Supplementary Material
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`

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

$$I^R[u, v] = F^I[u, v] \cdot F^D[u, v] \cdot M[u, v] \tag{1}$$

For calculating the reflectance intensity in Eq. 1, 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 $M[u, v]$ from Kashani et al. [2015] and used a standard 2D Gaussian distribution $M[u, v] \sim N(\mu, \sigma|s)$ to simulate the fine grains of the material. The μ, σ are the statistical parameters of different materials (e.g. asphalt, grass, etc as obtained from Kashani et al. [2015]) and s is the seed for the random number generator. The seed s was assigned based on the unique identification number of each 3D mesh instance in order to allow variability in the surface grain and the reflectance intensity accordingly.

Additionally, we considered two sources of intensity fall-off in our calculation of the reflectance intensity, namely inverse square law or light attenuation $F^D[u, v]$ and incidence angle $F^I[u, v]$ of LiDAR beams on surfaces.

Inverse Square Law: Light source intensity is inversely proportional to squared the distance between the source and the object, i.e. intensity decreasing at a quadratic rate as it travels from the light source according to Eq. 2. The attenuation coefficient C for the visible light source is 0.25. However, due to its narrow band of wavelengths, the laser beam has a Gaussian profile which diverges with small divergence angle $\theta \approx 0.3mrad$ as it travels beyond its Rayleigh range Damask [2004]. Thus, the attenuation coefficient C is as low as $\tan^{-2}\theta|_{\lambda \approx 532\text{nm}}$.

Laser light intensity is inversely proportional to the squared distance between the laser source and the object, i.e. intensity decreasing at a quadratic rate as it travels from the light source according to Eq. 2.

$$I = \frac{C * P}{\pi d^2} \tag{2}$$

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

where P and d are the power of the light source and the distance from the light source. which renders Eq. 2 to be;

$$I = \frac{P}{\tan^2 \theta \pi d^2} \tag{3}$$

When comparing $\tan^{-2} \theta|_{\theta \approx 532\text{nm}}$ in Eq. 3 of a laser beam to the 0.25 of a point light source in Eq. 2, the attenuation to the laser light source is infinitesimal. Thus, we considered it safe to assume that the intensity fall-off is linear and thus the fall-out image at pixel u, v is defined as;

$$F^D[u, v] = \frac{1}{||\ \mathbf{p}[u, v] - \mathbf{c}[u, v]\ ||} \tag{4}$$

where $\mathbf{p}$ and $\mathbf{c}$ are the coordinates of the intersection point on the surface and the sensor location (e.g. camera), respectively.

Incidence Angle The incidence angle of LiDAR laser beams on the surface of the object affects the intensity of the reflected beams. It is modelled with an inner product between the vector representing the laser beam and the surface normal of the object. The reflectance intensity map at each pixel u, v is calculated as;

$$F^I[u, v] = \left\langle \mathbf{n}[u, v], \mathbf{v}^T[u, v] \right\rangle \tag{5}$$

where $\mathbf{n}, \mathbf{v}$ are the surface normal and the incidence vector, respectively.

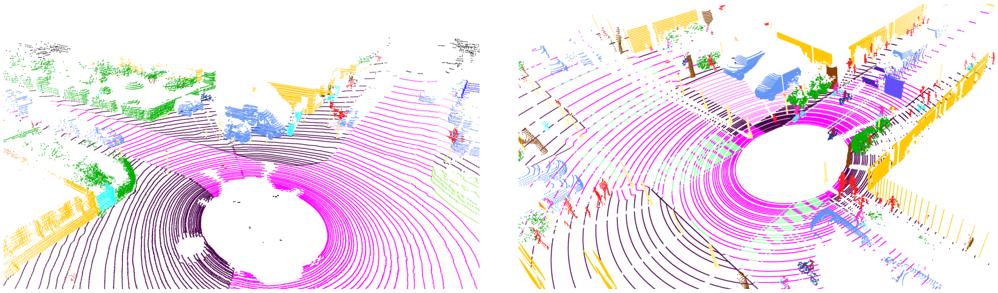
To that end and by expanding Eq. 1, the reflectance intensity map at each pixel u, v can be calculated as;

$$I^R[u, v] = \frac{\left\langle \mathbf{n}[u, v], \mathbf{v}^T[u, v] \right\rangle \cdot M[u, v]}{||\ \mathbf{p}[u, v] - \mathbf{c}[u, v]\ ||} \tag{6}$$

