Point Cloud Segmentation of Agricultural Vehicles using 3D Gaussian Splatting

Anonymous CVPR submission

Paper ID 42

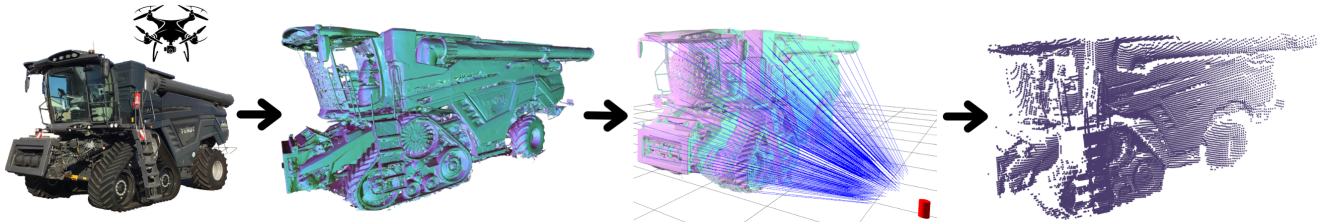


Figure 1. Illustration shows the pipeline created for generating the synthetic point cloud dataset. Step (1) shows the process of gathering images of a combine harvester. Step (2) shows the combine harvester mesh, extracted using GOF. Step (3) shows an example of a combine harvester mesh employed in the Gazebo LiDAR simulation environment, where the LiDAR sensor is visualized as the red cylinder with rays depicting the LiDAR scans. Step (4) shows an example of a final, synthetically generated, point cloud of a combine harvester.

Abstract

Training neural networks for tasks such as 3D point cloud semantic segmentation demands extensive datasets, yet obtaining and annotating real-world point clouds is costly and labor-intensive. This work aims to introduce a novel pipeline for generating realistic synthetic data, by leveraging 3D Gaussian Splatting (3DGS) and Gaussian Opacity Fields (GOF) to generate 3D assets of multiple different agricultural vehicles instead of using generic models. These assets are placed in a simulated environment, where the point clouds are generated using a simulated LiDAR. This is a flexible approach that allows changing the LiDAR specifications without incurring additional costs. We evaluated the impact of synthetic data on segmentation models such as PointNet++, Point Transformer V3, and OACNN, by training and validating the models only on synthetic data. Remarkably, the PTv3 model had an mIoU of 91.35%, a noteworthy result given that the model had neither been trained nor validated on any real data. Further studies even suggested that in certain scenarios the models trained only on synthetically generated data performed better than models trained on real-world data. Finally, experiments demonstrated that the models can generalize across semantic classes, enabling accurate predictions on mesh models they were never trained on.

1. Introduction

The demand for automation in the transportation sector for self driving vehicles has, in recent years, fueled research for having machines understand 3D environments [25]. As a format to capture 3D information, point clouds are increasingly popular due to their efficiency and direct representation of LiDAR data. Commonly, for the models to understand the composition of the environment represented by the point cloud, semantic segmentation is used as it provides per-point class-wise information, thus getting an understanding of the complete environment [14, 19–21, 31–33, 39]. An example of this is autonomous driving, where point cloud semantic segmentation is used to identify vehicles, road signs, pedestrians, etc. [25].

However, obtaining and annotating point clouds is a well-known tedious and expensive task, making the available training resources sparse compared to the training datasets for image recognition. This has led a wave of research into both using the point clouds more efficiently by augmenting them [4, 13], and methods for generating new, annotated point clouds synthetically using simulated environments relying on existing 3D assets [16, 30, 37]. The issue with using existing 3D assets arises when models need training for niche tasks with a sparse amount of available 3D assets. As a method to mitigate this issue, we propose the use of 3D Gaussian splatting (3DGS) to generate 3D assets that can be used to generate synthetic point cloud datasets. 3DGS is used due to the ease of only needing a digital camera for capturing useful data, and a few hours of GPU time to get a mesh. In this project, the method is tested as a use

case to generate and train point cloud semantic segmentation models in a self driving tractor scenario, with synthetic data acquired using the pipeline shown in Figure 1. This is a motivating domain for the proposed pipeline, as the existing 3D data is sparse and the meshes needed would be hard to model from scratch. As a proof of concept, the class count is reduced to: tractor, combine harvester, and other, as these vehicles often carry out collaborative tasks. Our key contributions are as follows:

