Toolchain for the Construction of Realistic Simulated Urban Environments

Pintér Balázs, Attila Ficsor

Abstract: Testing safety-critical autonomous vehicles (cars, trains, or trams) is an incredibly challenging task: those systems need to interact with an immensely complex and continuously changing environment, making systematic testing unfeasible. Moreover, physical test drives are highly costly, making simulator-based testing a favorable alternative. However, synthetic simulations may fail to provide realistic sensor input, thus hindering the effectiveness of the testing process. This paper proposes a toolchain for the semi-automated construction of realistic road networks using real-world maps. Therefore, better quality simulations can be implemented with higher productivity.

Keywords: simulation, realistic, ADAS, urban environment, analysis

1 Introduction

The promise of novel artificial intelligence and especially deep learning-based advanced driver-assistance systems (ADAS) is the increase of safety on our roads. Cameras and radar/lidar sensors observe the environment and collect information that is processed by computer vision and other sensor fusion applications. Based on the gained information these systems can send warning or even actuating signals. As we rely more and more on these ADAS technologies, ensuring their correct behavior is gaining even more importance. Various analysis techniques exist to verify the behavior of ADAS technologies. However, many of them requires that complex, realistic inputs have to be provided for the sensors: collecting sensory data from physical environments yields realistic data, but it is a tedious and expensive task. In addition, there might be a need to have data from dangerous situations, that is not feasible to collect from real, physical environments. On the other hand, simulators can serve as an alternative for sensory data generation without the risk of serious consequences of accidents and other damages.

Providing realistic environments for testing ADAS in simulators is a challenging but critical task. Manually constructing maps and the urban infrastructure is time-consuming. Artificially generating a road layout, and adding a basic environment around it can also be useful, but this approach might not generate realistically distributed lanes, curves, or junctions, hence during the test some faults might not occur, while others can be overrepresented.

Our objective was to find a scalable way to create a realistic simulator environment, with little manual effort. Based on the publicly available data of large cities, the urban environment can be virtually constructed. This approach requires less manual work than manually constructing the environment, and can give realistic results, as it is based on the real world. The results can then be used in various techniques such as testing or training ADAS technologies.

2 Overview of the approach

With the presented tools and technologies it is possible to create an accurate representation of a real world location, suitable for testing ADAS or autonomous vehicle software in a simulation environment. Our approach automated many time-consuming parts of the traditional virtual environment construction. Using public data instead of building the environment from scratch makes the environment building process much faster.

First, we extract the road geometry and the 3D environment from publicly available data sources. In the configuration step, we can fine-tune the different parts of the environment: road

network configurations or the editing of the 3D environment can be done in this step, manually. After the environment is constructed, we can export the environment into a standardized form, this way the widely used tools and simulators can work with developed environment.

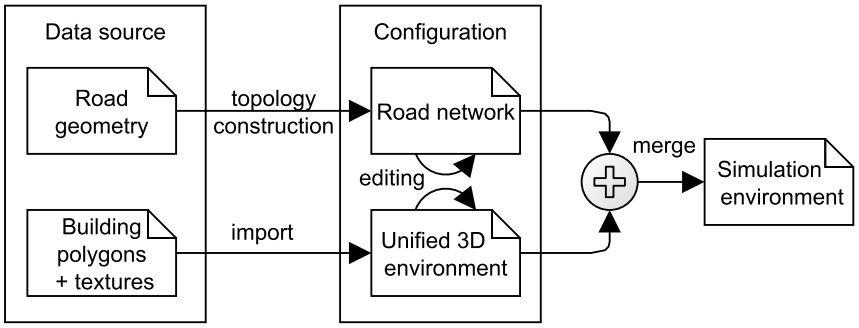


Figure 1: Simulator environment development workflow

2.1 Data source

Road Geometry The road geometry can be extracted from publicly available data sources (OpenStreetMap), then the map can be converted to the desired standard format (OpenDRIVE), which is widely supported by current test scenario generator and simulator tools.

