Proceedings of the Fifth International Conference on Railway Technology: Research, Development and Maintenance, Montpellier, France, August 22-25, 2022.

Graphic Simulation Framework of Railway Scenarios for LiDAR Dataset Generation

Gianluca D'Amico[1], Mauro Marinoni[1], Federico Nesti[1], Salvatore Sabina[2], Gianluigi Lauro[2], Giorgio Buttazzo[1]

[1]**Department of Excellence in Robotics & AI**
Scuola Superiore Sant'Anna, Pisa, Italy

[2]**Hitachi Rail STS, Italy**

Abstract

Guaranteeing safety and efficient traffic management in the railway network requires the correct execution of crucial and challenging tasks, such as localization, object detection, and obstacle avoidance. An increasing number of solutions are exploring the use of visual sensors to enhance accuracy without requiring a significant support at the infrastructure level. This paper proposes a simulation framework to generate LiDAR data for testing and validating novel algorithms in a railway scenario, where gathering these kinds of data in a real setting is impractical and time-consuming. Given a train trajectory, the framework exploits a graphic engine to generate the railway infrastructures in the surrounding area of the rail tracks, populating the virtual world with environmental objects. The point cloud of the LiDAR is generated through the ray-casting system of the graphic engine, taking into account the radiometric nature of the sensor, including the backscattered intensity for each point of the frame computed with the Lambertian–Beckmann model. Moreover, each LiDAR point comes with a semantic label that can be used to generate datasets for training and testing deep neural networks for object detection or segmentation tasks. The experiment results show the reliability of the LiDAR simulation to reproduce the sensor behaviour both in the distance measurements of the point cloud and in the backscattered intensity model.

Keywords: train simulation framework, lidar, visual dataset, backscattered intensity

1. Introduction

Train navigation systems play a crucial role in traffic management and security for estimating the position of the train along the railroad in real time. Currently, the train position is estimated based on balises, i.e., transponders placed along the railroad that, when detected, provides the absolute position of the train, and on wheel sensors, i.e., devices that read the angular velocity of the train wheels [1]. The speed acquired from the wheel sensor is integrated to compute the distance travelled by the train between two consecutive balises. However, wheel odometry suffers from drift error mainly caused by models inaccuracy and slip and slide phenomena. The drift accumulated during travelling reduces the accuracy of the train position from one balise to another. Moreover, balises have high deployment and maintenance costs, and are exposed to bad weather conditions and vandalism.

For these reasons, many sensors have been considered to increase the position estimation accuracy, such as inertial measurements units (IMUs), global navigation satellite system (GNSS), light detection and ranging (LiDAR), and camera, exploiting sensor fusion and deep learning methods to overcome the limitations of balises and wheel encoders [2]. Over the last decades, the scientific community primarily focused on the exploitation of visual sensors, creating accurate navigation systems and solving other tasks, such as object detection and tracking, obstacle avoidance, etc.

Despite of its higher cost, the LiDAR sensor allows overcoming several camera drawbacks, since it is less prone to weather conditions, is not affected by ambient light, and directly detects object distance without further computation. Unfortunately, gathering data on different railway scenarios to test and validate LiDAR-based algorithms and methods is still impractical and time-consuming, especially in critical operating conditions. For instance, acquiring a large dataset with thousands or millions of labelled data in real environments, in different scenarios, to train and test a deep neural network would take several months.

This paper presents a simulation framework based on Unreal Engine 4 (UE4) [3] to generate realistic railway scenarios semi-automatically and gather LiDAR labelled datasets to test and validate novel algorithms. Figure 1 presents a block diagram of the proposed simulation framework. It consists of two main components: the Environment Generation Tool, aimed at generating the environment surrounding the railroad and the environmental objects that populate the simulated world, and the Visualization Tool, aimed at creating the visual world, moving the train along a given trajectory, and exporting the obtained datasets.