- We present a novel way of generating high-quality LiDAR data from 3DGS.
- We demonstrate that by training a model purely on synthetic data it is possible to achieve high performance on real world data.
- We show that our method generalizes to objects not seen during training, highlighting its robustness and adaptability.
- Our method allows seamless modification of LiDAR-specific settings and the choice of mesh models used in the simulation, enabling greater adaptability to different sensing conditions and environments.

2. Related work

2.1. Point Cloud Semantic Segmentation

Segmentation of 3D points is a common problem in the field of 3D computer vision. Many methods have been proposed to tackle the challenge of segmenting the unstructured and irregular-spaced point clouds [14, 19–21, 31–33, 39]. These methods can be categorized into two categories: point-based models and voxel-based models [19]. Point-based methods directly manipulate the unstructured point clouds using point-wise operations. One such method is PointNet [20], which introduced a novel deep learning architecture used for point cloud semantic segmentation. However, the network cannot capture local structures due to the lack of neighborhood processing, limiting its ability to recognize fine-grained patterns and generalize to complex scenes. This was improved in PointNet++ [21] by partitioning the point set into local overlapping regions, and then recursively applying PointNet to obtain the local features. Dynamic Graph CNN, proposed by Wang *et al*. [31], is another method which expands upon the PointNet architecture, this time by implementing a new convolutional operation, EdgeConv, which allows the extraction of the local neighborhood features for each point. Another prominent point-based method is the transformer-based models, which leverages the attention mechanism [28] that has gained prominence across various domains, including point cloud segmentation. The Point Transformer V3 (PTv3) [33] advances point cloud segmentation by emphasizing scalability and efficiency. Unlike its predecessors, PTv1 [39] and PTv2 [32], which introduced local attention mechanisms, grouped vector attention, improved position encoding, and partition-based pooling, PTv3 focuses on the benefits of scaling rather than architectural complexity. It replaces the K-Nearest Neighbors (KNN) query with serialized patterns like Z-order and Hilbert curves, significantly reducing computational overhead, as KNN accounted for 28% of PTv2's forward time. Additionally, PTv3 refines the positional encoding, further enhancing the model's efficiency and ability to handle large-scale point clouds.

Voxel-based methods convert the point clouds into a voxel grid during data pre-processing. A recent paper by Peng *et al*. [19], introduced an Omni-Adaptive CNN which addressed the common problem of adaptivity for previous voxel-based methods [19]. They created an Adaptive Relation Convolution, and a corresponding adaptive aggregator, which dynamically adjusts the receptive field, allowing the model to focus more on parts with many defining features.

2.2. Synthetic data generation

Multiple methods have been proposed to generate synthetic annotated point cloud datasets [4, 13, 16, 26, 30, 34, 37]. These methods can be categorized into using augmented real point clouds or using simulated environments. A recent method that uses simulated point clouds that are then further processed by an augmentation process is suggested by Xiao *et al*. [34], proposing the use of the Unreal Engine to create a synthetic LiDAR segmentation dataset where they used an adversarial network to transform the synthetic point clouds that are acquired from the simulated environment, into point clouds that are closer to what would be sampled in reality.

Chen *et al*. [4] proposes to linearly interpolate between point clouds of the same classes to generate new training data. Another augmentation method by Li *et al*. [13] propose an augmentation neural network trained together with a classifier neural network in an adversarial manner. However, this method has not been tested on point cloud segmentation tasks.

Focusing on simulated environments, Ma *et al*. [16] uses building models to create synthetic annotated point clouds for training segmentation tasks. The point cloud sampling method used in the study resulted in uniformly sampled point clouds, whereas real point clouds would have non-uniform sparsity. This, according to the paper, is something that could be improved upon, as synthetic data should be similar to the real data.