Building polygons and textures There are multiple ways to acquire 3D models of real word buildings. They can be extracted from public databases, with additional software. It results in a triangle mesh file, which can be modified with any 3D modeling software.

From a public database (OpenStreetMap) we can extract the ground layout and the height of the buildings. This results in buildings with basic geometry, without any texture. Textures can be added manually, either using simple texture of materials, or real images.

3D buildings from Google Maps can also be extracted[1]. In the 3D model extracted from Google Maps the buildings are geometrically accurate, they also have realistic texture.

2.2 Configuration

Road network The map file has to be imported to the environment building software (RoadRunner), as it serves as the base of the environment, the spatial objects are adjusted to the map. The original OpenDRIVE map is sometimes adequate, but as Figure 2 shows, the file generated from the data source is not always complete: sometimes lanes and junction paths are missing and the road annotations and properties are not always correct. Fortunately, these are easy to fix and modify with the environment building software.

3D environment The extracted 3D models of the buildings are not always ready to be used directly in the environment building software. Usually, small modifications are needed in the 3D model, to get reusable photorealistic 3D components for the simulation environment, this can be done with a 3D computer graphics software. The importing method results in one comprehensive 3D model, containing everything, even the streets (and for the Google Maps approach, the vegetation or street objects), hence the buildings should be separated and the redundant elements must be pruned away. For the OpenStreetMap approach, we need to add texture to the walls, to reach realistic outlook.

[1]Tool and description: `https://github.com/eliemichel/MapsModelsImporter`

Advanced map and environment editing software are suitable for building complex environment above the map layer, including road materials, buildings, vegetation, traffic signs and other static objects. To make the scenes realistic in the simulation these objects are important, there are a lot of common assets already available in their asset library. In addition, everyone can make new assets or customize the existing ones, even importing new 3D objects or pictures is possible.

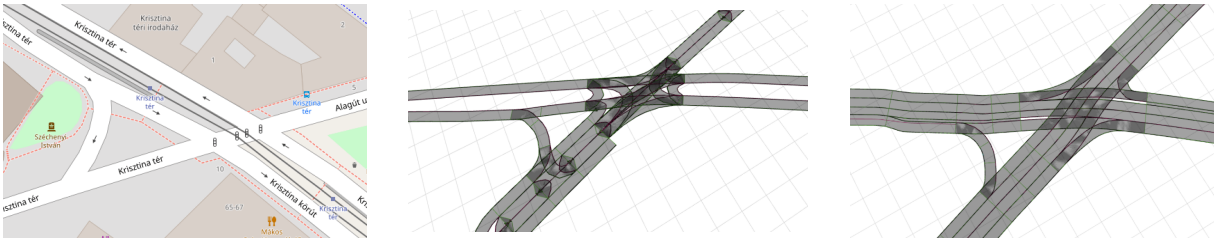
During the construction of the environment, we can label the objects with their type, this way it is possible to retrieve the semantic or instance segmentation data during the simulation, among the visual camera, depth map or other sensors. Our environment building process results in a realistic urban environment, which can be used for training or testing ADAS technologies or their components, in a simulated environment. For this purpose the input of the tested component (e.g. camera footage) and the ground truth (e.g. semantic segmentation) are also accessible, thanks to the our environment building process and the simulator.

3 Implementation

In this section we overview the main tools and technologies for the proposed realistic environment building workflow.

OpenStreetMap OpenStreetMap is an open source world map. OpenStreetMap contains detailed information about roads, it serves as a suitable starting point (even though due to the crowdsourcing data collection, it may contain some incorrectness). OpenStreetMap also contains simple 3D models of the buildings, which is useful for the environment building.

OpenDRIVE OpenDRIVE[2] is a widely used road network description standard, it describes the structure of the roads in XML syntax. The standard can describe roads, lanes, junctions, traffic signs, traffic lights, railways, switches, and many more. The standard can describe the road-related logic of the traffic, as the geometry of the road, signals, speed restriction, traffic lights, and other features implicate the traffic rules. It is supported (at least partially) by the tools we are working with: Roadrunner, Scenic [1], CARLA [2].



