VoxelScape: Large Scale Simulated 3D Point Cloud Dataset of Urban Traffic Environments

Khaled Saleh1  Mohammed Hossny2  Ahmed Abobakr3  Mohammed Attia4  Julie Iskander5
1Faculty of Engineering and IT, University of Technology Sydney, Australia  
2School of Engineering and IT, University of New South Wales, Australia  
3Faculty of Computers and Artificial Intelligence, Cairo University, Egypt  
4Medical Research Institute, Alexandria University, Egypt  
5Walter and Eliza Hall Institute of Medical Research, Australia  
khaled.aboufarw@uts.edu.au  mhossny@ieee.org  
a.abobakr@fci-cu.edu.eg  mohamed.hassan.attia@alexu.edu.eg  
iskander.j@wehi.edu.au

Abstract

Having a profound understanding of the surrounding environment is considered one of the crucial tasks for the reliable operation of future self-driving cars. Light Detection and Ranging (LiDAR) sensor plays a critical role in achieving such understanding due to its capability to perceive the world in 3D. Similar to 2D perception tasks, current state-of-the-art methods in 3D perception tasks rely on deep neural networks (DNNs). However, the performance of 3D perception tasks, specially point-wise semantic segmentation, is not on par with their 2D counterparts. One of the main reasons is the lack of publicly available labelled 3D point cloud datasets (PCDs) from 3D LiDAR sensors. In this work, we are introducing the VoxelScape dataset, a large-scale simulated 3D PCD with 100K annotated point cloud scans. The annotations in the VoxelScape dataset includes both point-wise semantic labels and 3D bounding boxes labels. Additionally, we used a number of baseline approaches to validate the transferability of VoxelScape to real 3D PCD for two challenging 3D perception tasks. The promising results have shown that training DNNs on VoxelScape boosted the performance of the 3D perception tasks on the real PCD. The VoxelScape dataset is publicly available at https://voxel-scape.github.io/dataset/

1 Introduction

Current self-driving cars rely on a number of on-board sensors to have a deep situational awareness of its surrounding. One of the key sensors that self-driving cars rely on is the LiDAR sensor. Unlike other optical sensors such as visible and IR cameras, LiDAR sensors are not affected by direct sunlight and do not need an external illumination source to operate. Therefore, the majority of the self-driving cars, tested on the roads nowadays, utilise LiDAR sensors in versatile perception tasks such as 3D semantic scene understanding and 3D object detection and tracking [Zhang et al. 2018, 2020]. Sun et al. [2019]. DNNs achieved current state-of-the-art results specially on 2D perception tasks, due to the availability of a large amount of labelled datasets, which DNNs exploit for training and evaluation [Shi et al. 2020], [Cortinhal et al. 2020]. However, for 3D perception tasks on PCD of LiDAR sensors, the number of publicly available annotated datasets in the context of autonomous driving is quite scarce. The reason for that is the difficulty and time-consuming

nature of the annotation process for LiDAR PCD, specially for tasks such as 3D point-wise semantic
segmentation. For instance, the time required to manually label the 3D points of a tile of 100m by
100m of an urban traffic environment is on average is 4.5 hours Behley et al. [2019]. Thus, recent
works started to explore the usage of game engines and 3D computer graphics software in order to
simulate and render annotated synthetic PCD of urban traffic environments, as shown in Dosovitskiy
et al. [2017], Griffiths and Boehm [2019], Yue et al. [2018].

While the synthetic PCD are used for the validation of trained machine learning models on real PCDs,
they still however suffer from some shortcomings. One of these shortcomings, is the lack of the
key properties that exists in real PCD coming from physical LiDAR sensors such as the returned
laser beams’ intensity/reflectivity values Dosovitskiy et al. [2017]. Another shortcoming, is the
negligence of simulating critical objects and scenarios which are of a great importance to self-driving
cars such as vulnerable road users (pedestrians, cyclists,... etc.) Griffiths and Boehm [2019] and
construction sites Yue et al. [2018]. Similarly, the small number of publicly available datasets of
annotated 3D PCD suffer from the lack of scenario diversity especially for the less frequent scenes
such as construction sites.

In this work, we tackle some of these challenges, which exist in both synthetic PCD from simulated
traffic environments and real PCDs from physical LiDAR sensors. We introduce a large scale and
diverse simulated 3D PCD in urban traffic environment, the VoxelScape dataset. In VoxelScape, we
provide more than 100K sequential LiDAR scans annotated with both 32 point-wise semantic labels
and 3D bounding boxes of 8 unique object classes.

To the best of our knowledge, this is considered the largest public 3D PCD with point-wise semantic
annotation across both simulated and real urban traffic environment datasets. Overall, the contribution
of this work is as follows:

• A large scale 3D PCD of simulated urban traffic environments with full detailed point-wise
semantic segmentation labels and 3D bounding boxes (BBox) annotation in 360°.