Another simulated approach by Wang *et al*. [30] uses a simulated city environment made with the CARLA simulation tool [6], where the LiDAR is simulated with ray casting to get a synthetic segmented LiDAR point cloud dataset. Similarly, Yue *et al*. [37] uses the video game Grand Theft Auto V to simulate an outdoor driving environment. Using plugins, a synthetic dataset is generated from the simulation. An approach that focuses on human body part segmen-

tation proposed by Takamz *et al*. [26], used mesh scenes from the ScanNet dataset[5] with human models, where the poses of the humans have been generated. These scenes are then sampled with a simulated depth camera to obtain annotated point clouds. This method relies on having a pre-existing dataset of scenes, and a model [38] to generate synthetic humans in the scene. Due to this, it is ill suited for making datasets of niche tasks.

2.3. Novel View Synthesis

Novel view synthesis aims to generate an unseen view of a scene from an arbitrary viewpoint. It can, however, also be used to capture the geometry of a scene. Mildenhall *et al*. [17] introduced Neural Radiance Field (NeRF), which uses a multi-layer perceptron (MLP) to represent a 3D scene, including view-dependent reflections, colors, and geometry. Although later enhancements in NeRF have improved the rendering speed [9, 22, 35], and anti-aliasing [1, 2], they are still implicit, as the scene is encoded in the weights of the model. This results in extra long inference times that can be reduced only by lowering the quality of the visuals. Another method used to synthesize novel views is 3D Gaussian Splatting, introduced by Kerbl *et al*. [11], which uses 3D Gaussians to represent a scene explicitly, thus allowing real-time rendering, editable scenes, and a more accurate extraction of geometry. Furthermore, in recent years NeRF and 3DGS methods have been used to generate synthetic data, though so far limited to dense image tasks such as stereo vision and optical flow [7, 15, 23, 27].

2.4. Mesh Extraction

Photorealistic rendering through 3DGS has shown remarkable efficiency compared to NeRFs, however, generating accurate geometric reconstructions from these scene representations, remains a challenging problem. This challenge arises from the inherently disconnected nature of individual Gaussian primitives, and the complexity of aligning these Gaussians with continuous surfaces for reconstruction. [8, 10, 36]

SuGaR, introduced by Guédon *et al*. [8], regularizes the 3D gaussians to align with surfaces, allowing them to compute the surface normals. Using the regularized 3D gaussians and the computed normals, a Poisson surface reconstruction is employed to generate the mesh. Huang *et al*. [10] proposes 2DGS, which, instead of the original 3D gaussians splats, uses 2D gaussians to recreate the radiance field. The meshes are then reconstructed through depth maps of the radiance field using Truncated Signed Distance Function.

Yu *et al*. [36] introduced Gaussian Opacity Fields (GOF), a state-of-the-art technique that directly extracts surface normals from the 3D Gaussian representations, without requiring prior regularization or conversion to 2D. This method mitigates the inevitable data loss that happens when regularizing or reducing the dimension of the radiance field, enhancing the quality in more detailed parts. Using the extracted normals, the final mesh extraction is done by utilizing tetrahedral grids and the Marching Tetrahedra algorithm.

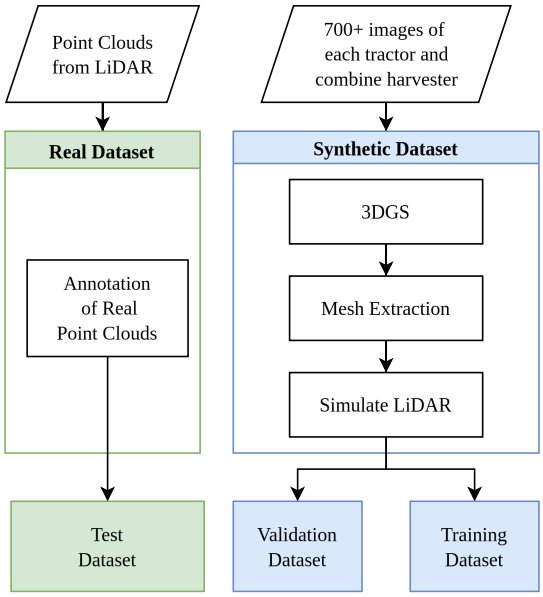


Figure 2. Illustration of the pipeline used to obtain synthetic datasets.