2. Methods

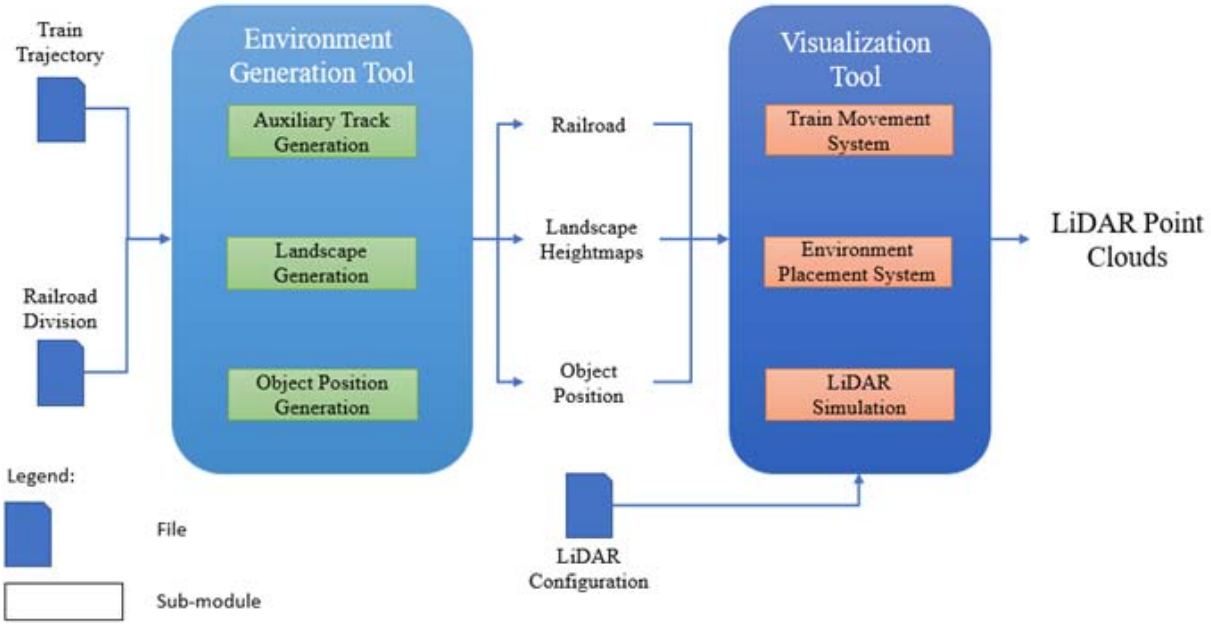

Figure 1: Block diagram of the proposed simulation framework.

The Environment Generation Tool reads the given train trajectory to build the surrounding environment of the railway. The trajectory consists of evenly-time-distributed positions and travelling directions of the train.

The train trajectory is divided into railroad blocks, each one labelled to indicate its type (e.g., curve, straight, tunnel, station, or bridge) and its starting point. The tool is composed of three sub-modules as follows.

1. The **Auxiliary Track Generation** sub-module loop-over the railroad blocks and generates auxiliary tracks parallel to the main trajectory, with a pseudo-random algorithm.
2. The **Landscape Generation** sub-module renders a heightmap based on the GeoGen application [4] that matches the height of the track waypoints in the surrounding, while the altitude of the farther area is obtained by modulating a Gaussian noise to attain a realistic scenario. Railroad blocks that represent bridge or station are explicitly treated, allowing the placement of rivers, stations, and buildings.
3. The **Object Position Generation** sub-module randomly generates the position of trees and rocks alongside the straight and curve blocks, and the position of buildings and roads alongside the station blocks.

The Environment Generation Tool produces three outputs containing the waypoints of the trajectory and the auxiliary tracks with the railroad division, the landscape heightmap, and the positions of the environmental objects. These are passed to the Visualization tool built on top of UE4, which consists of three main blocks:

1. **Train Movement System**: given the main trajectory, it moves the train to each waypoint, producing one frame for each waypoint. This method allows generating consistent data from different sensors (e.g., IMU, LiDAR, and camera) with respect to the same train position;
2. **Environment Placement System**: given the landscape heightmap and the environmental object positions, it physically places the landscapes in the virtual world and builds the whole scenario exploiting the Train Template UE4 plugin [5].
3. **LiDAR Simulation**: given the LiDAR specification, it emulates the LiDAR working principle by exploiting the ray-casting system of UE4. In particular, UE4 allows generating rays that return the first object hit in a given direction. Thus, the LiDAR is emulated spreading different rays accordingly with the scanning pattern of the sensor. The procedure also returns the type of object hit by the ray, which is used to compute the backscattered intensity of the ray.

3. Results

The proposed tool allows the generation of realistic railway scenarios, starting from the waypoints of the trajectory and the blocks' information. Figure 2 shows two examples of railway environment generated from a simulated train trajectory.

