Raphael: Realistic 3D Data Generation and Unsupervised Learning for Enhanced Human-Robot Detection in the Real World

Vlad Licaret[1,2]
vlli@norceresearch.no
Atle Aalerud[1]
ataa@norceresearch.no
Assia Belbachir[1]
assb@norceresearch.no
Nabil Belbachir[1]
nabe@norceresearch.no
Marius Leordeanu[1,2]
leordeanu@gmail.com

[1] NORCE Research AS,
Grimstad, Norway
[2] National University of Science and Technology POLITEHNICA Bucharest
Bucharest, Romania

Abstract

Ensuring safety in human–robot interaction requires reliable detection in industrial environments, yet the prohibitive cost of data annotation remains a major obstacle. We present a novel and cost-effective framework for automatic data generation and labeling, enabling unsupervised learning with state-of-the-art super-resolution LiDAR systems. Our method leverages recent advances in mesh reconstruction and animated 3D assets to create realistic synthetic datasets, surpassing existing augmentation strategies in fidelity and scalability. Beyond this, we introduce a second self-supervised stage that iteratively refines the model and bridges the virtual-to-real domain gap. Evaluations on real-world scenarios show substantial gains in performance.

1 Introduction

Ensuring safety in robotic systems is a critical challenge in environments where humans and large machines share confined spaces. Failures in detecting humans or robotic movements can result in severe accidents or fatalities. As collaborative robots proliferate, the demand for robust AI models capable of reliable detection has become urgent, yet progress is constrained by the scarcity of high-quality labeled data: collecting and annotating complex, cluttered, and dynamic industrial scenes is costly, labor-intensive, and rarely captures rare edge cases essential for safety validation.

Manual annotation at scale quickly becomes impractical, while traditional data collection fails to record high-risk scenarios, leaving safety-critical blind spots. To address data scarcity, simulation platforms such as CARLA [5] and synthetic city-scale datasets [8] generate LiDAR-like environments for autonomous driving. However, these approaches primarily focus on moving sensors in outdoor environments, where they typically maintain a horizontal viewpoint. In contrast, industrial sensors are often mounted overhead in relatively static environments. Moreover, these methods often fail to capture the fine-grained sensor effects of high-precision LiDAR systems, leading to a domain gap that limits real-world performance.

In this paper we propose Raphael, a framework for automated and realistic data generation and labeling, designed for detecting humans and large robots in 3D point clouds from state-of-the-art sensors [1]. Unlike prior work, our method directly addresses the challenges of indoor industrial environments by generating realistic and context-adapted scenarios, modeling sensor effects and employing a self-learning loop that iteratively adapts to real-world data.

We address these challenges through three key contributions:

1. Realistic 3D data generation: We introduce a novel 3D modeling algorithm that simulates the complex scanning process of the sensor and inserts realistic human and robot models into synthetic scenes. This automated approach drastically increases training data diversity and outperforms conventional data augmentation techniques.
2. Sensor effect simulation: We develop a method to process synthetic data by replicating sophisticated LiDAR effects, bridging the gap between virtual and real-world data. This allows us to train an initial "first-stage" model with enhanced robustness to sensor-specific noise and artifacts.
3. Self-learning adaptation: We introduce a self-learning phase, where the initial model generates predictions on real-world samples, which are then used as pseudo-labels for fine-tuning. This iterative process enables the model to gradually adapt to real-world conditions, even with minimal ground truth annotations.

Related Work Acquiring and annotating high-quality data remains a major challenge for training robotics and computer vision systems, particularly for point cloud processing. This has motivated alternatives to conventional data collection.

One line of research develops synthetic 3D environments to generate LiDAR-like point clouds via sensor simulation. CARLA [5] and AirSim [19] produce such data using raycasting, while Blensor [9] extends Blender to simulate range scanners. These frameworks are mainly designed for autonomous driving and standard LiDARs. In contrast, our work targets a Super-Resolution LiDAR in industrial settings, requiring new methodologies for realistic point cloud generation and annotation.