3. Methods

3.1. Data Acquisition & Processing Pipeline

To evaluate the influence of synthetic data on a point cloud semantic segmentation model, a *baseline* model is trained using only 'real' data acquired with an Ouster OS0 LiDAR [18]. A baseline model will be trained for each model tested in the paper. The effect of using synthetic data will be established based on a comparison between the baseline models, and models trained on both synthetic and real data. Figure 2 shows how data is acquired for the datasets, where the training and validation data is comprised of synthetic data. It also includes the pipeline for obtaining synthetic data, which begins with capturing 700+ images of each vehicle using a drone. These images are then processed to extract 3D meshes using GOF [36]. Finally, the meshes are imported into Gazebo [12], where a simulated LiDAR sensor is used to generate the synthetic data.

3.2. LiDAR captured dataset

The real dataset, which is split into training, validation and testing, is created by capturing LiDAR data from the real world. The data capture process is performed by driving

Figure 3. Drone-captured frames of a tractor (left) and a combine harvester (right) used in the 3D Gaussian Splatting mesh generation process.

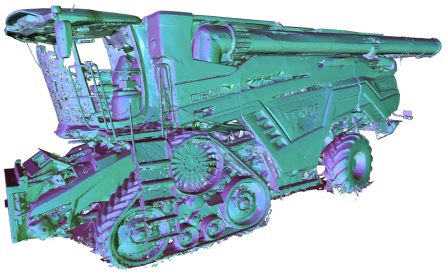


Figure 4. Figure shows a generated mesh where a single combine harvester has been cropped out, yielding a usable mesh for the Gazebo simulation.

around in a tractor, with the Ouster OS0-128 LiDAR sensor mounted on top, in a common agricultural scenario with multiple tractors and combine harvesters. Figure 3 presents an image of one of the tractors and an image of one of the combine harvesters. The data is collected in sequences of driving around for one minute, with the OS0-128 LiDAR sensor sampling point clouds at 10 Hz, yielding 600 point clouds per sequence. In total, 15 sequences from six different configurations have been acquired and annotated, where, for each scene configuration, the vehicles are moved to new positions. From the total amount of real data, the same five sequences, totaling 3,000 real point clouds, are used for the testing split for all tests performed. The five sequences are chosen such that the distribution of points per model is as even as possible across all possible tractor and combine models captured in the dataset. The remaining 6,000 real point clouds, which come from different vehicle configurations than those used in the testing set, are then used in the training of the real-only baseline models.

Each point cloud consists of around 50,000 points on average, where each point is labeled into separate classes. The dataset has three different classes: tractor, combine harvester, and other. Additionally, the average class-wise point distribution for each point cloud is 6.5% tractor, 12.6% combine harvester and 80.9% other. To annotate the point cloud the static environment used to capture the data is leveraged to create a combined point cloud for each sequence using KISS-ICP [29]. Clustering is then applied to each vehicle in the combined points cloud and these clusters are used to annotate the individual point clouds in the sequence.

3.3. Synthetic Data Generation

Multiple methods can be used to generate synthetic point cloud data, as mentioned in Section 2.2. Xiao *et al*. [34], showed that synthetic data, modified to close the sim-to-real gap, outperformed the purely synthetic data. This motivates the use of a simulated environment where points can be sampled in a LiDAR pattern compared to uniformly sampling points from the surfaces of the meshes as done by Ma *et al*. [16]. When producing synthetic data using our simulation, data is only generated for the three different classes also available in the real-world LiDAR captured dataset.

