Synthetic LiDAR-Labeled Data Set Generation to Train Deep Neural Networks for Object Classification in IoT at the Edge

Cristian Wisultschew, Rogelio Hernández, Carlos Pastor, and Jorge Portilla, *Senior Member, IEEE*

***Abstract*—Light detection and ranging (LiDAR) sensors are increasing in popularity due to the advantages they provide over 2-D sensors in IoT object detection and classification applications, because of their ability to provide very precise distances to objects. Deep learning algorithms need a huge amount of data during training to obtain high accuracy results. When using 2-D images, a vast quantity of data sets are publicly available, but this is not the case for LiDAR point clouds. Each LiDAR model generates a point cloud with unique properties, which causes the data sets not to be compatible between different LiDAR models. As a result, when using deep learning with LiDARs, it is necessary to generate the data sets manually. For this purpose, the data must be captured and then labeled one by one, which is a very time and cost-consuming process. To overcome this issue and to reduce the development time when using LiDAR sensors with deep learning algorithms, a methodology is proposed in this article to automatically generate point cloud data sets using a 3-D simulator for autonomous cars. In this regard, a data set can be generated for any LiDAR model by adding the specific LiDAR parameters to the simulator. Besides, custom scenarios can be designed and generated, based on the final deployment location, to provide a simulated solution very close to the final implementation. With the proposed methodology, a simulation can be performed to select the LiDAR that best fits certain application requirements, in contrast to the traditional approach where the LiDAR must first be purchased.**

***Index Terms*—3-D simulator, deep neural networks (DNNs), light detection and ranging (LiDAR), object classification, simulated data set.**

I. Introduction

THE CURRENT high demand for autonomous systems that are capable of recognizing the scenario in which they are located, requires the use of sensors that are as accurate as possible for this task. Nowadays, there is a trend of merging the information coming from several sensors of different types to obtain a very accurate scenario representation [1]. However, when IoT edge devices are considered, computational and power consumption constraints appear and, therefore, the number of sensors must be limited. In systems where a single sensor can be used, the use of 3-D sensors instead of 2-D sensors is essential to obtain precise spatial information. This is a critical requirement in applications that have to precisely locate objects in the space, such as autonomous vehicles or surveillance systems.

Among 3-D sensors, light detection and ranging (LiDAR) sensors are currently the most outstanding technology for several reasons. First, they provide very accurate distance measurements, which improves the detection and location of objects in the scene. This is the result of generating information about the scenario in the form of point clouds. Second, since they are based on laser technology, they are very robust in a variety of conditions: day or night, with or without reflections and shadows, rainy, and foggy [2], [3]. Third, regarding its price, even though some models are still very expensive today, their price is decreasing due to new technologies such as in the case of solid-state LiDARs [4]. Additionally, with this new solid state technology, it is also possible to reduce the power consumption, which allows LiDARs implementation in edge systems in a competitive manner.

In order to perform object classification tasks when processing the large amount of information generated by the LiDARs, deep neural networks (DNNs) are the primary solution adopted in the state of the art [5]. The reason for this is the high level of abstraction they provide when working with large amounts of data [6]. When working with DNNs, an initial stage of training with labeled data must be performed. To obtain high accuracy when classifying objects, DNNs require to be trained with a large amount of data [7].

When using RGB cameras, there are a wide variety of publicly available labeled data sets that can be used to train the DNN. However, there are not many data sets for LiDARs and, additionally, as each LiDAR generates a point cloud with different densities, the data sets are not compatible between different models of LiDARs. For this reason, when working with LiDARs, it is common to create a labeled data set from scratch for each LiDAR and for different scenarios. Thus, the LiDAR first has to be deployed, data have to be collected, and then the data have to be manually labeled one by one with all the objects within the scenario. Unfortunately, this labeling

Manuscript received 25 March 2022; revised 24 June 2022; accepted 25 July 2022. Date of publication 28 July 2022; date of current version 7 December 2022. This work was supported by the InSecTT European Project. InSecTT (www.insectt.eu) has received funding from the ECSEL Joint Undertaking (JU) under Agreement 876038. The JU receives support from the European Union's Horizon 2020 Research and Innovation Programme and Austria, Sweden, Spain, Italy, France, Portugal, Ireland, Finland, Slovenia, Poland, The Netherlands, and Turkey. *(Corresponding author: Cristian Wisultschew.)*

The authors are with the Centro de Electrónica Industrial, Universidad Politécnica de Madrid, 28006 Madrid, Spain (e-mail: cristian.wpuigdellivol@upm.es; r.hlorite@upm.es; carlos.pastor.molina@upm.es; jorge.portilla@upm.es).

Digital Object Identifier 10.1109/JIOT.2022.3194716

process is extremely time and cost consuming. Besides, a large number of objects must be labeled to train the network and to obtain accurate results [7].

In this work, a LiDAR point cloud generation methodology for automatically producing synthetic high-fidelity-labeled data sets is presented. Thus, the DNN training stage is carried out only with simulated data, significantly reducing the development time when implementing object classification applications with LiDAR sensors. Additionally, the cost is reduced since the most affordable LiDAR that meets the requirements of the application can be selected before the deployment stage, in contrast to the traditional approach where the LiDAR must be purchased first. The accuracy results obtained training the DNN with synthetic data are almost identical compared to the results achieved when training with real data. As far as the authors know, there are no works in the state of the art that propose this solution for training DNN only using simulated data and providing high accuracies for LiDAR sensors. In this regard, the main contributions of this article are described as follows.

1) Automation of the process of labeling synthetic data sets for LiDARs.
2) Addition of different types of LiDAR models for evaluating in predeployment which LiDAR model best fits the accuracy and cost requirements for each specific application.
3) Integration of custom scenarios to simulate the final location, which maximize the accuracy of the classification algorithm.
4) Addition of custom 3-D objects of new classes to the simulated scenario to better fit with the real scenario, aiming to improve the classification accuracy in the deployment stage.
5) To maximize the accuracy when performing object classification tasks using the point-cloud-based DNN implemented in this work, a methodology to calculate the DNN parameters is proposed.