(a) Data source (b) Generated road geometry (c) Fine tuned map

Figure 2: Map transformation steps

Developing maps: RoadRunner RoadRunner[3] is a road and environment design software by VectorZero, with powerful building and editing toolset, widely supported import and export formats and many flexible and good customizable assets. RoadRunner provides OpenDRIVE editing tools. Other road related properties and appearance can be configured in this software, for example the road surface, lane markings or speed limits are easily adjustable.

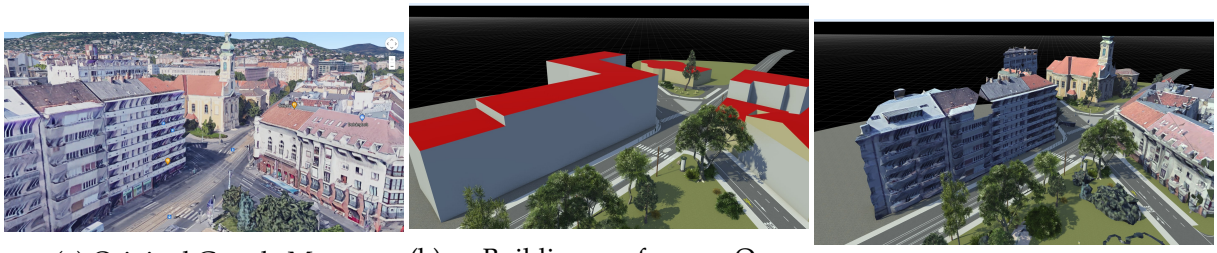
[2] https://www.asam.net/standards/detail/opendrive/

[3] https://www.mathworks.com/products/roadrunner.html

Unreal Editor 4 For executing the test scenarios, we chose the open-source CARLA simulator. CARLA simulator runs on Unreal Engine 4, and its editor Unreal Editor 4 provides rich functionality to manipulate the visual appearance and infrastructure (but not the OpenDRIVE map). Manually transforming the 3D objects, adding or removing street objects, vegetation is possible here.

For the 3D models we tried both the OpenStreetMap 3D buildings and the 3D view exported from Google Maps. These are shown in Figure 3, along with an image taken directly in Google Maps. Both approaches required manual modification of the 3D models. All vegetation was placed manually in Unreal Editor, using the variety of plants in the extensive blueprint library. The model from Google Maps contained some trees, but had too low resolution to be used. We used Blender to cut out these parts of the model, but we could still use the original version as a reference while placing the new, higher quality tree models.

Roadrunner and Unreal Editor allow us to label the objects, also every asset in their asset library has a type, accordingly, the simulator can provide semantic segmentation, and instance segmentation. CARLA's built-in semantic segmentation sensor encodes the type of an object in RGB values, but multiple objects can have the same color if their types match. Semantic segmentation images can give the ground truth for object detection, classification, or other similar tasks, but this is not always enough. The instance segmentation sensor encodes the type and the object's unique ID, this way we can make difference between the overlapping objects of the same type in the instance segmentation image.



(a) Original Google Maps (b) Buildings from OpenStreetMap (c) Buildings from Google Maps

Figure 3: 3D building comparison

4 Conclusion

In this paper, we proposed an approach with the necessary tools to support the efficient creation of real-world locations as virtual environments for simulators. We implemented a prototype of the toolchain and identified the steps that need manual fine tuning/configuration. The output of the toolchain provides realistic inputs for test scenario generation (which relies mostly on map-topology) and for the virtual sensors in the simulator.

References

[1] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: a language for scenario specification and scene generation," *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, Jun 2019.

[2] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning* (S. Levine, V. Vanhoucke, and K. Goldberg, eds.), vol. 78 of *Proceedings of Machine Learning Research*, pp. 1–16, PMLR, 13–15 Nov 2017.