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

Khaled Saleh[1] **Mohammed Hossny**[2] **Ahmed Abobakr**[3]
Mohammed Attia[4] **Julie Iskander**[5]

[1]Faculty of Engineering and IT, University of Technology Sydney, Australia
[2]School of Engineering and IT, University of New South Wales, Australia
[3]Faculty of Computers and Artificial Intelligence, Cairo University, Egypt
[4]Medical Research Institute, Alexandria University, Egypt
[5]Walter 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

Submitted to the 35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks. Do not distribute.

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. [1]Number of points is in millions, [2]Number of classes of point-wise semantic labels annotations/3D BBox object class annotations. (-) indicates that no information is provided or not available.

Data Type	Dataset	No. Scans	No. Points[1]	Annotation	No. Classes[2]	Sequential
Real	Semantic3D Hackel et al. [2017]	30	4009	point-wise	8/-	X
	Freiburg Behley et al. [2012]	77	1.1	point-wise	11/-	X
	Sydney Urban De Deuge et al. [2013]	588	-	point-wise	14/-	✓
	KITTI Geiger et al. [2012]	14999	1799	3D BBox	-/3	X
	nuScenes Caesar et al. [2020]	40000	2780	point-wise+3D BBox	16/23	✓
	Waymo Sun et al. [2020]	230000	40710	3D BBox	-/4	✓
	SemanticKITTI Behley et al. [2019]	43552	4549	point-wise	28/-	✓
	SemanticPOSS Pan et al. [2020]	2988	216	point-wise	14/-	✓
Synthetic	GTA-LiDAR Yue et al. [2018]	8585	-	point-wise	3/-	✓
	SynthCity Hackel et al. [2017]	75000	367.9	point-wise	9/-	✓
	PreSIL Hurl et al. [2019]	50000	-	3D BBox	-/12	✓
	VoxelScape (ours)	100000	13340	point-wise+3D BBox	32/9	✓

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 nuScenesCaesar 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. [2020]. 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

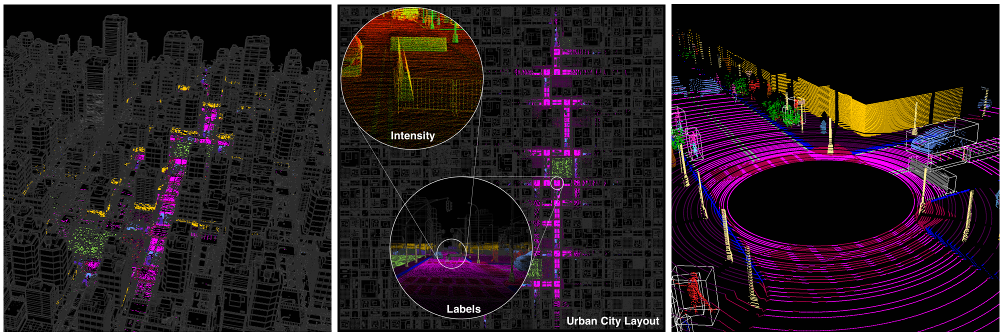

Figure 1: A sample 3D (left) and top views (middle) layout of the procedurally generated urban city with labelled point cloud accumulated along the vehicle path. The highlighted parts on the right showcase samples of the label annotation and calculated intensity. Bounding boxes (right) for vehicles,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

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 BBoxs 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.

3.1.1 Calculating Point Cloud 3D Coordinates

There are many topological formations to represent a sphere. Yet the UV-sphere was chosen because of its simplicity in mapping between Euclidean and spherical coordinates using trigonometric functions as follows;