This work is used as part of a project in which objects that pass through a railway level crossing have to be detected and classified. A static LiDAR will be used as an input sensor as it provides very accurate spatial information, which is essential to ensure a robust and reliable solution. The system has to run on an IoT node, therefore, computing resources and power consumption constraint limitations have to be considered. This work is an improvement of the work presented in [8], in which the detection of objects crossing a certain critical region surrounding a railway level crossing is performed using a LiDAR sensor and processing the information in an IoT node. The detection starts by projecting the LIDAR point cloud into a 2-D image and then this image is used as input of an object detection and tracking algorithm. This algorithm detects the objects by identifying the changes compared to a previously generated background image. This processing was performed using an IoT edge node with a 32-bit medium-performance, low-power ARM Cortex-A5 core. In this regard, with the simulated data sets generated in this work, the aim is to train a DNN to add the object classification capability to the system proposed in [8].

The remainder of this article is structured as follows. In Section II, related works which generate simulated data sets to train DNN are described. The technical background about the real point cloud data sets used to evaluate the system, the different LiDAR types involved, 3-D design platforms, and the DNN algorithm implemented for point cloud object classification tasks are presented in Section III. The process for the generation of the simulated data sets is detailed in Section IV. Experimental results are discussed in Section V. Finally, conclusions and future lines of work are provided in Section VI.

II. Related Work

With the increasing number of applications that use DNN with LiDARs as input for object classification and detection tasks, there is an increasing effort to address the lack of labeled point cloud data sets. With the additional challenge that each data set depends on each specific LiDAR brand and model, they have different properties [9]. This section aims to present the current state of the art about LiDAR synthetic data sets used for object classification, object detection, and scene segmentation tasks.

The first simulated LiDAR data set generated automatically used for augmenting real training data sets on scene segmentation applications is presented by Yue *et al.* [10]. For the simulator, they select the virtual world in Grand Theft Auto V (GTA V), a popular video game, to obtain high fidelity simulated point clouds. Their analysis is based on SqueezeSeg [11], a DNN-based model for point cloud segmentation. They show improvements in the accuracy when augmenting the KITTI data set [12] with their simulated LiDAR point clouds before the training stage. Additionally, the scene images can be captured simultaneously for future sensor fusion tasks. The results show that for a point cloud segmentation task, synthesized data help improve the Intersection-over-Union metric [10] by 9% on the KITTI benchmark. However, when using the methodology proposed by Yue *et al.* [10], the data set generation stage is still maintained, as opposed to the methodology proposed in this work in which the data set generation stage is completely avoided. The goal of the work proposed in [10] is to increase the accuracy of the model by using simulated data along with real data. However, the goal proposed in this work is to remove the data set generation step while maintaining the same accuracy compared to using real data.

A LiDAR simulator that augments real point cloud with synthetic obstacles (e.g., cars, pedestrians, and other movable objects) is presented by Fang *et al.* [13]. The augmented simulator avoids the requirement to create background 3-D models, unlike other simulators that entirely rely on 3-D models and game engines. Instead, a vehicle with a LiDAR scanner can be deployed to scan the region of interest and obtain the background. Then, a labeled point cloud object can be automatically generated and added to the data set of the previously scanned scenario. The specific LiDAR properties for the simulated data can be modified, such as the channels number, range, horizontal and vertical filed-of-view, and angular and vertical resolution. Their solution is evaluated for 3-D object

detection and scene segmentation tasks using real and simulated data. The DNN used in their experiments are based on SECOND [14] for 3-D object detection, while for scene segmentation, an accelerated version of MV3D [15] is used. Fang *et al.* [13] showed that the accuracies obtained with real data can be slightly improved by training with simulated data together with real data. They also prove that by using real background data augmented with simulated data during training, accuracies close to those achieved by training with real data can be obtained. However, extremely low accuracies are obtained when training only with simulated data and without real backgrounds. In the work carried out by Fang *et al.* [13], real data were combined with simulated data aiming to increase the accuracy of the model compared to the one trained only with real data. However, when testing the same DNN model trained using only simulated data, the accuracy of the model is drastically reduced in contrast to this work in which very similar accuracies are obtained when training only with simulated data compared to train with real data.

Hurl *et al.* [16] provided a method to generate precise LiDAR point clouds which accurately represent people (pedestrians and cyclists) using the video game GTAV. They provide a large data set (50 000+ frames) in the KITTI data set format. As Hurl *et al.* [16] were focused on people detection, the AVOD-FPN [17] DNN architecture is used for testing. AVOD-FPN is a top-5 performer on the KITTI 3-D object detection benchmark challenge [18] under the pedestrian category. They demonstrate the effectiveness of their simulated data set by showing an improvement of up to 5% average precision (metric defined in [16]) on the KITTI 3-D Object Detection benchmark challenge when an object detection DNN was trained with their simulated data sets along with KITTI data sets. In contrast, in the presented work, the DNN training is performed only with simulated data sets, as opposed to Hurl *et al.* [16] proposed, who obtain high accuracy results when training with the simulated data sets along with the KITTI data sets. In [16], the accuracies obtained when training only with simulated data are up to 14.13% average precision. In the proposed work, high accuracies are obtained when performing object classification tasks training only with simulated data sets. This has the advantage of avoiding the dependence on real data to obtain high accuracy, which means not having to rely on the availability of a labeled data set or to create a data set from scratch. Furthermore, the proposed work focuses on object classification tasks, opposed to [16] in which object detection is performed. So, they are not directly comparable according to the metrics used in each case.

In summary, every work that can be found in the state of the art related to object detection and classification tasks using LiDAR simulated data sets aiming to increase the accuracy provided by the DNNs by using simulated data along with real data during the training stage. When using simulated data together with real data, there are works that significantly improve the object detection by improving parameters such as Intersection-over-Union [10]. Additionally, other works improve the accuracy during object classification stage [13], [16]. In all these works, an improvement in accuracy is achieved, however, the data labeling stage remains, which is the most time and cost-consuming stage. The problem arises when using only simulated data to train the DNNs as the model accuracies drop drastically [13], [16]. As far as the authors know, no work obtains high accuracy results when performing object classification tasks using only simulated data sets for the training stage contrary to the methodology proposed in this work in which high accuracies are obtained by using only simulated data for the training stage. Besides, the properties of each LiDAR along with the scenarios and objects of the simulated data can be customized in this work, to be as similar as possible to the final deployment.