Mesh generation: To simulate the environment, which synthetic point clouds will be extracted from, it is of utmost importance to obtain the best possible meshes of the vehicles in the scene. The better the meshes resemble the real-world vehicles, the better the simulated LiDAR will be at sampling synthetic point clouds close to an actual real-life scene. The method for generating the meshes starts with capturing images of a scene where the vehicle is focused in the middle. This was done by capturing a video with a drone flying slowly around the vehicle, then sampling images from the video at a consistent interval.

Using the captured images, the initial sparse point cloud is computed using the Structure-from-Motion (SfM) implementation in COLMAP [24]. Following this, the sparse SfM point cloud is utilized for the 3DGS mesh extraction algorithm. Through initial experimentation, it was found that, out of SuGaR [8], 2DGS [10] and GOF [36], the meshes extracted using GOF yielded the best results, with the highest degree of fidelity. Meshes were then generated for all the different vehicles, which includes seven different tractor models for the tractor class of the dataset, and three different combine harvesters for the combine class. An example of a combine harvester mesh can be seen in Figure 4. A bit of post-processing is done on the meshes, since the output from the 3DGS mesh extraction is one big mesh of the whole scene. The post-processing comprises of cropping out everything except the specific vehicle in question. The meshes are then employed in the Gazebo simulation to generate the synthetic point clouds. Additionally, to achieve a simulation environment resembling the real world as best as possible, grass and other miscellaneous object meshes from the 3DGS meshes are utilized as well.

Simulation Environment: Gazebo is an open-source simulation tool designed to simulate robotic applications. It is built around the Ogre2 engine, and has an integrated LiDAR plugin. Using the LiDAR plugin, it is possible to simulate any real LiDAR sensor geometrically. For the data genera-

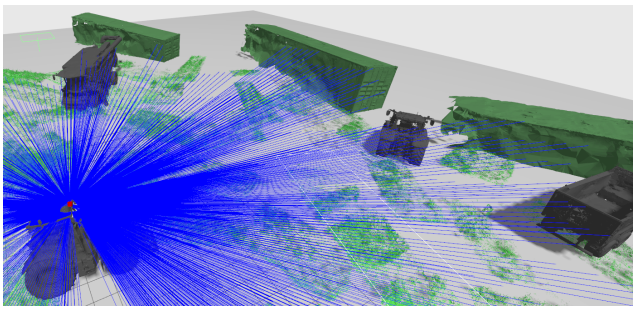


Figure 5. The Gazebo simulation where the target assets have moved to random positions. The LiDAR sensors position is marked by the red cylinder. The blue rays visualize a sparse version of the LiDAR rays

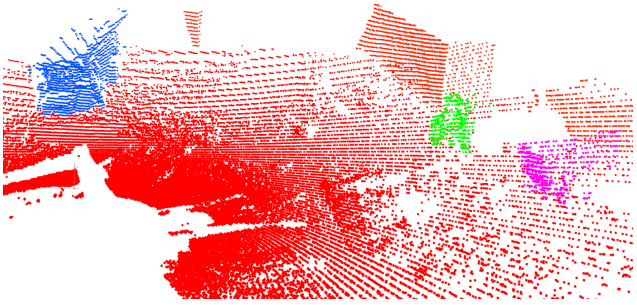


Figure 6. An annotated point cloud obtained from the gazebo simulation containing tractors (green), combine harvesters (blue), trailers(pink), and other (red).

tion the Ouster OS0-128 LiDAR is simulated, as it was used to capture the real dataset. Due to this, the synthetically extracted point clouds also end up being roughly the same size, with around 62,000 points on average per point cloud, with the average class-wise point distribution for each point cloud being 88.3% other, 5.0% tractor and 6.7% combine harvester.

To use Gazebo as a dataset generator, a custom plugin is used to move the meshes of different classes and the LiDAR to random positions. To avoid meshes being placed out of range for the LiDAR or overlapping with one another, placement rules were made for the meshes and the LiDAR sensor. The result of this is depicted on Figure 5 and 6.