• Realistic simulation of physical LiDAR sensor properties (i.e intensity/reflectivity) and
diverse simulation of less-frequent scenarios that are missing in real 3D PCDs in urban
traffic environments

• We additionally provide an evaluation of the applicability of synthetic PCDs in real scenarios
captured by physical 3D LiDAR sensors for two 3D perception tasks for self-driving cars.

The remainder of the paper is structured as follows. In Section 2, a brief overview of related work
from the literature is presented. The description of the pipeline utilised in generating our diverse
VoxelScape dataset is outlined in Section 3, along with details of our VoxelScape dataset. The
evaluation of state-of-the-art methods for point-wise semantic segmentation and 3D object detection
of 3D PCD on our VoxelScape dataset is described in Section 4 and Section 5. Finally, we conclude
our paper in Section 6.

2 Related Work

Thanks to the plethora of the publicly available 2D image datasets, there has been a huge leap in the
performance of 2D computer vision tasks such as image classification and semantic segmentation Lin
et al. [2014], Krizhevsky et al. [2017]. On the other hand, in the 3D computer vision field, the number
of available 3D LiDAR datasets is not quite on a par with their 2D counterparts specially in the
context of self-driving cars and urban traffic environment.

In Table 1, we listed a number of the relevant 3D LiDAR datasets which were captured in urban traffic
environments and made publicly available. We categorised the datasets into two main categories
based on the capturing procedure, whether it was using a real physical sensor (real data) or a simulated
virtual sensor (synthetic data). In the following, we will discuss some of the datasets under each one
of the aforementioned categories.

2.1 Real Datasets

One of the early 3D LiDAR real datasets was the Freiburg LiDAR dataset Behley et al. [2012]. which
was captured in an urban traffic environment inside the campus of University of Freiburg. The dataset
contains a total of 77 3D LiDAR scans captured using a SICK LMS LiDAR sensor mounted on a pan-tilt module. The dataset was manually annotated with point-wise semantic labels with a total number of 11 classes. Another 3D LiDAR dataset with point-wise annotation is the semantic3D dataset [Hackel et al. 2017]. It was captured using a Terrestrial Laser Scanner (TLS) in an urban traffic environment, which is commonly used in surveying applications for its highly dense PCD. The dataset contains only 30 scans with point-wise annotation for 8 classes. The majority of class labels belong to static objects of an urban city (such as terrain, building vegetation, etc) and without any labels for dynamic objects such as pedestrians and cyclists.

In 2012, the KITTI benchmark was released which is considered the first benchmark for a number of perception tasks focused mainly on self-driving cars. The KITTI contained a 3D LiDAR dataset for the task of 3D object detection which consisted of roughly 15K 3D LiDAR scans captured using a Velodyne HDL-64E sensor. The dataset had 3D BBox annotation for three classes, namely cars, pedestrians and cyclists. Similar to KITTI, both the nuScenes [Caesar et al. 2020] and Waymo [Sun et al. 2020] datasets contained 3D BBox annotations for 3D LiDAR scans. These two datasets were the first largest datasets released by two major self-driving car companies (Motional and Waymo respectively), with 40K scans in nuScenes and 230K in Waymo.

Recently, three larger 3D LiDAR datasets were released with point-wise semantic annotations, namely SemanticKITTI [Behley et al. 2019], SemanticPOSS [Pan et al. 2020] and nuScene [Caesar et al. 2020]. Both of the SemanticKITTI and nuScenes datasets contained fairly large number of class labels with 28 and 16 labels for SemanticKITTI and nuScenes respectively. However, the number of labels of vulnerable road users such as pedestrians and cyclists/motor-bikers were quite small in comparison to other class labels. For example, in the SemanticKITTI dataset, the total number of pedestrians and bicyclists objects are roughly 900 and 350 respectively, whereas the total number of cars is roughly 10K Behley et al. [2019]. Moreover, both the SemanticKITTI and the SemanticPOSS datasets did not contain any of the critical scenarios that are crucial for safe and reliable situational awareness of the self-driving cars in urban traffic environments such as roadwork scenarios or heavily cluttered spaces with pedestrians. Furthermore, the nuScenes dataset contains much sparser point cloud scans in comparison to the SemanticKITTI as it was captured using a Velodyne LiDAR sensor with only 32 vertical channels.