A second area addresses the domain gap between simulated and real LiDAR data. Prior work [11, 17] highlights discrepancies such as missing echoes and unreturned rays, but largely within open-loop driving scenarios. Our focus is domain adaptation with dynamic actors in real industrial environments.

A third line concerns data augmentation. Methods like SECOND [26], PPBA [4], PseudoAugment [13], and PolarMix [23] improve generalization by planting objects, self-training, or pseudo-labeling. Different from those works, our approach extends the concepts of planting objects in the scene and self-training by extending data augmentation with realistically modeling sophisticated 3D scanning systems.

Further advances address realism through occlusion handling [6, 18], generative strategies for synthetic scans [3, 15, 27, 29], or hybrid approaches mixing rendering and learning [14, 16]. Unlike these methods, which often rely on secondary models, we directly model

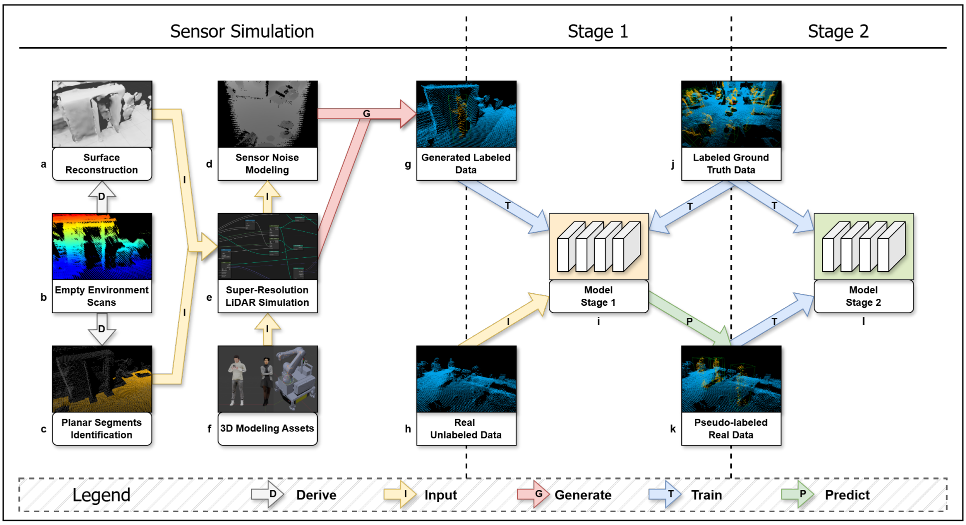


Figure 1: Illustration of the developed methodology for our realistic human-robot 3D data generation and pseudo-labeling (Raphael).

the real sensor for data generation and employ pseudo-labeling, enabling a more faithful alignment of synthetic and real domains.

Main technical contribution To the best of our knowledge, this is the first effective approach for automatic data generation and augmentation that realistically models advanced 3D scanning systems for unsupervised learning of human and robot detection in real-world scenarios. By addressing the limitations of existing simulation platforms and overcoming the domain gap, our approach sets a new standard for safety-driven AI systems. We will make our code available to accelerate research in safe human-robot collaboration.

2 System Description

Next we present our system for realistic human-robot data generation and pseudo-labeling, termed Raphael. As previously mentioned, Raphael is designed to address the challenges associated with acquiring high-quality labeled point cloud data and leveraging the abundance of unlabeled data available in real-world scenarios. Our approach is divided into three main steps, each focusing on either data generation or staged model training (see Fig. 1).

A real Super-Resolution LiDAR is deployed in a new environment where it has not been previously calibrated. Our approach builds an automated data generation pipeline using a virtual sensor and a simulated copy of the scene to fabricate synthetic training samples. These samples, combined with a small set of previously labeled ground truth from past testing scenarios, train an initial model. Once operational, the model generates pseudo-labels for incoming unlabeled data, which are then used in a second stage to refine the model, improving performance while mitigating virtual-to-real domain gap effects.