4. Experimental Design

To evaluate the effectiveness of our synthetic point cloud generation method, we conduct a series of experiments focused on 3D point cloud segmentation. This section focuses on outlining the 3D point cloud segmentation models used, the conducted tests and the dataset compositions used.

4.1. 3D Point Cloud Segmentation Model Selection

All tests performed will be done on three different models, specifically, PointNet++, PTv3 and OACNN. These models were chosen on the basis that all three approach the point cloud segmentation problem quite differently in their point cloud processing and model architectures. The hyperparameters used for the three different models are based upon the original paper implementations of the respective models. We performed a small search on the learning rate and number of epochs, as shown in the supplementary materials.

4.2. Synthetic Only Training

Recent advances in Gaussian splatting scene representation have allowed for easy generation of highly detailed meshes of custom objects which are very difficult and time consuming to model from scratch. The mesh representation can be used to generate highly realistic semantic segmentation LiDAR datasets for training, as it is easy to model custom LiDAR scanning patterns. Given the relative ease in generating synthetic datasets, it is interesting to test the performance when only utilizing a synthetically generated dataset to train a segmentation model.

To assess the models' ability to generalize to real scenes, when trained exclusively on synthetic point clouds, a test is proposed. This test also assesses how accurate the simulation replicates real-world environments. The models are trained on 10,000 synthetic point clouds and validated on 2,000 synthetic only point clouds, to ensure the models trained using this approach, have never been exposed to any real point clouds until it is tested. Lastly, the models are tested on the test set containing 3,000 real point clouds, outlined in Section 3.2.

4.3. Tractor Generalization Test

It can be hypothesized that this method of using synthetically generated data can generate otherwise hard to acquire datasets of custom objects we have physical access to, with sufficient fidelity, such that the models trained on the synthetic datasets will be able to generalize to new unseen objects of the same semantic category, when captured with real LiDAR sensors.

To test this hypothesis, multiple synthetic datasets were created in which a tractor was removed from the available assets during the generation process. As a result, each dataset lacked one specific tractor model. Individual segmentation models were then trained on these modified datasets and had their performance evaluated on the same 3,000 real point clouds as previous tests, which contains all the different tractor models.

4.4. Extended Synthetic Dataset Test

Given the ease of generating synthetic point cloud datasets, it is valuable to investigate how expanding the synthetic

Class	PointNet++ Baseline	PointNet++ Synth only	PTv3 Baseline	PTv3 Synth only	OACNN Baseline	OACNN Synth only
Tractor	0.5580	0.6430	0.8675	0.7957	0.9052	0.7755
Combine	0.8180	0.7580	0.9145	0.8853	0.9273	0.8857
Other	0.9720	0.9670	0.9878	0.9790	0.9885	0.9793
mIoU	0.7824	0.7893	0.9232	0.8867	0.9403	0.8802

Table 1. Table displays the individual IoU's and mIoU for each model traing two different datasets. Baseline is trained on all the available real data, and Synth only, is a synthetic only dataset with 10k point clouds.

dataset used in Section 4.2 affects the performance of models trained exclusively on synthetic data. To explore this, we conduct an experiment in which the number of synthetic point clouds used for training is significantly increased, from the initial 10,000 to approximately 65,000, while keeping the model architecture and training pipeline consistent.

4.5. Prediction Visualization for Analysis

As the testing scenes are static, the transforms between each point cloud can be found using KISS-ICP [29]. The transforms can be used to align the point clouds for a testing sequence creating a combined point cloud. Visualizing this point cloud provides insight into how the models generally segment the point clouds. This is done to qualitatively asses the segmentation quality of the different models.

5. Results

5.1. Synthetic Only Test

Table 1 presents the results for the synthetic only test, which tests the performance of training and validating the models without any real data. It can be seen that the mIoU across the models is on average 2.99 percentage points worse for the synthetic only models compared to the real only. The class specific IoU results reveal that all models struggle the most with the "tractor" class, which is consistent with the class-wise point distributions described in Section 3.3.