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = r \begin{pmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{pmatrix}, \tag{1}$$

where r is the radius of the sphere. In Hossny et al. [2020], they replaced r with depth values obtained from the z-buffer of a $360\deg$ rendering of the scene. This, in return, allowed them to render a multi-channel equirectangularly unwrapped image of the scene $I^{D,P_i}[u,v]$ where D, P_i are per-point depth and other properties of the scene where u and v serve as both texture and spherical coordinates. The depth is used to determine the distance between the sensor and the point on the sphere surface while the properties P_i are used to represent the labels, reflectivity, etc for each point. In this work, we chose $P_i = L, R$ to represent per-point labels and material reflectivity, respectively. Therefore, each rendered pixel at $[u,v]$ produces a labelled 3D point in the rendered point cloud as;

$$\begin{pmatrix} x_{u,v} \\ y_{u,v} \\ z_{u,v} \end{pmatrix} = I^D[u,v] \begin{pmatrix} \sin\theta_u\cos\phi_v \\ \sin\theta_u\sin\phi_v \\ \cos\theta_u \end{pmatrix} \tag{2}$$

$$l_{u,v} = I^L[u,v], \tag{3}$$

$$i_{u,v} = I^R[u,v], \tag{4}$$

where $l_{u,v}, i_{u,v}$ is the label and intensity for all texture coordinates u, v and $I^{D,L,R}$ is the equirectangular depth, label and material reflectance maps, respectively.

3.1.2 Calculating Reflectance Intensity $I^R[u,v]$

We expanded the work in Hossny et al. [2020] to simulate the reflection intensity of different surfaces by incorporating the incidence vector from the simulated sensor and the surface. We obtained the reflectance parameters of the different surface materials from Kashani et al. [2015] and used a standard 2D Gaussian distribution to simulate the fine grains of the material. Additionally, we considered two sources of intensity fall-off, namely light attenuation and incidence angle of LiDAR beams on surfaces. We have included a detailed description and equations about the procedure we followed to calculate these two sources of intensity fall-off as part of the supplementary material.

3.1.3 Assigning Labels $I^L[u,v]$

During rendering, labels are assigned to each pixel u, v based on the type of the 3D object (e.g. vehicle, road, etc) and also the material of different parts of a 3D object (e.g. tyres, car frame, asphalt, side-walk, etc). In addition, material is then mapped to the mesh geometry of the 3D asset using local texture coordinates. This part is discussed further in the annotation subsection later.

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.

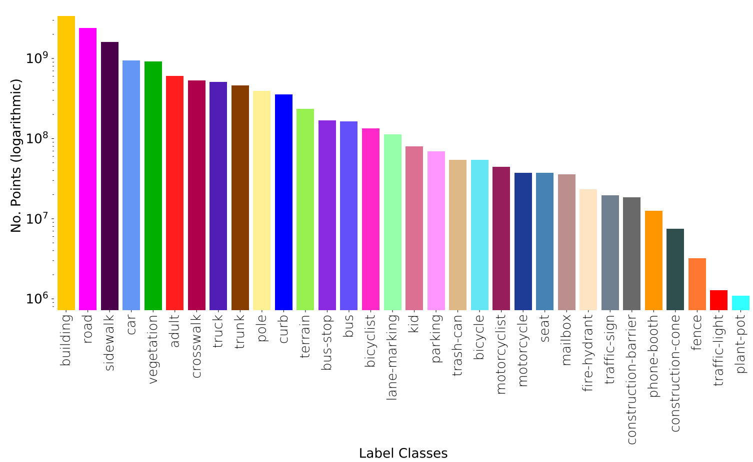

Figure 2: Label distribution of our VoxelScape dataset. The number of labelled points per class is shown.

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,

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

Object Class	No. BBox
Pedestrian (Adult)	537850
Pedestrian (Child)	132346
Cyclist	101342
Motorcyclist	28825
Car	88845
Truck	35843
Bus	3774
Construction-Barrier	6361
Construction-Cone	14670
Total	949856

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.

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.

Table 3: Evaluation of the baseline DNN models trained on our VoxelScape dataset (w/wo using the intensity (INT) values) when tested on the validation split of the SemanticKITTI Behley et al. [2019].