The rest of this section is organized as follows. The first subsection named **Sensor Simulation** deals with various aspects of modeling a high-resolution LiDAR and the world

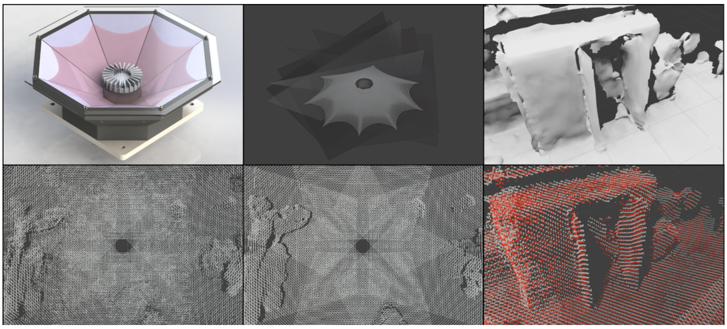


Figure 2: Reality vs. simulation. The left column shows the physical sensor and a real scan with its characteristic star pattern. The middle column depicts the simulated sensor and its virtual scan of the same environment. The right column presents a close-up of the reconstructed mesh with overlaid point clouds (red: real, white: simulated).

around it with the aim of extracting virtual data samples in an automated way. Here, **Super-Resolution LiDAR** concerns the actual sensor and **Scene Reconstruction** explains how virtual environments are derived from real point clouds. The next two paragraphs address the objects we aim to detect: **Mobile Robot Simulation** covers the kinematic modeling and integration of a robotic arm while **3D People Modeling** outlines the generation and placement of realistic human models. Closing this subsection, **Bridging the Domain Gap** presents techniques for aligning synthetic and real data by simulating various real-world phenomena. The next two entries, **Stage 1 Training** and **Stage 2 Distillation**, describe the initial learning phase using the previously generated data and the subsequent refinement stage, where pseudo-labels from Stage 1 guide the training of a second model.

2.1 Sensor simulation

Super-Resolution LiDAR The real data is acquired using a high-performance LiDAR system developed from the prototype in [1, 2], shown in Fig. 2. This sensor features eight mirror segments inclined at 34°, enabling accurate 3D data capture of people and robots in industrial environments. The specific model is the Ouster OS1-128,[1] which employs 128 laser emitters distributed over a 45° span, producing 128×4096 data points per frame. For our dataset we configure it at 128×2048 points and 10 frames per second, yielding detailed, real-time point clouds suitable for safety-critical applications.

To replicate this system virtually, we construct a Super-Resolution LiDAR simulation in Blender (Fig.1e, Fig.2), as it provides raycasting, surface intersection and normal estimation modules, together with Python scripting and easy integration of external assets. Our virtual sensor pipeline is built from scratch using geometry nodes to handle ray emission, mirror reflections, and sensor configuration. The model initializes arrays of rays in a user-defined spatial pattern, directs them toward configurable planar mirrors, computes reflections based on surface normals and, finally, records point intersections with scene objects. This design allows flexible modeling of LiDAR structure and optical effects in a virtually unlimited en-

[1]Datasheet: https://data.ouster.io/downloads/datasheets/datasheet-rev7-v3p1-os1.pdf, accessed October 2025.

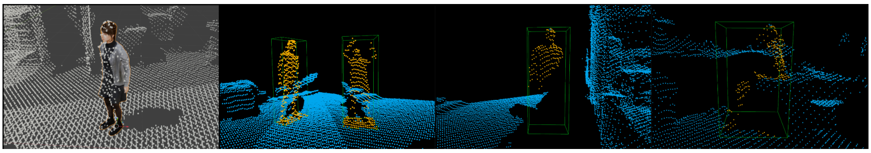


Figure 3: Examples of generated data. The first image shows virtual LiDAR readings interacting with a 3D model and its surrounding scene. Subsequent images display samples from our generated database, including automatically labeled point clouds with various poses and occlusions.

vironment. At this time we omit simulating the LiDAR intensity channel as experimentation on the current dataset and task did not demonstrate its usefulness. A visual comparison between real and simulated point clouds is presented in Fig. 2.