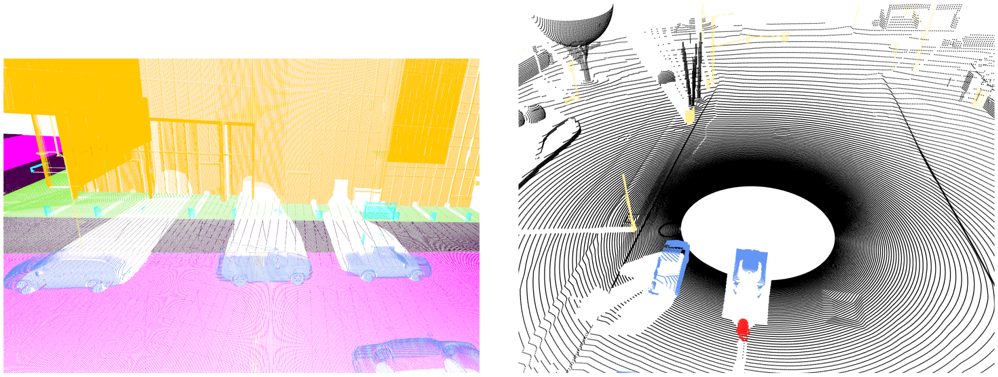
where $\mathbf{n}[u, v], \mathbf{v}[u, v], \mathbf{p}[u, v], \mathbf{c}$ are the normal, incidence, intersection maps (equirectangular as suggested in Hossny et al. [2020]); $M[u, v]$ is randomly generated with μ, σ obtained from Kashani et al. [2015]; and $< \cdot, \cdot >$ is the inner product between two vectors (dot product).

2 Dataset Realism Comparison

In this section, we demonstrate and compare the realism of our VoxelScape dataset in comparison to both other synthetic point cloud datasets and real point cloud datasets. Unlike other synthetic point cloud datasets, our VoxelScape dataset was generated using an accurate simulated physical 3D LiDAR sensor (i.e. Velodyne HDL-64E) which make the generated point cloud scans exhibit some of the key attributes of real point cloud scans from real physical LiDAR sensors. For example, in Figure 1, we can qualitatively assess the realism of the sample point cloud scan from our VoxelScape dataset and how it mimics the real point cloud scan from SemanticKITTI in terms of vertical/horizontal resolution of the capturing LiDAR sensor and the missing points due to occlusion of background/foreground objects. On the other hand, the sample point cloud scan from the synthetic SynthCity point cloud dataset Hackel et al. [2017], it lacks the accurate emulation of a physical LiDAR sensor, which in return make the points in their generated point cloud scan seems to be uniformly distributed without taking into consideration any of the physical attributes of a real LiDAR sensor. Another synthetic LiDAR point cloud dataset shown in Figure 1, is the GTA-LiDAR dataset Yue et al. [2018]. Despite the more accurate emulation of a physical LiDAR sensor of this dataset, it still suffers from lack of realism due to its simplification of the 3D meshes of asset objects while they are rendering their traffic scenes. This can be demonstrated by their oversimplified spherical meshes of trees and shrubs in the top-left corner of the sample point cloud scan of GTA-LiDAR in Figure 1. Another limitation of the GTA-LiDAR dataset, is the low number of ground-truth semantic labels they provide, where they only provide 3 class labels (cars, persons and poles).



(a) SemanticKITTI Behley et al. [2019] (b) VoxelScape (ours)



(c) SynthCity Hackel et al. [2017] (d) GTA-LiDAR Yue et al. [2018]

Figure 1: Qualitative realism comparison between real point cloud dataset and simulated LiDAR point cloud datasets.

Table 1: Statistical measure of the range and the intensity values of all point cloud scans in our VoxelScape dataset and the SemanticKITTI dataset Behley et al. [2019].

		Range	Intensity
SemanticKITTI Behley et al. [2019]	Mean	12.12	0.21
	± Std	12.32	0.16
VoxelScape (ours)	Mean	10.30	0.15
	± Std	10.95	0.01

Furthermore, in Table 1, we report the statistical measure in terms of mean and standard deviation (Std) of the range and intensity values across all the point cloud scans of our VoxelScape dataset and the real point cloud dataset, SemanticKITTI Behley et al. [2019]. As it can be shown from the table, the mean and Std measures of our VoxelScape dataset is quite close to the measures from the SemanticKITTI dataset, which further proves the realism of our VoxelScape dataset.

3 Class Labels Definition

In this section, we provide a full description of the total 32 class labels of our VoxelScape dataset. The full description can be found in next in Table 2.