Approach	mAcc	mIoU
SqueezeSeqV2(-INT) Wu et al. [2019]	25.4	7.4
SqueezeSeqV2(+INT) Wu et al. [2019]	32.4	9.5
Darknet53(-INT) Milioto et al. [2019]	29.1	7.9
Darknet53(+INT) Milioto et al. [2019]	40.7	10.2

4.1 Baseline DNN Models

The DNN models using convolutional neural networks (ConvNets), have became the state-of-the art for the point-wise semantic segmentation of the PCD Milioto et al. [2019], Wu et al. [2019]. We chose two two architectures as baseline: SqueezeSegV2 model Wu et al. [2019] and DarkNet53 model Milioto et al. [2019]. 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 up 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 model is a modified encoder-decoder fully ConvNet model similar to typical semantic segmentation ConvNet models for 2D RGB images. One of the unique components of the SqueezeSegV2 is the added Context Aggregation Module (CAM), that helps in reducing the effect of missing points from 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° 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.

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-tunning 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].

4.2.1 Intensity Evaluation Experiment

For our first experiment, we trained the baseline DNN models twice, one while using the full PCD values (x, y, z) and intensity values from our VoxelScape dataset. The other model, it was trained only with the (x, y, z) values without the intensity values. Then, we evaluate the performance of these models on real PCD from physical LiDAR sensors. 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 4), 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). In Milioto et al.

Table 4: 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-TF) and another one without the fine-tuning.

Approach	mIoU	road	sidewalk	parking	other-ground	building	car	truck	bicycle	motorcycle	other-vehicle	vegetation	trunk	terrain	person	bicyclist	motorcyclist	fence	pole	traffic sign
SqueezeSeqV2 Wu et al. [2019]	35.1	89.6	72.7	32.2	0.9	68.8	78.8	15.9	13.1	17.6	18.1	70.6	22.9	67.5	8.1	27.2	0.0	33.5	13.4	16.1
SqueezeSeqV2 (VS-FT) Wu et al. [2019]	**36.5**	90.1	73.3	35.0	0.5	69.7	79.6	26.0	11.9	21.6	16.6	71.6	26.1	67.6	12.4	30.7	0.0	35.3	14.0	12.2
DarkNet53 Milioto et al. [2019]	36.5	85.1	69.4	14.7	0.0	74.8	80.9	17.9	8.1	4.0	9.6	78.8	28.0	70.9	8.3	33.8	0.0	47.4	32.1	28.9
DarkNet53 (VS-FT) Milioto et al. [2019]	**39.8**	92.0	76.2	45.2	0.1	72.3	80.6	39.0	13.0	20.8	20.4	75.1	31.2	71.0	13.6	38.8	0.0	30.5	17.6	19.2

[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 first experiment evaluated on the validation split of the SemanticKITTI dataset using two different evaluation metrics, namely the mean accuracy (mAcc) and the mean intersection over-union (mIoU) metrics Everingham et al. [2015]. As it can be noticed from Table 3, we have two versions of the two baseline DNN models (SqueezeSegV2 and Darknet53); one is trained on our VoxelScape dataset without intensity values (-INT), and the other one with the intensity values (+INT). From the reported results, it can be noticed that the simulated intensity values of our VoxelScape dataset helped in improving the performance of the two baseline DNN models when tested on the real PCD SemanticKITTI. It is worth noting that the scores for both mAcc and mIoU were calculated across each class from the 19 classes of SemanticKITTI.

4.2.2 Generalisation Evaluation Experiment

In Table 4, 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) and DarkNet53 (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, the SqueezeSegV2 (VS-FT) model achieved only a total mIoU score of 36.4% and outperformed the SqueezeSegV2 which scored only 35.1%. The main deduction from the results in Table 4, 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]

Table 5: Evaluation of the 3D detection results on the validation split of KITTI Geiger et al. [2012]. [1]VS-FT denotes that the model was fine-tuned using our VoxelScape dataset.

Method	Training Data	mAP	Car			Pedestrian			Cyclist		
			Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
PointPillars Lang et al. [2019]	VoxelScape	14.96	21.01	17.68	17.48	11.61	10.61	10.43	21.29	17.73	16.97
	KITTI	45.55	74.67	62.63	57.16	41.83	38.61	36.51	54.64	45.79	42.99
	KITTI (VS-FT)[1]	**58.06**	80.35	71.73	65.96	60.71	54.53	52.84	70.07	60.76	55.39

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 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 three 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 two 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 last two models that, one model was fine-tuned using the weights from the trained PointPillars on our VoxelScape dataset, while the other was not. In Table 5, we evaluate the performance of the trained three 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 5 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 5, 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.