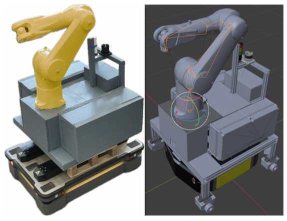
Scene reconstruction Acquiring labeled data is often difficult, so we propose an alternative: scanning the empty environment, reconstructing meshes using recent surface reconstruction methods, and then populating the scene with virtual people. To improve accuracy, we aggregate several point cloud scans within a short time window and compute the median for each point (Fig.1b). The resulting refined cloud serves two purposes: (i) positioning virtual humans in realistic locations, and (ii) reconstructing a 3D surface to accelerate ray–mesh intersections (Fig.1a). For the latter, we rely on an off-the-shelf mesh reconstruction method.

For the former, we categorize the point cloud into three classes: (a) ground, representing walkable areas; (b) desk, representing seating surfaces; and (c) other, including walls, objects, and remaining structures (Fig. 1c). This prevents unrealistic placements that could blur the distinction between humans and background. Ground and desk classes are extracted using RANSAC-based plane detection followed by DBSCAN clustering to isolate horizontal planar segments. Large segments are classified as ground, while desk segments are identified at approximately 0.7 m height. From these, we generate voxel maps indicating likely human positions, similar to [6].

Modeling the mobile robot To support early-stage development, we constructed a mockup of the target robotic system (Fig. 4a). The final platform combines a Stäubli TX90L manipulator with a Ridgeback Automated Guided Vehicle (AGV) from Clearpath Robotics, but only the manipulator and its control equipment are modeled here, as the AGV is not visible in the bird's-eye view used in simulation.

The links and kinematic structure of the robotic arm, provided by the manufacturer, serve as the foundation for the simulation. Using this data, we implement the kinematic model in Blender to replicate the movement and constraints of the real-world manipulator. Random joint configurations are then generated algorithmically within physical limits, ensuring that simulated movements remain operationally valid and consistent with the real system.

3D people Modeling We require 3D human models to accompany the robot in the simulated scene (Fig.1f). While recent generative methods exist [24, 25, 28], we opt for using animated human assets from a publicly available 3D model pack for efficiency. Before utilizing these assets, we conduct a brief analysis of person sizes in our training set and align the


(a)

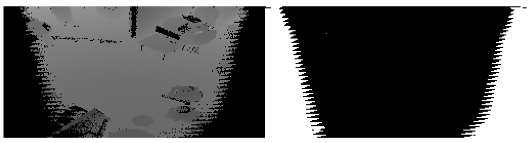

(b)

Figure 4: (a) The real arm robot (left) and its virtual counterpart (right). (b) Mirror edge drop. (left) a real scan, (right) a sampled edge drop mask where white regions represent dropped readings. In our case this pattern repeats eight times, once for each mirror.

3D models accordingly to further narrow the gap between the virtual and real domains. Having previously extracted sensible locations, we can now place virtual people into the scene while varying the size, placement, pose or animation keyframe, and, of course, the models themselves (Fig. 3). With these elements, we can generate large numbers of diverse training samples (Fig. 1g), expanding the dataset while emphasizing the target environment.

