculation errors during the simulation. It can be seen from the table that the LIDAR simulation based on the high-precision scene model has a small calculation error in the simulation process. The main factor affecting the accuracy of LIDAR simulation is the modeling accuracy of the scene.
6. Conclusion
This paper presents a method for digital simulation of vehicle-mounted LIDAR using computer graphics. Use GPU to perform depth calculations on high-precision traffic environment scenes. Experiments show that this method can effectively simulate the working characteristics of LIDAR in real time. In the follow-up work, the simulation of LIDAR in different weather conditions will be studied to meet the all-weather digital testing requirements of autonomous cars.

7. References
[1] Niklas Peinecke, Thomas Lueken, Bernd R. Korn. LIDAR Simulation using Graphics hardware acceleration. Published in: 2008 IEEE/AIAA 27th Digital Avionics Systems Conference (09 December 2008) St. Paul, MN, USA DOI: 10.1109/DASC.2008.4702838
[2] Wang Ying, Huang Jianming, Liu Yu, Chen Feng, Wei Xiangquan. Simulation of LIDAR imaging for space target. Infrared and Laser Engineering (25 Sept. 2016) vol 45 no. 9 (0906002, 6 pp) Journal Paper DOI: 10.3788/IRLA201645.0906002
[3] Yang Bing, Yang Xiaoli, Zhang Wei. Computer Simulation of Imaging Laser Radar. Proceedings of the 2007 conference on communication and information technology for academic exchange of university graduate students in Beijing area (2008-1-1) China conference
[4] Wang Hong, Zhang Zhiyu, Zhou Hui, Li Song, Ma Yue, Zhang Wenhao. Simulation of spaceborne large spot LIDAR echo for forest vegetation. Geomatics and information Science of Wuhan University (May 2018) Vol 43 No 5 DOI: 10.13203/j.whugis20160375
[5] Valeriano Mendez, Heliodoro Catalan, Joan R. Rosell-Polo, Jaume Arno, Ricardo Sanz. LIDAR simulation in modelled orchards to optimise the use of terrestrial laser scanners and derived vegetative measures. Biosystems Engineering, (May 2013) v 115, n 1, p 7-79 Journal article (JA)
[6] Angela M. Kim and Richard C. Olsen. Simulated LIDAR waveforms for the analysis of light propagation through a tree canopy. American Society for Photogrammetry and Remote Sensing Annual Conference 2011, p 249-258, 2011, American Society for Photogrammetry and Remote Sensing Annual Conference 2011. Conference article (CA)
[7] Brett Browning, Jean-Emmanuel Deschaud, David Prasser and Peter Rander. 3D Mapping for high-fidelity unmanned ground vehicle LIDAR simulation. International Journal of Robotics Research, v 31, n 12, p 1349-1376, October 2012; ISSN: 02783649, E-ISSN: 17413176; DOI: 10.1177/0278364912460288; Publisher: SAGE Publications Inc.
[8] Jean-Emmanuel Deschaud, David Prasser, M. Freddie Dias, Brett Browning, Peter Rander. Automatic Data Driven Vegetation Modeling for LIDAR Simulation. Published in: 2012 IEEE International Conference on Robotics and Automation (28 June 2012). Saint Paul, MN, USA DOI: 10.1109/ICRA.2012.6225269
[9] Seongjo Lee, Dahyeon Kang, Seoungjae Cho, Sungdae Sim, Yong Woon Park, Kyhyun Um and Kyungun Cho. LIDAR simulation method for low-cost repetitive validation. Advances in Computer Science and Ubiquitous Computing. Lecture Notes in Electrical Engineering, vol 373. Springer, Singapore (18 December 2015) ISBN: 978-981-10-0280-9 DOI: https://doi.org/10.1007/978-981-10-0281-6_34