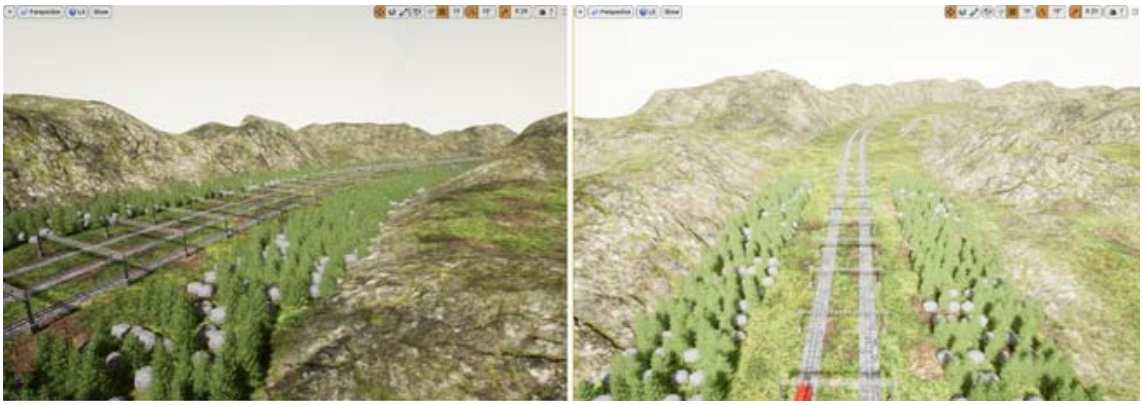

Figure 2: Example of railway environments generated by the simulation framework.

The final output is a dataset ready to be used in different contexts. For example, Figure 3 shows a visualization from RViz [6] of a simulated LiDAR frame.

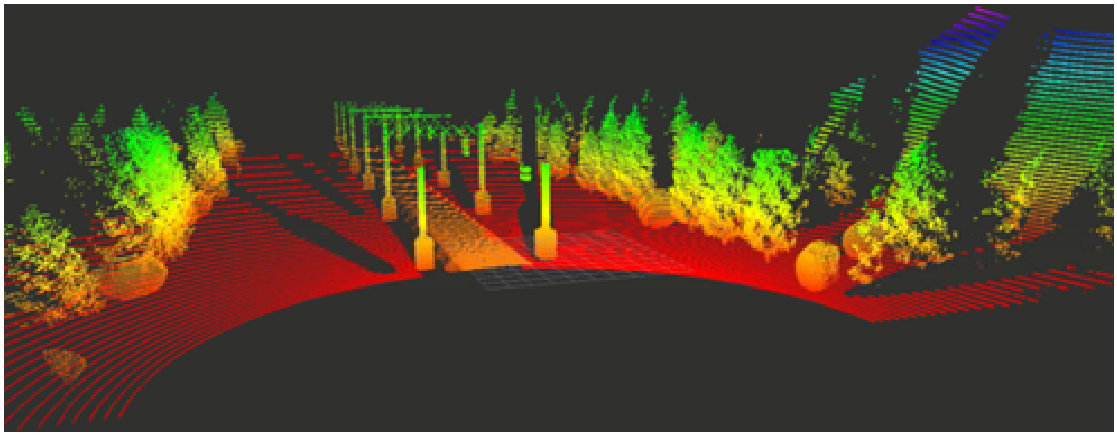

Figure 3: Simulated point cloud of a generated railway environment visualized with RViz.

To validate the accuracy of the simulation, two experiments have been performed. Following the approach presented in [8], a Velodyne LiDAR VLP-16 [9] was used in a scene with a cubic object with low specular reflection. Table 1 shows the main characteristic of the sensor.

Horizontal resolution	Vertical resolution	Wavelength	Beams	Precision
0.2°	2°	905 nm	16	±3 cm

Table 1: VLP-16 LiDAR specification.

In the two experiments the distance between the LiDAR and the object has been set to 70 cm and 150 cm, respectively. The scene was reproduced in the simulation environment, and the RMSE was computed between the positions returned by the real LiDAR and those computed by the simulator. As shown in Figure 4 and 5, the RMSE resulted to be ~1 cm, which is comparable with the precision of the VLP-16.