Bridging the Domain Gap Real LiDAR data is influenced by various environmental and sensor-related phenomena, leading to inaccuracies such as measurement imprecision and dropped readings. Previous studies, such as [17], have attempted to replicate these effects on synthetic data, demonstrating the value in doing so. In line with this, we adopt a methodology similar to [14]. We simulate three relevant effects, the first two having also been found as most impactful in prior research [17], and the third specific to our sensor type:

- **Ray drop (missed readings).** Repeated scans of the same scene naturally yield missing returns. We estimate drop ratios from the total readings and aggregate probabilities across bins defined by beam range d and incidence angle ϕ, computing $\bar{d}$, $\bar{\phi}$ and $\bar{P}$ for each. To generalize beyond discrete bins, we train a small MLP to predict drop probability conditioned on synthetic (d,ϕ), similar to [14], and sample from it at runtime.
- **Range noise.** ToF variance is measured by comparing multiple readings against their median. The error distribution (Δ) approximates but does not perfectly fit normality (Shapiro–Wilk statistic 0.88). We nonetheless model it as Gaussian: bins of $(\bar{d},\bar{\phi},\bar{\sigma})$ are used to train an MLP that outputs σ given (d,ϕ). At runtime, we sample from this distribution to inject a $\Delta Range$ perturbation as a ToF error along the beam direction.
- **Mirror edge drop.** A systematic loss at reflective boundaries arises from beam splitting, causing persistent gaps in returns. Because this is sensor-specific and largely constant, we precompute high-probability ($> 90\%$) drop masks per (x,y) pixel in the 128×2048 range image. At runtime, we sample M_{edge} from these masks and remove the corresponding points. See Fig. 4b for an example, Fig. 1d for integration.

2.2 First stage training

With the small amount of human labeled ground truth data we have available (Fig. 1j) and the newly generated data (Fig. 1g), we can now train a preliminary model. We name this to be our **Stage 1** (Fig. 1i) model assuming that we could not, in spite of our efforts, truly

	Method	People AP $IoU_{3D} > 50\%$	People AP $IoU_{BEV} > 25\%$	Robot AP $IoU_{3D} > 50\%$	Robot AP $IoU_{BEV} > 50\%$
DSVT	Baseline (no scene D)	0.01	0.46	N/A	N/A
	GT-Aug (with D_{empty})	71.34	74.47	N/A	N/A
	Ours Stage 1 ($D_{empty} + D_{virtual}$)	81.02	85.80	77.13	77.13
	Raphael - Ours Stage 2 (D_{pseudo})	**82.80**	**88.08**	**84.28**	**84.28**
PointPillar	Baseline (no scene D)	0.01	6.90	N/A	N/A
	GT-Aug (with D_{empty})	74.21	82.63	N/A	N/A
	Ours Stage 1 ($D_{empty} + D_{virtual}$)	75.81	84.08	77.61	78.56
	Raphael - Ours Stage 2 (D_{pseudo})	**78.46**	**85.64**	**80.67**	**79.85**

Table 1: Experimental results. We report average precision (AP) for 3D bounding boxes and bird's-eye-view (BEV) 2D boxes across IoU thresholds. (a) *Baseline*: trained only on scenes A, B, C, without Copy/Paste augmentation, showing poor generalization to scene D; performance is further reduced by sensor placement in D at roughly twice the height of prior setups, altering LiDAR range and density. (b) *GT-Aug*: Copy/Paste augmentation of people from A, B, C into D_{empty} (robot class absent in training). (c) *Ours Stage 1*: adds our automatically generated $D_{virtual}$ labels to the Baseline setting. (d) *Raphael*: final Stage 2 model trained on ground truth and pseudo-labels from D_{pseudo}, produced by the Stage 1 model on real unlabeled data.

replicate reality. However, using this first stage as an intermediary one, we aim to further improve results by leveraging it during a second iteration.

2.3 Second stage distillation

Our objective here is twofold: (a) to incorporate new unlabeled samples as they become available, (Fig. 1h) utilizing our Stage 1 model as a teacher, and (b) to mitigate the domain gap penalty introduced by the initial training with synthetic data.