2.2 Synthetic Datasets

With recent work on domain adaptation of DNNs models trained on synthetic datasets, it was shown that synthetic datasets could help in boosting the performance of DNNs models when tested on real datasets [Ros et al. 2016], [Saleh et al. 2019], [Wu et al. 2019]. Ros et al. [Ros et al. 2016] introduced the SYNTHIA dataset which is a synthetic 2D image dataset for the semantic segmentation task. When DNN models were trained on SYNTHIA with parts from a real semantic segmentation dataset, the trained model achieved more accurate results when tested on real datasets in comparison to those models trained solely on a real dataset. On the other hand, in 3D perception tasks, Saleh et al. [Saleh et al. 2019] obtained higher average precision scores when training a model on birds eye view images from both synthetic 3D PCD and KITTI 3D PCD for the task of 3D car detection. The trained DNN

Table 1: Comparison between the publicly available 3D PCDs with annotations. Our VoxelScape dataset is the largest dataset with both point-wise and 3D bounding box (3D BBox) annotations. Number of points is in millions, Number of classes of point-wise semantic labels annotations/3D BBox object class annotations. (-) indicates that no information is provided or not available.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Dataset</th>
<th>No. Scans</th>
<th>No. Points</th>
<th>Annotation</th>
<th>No. Classes</th>
<th>Sequential</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td>Semantic3D [Hackel et al. 2017]</td>
<td>30</td>
<td>4009</td>
<td>point-wise</td>
<td>8/-</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Freiburger [Behley et al. 2019]</td>
<td>77</td>
<td>1.1</td>
<td>point-wise</td>
<td>1/-</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Sydney Urban [De Deuge et al. 2013]</td>
<td>588</td>
<td>-</td>
<td>point-wise</td>
<td>14/-</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>nuScenes [Caesar et al. 2020]</td>
<td>40000</td>
<td>2780</td>
<td>point-wise+3D BBox</td>
<td>16/23</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>WaymoPreSIL [Hurl et al. 2019]</td>
<td>230000</td>
<td>40710</td>
<td>3D BBox</td>
<td>-4/4</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>SemanticKITTI [Behley et al. 2019]</td>
<td>43552</td>
<td>4549</td>
<td>point-wise</td>
<td>28/-</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>SemanticPOSS [Pan et al. 2020]</td>
<td>2988</td>
<td>216</td>
<td>point-wise</td>
<td>14/-</td>
<td>✓</td>
</tr>
<tr>
<td>Synthetic</td>
<td>GTA-LiDAR [Yue et al. 2018]</td>
<td>8585</td>
<td>-</td>
<td>point-wise</td>
<td>3/-</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>SynthCity [Hackel et al. 2017]</td>
<td>75000</td>
<td>367.9</td>
<td>point-wise</td>
<td>9/-</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>PreSIL [Hurl et al. 2019]</td>
<td>50000</td>
<td>-</td>
<td>3D BBox</td>
<td>-12/12</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>VoxelScape (ours)</td>
<td>100000</td>
<td>13340</td>
<td>point-wise+3D BBox</td>
<td>32/9</td>
<td>✓</td>
</tr>
</tbody>
</table>
model has shown more promising results when trained on both synthetic and real 3D PCD similar to SYNTHIA DNN models. That being said, the number of synthetic 3D LiDAR datasets with annotated point-wise and/or 3D BBox annotations are still rather limited. Additionally, the publicly available synthetic 3D LiDAR datasets are not diverse with their point-wise semantic annotations. For instance, the GTA-LiDAR and PreSIL datasets Yue et al. (2018), Hurl et al. (2019), which were obtained using a plugin interfaced with the famous Grand Theft Auto (GTA) game to simulate a virtual 3D LiDAR sensor based on ray casting, only contains labels for 3 classes, namely cars, pedestrians and cyclists. Recently, the SynthCity dataset Hackel et al. (2017) was presented, which include 75K 3D point cloud scans with point-wise annotations of 9 class labels of infrastructure objects of urban traffic environments excluding vulnerable road users such as pedestrians and cyclists. The dataset was generated using a sensor simulation plugin (Blensor) for the open source 3D computer graphics software, Blender Gschwandtner et al. (2011).

3 The VoxelScape Dataset

Unlike other synthetic 3D LiDAR datasets, our introduced VoxelScape dataset contains a large scale 3D point cloud scans of more than 100K scans with full-detailed point-wise semantic annotations for 32 class labels. Additionally, our synthetic 3D PCD was generated using an emulation of a Velodyne HDL-64E 3D LiDAR sensor which enabled us to not only obtain (x, y, z) coordinates of the points like other synthetic 3D LiDAR datasets but also obtain the intensity values for each returned laser beam hitting an object in the scene. It is also worth noting that unlike the SynthCity dataset, the rendering for each scan in 360° in our dataset takes only roughly 2 seconds rather than the 330 seconds in SynthCity dataset. Additionally, in contrast to real 3D LiDAR datasets, our VoxelScape datasets not only contains a larger number of point-wise semantic labels but it also contains 3D BBox annotations for 9 object classes. Furthermore, as described later, our dataset simulates some corner scenarios which are missing in the available real 3D LiDAR datasets. Next, we will describe the pipeline we utilised for generating our VoxelScape dataset. Then, we will provide a thorough discussion of the details of the dataset and the provided annotations.