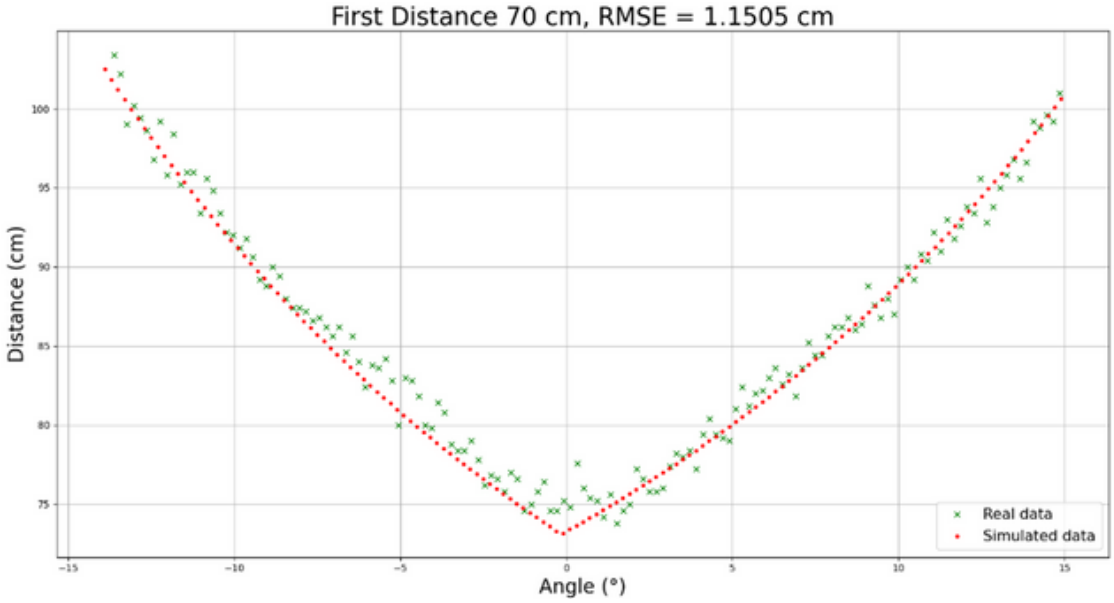

Figure 4: Experiment at 70 cm, showing the distances acquired by the LiDAR central beam (green crosses) and those computed by the simulator (red dots).

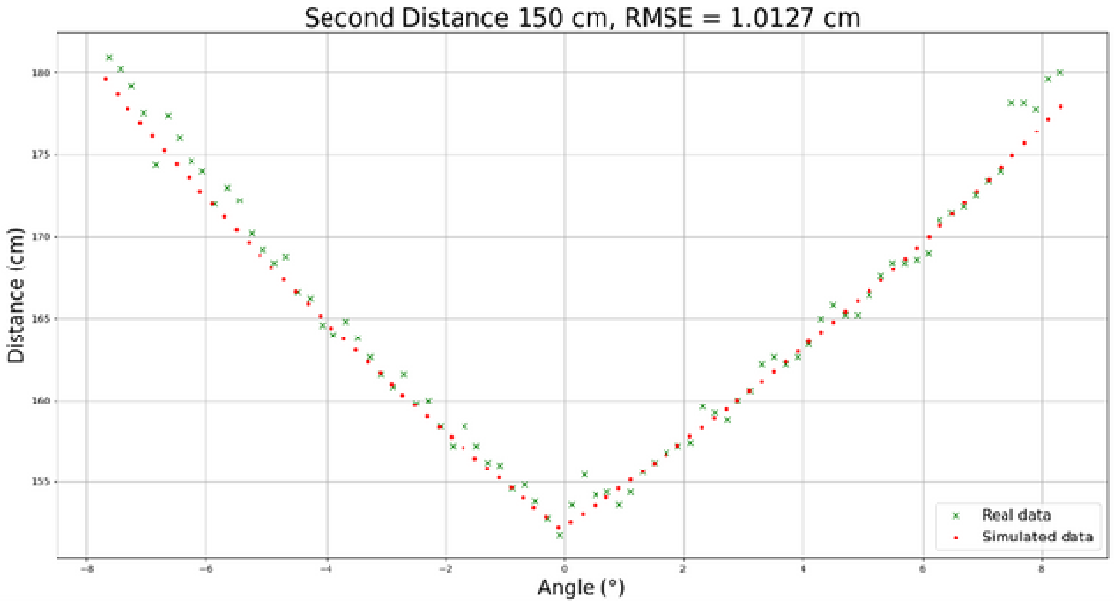

Figure 5: Experiment at 150 cm, showing the distances acquired by the LiDAR central beam (green crosses) and those computed by the simulator (red dots).

To evaluate the intensity model, another experiment has been performed by placing a sheet of aluminium on a rough wall at 75°, and comparing that acquired data with those computed by simulation in a similar setting. The backscattered intensity of each point is computed using the Lambertian–Beckmann model as a function of the characteristics of the object and the incidence angle, using the constants of wood for the wall and the coefficients of car shell for the aluminium sheet defined in [7].