In summary, every work that can be found in the state of the art related to object detection and classification tasks, using LiDAR-simulated data sets aiming to increase the accuracy provided by the DNNs by using simulated data along with real data during the training stage, obtain high-accuracy results by augmenting real data with simulated data. However, the data labeling stage remains, which is the most time and cost-consuming stage. As far as the authors know, no work obtains high-accuracy results when performing object classification tasks using only simulated data sets for the training stage. Besides, the properties of each LiDAR along with the scenarios and objects of the simulated data can be customized in this work, to be as similar as possible to the final deployment.

III. Technical Background

In this section, an overview of the LiDAR sensor technology considered along with the data sets generated by them is provided. Additionally, the object classification DNN architecture implemented and the reasons for its selection are detailed.

A. LiDAR Sensor

A LiDAR is a device that measures the distance from the laser emitter to a target using a laser beam. The distance to the object is determined by measuring the time between the emission of the pulse and its detection through the reflected signal. In the case of 3-D-LiDAR, there are several lasers placed in a column configuration. The column rotates 360° to generate a point cloud map of the surrounding scenario. This sensor provides information about the coordinates of each point in XYZ format along with the reflectivity value.

In this work, the Velodyne VLP-16 [19] was used to collect real data in the railway-level crossing scenario, which is the same location for the final deployment of this system as shown in Fig. 1(a). The VLP-16 is a low-resolution mechanical LiDAR with a reduced price compared to other high-resolution LiDARs from Velodyne. The LiDAR configuration properties for the VLP-16 are presented in Table I.

B. Data Sets From Real Data

Two different point cloud data sets containing real data are used in this work to validate the DNN trained with simulated data. On the one hand, a data set was generated with the Velodyne VLP-16, which has been manually collected and labeled. On the other hand, the KITTI data set will be used to

TABLE I
VELODYNE VLP-16 CONFIGURATION PROPERTIES

	Properties	VLP-16
Sensor	Measurement range	up to 100 m
	Points per second	up to 0.3×10^6
	Lasers number	16
	Accuracy	+/- 3 cm
	Horizontal field of view	360º
	Horizontal resolution	0.1º - 0.4º
	Vertical field of view	30º (-15º to 15º)
	Vertical resolution	2º
	Rotation rate	5 - 20 Hz
Laser	Laser safety	Class 1
	Laser wavelength	903 nm
	Pulse duration	6 ns
Mechanical and electrical specifications	Power consumption	8 W
	Operating voltage	9 - 32 VDC
	Weight	830 g
	Size (diameter x height)	103 x 72 mm
	Spin rate	300 - 1200 rpm
	Operating temperature	-10 ºC to +60 ºC

compare the results of this work with a data set widely used in the state of the art.

The VLP-16 data set was collected in the railway-level crossing scenario shown in Fig. 1(a) which is situated next to the train station of Langau, which is a town in the district of Horn in Lower Austria. This data set contains different object classes, such as pedestrians, bicycles, cars, trucks, and buses. Once the data set was collected, 3000 frames were randomly selected. Then, all the objects that appeared in each frame were manually labeled. For this purpose, the LiDARLabeler tool [20] was used, which is specific for the labeling of point clouds. This tool is a MATLAB package that allows interactive point cloud labeling of Velodyne LiDARs. These data will later be exported to be used with the DNN. The average labeling time when using this tool on data generated by the Velodyne VLP-16 is 80 labeled objects per hour. Additionally, it is necessary to take into account the time it took to capture the data during the deployment, which was 4 h for this data set. Therefore, the total amount of time used for the generation of this manually labeled data set took 26 h approximately.

Once all the objects in the frame were labeled, the objects outside a certain critical region were removed. This region consists of a rectangular region of 200-m long by 6-m wide, covering the road in the railway level crossing. This region was selected as it is where vehicles and pedestrians circulate in the real scenario of the railway level crossing. In the 3000 selected frames, a total of 1739 objects, including 1291 pedestrians, 59 cyclists/motorcycles, 368 cars, and 21 trucks/buses were labeled inside the critical region. The cyclists and motorcycles are grouped in the same class as they are extremely difficult to distinguish by the DNN since they share a high level of similarity. Besides, the low resolution provided by VLP-16 also increases the difficulty to distinguish them. This grouping is also applicable to truck and bus classes for the same reason. The object classification DNN implemented in this work uses objects as inputs instead of a complete point cloud from a scenario as object detection DNN use. For this reason, the data-set objects were separated from their scenario.

Additionally, the KITTI point cloud data set has also been used in this work to compare the results obtained when using simulated data. The KITTI data set was generated using the Velodyne HDL-64 [19], which is a high-resolution 360° mechanical LiDAR. The scenarios captured in the KITTI data set are diverse, covering real-world traffic situations, ranging from freeways over rural areas to inner city scenes with many static and dynamic objects. This data set contains different classes of labeled objects, such as pedestrians, cyclists, cars, vans, trucks, and trams. The presented work is focused on object classification purposes, for this reason, this data set was preprocessed to obtain the objects separated from the scenarios. As well as for the VLP-16 data set, the vans and truck classes are grouped in the same class as they are extremely difficult to distinguish using point clouds due to their high level of similarity. Besides, the trams class is discarded since it is not found in the data sets collected with the VLP-16.

It should be noted that the KITTI data set is generated with the Velodyne HDL-64, differently from the presented work, which uses the VLP-16, being a lower cost and lower resolution version. It provides similar properties as the VLP-16 but with a higher resolution since it has 64 lasers instead of 16. As a result, the point clouds have a different density, which means that the data sets cannot be directly compared. To solve this issue, the HDL-64 point cloud has been transformed to be compatible with the VLP-16 by taking the information from 16 of the 64 available lasers. The 16 lasers selected are in the same arrangement as the VLP-16 lasers, making them fully compatible. For this reason, this adaptation of the KITTI data set is called KITTI16 in this work.