3.1 LiDAR Simulation

In this work, we utilised the equirectangular UV spherical mapping method presented by Hossny et al. at Hossny et al. (2020). Their method unfolds in three stages. Firstly, a 360 degrees equirectangular depth map is rendered. Secondly, the rendered depth map is texture mapped on a sphere using spherical UV coordinates to produce a spherical point cloud. Finally, the spherical point cloud is carved based on the depth values in the rendered depth map. More information about these three stages along with the intensity simulation calculations can be found in the supplementary material.

Figure 1: A sample 3D (left) and top views (middle) layout of the procedurally generated urban city with labelled point cloud accumulated along the vehicle path. The highlighted parts on the right showcase samples of the label annotation and calculated intensity. Bounding boxes (right) for vehicles, cyclists and pedestrians as well as roadwork (concrete barriers and metal fences) are filtered based on the distance from the sensor.
3.2 Procedural Urban City Generation

According to Compton [2019], procedural content generation (PCG) has become a common technique in computer games. The rationale behind using PCG in computer games also varies across different use-cases starting with cost reduction and ending with producing infinite game play experiences. There are several schools of thought about PCG but perhaps the most common one is based on stacking parameterised building blocks where the values of different parameters are chosen randomly according to a statistical distribution [Compton 2019]. This approach is particularly useful for automation rather than presenting infinite experiences. In this work, we chose this approach to generate the urban scenes in three major stages. First, a layout of the city is generated where roads and intersections are laid out. Second, the laid out roads are used to generate buildings on both sides. Finally, the road segments are populated with agents (e.g. pedestrians, cyclists and vehicles), vegetation (e.g. trees and shrubs) and road signs. Figure 1 shows a sample of the procedurally generated urban city with labels and the associated labelled point cloud and the reflectance intensity of the LiDAR points when projected on different surfaces.

3.2.1 Urban Scene Generation

The city layout was derived using recursive partitioning of a 2D rectangular area with the size of 320 m$^2$ using quad tree decomposition with random number generator. The number generator decides whether to subdivide a sub-rectangle while maintaining a maximum and minimum dimensions of each building block. The resulting partitioning map then serves as the blueprint for placing the roads and intersections. The buildings were placed alongside the laid roads and they were randomly selected from a library of 3D building assets. They were subjected to discrete rotation of 0, 90, 180 degrees around the z-axis (up) while grass patches and pedestrians were randomly rotated with angles in range of $[0, 180]$ degrees. Each 3D building is equipped with areas to spawn trees and street props (e.g. mailboxes, trash cans, seats, phone booths). We also included two different special blocks to allow for a green area with pedestrians. The green area is another procedurally generated terrain with random deformation and grass patches. Pedestrian spawning follows a more articulate procedural generation which takes place on two stages allowing to choose the population density and then randomly choosing digital manikins from a library of assets. As shown in Figure 1-right, we designed two road portions to simulate normal and roadwork scenarios. In normal scenarios, each road portion is subdivided into 7 areas for spawning trees, shrubs, pedestrians, cyclists, vehicles, road signs, and lamps. For roadwork scenarios, the vehicle and cyclist spawning areas are merged to spawn a road work area. The roadwork area itself is subdivided into three areas which spawn different kinds of barriers (e.g. concrete, metal fence and cones). The spawning of different 3D assets is done randomly according to a selected seed.

3.2.2 Seed Selection

Each generated sequence (city), is uniquely identifiable by an initial seed. We chose 100 different 32-bits prime numbers with a 50% zeros to ones ratio and maximum 3 consecutive similar digits as our sequence seeds. This seed is then used to generate subsequent unique seeds for the random number generators controlling the city layout, selected assets and as well as their transformation matrix.

3.2.3 Annotation

Labels were generated on two levels for each 3D asset in the generated scene. First, an object label is assigned to the overall asset which is defined by its bounding box. Second a sub-label is also assigned at the mesh level to facilitate more articulation of the different parts of the asset. For example, a cyclist or a biker are assigned sub-labels different from the bicycle or the motorcycle. In the implementation, the label was assigned based on the object type characterised by the object name prefix. Sub-labels, on the other hand, were assigned using the material identifier of different parts of the mesh. The use of materials also allowed us to implement different reflectivity response as described above. For example, the body and tyres of a vehicle reports different levels of reflective intensity. Another critical example is the ground where road lines are more reflective than the asphalt materials.
Figure 2: Label distribution of our VoxelScape dataset. The number of labelled points per class is shown.

Table 2: Number of 3D BBox annotations for each of the 9 object classes, which exist in our VoxelScape dataset.