Table 2: Class definitions.

class	definition
building	Our city generator includes three different types of buildings (residential, commercial and industrial). This class label includes all three different building types including their doors, windows and walls.

road	This class refers to any area that a motorised vehicle (car, bus, truck and motorbike) can drive on (typically asphalt areas, excluding lane-lines and crosswalks).
sidewalk	This class refers to areas made for vulnerable road users (pedestrians and bicyclists) to walk/move on excluding the curb.
car	Any motorised vehicle other than trucks, buses and motorbikes. We have a variety of car models including SUV, sedan and hatch-back cars .
vegetation	This class includes all greenery foliage or plants and trees' bushes and shrubs .
adult	This class refers to adult pedestrian.
crosswalk	Zebra marking on the road for pedestrian crossing.
truck	This class includes all vehicles with a body that is separate from the driver cabin such as ute (in Australia) or pickup trucks.
trunk	This refers to the tree trunk excluding all its greenery parts (i.e. vegetation) .
pole	This class refers to street light poles as well as traffic-light/sign poles .
curb	This class refers to the concrete edge of a sidewalk.
terrain	This class refers to all planar surfaces that typically not an asphalt nor a sidewalk such as: grass and gravel.
bus-stop	This class refers to the bus-stop structure including all its metal, plastic/glass parts and seats.
bus	This class refers to all different types of buses including school buses.
bicyclist	This class refers to the person riding the bicycle.
lane-marking	This class refers to all line markings on the road including dashed and solid ones.
kid	This class refers to child pedestrians.
parking	This class refers to areas in the road that are dedicated for parking vehicles.
trash-can	This class refers to all different shapes of trash cans that typically exist on the sides of the road.
bicycle	This class refers to the bicycle itself excluding the bicyclist person.
motorcyclist	This class refers to the person riding the motorcycle.
motorcycle	This class refers to the motorcycle itself excluding the motorcyclist person.
seat	This class refers to street prop's seat excluding the bus stop seat .
mailbox	This class refers to the mailbox container that commonly exist on the sides of the road.
fire-hydrant	This class refers to the fire-hydrant or firecock that commonly exist on the sides of the road. .
traffic sign	This class refers to all road signs excluding its mounting pole .
construction-barrier	This class refers to all temporary barriers that are used during roadworks.
phone-booth	This class refers to the full structure of street phone-booth.
construction-cone	This class refers to all temporary red/orange cones that are used during roadworks .
fence	This class refers to any permanent separators or barriers inside or on the sides of the road.

traffic-light	This class refers to the box that contains the traffic lights excluding its mounting pole.
plant-pot	This class refers to street pots that contains plants excluding the plants themselves that typically exist on the sides of the road.

4 Baseline Setup

4.1 Point-wise Semantic Segmentation Task

In our two baseline deep neural network (DNN) models, SqueezeSegV2 model Wu et al. [2019] and DarkNet53 model Milioto et al. [2019], the input point cloud data to them were mapped to a 2D representations using sphere projection following the procedure introduced in Milioto et al. [2019] and no data augmentation were utilised. The resolution of the 2D projected point cloud scan for both models were 1024(W) X 64(H). For all the SqueezeSegV2 models, we trained them using PyTorch and a stochastic gradient descent (SGD) as an optimiser with a learning rate of 0.002 and momentum of 0.9 for a maximum 200 epochs and we used a batch size of 20. For the DarkNet53 models, we trained them using PyTorch and SGD with a learning rate of 0.01 and momentum of 0.9 for a maximum 150 epochs and we used a batch size of 16. These parameters were empirically chosen based on the experimental results reported in Behley et al. [2019]. All the experiments were run on a desktop with Intel i7 CPU and a NVIDIA Quadro RTX 5000 GPU.

4.2 3D Object Detection Task

The input to our baseline DNN model, PointPillars Lang et al. [2019] is the 4-channel (x, y, z, intensity) raw point cloud scans and no data augmentation were utilised for the three models outlined in Table 5 of the main paper. The architecture parameters such as: xy resolution, max number of pillars and max number of points per pillar, ..etc were adopted from the settings for the KITTI experiment defined in the PointPillars paper in Lang et al. [2019]. To comply with the point cloud range for the KITTI dataset, the detection range for the three models was clipped to [0..70] in x, [-40..40] in y and [-3..1] in z for all three object classes. We trained the three PointPillar models using PyTorch and Adam optimiser with initial learning rate of 0.001 and weight decay of 0.01 for a maximum 150 epochs and we used a batch size of 4. Regarding the evaluation metric, the average precision (AP), we used the 3D IoU as the matching criteria, where we used 3D IoU threshold of 0.5 for pedestrians and cyclists and 0.7 for cars. All the experiments were run on a desktop with Intel i7 CPU and a NVIDIA Quadro RTX 5000 GPU.