C. DNN

For the DNN selection, it must be taken into account that the DNN will be implemented on an IoT edge node that has limited computational resources. Thus, it is essential to select a lightweight DNN that provides high performance. Among the point cloud-based DNN architectures studied in [21], the one that offers the highest performance along with the smaller size in memory is the VoxNet architecture. For this reason, it is the DNN architecture selected for the implementation of this work.

VoxNet was presented in [22] by Maturana and Scherer aiming to classify 3-D volume objects with the minimum resolution using a voxel grid representation [23]. The voxel grid representation is a 3-D matrix space composed of a fixed number of voxels. A voxel consists of a representation similar to a pixel, but instead of representing a 2-D plane, it represents a volume feature on a 3-D grid.

When using point cloud as voxels there is a preprocessing stage required to convert the point cloud into voxels as it is explained in [22]. This preprocessing stage is extremely lightweight in computational terms, which is essential for systems with low computational resources. When using VoxNet it is necessary to define the voxel grid representation

size, which will be the same for all objects in the data set since the input size when working with DNNs is fixed. This size is quantitatively calculated in Section V-B.

To carry out the training of the DNNs used in this work, the TensorFlow framework [24] is selected since it provides tools to create complex topologies such as 3-D point cloud-based DNNs. TensorFlow was created by Google and is one of the most widely used frameworks to design, train, validate, and deploy DNNs. TensorFlow supports Python, C++, and Java programming languages. However, its Python API is much more efficient when developing an end-to-end solution as it provides a large number of libraries for data preprocessing. For this reason, in this work, both training and inference stages are performed using Python.

IV. Generation of the Simulated Data Set

A. SVL Simulator

As it was mentioned before, the data labeling process is extremely time and cost consuming. To solve this issue, a simulator is used to automatically generate a labeled data set. When selecting the simulator, certain basic requirements must be taken into account. First, an opensource simulator is required, since it allows the modification of the source code to adapt it to the requirements of this work. Second, it is essential that it can be controlled with an API since the process must be fully automatic. Third, it has to be based on a game engine for interactive media creation. Thus, it will be possible to implement custom objects and scenarios according to the specifications of each specific problem. Fourth, it has to allow the parameter customization of the LiDAR sensors. In this regard, any LiDAR model can be implemented.

SVL [25] and CARLA [26] simulators are similar and both meet the requirements for this work, however, they are built with different game engines. SVL Simulator has been selected since it is Unity based [27], which is more user friendly and accessible than CARLA that uses the Unreal Engine [28]. The SVL Simulator is an end-to-end autonomous vehicle simulation platform developed by LG Electronics America Research and Development Lab. This simulator has been selected also because it is opensource and has a Python API for its use. This simulator includes predefined objects, such as pedestrians, cars, trucks, and buses that can be controlled. Being Unity based, it also supports the creation of custom objects, scenarios, and sensors. The SVL simulator provides a Python API to allow the automation of the entire process.

B. Simulated Sensors

To generate a simulated data set, two different kinds of simulated sensors must be used. First, a simulated LiDAR, and second, a ground truth sensor. The LiDAR will provide point cloud information from its surroundings that can be stored using the Python API. The timing of the point cloud capture can be controlled. Velodyne VLP-16 has been selected as LiDARs, as was explained in Section III-A. The LiDAR model configuration properties have to be provided to the simulator. These parameters are shown in the sensor section of Table I. Moreover, the ground-truth sensor will provide information about the location, size, and orientation for each of the objects presented in the simulation. The output of this sensor is only available through ROS [29]. Combining the data from these two simulated sensors, it is possible to identify whether a point from the point cloud belongs to any of the objects of the simulation or if it is just background. This is essential in order to be able to label the data.

C. Virtual Scenarios and Objects Creation

Game engines, such as Unity, allow importing 3-D objects previously modeled in external programs. These objects can be vehicles, pedestrians, sensors, or any custom object. Additionally, maps can be imported as well. Adding 3-D objects stored in Unity to the simulator is straightforward. The simulator recognizes the names of each object and then allows them to navigate through the maps.

To create a realistic environment faster and in a systematic way, the blender-osm plugin [30] has been used. Blender-osm provides one-click download and import of OpenStreetMap [31] and terrain data from satellite information with global coverage. This information is easily transferable to a 3-D model. The information obtained through this process is insufficiently accurate, but it provides a basis to start working. The final adjustments and details are modeled manually.

D. Simulation

The simulation consists of a setup, where a LiDAR sensor, a scenario, and objects have to be selected. The simulator provides different road maps by default; however, custom maps can be also be integrated. Besides, the community uploads custom scenarios that can be used. For the proposed work, a customized scenario of a railway level crossing was designed based on the scenario where the real data were collected as mentioned in Section III-B. Fig. 1(a) shows an image of the real scenario and Fig. 1(b) shows the simulated scenario created.

Then, objects are randomly generated according to the AI paths that are selected for each scenario. Some new objects, such as bicycles and motorcycles have been included due to the simulator does not include them and these data are critical for real usage. Fig. 1(c) shows the objects placed in the scenario.

Finally, the position where the sensor will be placed must be defined. The sensor will be fixed and located off the road as it will be in the final deployment. In this case, the LiDAR was located where the real data were collected as shown in Fig. 1(a).