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, fined-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

Jesús Balado, Joaquín Martínez-Sánchez, Pedro Arias, and Ana Novo. Road environment semantic segmentation with deep learning from mls point cloud data. *Sensors*, 19(16):3466, 2019.

Jens Behley, Volker Steinhage, and Armin B Cremers. Performance of histogram descriptors for the classification of 3d laser range data in urban environments. In *2012 IEEE International Conference on Robotics and Automation*, pages 4391–4398. IEEE, 2012.

Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9297–9307, 2019.

Jens Behley, Andres Milioto, and Cyrill Stachniss. A benchmark for lidar-based panoptic segmentation based on kitti. *arXiv preprint arXiv:2003.02371*, 2020.

Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020.

Katherine Compton. *Casual Creators: Defining A Genre Of Autotelic Creativity Support Systems*. PhD thesis, University of California Santa Cruz, 2019.

Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds for autonomous driving. 2020.

Mark De Deuge, Alastair Quadros, Calvin Hung, and Bertrand Douillard. Unsupervised feature learning for classification of outdoor 3d scans. In *Australasian Conference on Robitics and Automation*, volume 2, page 1, 2013.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

David Griffiths and Jan Boehm. SynthCity: A large-scale synthetic point cloud. In *ArXiv preprint*, 2019.

Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. Blensor: Blender sensor simulation toolbox. In *International Symposium on Visual Computing*, pages 199–208. Springer, 2011.

Timo Hackel, Nikolay Savinov, Lubor Ladicky, Jan D Wegner, Konrad Schindler, and Marc Pollefeys. Semantic3d. net: A new large-scale point cloud classification benchmark. *arXiv preprint arXiv:1704.03847*, 2017.

Mohammed Hossny, Khaled Saleh, Mohammed Attia, Ahmed Abobakr, and Julie Iskander. Fast synthetic lidar rendering via spherical uv unwrapping of equirectangular z-buffer images. *arXiv*, pages arXiv–2006, 2020.

Braden Hurl, Krzysztof Czarnecki, and Steven Waslander. Precise synthetic image and lidar (presil) dataset for autonomous vehicle perception. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2522–2529. IEEE, 2019.

A. G. Kashani, M. J. Olsen, C. E. Parrish, and N. Wilson. A review of lidar radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration. *Sensors*, pages 28099–28128, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4213–4220. IEEE, 2019.

Yancheng Pan, Biao Gao, Jilin Mei, Sibo Geng, Chengkun Li, and Huijing Zhao. Semanticposs: A point cloud dataset with large quantity of dynamic instances. *arXiv preprint arXiv:2002.09147*, 2020.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243, 2016.

Khaled Saleh, Ahmed Abobakr, Mohammed Attia, Julie Iskander, Darius Nahavandi, Mohammed Hossny, and Saeid Nahvandi. Domain adaptation for vehicle detection from bird's eye view lidar point cloud data. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. 2019.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020.

Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018.

Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4376–4382. IEEE, 2019.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

Xiangyu Yue, Bichen Wu, Sanjit A Seshia, Kurt Keutzer, and Alberto L Sangiovanni-Vincentelli. A lidar point cloud generator: from a virtual world to autonomous driving. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, pages 458–464, 2018.

Chris Zhang, Wenjie Luo, and Raquel Urtasun. Efficient convolutions for real-time semantic segmentation of 3d point clouds. In *2018 International Conference on 3D Vision (3DV)*, pages 399–408. IEEE, 2018.

Feihu Zhang, Chenye Guan, Jin Fang, Song Bai, Ruigang Yang, Philip HS Torr, and Victor Prisacariu. Instance segmentation of lidar point clouds. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9448–9455. IEEE, 2020.

Sicheng Zhao, Yezhen Wang, Bo Li, Bichen Wu, Yang Gao, Pengfei Xu, Trevor Darrell, and Kurt Keutzer. epointda: An end-to-end simulation-to-real domain adaptation framework for lidar point cloud segmentation. *arXiv preprint arXiv:2009.03456*, 2020.

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 include 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]