<table>
<thead>
<tr>
<th>Object Class</th>
<th>No. BBox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pedestrian (Adult)</td>
<td>537850</td>
</tr>
<tr>
<td>Pedestrian (Child)</td>
<td>132346</td>
</tr>
<tr>
<td>Cyclist</td>
<td>101342</td>
</tr>
<tr>
<td>Motorcyclist</td>
<td>28825</td>
</tr>
<tr>
<td>Car</td>
<td>88845</td>
</tr>
<tr>
<td>Truck</td>
<td>35843</td>
</tr>
<tr>
<td>Bus</td>
<td>3774</td>
</tr>
<tr>
<td>Construction-Barrier</td>
<td>6361</td>
</tr>
<tr>
<td>Construction-Cone</td>
<td>14670</td>
</tr>
<tr>
<td>Total</td>
<td>949856</td>
</tr>
</tbody>
</table>

3.3 Dataset Overview

Given the aforementioned data-generation pipeline, we obtained our VoxelScape dataset, which contains a total of 100 sequences (each with 1000 point cloud scans) covering different parts of our procedurally generated city. The generated cities include common diverse scenarios in urban traffic environments. In total, we have 100K point cloud scans with a total number of 13340 million points. Each scan contains four main components, which are the \((x, y, z)\) coordinates of the point and its reflection intensity \(i\). Furthermore, each scan is annotated with two different annotations, namely point-wise semantic labels and 3D object BBox. The number of semantic labels is 32 class labels (shown in Figure 2), which covers a wide range of elements found in any typical urban traffic environment. The 3D BBox annotations covers 9 different class objects, namely cars, adult-pedestrians, child-pedestrians, cyclists, motorcyclists, truck, bus, construction-cones and construction-barriers. In Figure 2, the distribution of the 32 point-wise semantic labels is presented. Similar to real PCDs [Behley et al. [2019], Pan et al. [2020]], the majority of class labels belong to ‘building’, ‘road’ and ‘sidewalk’ classes. The number of 3D BBox annotations per class is presented in Table 2. The dataset contains a large number of BBox (approximately 950K BBox) with a focus on vulnerable road users (pedestrians, cyclists, ... etc.) which is a unique characteristic of our VoxelScape dataset that is missing in other real PCDs [Geiger et al. [2012], Caesar et al. [2020]].

4 VoxelScape for Point-wise Semantic Segmentation Task

Since our end goal is to bridge the gap between synthetic and real PCDs, for 3D perception tasks. Therefore, in this section, we are going to validate the applicability and the realism of our presented
VoxelScape dataset for real 3D perception tasks. In order to do so, we chose one of the challenging tasks in the 3D perception domain which is the point-wise semantic segmentation of PCD. The VoxelScape dataset was used and the results were analysed to calculate the improvement (if any exist) in the performance of the methods developed for this task. This strategy is motivated by the promising work in [Ros et al., 2016]. In their work, the DNN models trained for 2D image segmentation with the synthetic RGB images had enhanced the performance when tested on real RGB images. Similarly, in our case, we will be relying on baseline DNN models to carry out number of experiments to evaluate the performance of these models when trained using our VoxelScape dataset for the point-wise semantic segmentation task. In the following, we will first start with presenting the baseline DNN models that will be utilised in our experiments. Then, we will discuss the setup for the experiments and analyse their results.

4.1 Baseline DNN Models

The DNN models have become the state-of-the art methods for the point-wise semantic segmentation of the PCD [Hu et al., 2020; Milioto et al., 2019; Wu et al., 2019]. Generally speaking, there are two different types of DNNs that it comes to point-wise semantic segmentation. The first type is projection-based networks where it utilises 2D CNNs architectures by projecting the PCD to 2D images. The second type is point-based networks where it works directly on raw PCD by learning per-point local features. From the first type we chose two models as baseline: SqueezeSegV2 model [Wu et al., 2019] and DarkNet53 model [Milioto et al., 2019]. While from the second type, we chose the RandLANet model [Hu et al., 2020] as our third baseline. SqueezeSegV2 [Wu et al., 2019] is the first baseline and is one of the commonly utilised models for the task of point cloud-based segmentation due to its real-time inference and its relative accurate results [Zhao et al., 2020; Balado et al., 2019]. The architecture of SqueezeSegV2, as the name implies, is build on the SqueezeSegV1 architecture [Wu et al., 2018], which takes the point cloud data as a spherically projected 2D image as input. The 2D image consists of 5 channels namely: range, \( x \), \( y \), \( z \), and intensity of the input point cloud. The second DNN model is the DarkNet53 model [Milioto et al., 2019], which was one of the well-performing DNN model for point-wise semantic segmentation over the SemanticKITTI dataset [Behley et al., 2019]. The underlying architecture of DarkNet53 is a fully ConvNet architecture with Yolov3’s backbone architecture DarkNet53 [Redmon and Farhadi, 2018]. We utilised the implementation of DarkNet53 that was introduced in Milioto et al. [2019], which projects the 360\(^\circ\) point cloud scan and unwrap it into 2D image with 5 channels that corresponds to range, \( (x, y, z) \) coordinates and intensity values of each point in the scan similar to the SqueezeSegV2 model. The final baseline DNN model is the RandLANet model [Hu et al., 2020], which consists only of shared multilayer perceptrons (MLPs) that can learn local inter-point features within randomly sampled points from the original raw PCD.