Accordingly, we utilize the Stage 1 model to generate predictions for real unlabeled data, applying a thresholding operation on the confidence of each prediction. Following this, we merge the newly constructed pseudo-labeled training set (Fig. 1k), comprising all sufficiently confident detections, with the original ground truth. Finally, we train (or fine-tune) the **Stage 2** model (Fig. 1l) using this combined dataset comprised entirely of real-world samples.

3 Experiments and Results

Our ground-truth dataset consists of three training scenes (A_{train}, B_{train}, C_{train}), manually labeled, and one testing scene (D_{test}). The training scenes contain on average 2.7, 23.2, and 3.1 people per frame, respectively, for a total of 2652 frames. Unfortunately, none of these training sets include the robot class. The test set (D_{test}) comprises 600 frames from a 5-minute recording of the robot and roughly five people, annotated at 2 fps. From the same environment we additionally collect D_{empty}, a small scan of the empty scene during off hours. This scan, together with synthetic data generation and Stage 1 training, can be viewed as an initialization step.

Following from D_{empty}, we generate 1500 virtual training samples using the sensor-simulation pipeline described in Section 2.1. Each frame averages 4.4 persons and 1 robot, forming our $D_{virtual}$ labeled set. Human figures are drawn from the Procedural Crowds

Method	Detection AP $IoU_{3D} > 70\%$	Detection AP $IoU_{BEV} > 50\%$
SR LiDAR Simulation	75.98	76.84
+ Noise Modeling (Stage 1)	77.61	78.55
Raphael (Stage 2)	**80.67**	**79.85**

Table 2: Ablation study demonstrating the impact of each stage on overall performance, specifically for the Robot class trained solely on synthetic data.

Standard database [20], which provides 22 models and 9 animations; for every sample we randomize count, size, placement, rotation, animation, and keyframe. For the robot, beyond altering position within the virtual space, the animated arm is randomized across its six pivot joints. To ensure physical plausibility, a self-intersection check prevents unrealistic overlaps, but otherwise the aim is maximum variability, driving the robot through a broad range of motion. In terms of efficiency, generating one virtual frame averages 27.6 seconds, compared to 6.2 minutes for manual annotation of a real frame under similar conditions (same scene, similar number of objects). As the process is offline, runtime optimization is not essential. For instance, an automated pipeline could scan an empty environment overnight, generate a small set of virtual annotations, and fine-tune the model by morning.

Environmental surfaces are reconstructed using NKSR [10], producing a high-quality 3D mesh. To bridge the domain gap, we then simulate three error types. For the first two (ray drop and range noise), we bin usable ranges d (2–16 m, 56 bins) and incidence angles ϕ (16 bins). For each bin we compute drop probabilities and range-error standard deviations, discarding bins with fewer than 100 samples, and train two MLPs to predict these statistics from $(\bar{d}, \bar{\phi})$. For mirror edge drops no predictor is needed as we directly sample probabilities following the procedure described in Section 2.1.

All models and detection evaluations are implemented in OpenPCDet [21]. For augmentation we adopt the built-in Copy/Paste (GT-Aug) procedure, derived from Ghiasi et al. [7] and widely used in 3D detection [4, 12, 26]. GT-Aug builds an object database during preprocessing by extracting all labeled instances, then pastes a random subset onto training samples up to a class-specific cap N. In total, we obtain 24,687 person instances from ground truth and 8,493 instances (persons & robot) from our generated dataset. While ground-truth instances are abundant, many are redundant due to the 2 fps capture rate and repeated scenes; in contrast, our synthetic data avoids this by randomizing parameters across assets.

We evaluate two detectors: DSVT [22] and PointPillars [12]. Both are trained with a scene size of $(32, 32, 4)$ m and voxel size $(0.1, 0.1, 4)$ m, using NMS=0.25, batch size 4, and standard flip/rotation/translation augmentations. Optimizer (Adam + OneCycle, lr=10^{-3}) and other hyperparameters follow defaults. To observe maximum performance, no validation split is used; instead, we track test-set results directly and report the best stable performance. Final models are obtained by weight-averaging the last five epochs once metrics converge.