5 Results of Baseline DNN Intensity (+/-INT) Models

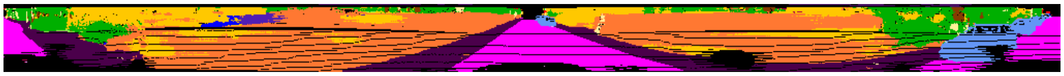
In our first experiment, we were interested in assessing the effect of our simulated intensity values and whether it could influence the performance of the trained DNN models. Having intensity values as part of the input to DNN models were shown to help boosting their performance in 3D perception tasks such as 3D object detection Yan et al. [2020], Qi et al. [2018] and semantic segmentation Wu et al. [2018], Zhao et al. [2020]. To that end, we trained two versions (w/wo using the intensity (INT) values) of the baseline DNN models using our VoxelScape dataset and tested them directly without any fine-tunning on real datasets i.e. SemanticKITTI Behley et al. [2019]. In Table 3, we expand the reported results from the main paper in Table 3, to include a more detailed intersection-over-union (IoU) scores over each class from the validation split of the SemanticKITTI dataset. As it can be noticed fro the scores in the table, the DNN models tend to be benefiting from the added intensity values in general. More specifically, extremely higher and lower reflective distinctive objects such as cars and roads, both the trained baseline DNN models with our VoxelScape intensity values (+INT), achieved higher IoU scores on them.

6 Qualitative Results

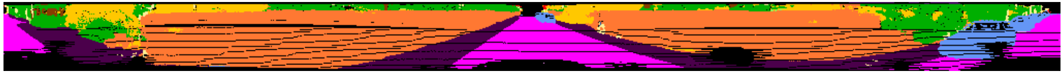
In Figure 2, we present some qualitative results of the fine-tuned baseline DNN model, SqueezeSeqV2 (VS-FT), on the SemanticKITTI dataset which was initially trained on our VoxelScape dataset. We

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].

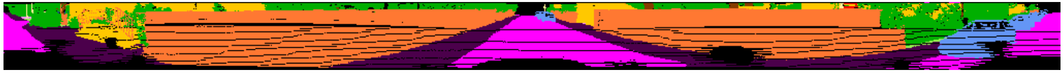
Approach	mIoU	road	sidewalk	parking	other-ground	building	car	truck	bicycle	motorcycle	other-vehicle	vegetation	trunk	terrain	person	bicyclist	motorcyclist	fence	pole	traffic sign
SqueezeSeqV2(-INT) Wu et al. [2019]	7.4	20.3	17.5	3.5	0.0	20.4	14.4	0.6	0.2	0.6	1.4	16.5	3.3	27.8	0.7	0.9	0.1	3.3	7.6	1.3
SqueezeSeqV2(+INT) Wu et al. [2019]	**9.5**	30.2	19.5	1.5	0.0	25.4	43.7	0.3	0.1	0.2	2.2	14.1	2.0	31.4	0.6	1.4	0.0	5.6	0.7	1.4
Darknet53(-INT) Milioto et al. [2019]	7.9	24.7	17.7	3.5	0.0	29.5	14.1	0.4	0.2	0.6	0.1	20.3	5.4	21.2	1.4	1.5	0.0	1.8	7.2	1.4
Darknet53(+INT) Milioto et al. [2019]	**10.2**	26.3	17.6	0.1	0.0	29.0	46.3	0.6	0.0	0.7	3.8	26.8	4.6	31.8	0.4	1.7	0.0	0.9	2.4	1.3



SqueezeSeqV2 Wu et al. [2019]



SqueezeSeqV2 (VS-FT) Wu et al. [2019]



Ground Truth

🟪 road
🟪 sidewalk
🟪 parking
🟦 car
🟦 motorcycle
🟨 pole
🟩 vegetation
🟩 terrain
🟫 trunk
🟨 building
🟦 other-object
⬛ unlabeled

Figure 2: Sample prediction of our fine-tuned SqueezeSeqV2 (VS-FT) in comparison to the SqueezeSeqV2 model that was trained only on the SemanticKITTI without any fine-tuning.

compare it also against the same SqueezeSeqV2 but without utilising the initial weights from the trained model on our VoxelScape dataset. As it can be shown, the predictions of the fine-tuned model SqueezeSeqV2 (VS-FT), tends to provide more consistent predictions with lower false positive predictions.