4.2 Experimental Results

In our validation study, we carried out two experiments in order to validate the utility of our VoxelScape dataset. In our first experiment, our goal is to assess whether our simulated intensity values (that are missing from all synthetic LiDAR datasets in the literature) would make a difference in the overall performance of the trained DNN models. On the other hand, in our second experiment, our goal is to evaluate the generalisation capabilities of the trained baseline DNN models on our VoxelScape dataset, when they are both tested directly on real PCD, and when their weights are utilised to fine-tune the DNN models on real PCD. Fine-tuning DNN models is considered one form of transfer learning, which was shown to be helping in both reducing the time required for DNN models to converge and boosting its overall performance as it was shown in [Yosinski et al., 2014; Ros et al., 2016]. Due to the content page limit, we will move the first experiment to the supplementary material and will focus in the following on the second experiment. We used the, recently released, real point cloud scans from SemanticKITTI [Behley et al., 2019] for our experiments. The justification of this choice is that SemanticKITTI is considered (to the best of our knowledge) the second largest PCD with point-wise annotations after our proposed VoxelScape dataset. In order to conform with the number of labels exist in the SemanticKITTI evaluation benchmark (which are only 19 classes defined in Table 3), we only trained our baseline models on their corresponding labels in our VoxelScape dataset. The SemanticKITTI consists of 22 sequences divided into three parts (from seq. 00 to 10 except seq. 08 is for training; seq. 08 for validation and from seq. 11 to 21 is for testing)
Table 3: Evaluation of the baseline DNN models on the validation split of the SemanticKITTI Behley et al. [2019]. Each baseline model has two versions, one is fine-tuned using our VoxelScape (VS-FT) and another one without the fine-tuning.

<table>
<thead>
<tr>
<th>Approach</th>
<th>mIoU</th>
<th>road</th>
<th>sidewalk</th>
<th>parking</th>
<th>building</th>
<th>car</th>
<th>truck</th>
<th>bus</th>
<th>motorcycle</th>
<th>other-vehicle</th>
<th>vegetation</th>
<th>tree</th>
<th>truck</th>
<th>fence</th>
<th>pole</th>
<th>traffic sign</th>
</tr>
</thead>
<tbody>
<tr>
<td>SqueezeSeqV2 Wu et al. 2019</td>
<td>35.1</td>
<td>89.6</td>
<td>72.7</td>
<td>32.2</td>
<td>0.9</td>
<td>68.8</td>
<td>78.8</td>
<td>15.9</td>
<td>13.1</td>
<td>17.6</td>
<td>18.1</td>
<td>70.6</td>
<td>22.9</td>
<td>67.5</td>
<td>8.1</td>
<td>27.2</td>
</tr>
<tr>
<td>SqueezeSeqV2 (VS-FT) Wu et al. 2019</td>
<td>36.5</td>
<td>90.1</td>
<td>73.3</td>
<td>35.0</td>
<td>0.5</td>
<td>69.7</td>
<td>79.9</td>
<td>26.0</td>
<td>11.9</td>
<td>21.6</td>
<td>16.6</td>
<td>71.6</td>
<td>26.1</td>
<td>67.6</td>
<td>12.4</td>
<td>30.7</td>
</tr>
<tr>
<td>DarkNet53 Milioto et al. 2019</td>
<td>36.5</td>
<td>90.1</td>
<td>69.4</td>
<td>14.7</td>
<td>0.0</td>
<td>74.8</td>
<td>80.9</td>
<td>17.9</td>
<td>8.1</td>
<td>4.0</td>
<td>9.6</td>
<td>78.8</td>
<td>28.0</td>
<td>70.9</td>
<td>8.3</td>
<td>33.8</td>
</tr>
<tr>
<td>DarkNet53 (VS-FT) Milioto et al. 2019</td>
<td>39.8</td>
<td>92.0</td>
<td>76.2</td>
<td>45.2</td>
<td>0.1</td>
<td>72.3</td>
<td>80.6</td>
<td>39.0</td>
<td>13.0</td>
<td>20.8</td>
<td>20.4</td>
<td>75.1</td>
<td>31.2</td>
<td>13.6</td>
<td>38.8</td>
<td>0.0</td>
</tr>
<tr>
<td>RandLANet Hu et al. 2020</td>
<td>52.9</td>
<td>91.2</td>
<td>75.9</td>
<td>42.1</td>
<td>0.1</td>
<td>87.8</td>
<td>91.9</td>
<td>60.0</td>
<td>12.2</td>
<td>29.0</td>
<td>44.4</td>
<td>84.4</td>
<td>59.5</td>
<td>74.1</td>
<td>51.5</td>
<td>67.6</td>
</tr>
<tr>
<td>RandLANet(VS-FT) Hu et al. 2020</td>
<td>53.9</td>
<td>93.0</td>
<td>77.3</td>
<td>45.6</td>
<td>1.0</td>
<td>85.8</td>
<td>93.4</td>
<td>71.9</td>
<td>15.5</td>
<td>34.6</td>
<td>47.4</td>
<td>82.4</td>
<td>60.3</td>
<td>72.4</td>
<td>55.7</td>
<td>72.1</td>
</tr>
</tbody>
</table>