E. Outputs Provided by the Simulator

The simulation outputs are generated in each frame and are composed on the one side by the information of the complete point cloud provided by the LiDAR in XYZ format as it is shown in Fig. 1(d). On the other side, the location and size of each bounding box are generated for all the objects in the scenario as illustrated in Fig. 1(e). These data are given in meters. Note that no information regarding reflectivity is provided by the simulator. Then, these data have to be processed to obtain the point cloud of each object separated from the

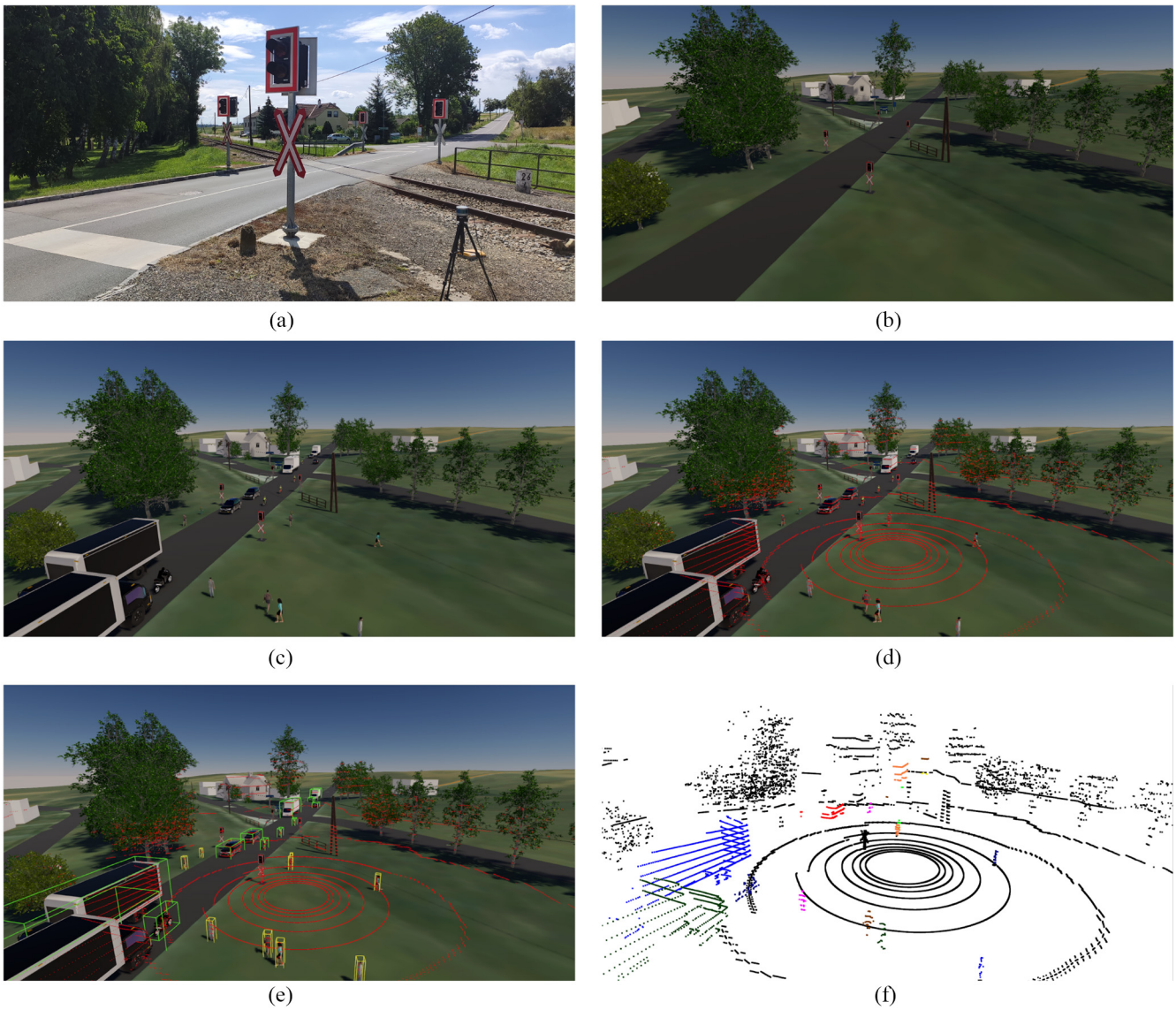

Fig. 1. Simulated data set generation workflow. (a) Real image of the railway level crossing. (b) Simulated scenario without objects. (c) Simulated scenario with objects. (d) Simulated scenario with simulated point cloud provided by the LiDAR. (e) Simulated scenario, and LiDAR with objects inside bounding boxes. (f) Simulated LiDAR with objects point clouds of colored objects by classes.

rest of the frame points. Fig. 1(f) shows the point clouds of each object with a different color. Finally, each of these objects will be converted into voxels, which is the format supported by VoxNet.

V. Object Classification Experiments

In this section, experiments are carried out to demonstrate that training exclusively with simulated data provides very similar results in terms of accuracy compared to training with real data. Once the simulated data are generated, the voxel grid size together with the data set size needed to obtain the maximum accuracy is calculated in this section.

A. Experimental Setup

The object classification performed in this work is focused on a critical region of 10-m radius from the LiDAR center corresponding to the region to be analyzed in the surroundings of the railway level crossing following the project specifications as it was explained in Section III-B. For this reason, all objects that fall outside this critical region are not taken into account for all data sets used, both simulated and real. In this regard, the best results in terms of accuracy when deploying the system at the railway level crossing will be obtained since the network becomes more specialized in data similar to the final deploy.

The accuracy averaged per class is used as a metric to evaluate each experiment. This metric is defined by the number of correctly classified objects divided by the total number of samples.

Training with simulated and real point cloud data sets was carried out during the experiments of this work. For the DNN training stage, the data sets have been divided into training and validation. The training and validation data were split in a random manner, representing 80% and 20% of the total data set, respectively. The number of objects per class contained in each

TABLE II
OBJECTS NUMBER BY CLASSES FOR EACH DATA SET

Class	Datasets		
	Simulated dataset	VLP-16	KITTI
Pedestrian	43683	1291	4600
Car	4805	368	1361
Truck/Bus	1624	21	469
Cyclist/Motorbike	3219	59	231
Total	53331	1739	6661

TABLE III
ACCURACIES FOR DIFFERENT VOXEL GRID SIZES

Voxel grid size	Inference time	Accuracy
4x4x4	4.61 ms	85.39 %
8x8x8	5.62 ms	90.51 %
12x12x12	6.35 ms	**96.84 %**
16x16x16	7.24 ms	95.80 %
24x24x24	8.87 ms	96.61 %
32x32x32	12.53 ms	96.03 %

TABLE IV
ACCURACIES WHEN TRAINING WITH DIFFERENT OBJECT NUMBERS

Frames number	Objects number	Generation time	Accuracy
200	1111	3 min	77.93 %
500	2402	7 min	85.28 %
1000	5359	14 min	91.37 %
5000	26440	1 h 8 min	94.08 %
10000	53338	2 h 17 min	96.84 %
15000	81448	3 h 25 min	97.18 %
20000	109148	4 h 31 min	**97.24 %**
35000	196957	7 h 52 min	96.01 %
50000	281919	11 h 20 min	96.14 %

TABLE V
ACCURACY WHEN TESTING WITH THE VLP-16 DATA SET

DNN trained with	Accuracy
Simulated dataset	97.24 %
VLP-16	97.38 %*
KITTI16	83.95 %

data set, which is presented in Table II, impacts the learning capability of the DNN as will be explained in the following section.