For Stage 2, we use the Stage 1 model to generate predictions on 1200 unseen unlabeled frames from scene D (Ours Stage 1 in Table 1). Applying a confidence threshold of 0.6 yields the pseudo-labeled set D_{pseudo}. A new model is then trained on the union of ground-truth data (scenes A, B, C) and D_{pseudo}, initialized from Stage 1 weights; training proceeds until test metrics stabilize. We denote this final Stage 2 model as *Raphael* (Table 1).

Table 1 shows that both components of our pipeline—high-resolution sensor simulation (Stage 1) and pseudo-label distillation (Stage 2)—outperform the standard GT-Aug (Copy/Paste) baseline. Notably, the robot class achieves competitive performance even when trained only on synthetic data. We further analyze the impact of our iterative approach for progressively improving detection by considering an ablation study in Table 2.

4 Conclusion and Future Work

We introduced *Raphael*, a framework for automated, high-fidelity data generation and labeling for human and robot detection in 3D point clouds. Using a high-resolution LiDAR system, we construct detailed virtual datasets that complement scarce ground truth and significantly improve detection performance. A second stage of self-supervised training further boosts results by leveraging pseudo-labels from real data, effectively distilling synthetic knowledge while narrowing the virtual-to-real gap. Our experiments demonstrate both strong empirical gains and practical utility for rapid deployment of safe, human-centric AI in new environments. In sum, Raphael provides a concrete path toward scalable 3D data generation, annotation, and domain adaptation in complex real-world settings.

Acknowledgements

The COGNIMAN project[2], leading to this paper, has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101058477. Marius Leordeanu was also supported in part by project "Romanian Hub for Artificial Intelligence - HRIA", Smart Growth, Digitization and Financial Instruments Program, 2021-2027 (MySMIS no. 334906).

References

[1] Atle Aalerud, Joacim Dybedal, and Dipendra Subedi. Reshaping field of view and resolution with segmented reflectors: Bridging the gap between rotating and solid-state lidars. *Sensors*, 20(12), 2020. ISSN 1424-8220. doi: 10.3390/s20123388. URL `https://www.mdpi.com/1424-8220/20/12/3388`.

[2] Assia Belbachir, Antonio M. Ortiz, Atle Aalerud, and Ahmed Nabil Belbachir. Advancing point cloud perception: A focus on people detection. *SN Computer Science*, 6(698), 2025. doi: 10.1007/s42979-025-04221-9. URL `https://link.springer.com/article/10.1007/s42979-025-04221-9`.

[3] Lucas Caccia, Herke Van Hoof, Aaron Courville, and Joelle Pineau. Deep generative modeling of lidar data. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5034–5040. IEEE, 2019.

[4] Shuyang Cheng, Zhaoqi Leng, Ekin Dogus Cubuk, Barret Zoph, Chunyan Bai, Jiquan Ngiam, Yang Song, Benjamin Caine, Vijay Vasudevan, Congcong Li, et al. Improving 3d object detection through progressive population based augmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*, pages 279–294. Springer, 2020.

[2] `www.cogniman.eu`

[5] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 1–16. PMLR, 13–15 Nov 2017. URL https://proceedings.mlr.press/v78/dosovitskiy17a.html.

[6] Jin Fang, Xinxin Zuo, Dingfu Zhou, Shengze Jin, Sen Wang, and Liangjun Zhang. Lidar-aug: A general rendering-based augmentation framework for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4710–4720, 2021.

[7] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D. Cubuk, Quoc V. Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2918–2928, June 2021.

[8] David Griffiths and Jan Boehm. Synthcity: A large scale synthetic point cloud, 2019. URL https://arxiv.org/abs/1907.04758.

[9] Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. Blensor: blender sensor simulation toolbox. In *Proceedings of the 7th International Conference on Advances in Visual Computing - Volume Part II*, ISVC'11, page 199–208, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 9783642240300.