In Milioto et al. [2019], they have a number of variants for both the DarkNet53 and SqueezeSegV2 architectures. The difference between these variants are the resolutions of the projected 3D LiDAR point cloud into the input 2D image. The resolutions are 2048(W) X 64(H), 1024(W) X 64(H) and 512(W) X 64(H). For computational purposes, we chose to utilise the 1024(W) X 64(H) resolution for both the DarkNet53 and the SqueezeSegV2 models in our experiments.

In Table 3, we report the results of our second experiment where we evaluate the performance of the fine-tuned baseline DNN models using our VoxelScape dataset on the SemanticKITTI dataset (namely SqueezeSegV2 (VS-FT), DarkNet53 (VS-FT) and RandLANet(VS-FT)). Additionally, we also evaluate the same baseline DNN models when trained only on the training split of the SemanticKITTI without any fine-tuning from the trained DNN models on our VoxelScape dataset. Similar to Behley et al. [2019], Qi et al. [2017], Milioto et al. [2019], the evaluation metric we used is mIoU. The results show that the Darknet53 (VS-FT) model that was fine-tuned based on the weights of the pre-trained Darknet53 (+INT) model on our VoxelScape dataset, achieved a total mIoU score of 39.8% and outperformed the Darknet53 model with a significant margin which scored only 36.5%. On the other hand, both the SqueezeSegV2 (VS-FT) and the RandLANet(VS-FT) models achieved a total mIoU score of 36.5% and 53.9% respectively which is slightly better than their counterpart models without the fine-tuning. The main deduction from the results in Table 3 is that the fine-tuned models using our VoxelScape dataset have achieved higher mIoU scores than their counterparts model without the fine-tuning. This can be further demonstrated by the mIoU scores on the vulnerable road users (persons, bicyclists,...etc) which we have multiple instances of them in our VoxelScape dataset, which in return helped in making the fine-tuned DNN models scored better IoU scores when compared with the DNN models without any fine-tuning. More details about the experiments setup can be found in the supplementary material.

5 VoxelScape for 3D Object Detection Task

In order to further demonstrate the utility of our VoxelScape dataset for real 3D perception tasks. In this section, we will be investigating the performance of our VoxelScape dataset when utilised for the 3D object detection task on real LiDAR point cloud dataset.

5.1 Baseline DNN Model

The baseline DNN we will be relying on for the 3D object detection task will be the LiDAR-based 3D object detection method, PointPillars Lang et al. [2019]. PointPillars is one of the best performing and fastest 3D object detectors on real LiDAR PCD datasets such as KITTI Geiger et al. [2012] and nuScenes Caesar et al. [2020]. More details about the baseline setup can be found in the supplementary material.

5.2 Experimental Results

Similar to the second experiment from the point-wise semantic segmentation task, we would like to evaluate the generalisation capabilities of the trained baseline PointPillars model on our VoxelScape dataset, when it is both tested directly on real PCD, and when its weights are used to fine-tune the DNN model on real PCD. In this experiment, we will be utilising the real PCD from KITTI.
Table 4: Evaluation of the 3D detection results on the validation split of KITTI [Geiger et al., 2012].

<table>
<thead>
<tr>
<th>Method</th>
<th>Training Data</th>
<th>Car mAP Easy</th>
<th>Car mAP Moderate</th>
<th>Car mAP Hard</th>
<th>Pedestrian mAP Easy</th>
<th>Pedestrian mAP Moderate</th>
<th>Pedestrian mAP Hard</th>
<th>Cyclist mAP Easy</th>
<th>Cyclist mAP Moderate</th>
<th>Cyclist mAP Hard</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointPillars [Lang et al., 2019]</td>
<td>VoxelScape</td>
<td>14.96</td>
<td>21.01</td>
<td>17.68</td>
<td>11.61</td>
<td>10.61</td>
<td>10.43</td>
<td>21.29</td>
<td>17.73</td>
<td>16.97</td>
</tr>
<tr>
<td>PointPillars</td>
<td>KITTI</td>
<td>45.55</td>
<td>74.67</td>
<td>62.63</td>
<td>57.16</td>
<td>41.83</td>
<td>38.61</td>
<td>36.51</td>
<td>54.64</td>
<td>45.79</td>
</tr>
<tr>
<td>PointPillars</td>
<td>KITTI (Aug)</td>
<td>52.74</td>
<td>82.08</td>
<td>70.89</td>
<td>66.70</td>
<td>43.80</td>
<td>40.22</td>
<td>38.14</td>
<td>69.19</td>
<td>56.77</td>
</tr>
<tr>
<td>PointPillars</td>
<td>KITTI (VS-FT)</td>
<td>58.06</td>
<td>80.35</td>
<td>71.73</td>
<td>65.96</td>
<td>60.71</td>
<td>54.53</td>
<td>52.84</td>
<td>70.07</td>
<td>60.76</td>
</tr>
</tbody>
</table>