The hardware used for the development of the experiments presented in this section consist, on the one hand, by a desktop PC with an Intel i5-10600K processor, 8-GB RAM memory, and a Nvidia GeForce GTX 1060 for the generation of the simulated data set. On the other hand, the inference time results provided by the DNN experiments were obtained using a desktop PC with an Intel i7-8700K processor, 32-MB RAM memory, and without using a GPU. GPU is not used for making inference with DNN as it requires adapting the DNN models and is out of the scope of this work.

B. Experimental Results

Since the main goal of this work is to implement the system on an IoT edge node, the smallest voxel grid size should be selected as it will provide maximum performance. However, it should be noted that a too small grid size may cause the accuracy to drop significantly. Table III shows the experiments when training different voxel grid size DNNs with simulated data. The accuracy results shown in Table III are calculated when testing the DNNs with the VLP-16 real data set. The DNN was trained with a simulated data set of 10 000 frames. Besides, Table III shows results about the performance when making inference with a batch size of one object in order to analyze the influence of the processing time compared to the size of the object. The inference times shown is an average value of the time it takes to perform inference on the 1739 objects contained in the VLP-16 real object database. It is calculated as the average value since there are slight differences in the measurements when processing each object individually. This occurs because the experiments were executed on a Linux operating system that performs other processes at the same time.

The voxel grid size of $12 \times 12 \times 12$ is the one that obtains the best results in terms of accuracy, for this reason, it will be used in the following experiments. An experiment must be performed to select the number of frames generated by the simulator to obtain the best results in terms of accuracy. For this purpose, data sets have been generated using simulations with a different number of frames. Each of these data sets was then used to train the DNN and perform inference using the real VLP-16 data set. The accuracy results when performing the inferences are shown in Table IV. Additionally, results about the times for the data set generation are presented in Table IV. Within this time, both the simulation time and the postprocessing time to adapt the database to the training format are included, representing 39% and 61% of the total time, respectively.

Once the optimal DNN model trained with simulated data is obtained, it is compared to models trained only with real data. In Table V, the results when training the same VoxNet DNN using real and simulated data sets are presented. VLP-16 represents the data set generated and labeled in this work with the VLP-16 LiDAR. On the other hand, KITTI16 refers to the KITTI data set adapted to 16 lasers as it was explained in Section III-B. All the accuracy results shown in Table V were calculated making inference using the real data sets of the VLP-16. In this regard, an appropriate comparison of the results can be achieved. The real data from VLP16 have been used for the experiments since they are the data collected in the final deployment scenario of the use case.

It should be noted that the accuracy results when training with the VLP-16 data set were calculated with the validation data used during training, which represent 20% of the total data set. The purpose of this is to avoid calculating accuracy values with data used for training. However, for the simulated and KITTI16 data sets results, the accuracy is

calculated using the full VLP-16 data set. For this reason, the accuracy value of VLP-16 is marked with the * symbol in Table V.

C. Analysis of Results

The voxel grid size that provides the best results in terms of accuracy is $12\times12\times12$ as it is shown in Table III. It could be expected that for larger voxel grid sizes, higher accuracies should be obtained since they have higher resolution. However, this is not the case due to the point cloud spreading caused by the low resolution of the VLP-16 LiDAR. As it is shown in Fig. 1(d), the point cloud produced by VLP-16 presents a low resolution along the vertical axis. This fact provokes voxel empty spaces inside the objects, resulting in a depreciation of the DNN learning during training.

Regarding the number of simulated frames used to train the DNN, different results in terms of accuracy were obtained when varying the number of frames as it is shown in Table IV. The data set size that provides the best results in terms of accuracy depends on various factors. First, the architecture of the DNN used, in this case, is the VoxNet architecture. Second, the statistical characteristics of the data, which are point clouds provided by the Velodyne VLP-16. Finally, in the number of different classes that the DNN is trained to classify, in this case, there are four classes. Adding more samples to the data set when training with DNN increases the diversity and decreases the generalization error. Thus, using more data implies an increase in accuracy up to a certain limit. However, using a reduced number of samples affects the accuracy. There is a minimum sample size after which the network starts to lose accuracy. For the simulated data set generated in this work, the limiting size where the accuracy starts to decrease significantly is 10 000 frames as it is shown in Table IV. Beyond this limit, the accuracy slightly oscillates.

In the use case presented in this work, it takes about 26 s to generate and label the VLP-16 database with four classes and obtain good results during training. By using the simulated data set, this process is avoided and this time can be saved. When increasing the number of classes identified by the DNN, it is necessary to train with a data set with more samples in each class [7]. Thus, when using the proposed methodology, as more objects need to be classified, more time will be saved by using simulated data.

No comparison to the impact of other data sets on DNN accuracy is possible at this time due to their current lack of availability for object classification applications that train the DNN only with simulated data. Most object classification algorithms use 2-D images as input and when working with this type of data, it does not make sense to use simulated data sets as there is a large number of labeled image data sets available online. However, it does not occur when using LiDAR sensors, as each LiDAR model produces a point cloud with different properties. In this regard, if an object classification DNN should be implemented in a LiDAR system, the data set used to train the DNN has to be generated with the same LiDAR model that will be used in the deployment to obtain high accuracies when classifying objects.

D. Discussion of the Data Set

A fine analysis between the simulated data and the real data reveals a few differences. The most notable one is related with the luminosity information provided by the LiDAR sensors, since in the simulated data, this information is not available. These data are relevant because they vary depending on the type of material on which the laser is reflecting. Aksoy *et al.* [32] used these data to improve the accuracy of a point-cloud-based DNN model in object detection and classification tasks. Since this parameter cannot be simulated using the methodology proposed in this work, these data have not been used in the DNN models implemented in this work.