[10] Jiahui Huang, Zan Gojcic, Matan Atzmon, Or Litany, Sanja Fidler, and Francis Williams. Neural kernel surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4369–4379, 2023.

[11] Sebastian Huch, Luca Scalerandi, Esteban Rivera, and Markus Lienkamp. Quantifying the lidar sim-to-real domain shift: A detailed investigation using object detectors and analyzing point clouds at target-level. *IEEE Transactions on Intelligent Vehicles*, 8(4): 2970–2982, April 2023. ISSN 2379-8858. doi: 10.1109/tiv.2023.3251650.

[12] 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.

[13] Zhaoqi Leng, Shuyang Cheng, Benjamin Caine, Weiyue Wang, Xiao Zhang, Jonathon Shlens, Mingxing Tan, and Dragomir Anguelov. Pseudoaugment: Learning to use unlabeled data for data augmentation in point clouds. In *European conference on computer vision*, pages 555–572. Springer, 2022.

[14] Chenqi Li, Yuan Ren, and Bingbing Liu. Pcgen: Point cloud generator for lidar simulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11676–11682. IEEE, 2023.

[15] Ruihui Li, Xianzhi Li, Pheng-Ann Heng, and Chi-Wing Fu. Pointaugment: an auto-augmentation framework for point cloud classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6378–6387, 2020.

[16] Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. Lidarsim: Realistic lidar simulation by leveraging the real world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[17] Sivabalan Manivasagam, Ioan Andrei Bârsan, Jingkang Wang, Ze Yang, and Raquel Urtasun. Towards zero domain gap: A comprehensive study of realistic lidar simulation for autonomy testing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8272–8282, October 2023.

[18] Petr Šebek, Šimon Pokornỳ, Patrik Vacek, and Tomáš Svoboda. Real3d-aug: Point cloud augmentation by placing real objects with occlusion handling for 3d detection and segmentation. *arXiv preprint arXiv:2206.07634*, 2022.

[19] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017. URL https://arxiv.org/abs/1705.05065.

[20] Difffuse Studio. Procedural crowds. https://blendermarket.com/products/procedural-crowds, 2024. Available at: https://blendermarket.com/products/procedural-crowds.

[21] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet, 2020.

[22] Haiyang Wang, Chen Shi, Shaoshuai Shi, Meng Lei, Sen Wang, Di He, Bernt Schiele, and Liwei Wang. Dsvt: Dynamic sparse voxel transformer with rotated sets. In *CVPR*, 2023.

[23] Aoran Xiao, Jiaxing Huang, Dayan Guan, Kaiwen Cui, Shijian Lu, and Ling Shao. Polarmix: A general data augmentation technique for lidar point clouds. *Advances in Neural Information Processing Systems*, 35:11035–11048, 2022.

[24] Yuliang Xiu, Jinlong Yang, Dimitrios Tzionas, and Michael J. Black. ICON: Implicit Clothed humans Obtained from Normals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13296–13306, June 2022.

[25] Yuliang Xiu, Jinlong Yang, Xu Cao, Dimitrios Tzionas, and Michael J. Black. ECON: Explicit Clothed humans Optimized via Normal integration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023.

[26] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10), 2018. ISSN 1424-8220. doi: 10.3390/s18103337. URL https://www.mdpi.com/1424-8220/18/10/3337.

[27] Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. Unisim: A neural closed-loop sensor simulator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1389–1399, 2023.

[28] Zhuoqian Yang, Shikai Li, Wayne Wu, and Bo Dai. 3dhumangan: 3d-aware human image generation with 3d pose mapping. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23008–23019, 2023.

[29] Vlas Zyrianov, Xiyue Zhu, and Shenlong Wang. Learning to generate realistic lidar point clouds. In *European Conference on Computer Vision*, pages 17–35. Springer, 2022.