7 Dataset Usage

The two main tasks that the VoxelScape dataset can be utilised for are: semantic segmentation and 3D object detection on point cloud data. Other use case for the dataset is the unsupervised domain-adaptation between real and synthetic point cloud datasets to bridge the gap between the two domains.

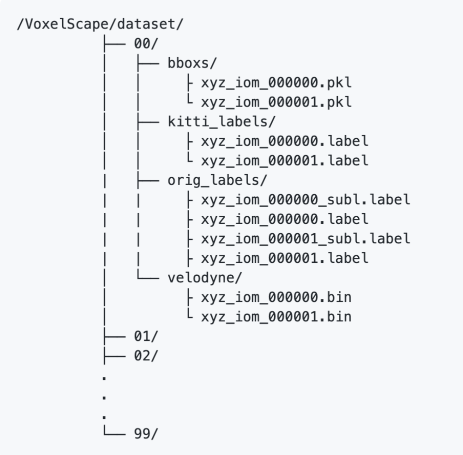
8 Dataset Access and Maintenance

should include download linke and assurance for the availability as well as the folder structure after the download

The dataset is publicly available through `https://voxel-scape.github.io/dataset/`. The dataset is hosted on a business Office 365 OneDrive account which is maintained by the University of Technology Sydney (UTS)/Australia. The hosted OneDrive repository can be publicly accessed with long-term availability (5 years that can be extended).

9 Dataset License

The dataset is published under the Creative Commons Attribution-Non Commercial 4.0 International License `https://creativecommons.org/licenses/by-nc/4.0/`. This means that it can be used for research and educational purposes but appropriate credit must be given, a link to the license must be provided, and an indication if changes were made. This may be done in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. We make no representations or warranties.



```
/VoxelScape/dataset/
          ├── 00/
          │    ├── bboxs/
          │    │      ├ xyz_iom_000000.pkl
          │    │      └ xyz_iom_000001.pkl
          │    ├── kitti_labels/
          │    │      ├ xyz_iom_000000.label
          │    │      └ xyz_iom_000001.label
          │    ├── orig_labels/
          │    │      ├ xyz_iom_000000_subl.label
          │    │      ├ xyz_iom_000000.label
          │    │      ├ xyz_iom_000001_subl.label
          │    │      ├ xyz_iom_000001.label
          │    └── velodyne/
          │           ├ xyz_iom_000000.bin
          │           └ xyz_iom_000001.bin
          ├── 01/
          ├── 02/
          .
          .
          .
          └── 99/
```

Figure 3: Folder structure of our VoxelScape dataset.

10 Dataset Structure

After downloading and un-zipping each sequence folder from the above access link into '/VoxelScape/dataset/' directory. The data should be organized as in Figure 3. As it can be shown, internally each sequence folder contains four sub-folders, namely:

- **velodyne:** contains the point clouds for each scan in each sequence. Each .bin scan is a list of float32 points in [x,y,z,intensity] format.
- **bboxs:** contains the 3D bounding boxes annotation of the 9 object classes in the dataset. Each '.pkl' file contains the 8-vertices of the 3D bounding box and class labels for each object exist in the corresponding .bin scan.
- **kitti_labels:** contains only 19 merged/subset semantic class labels, which correspond to the labels exist in the SemanticKITTI dataset, for each scan in each sequence. Each .label file contains a uint32 label for each point in the corresponding '.bin' scan.
- **orig_labels:** contains the total 32 semantic class labels introduced in the VoxelScape dataset. Each '_subl.label' file contains a fine-grained uint32 label for different attributes of each object in the corresponding '.bin' scan.

11 Dataset Visualization

In order to facilitate the process to work with the VoxelScape dataset, we have created a devkit repository for it on github. The devkit repo contains helper scripts to open, visualize and process point clouds and annotations from the VoxelScape dataset. The github repo can be accessed through: `https://github.com/voxel-scape/voxelscape-devkit`

References

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.

Jay N. Damask. *Polarization Optics in Telecommunications*. Springer, 2004. ISBN 0-387-22493-9.

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.

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 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.

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.

Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018.

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.

Lin Yan, Kai Liu, Evgeny Belyaev, and Meiyu Duan. Rtl3d: real-time lidar-based 3d object detection with sparse cnn. *IET Computer Vision*, 14(5):224–232, 2020.

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.

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.