One scenario in which the simulator is not able to correctly simulate the point cloud consists of environments with adverse weather conditions, such as rain, snow, or fog. The number of points generated by a real LiDAR decreases in such scenarios [2], [3], which may affect the accuracy of the DNN model. However, the LiDAR points generated by the simulator are not affected by these factors.

The main advantages of using simulated data are related to the reduction of development time when point-cloud-based data sets must be generated, and the possibility to simulate the system in the early stages of the project development. In this regard, it is possible to provide the LiDAR model or the DNN architecture that best suits the requirements of the application.

Besides, there are advantages when it is necessary to simulate objects that are difficult to record in the real world. This is the case for certain objects, which are very difficult to record. For instance, certain vehicles, such as large trucks, buses, or even certain objects such as animals. In addition, by using simulated data, it is possible to make them move through the parts of the scenario that are required. This allows to obtain a larger amount of data of these classes from many different angles, which allows to get a richer DNN model and, thus, improve its accuracy. Table II shows that the number of objects from the truck/bus and cyclist/motorbike classes are very small compared to the pedestrian and car classes. This is because in the scenario where the real data sets were generated using the LiDAR, there were few vehicles of those classes. This is not the case with the simulated data, since the number of objects of each class appearing in the scenario can be controlled along with their movements.

VI. Conclusion

This work presented an approach for automatically generating synthetic-labeled LiDAR point cloud data sets using a simulator. The generated data sets were ready to be used for DNN training. It was demonstrated that using these simulated data sets to train DNNs for point cloud object classification tasks provides almost identical results compared to training with real data. Besides, by using only simulated data for training, the manual data set generation and labeling step was avoided. This leads to considerable time and costs savings in the development stage of the traditional point cloud object classification applications where a data set must be generated and labeled. As far as the authors know, there is no work in the state of the art that obtain high accuracy results when

training DNN only with simulated point cloud data for object classification tasks.

A wide range of LiDAR models is available on the current market. Each one of them has different properties, which imply that the point clouds generated by them are different. This causes the generated data sets to be dependent on each LiDAR model. In the methodology proposed in this work, the simulated LiDAR parameters can be modified, allowing the simulated data set to be compatible with any LiDAR model. This methodology also allows the analysis of different LiDARs models using simulations in order to identify the LiDAR parameters that best fit the requirements of each application in terms of cost and accuracy, in contrast to the traditional approach where the LiDAR must first be purchased.

The simulated scenario design using the methodology proposed in this work allows a very accurate estimation of the results that will be obtained in the final deployment. In this regard, it is also possible to select the LiDAR location that provides the best accuracy results for each specific scenario before the deployment.

The DNN trained with simulated data presented in this work will be deployed in the railway level crossing shown in Fig. 1(a). The simulated scenario was designed using this railway level crossing as a reference to obtain the best accuracy results when deploying the DNN. Additionally, the simulated LiDAR sensor is the same model as the one that will be implemented in the final deployment, the Velodyne VLP-16. The system will be run on an edge IoT node and will be able to detect and classify any object that passes through a certain critical region in the surrounding of the railway level crossing.

One of the limitations of the simulated data generated with the proposed methodology is related to the luminosity data, which are provided by the LiDAR sensors. The information provided by this parameter could be relevant and, therefore, could provide higher accuracies during the object classification stage. However, the SVL simulator currently does not allow to simulate this information. As future work, a study of the reflectivity provided by different materials will be carried out to be able to provide a realistic value of luminosity in the simulated data. Besides, with the new solid-state LiDARs technology, the information provided by this parameter is richer compared with traditional mechanical LiDARs. In this regard, using this parameter for training the DNN may imply an increase in classification accuracy.

In the proposed work, the main goal was to provide object classification capabilities, however, no object detection was performed. As future work, another goal is to validate the methodology proposed in this work with a DNN that performs both object classification and detection. Besides, scenarios like those of the KITTI data set can also be simulated along with objects of their same classes. Thus, it will be possible to use the simulated data sets to compare them directly with the KITTI data set and test if similar results can be obtained when training DNNs that perform object detection and classification.

The simulator used in this article also allows the generation of RGB images around the sensor. As another future line of work, these labeled images may be used to generate simulated data sets with any class of object moving around any specific scenario since some objects and scenarios are difficult to record in the real world. The 2-D simulated images generated can be merged with the LiDAR point cloud data to improve the accuracy obtained by the DNNs.

ACKNOWLEDGMENT

The document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7345–7353.

[2] R. Heinzler, P. Schindler, J. Seekircher, W. Ritter, and W. Stork, "Weather influence and classification with automotive lidar sensors," in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2019, pp. 1527–1534.

[3] A. Filgueira, H. González-Jorge, S. Lagüela, L. Díaz-Vilariño, and P. Arias, "Quantifying the influence of rain in LiDAR performance," *Measurement*, vol. 95, pp. 143–148, Jan. 2017.

[4] T. Raj, F. H. Hashim, A. B. Huddin, M. F. Ibrahim, and A. Hussain, "A survey on LiDAR scanning mechanisms," *Electronics*, vol. 9, no. 5, p. 741, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/5/741

[5] Y. Li *et al.*, "Deep learning for LiDAR point clouds in autonomous driving: A review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3412–3432, Aug. 2021.

[6] K. Ota, M. S. Dao, V. Mezaris, and F. G. B. D. Natale, "Deep learning for mobile multimedia: A survey," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 13, no. 3s, p. 34, Jun. 2017. [Online]. Available: https://doi.org/10.1145/3092831

[7] T. Liu, A. Abd-Elrahman, J. Morton, and V. L. Wilhelm, "Comparing fully convolutional networks, random forest, support vector machine, and patch-based deep convolutional neural networks for object-based wetland mapping using images from small unmanned aircraft system," *GISci. Remote Sens.*, vol. 55, no. 2, pp. 243–264, 2018. [Online]. Available: https://doi.org/10.1080/15481603.2018.1426091

[8] C. Wisultschew, G. Mujica, J. M. Lanza-Gutierrez, and J. Portilla, "3D-LIDAR based object detection and tracking on the edge of IoT for railway level crossing," *IEEE Access*, vol. 9, pp. 35718–35729, 2021.