Figure 6 shows both point clouds coloured with the intensity values.

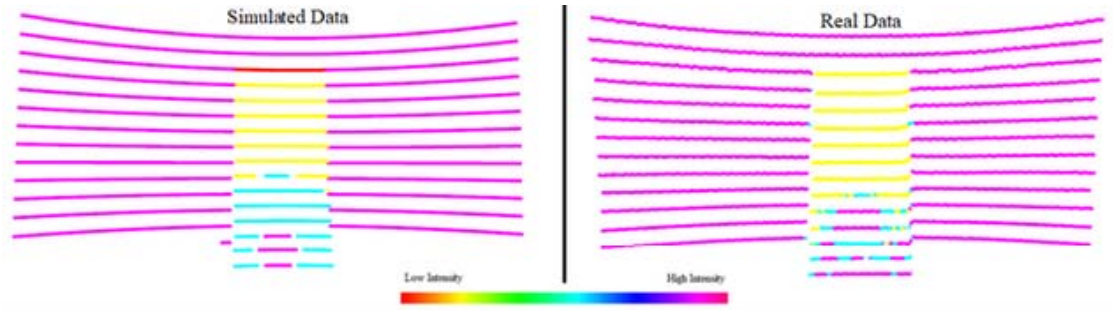

Figure 6: Intensity coloured point clouds.

Note that the central aluminium sheet shows how the backscattered intensity system follows the Lambertian–Beckmann model, where the highest intensity corresponds to a lower incidence angle.

4. Conclusions

This paper presented a graphic simulation framework to generate LiDAR frame datasets, for testing and validating new navigation solutions, as well as object detection and tracking algorithms. The tool allows reproducing different train operating conditions, generating railway environments in a pseudo-random manner, and simulating critical case studies in the railway environment. The framework is focused on the generation of LiDAR datasets exploiting the ray casting system of UE4.

The preliminary experiments carried out to compare the simulated data with the real ones were encouraging, reporting a difference of 1 cm between the two measurements, which is comparable with the error of the Velodyne VLP-16 LiDAR used in the tests. Also, the reflection model implemented in the system allows simulating the LiDAR ray-dropping behaviour, thus enhancing the realism of the obtained point clouds. Furthermore, the ray casting system directly provides the label of the object for each point in the point cloud, thus simplifying the creation of datasets for training deep neural networks in tasks such as object detection and segmentation.

As a future work, we aim at enhancing the LiDAR model, comparing it with real gathered datasets to adjust the reliability of the ray-dropping system and the distance error model. We also plan to add the possibility to simulate different weather conditions, in order to test and stress novel algorithms in multiple operating conditions. At last, the simulation framework will include the possibility to gather datasets from cameras that can be directly acquired from the graphic engine.

5. References

[1] A. Mirabadi, N. Mort, F. Schmid, "Application of sensor fusion to railway systems", Proc. of the IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems (Cat. No.96TH8242), pp. 185-192, 1996.

[2] F. Tschopp et al., "Experimental Comparison of Visual-Aided Odometry Methods for Rail Vehicles", IEEE Robotics and Automation Letters, Vol. 4, No. 2, pp. 1815-1822, 2019.

[3] Epic Games, "Unreal engine." [Online]. Available:https://www.unrealengine.com.

[4] Matěj Zábský, "Geogen studio." [Online]. Available: https://code.google.com/archive/p/geogen/.

[5] ZerstorenGames, "Train template." [Online]. Available: https://www.unrealengine.com/marketplace/en-US/product/train-template.

[6] Open Robotics, "RViz Robotic Operating System package" [Online], Available: http://wiki.ros.org/rviz.

[7] G. F. Gusmão, C. R. H. Barbosa, A. B. Raposo. "Development and validation of LiDAR sensor simulators based on parallel raycasting", Sensors, 20(24), 7186, 2020.

[8] "VLP-16 User Manual", Velodyne Lidar, [Online]. Available https://velodynelidar.com/wp-content/uploads/2019/12/63-9243-Rev-E-VLP-16-User-Manual.pdf.

[9] W. Tian, L. Tang, Y. Chen, Z. Li, J. Zhu, C. Jiang, J. Hyyppä, "Analysis and Radiometric Calibration for Backscatter Intensity of Hyperspectral LiDAR Caused by Incident Angle Effect.", Sensors, 21(9), 2960, 2021.