dataset [Geiger et al., 2012] for the 3D object detection task. The reason for choosing the KITTI dataset is because it was captured using a Velodyne HD-64E 3D LiDAR which is similar to our simulated LiDAR sensor. As we have shown in Table 1, the KITTI dataset has only 3D Bbox annotations for three object classes, namely Cars, Pedestrians and Cyclists. In order to conform with KITTI, we only trained our baseline model, PointPillars, on the aforementioned class labels in our VoxelScape dataset for the 3D object detection task. In total, we have trained four PointPillars models with the same architecture configuration (more about it can be found in the supplementary material). The first model is using our VoxelScape dataset as its sole input training data. Whereas, the three other models are utilising the first 3712 point cloud scans from the training split of the KITTI dataset as their input training data. The only difference between the second and fourth model is that, one model ‘KITTI (VS-FT)’ was fine-tuned using the weights from the trained PointPillars on our VoxelScape dataset, while the other ‘KITTI’ was not. On the other hand, the third model ‘KITTI (Aug)’ was trained with some additional data augmentation techniques on the original PCD that was not present in any of the other three baseline models. These data-augmentation techniques includes: object sampling, object range filter and random point shuffling. In Table 4, we evaluate the performance of the trained four baseline PointPillars models on the 3769 point cloud scans of the validation subset (which is the rest of the 7481 scans from the training split) of the KITTI 3D object detection benchmark. We report the results according to the KITTI’s validation criteria which is the average precision (AP) in 3D. Since the KITTI dataset has also further annotated each 3D BBox with one of three difficulty levels (easy, moderate, hard), we have categorised the AP scores in Table 4 for each class into those three difficulty levels. Additionally, we have reported the overall mean value over the AP of all classes for all difficulty levels in the third column (mAP). Similar to the 3D semantic segmentation task, we can notice that in the 3D object detection task, the last baseline DNN model in Table 4 when was fine-tuned using the weights from the trained model on our VoxelScape dataset, the performance was boosted by more than 12% in the mAP score when compared with the model that was not fine-tuned. Additionally, it can be noticed also that our fine-tuned model outperformed the model that was utilising data-augmentation techniques during the training by more than 5% in mAP.

6 Conclusion

In this work we have presented the VoxelScape dataset, a novel large scale simulated PCD of diverse urban traffic environment. The dataset is provided with 32 point-wise semantic labels and 3D bounding boxes annotations of 9 object classes for 3D perception tasks in the context of self-driving vehicles. Our unique efficient 3D LiDAR simulation approach combined with procedural urban city generation enabled us to achieve 100K point cloud scans of articulated scenes with a total of 13340 million annotated points. In our experiments, we validated the realism and utility of the proposed dataset for two 3D perception tasks using the 3D point cloud scans. We trained baseline DNNs on our VoxelScape dataset and fine-tuned them with real PCD. The results have shown that our simulated intensity values helped in improving the accuracy of DNN models by more than 10%. Additionally, fine-tuned DNN models using our VoxelScape dataset achieved both higher mean intersection over-union and mean average precision scores over the DNN models that were not utilising it. In our future work, we will focus on synthesising more corner-case scenarios in highway traffic scenes (such as crossing wild animals). Furthermore, we will explore more domain-adaptation techniques to further decrease the gap between synthetic and real PCDs for other 3D perception tasks.
References


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]. Please check the conclusion section.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]. Since our VoxelScape dataset is synthetically generated and all its assets are simulated ones, so we are confident that our paper does not violate any ethical standards.

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]. Some these details are described in Section 4.2 and the rest are included in the supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]. Those are included as part of the supplemental material.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]. All the assets used in generating our VoxelScape dataset was either created/designed from scratch (city layout, etc.) or purchased directly (different car/bus/truck/motorcycle models.. etc.) from 3D asset stores.
   (b) Did you mention the license of the assets? [N/A]. See above.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]. The 3D assets themselves aren’t included, but our fully generated VoxelScape dataset is publicly available from the link in the Abstract.
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [N/A]. All the agents in our dataset are simulated ones.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]. All the agents in our dataset are simulated ones.

5. If you used crowdsourcing or conducted research with human subjects...
(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]