[9] D. J. Yeong, G. Velasco-Hernandez, J. Barry, and J. Walsh, "Sensor and sensor fusion technology in autonomous vehicles: A review," *Sensors*, vol. 21, no. 6, p. 2140, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/6/2140

[10] X. Yue, B. Wu, S. A. Seshia, K. Keutzer, and A. L. Sangiovanni-Vincentelli, "A LiDAR point cloud generator: From a virtual world to autonomous driving," in *Proc. ACM Int. Conf. Multimedia Retrieval*, 2018, pp. 458–464. [Online]. Available: https://doi.org/10.1145/3206025.3206080

[11] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3D lidar point cloud," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018, pp. 1887–1893.

[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013. [Online]. Available: https://doi.org/10.1177/0278364913491297

[13] J. Fang *et al.*, "Augmented LiDAR simulator for autonomous driving," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1931–1938, Apr. 2020.

[14] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[15] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1907–1915.

[16] B. Hurl, K. Czarnecki, and S. Waslander, "Precise synthetic image and LiDAR (PreSIL) dataset for autonomous vehicle perception," in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2019, pp. 2522–2529.

[17] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3D proposal generation and object detection from view aggregation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2018, pp. 1–8.

[18] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.

[19] "Velodyne: Velodyne LiDARs Documentation." [Online]. Available: https://velodynelidar.com/downloads/
[20] "MathWorks Team: Using Ground Truth for Object Detection." MATLAB Central File Exchange. Dec. 15, 2021. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/69180-using-ground-truth-for-object-detection
[21] J.-C. Su, M. Gadelha, R. Wang, and S. Maji, "A deeper look at 3D shape classifiers," in *Proc. Eur. Conf. Comput. Vis. (ECCV) Workshops*, Sep. 2018, pp. 1–20.
[22] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2015, pp. 922–928.
[23] E. Ahmed *et al.*, "Deep learning advances on different 3D data representations: A survey," Aug. 2018, *arXiv:1808.01462v1*.
[24] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, Nov. 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi
[25] G. Rong *et al.*, "LGSVL simulator: A high fidelity simulator for autonomous driving," in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst. (ITSC)*, 2020, pp. 1–6.
[26] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, vol. 78, 2017, pp. 1–16. [Online]. Available: https://proceedings.mlr.press/v78/dosovitskiy17a.html
[27] "Unity Technologies." Unity. [Online]. Available: https://unity.com
[28] "Unreal Engine." Epic Games. [Online]. Available: https://www.unrealengine.com
[29] A. Koubâa, *Robot Operating System (ROS)*, vol. 1. Cham, Switzerland: Springer, 2017.
[30] "Blender-OSM: Open Street Map and Terrain for Blender." MATLAB Central File Exchange. Dec. 15, 2021. [Online]. Available: https://github.com/vvoovv/blender-osm/wiki/Documentation
[31] J. Bennett, *OpenStreetMap*. Birmingham, U.K.: Packt Publ., 2010.
[32] E. E. Aksoy, S. Baci, and S. Cavdar, "SalsaNet: Fast road and vehicle segmentation in LiDAR point clouds for autonomous driving," in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2020, pp. 926–932.

Cristian Wisultschew received the B.S. and M.Sc. degrees in industrial electronics from the Universidad Politécnica de Madrid (UPM), Madrid, Spain, in 2017 and 2018, respectively, where he is currently pursuing the Ph.D. degree.

He carries out his research activity with the Centro de Electrónica Industrial, UPM. He is participating in two European H2020 Research Projects, SCOTT and InSecTT, related to real-time object detection and classification systems deployed at the edge of IoT for railway-level crossing applications. He is also participating in a Spain Government Funded Project, PLATINO, related to accelerating the processing of deep learning algorithms in embedded systems using specific deep learning neural accelerators. He has authored four papers published in international conferences and journals. His research interests are focused on sensor system integration, digital embedded systems, embedded deep learning, deep learning HW accelerators, LiDAR sensors, and Internet of Things.

Rogelio Hernández received the B.S. degree in industrial electronics from the Universidad Politécnica de Madrid (UPM), Madrid, Spain, in 2019, where he is currently pursuing the M.Sc. degree.

He carries out his research activity with the Centro de Electrónica Industrial, UPM. He is participating in InSecTT Project, which is a European H2020 Project, related to real-time object detection and classification systems deployed at the edge of IoT for railway-level crossing applications. His research interests are focused on simulated data set generation, 3-D design, point cloud object classification, and LiDAR sensors.

Carlos Pastor received the B.S. and M.Sc. degrees in industrial engineering from the Universidad Politécnica de Madrid (UPM), Madrid, Spain, in 2020 and 2022, respectively.

He carries out his research activity with the Centro de Electrónica Industrial, UPM. His research interests are focused on simulated data set generation using game engines, deep learning, and LiDAR sensors.

Jorge Portilla (Senior Member, IEEE) received the M.Sc. degree in physics from the Universidad Complutense de Madrid, Madrid, Spain, in 2003, and the Ph.D. degree in electronic engineering from the Universidad Politécnica de Madrid (UPM), Madrid, in 2010.

He was a Visiting Researcher with the Industrial Technology Research Institute, Hsinchu, Taiwan, in 2008, and also with the National Taipei University of Technology (Taipei Tech), Taipei, Taiwan, in 2018, working on wireless sensor networks hardware platforms and network clustering techniques. He is currently an Associate Professor with UPM. He carries out his research activity with the Centro de Electrónica Industrial, UPM. He has participated in more than 30 funded research projects, including the European Union FP7 and H2020 Projects, and Spain Government Funded Projects, as well as private industry funded projects, mainly related to wireless sensor networks and Internet of Things. He has numerous publications in prestigious international conferences as well as in journals with impact factor. His research interests are focused on wireless sensor networks, Internet of Things, digital embedded systems, and reconfigurable FPGA-based embedded systems.