5.2. Tractor Generalization Test

The results from testing the ability to generalize across individual tractor models within the same semantic class are presented in Figure 7. The figure displays each individual model that has been trained on a dataset missing the displayed tractors, these models are also compared to the synthetic only model, seen in Section 5.1, as a baseline comparison which has been trained on all available tractors. The results show a general tendency towards a drop in performance when missing a tractor by, on average, 3.65% percentage points.

The mean of all the unseen tractor IoU's is used as a metric to compare the mean tractor IoU for the seen tractors, as this gives an image of how well the models generalize to unseen tractors compared to seen tractors. This is represented in Table 2, where it can be seen that there is a average drop in accuracy for unseen tractors, but the effect is limited, which shows that the models are able to generalize on the tractor class.

An overall view of the difference between the model having seen the tractor in the training set is presented in Figure 8. The matrix displays a small correlation between the tractor missing in the training set, and a lower IoU of the tractor in the testing set, however it is not always the case.

Model	PTv3	OACNN
Mean IoU of unseen tractors	0.7273	0.7652
Mean IoU of seen tractors	0.7603	0.8052

Table 2. Figure shows the mean IoU of tractor models not included in the training process compared to tractors which are included in the training process.

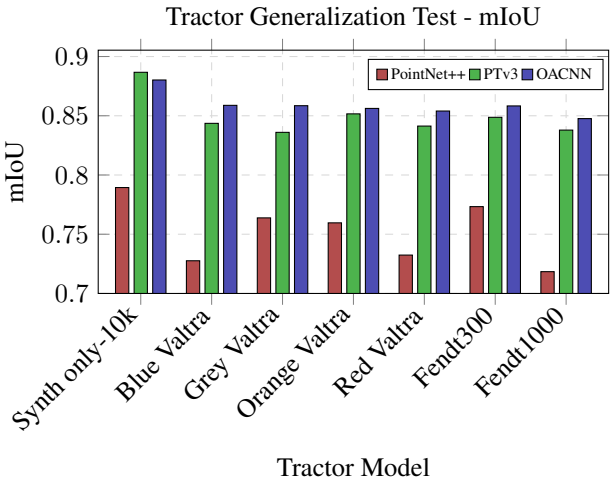


Figure 7. Figure shows the mIoU results for each test in which the specified tractor model was not included in the training/validation data. The "Synth only-10k" entry contains all the tractor models.

5.3. Extended Synthetic Dataset Test

The results of the test can be seen on Figure 9, which displays the results marked with the "Synth only-65k" label. It can be seen that performance increases significantly with a larger dataset, where PTv3 and OACNN achieve over 90% in mIoU when trained only on 65k synthetic point clouds. Additionally, it can be seen that both of the two synthetically trained PointNet++ models outperform the baseline, which is trained only on real data. This could potentially be due to the lack of augmentations in the PointNet++ implementation as opposed to PTv3 and OACNN which use several augmentation techniques.

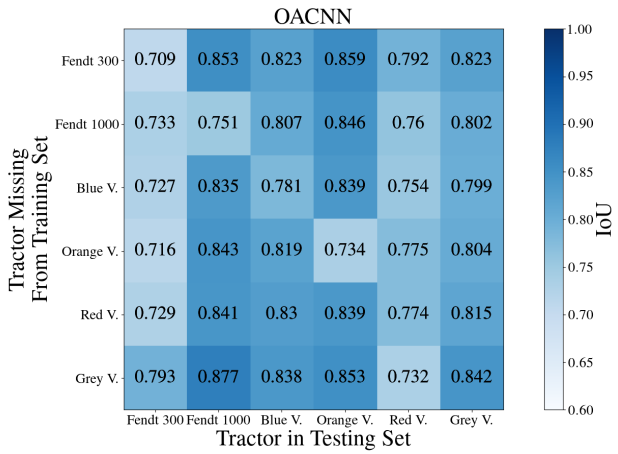


Figure 8. OACNN IoU of tractors when not in training set

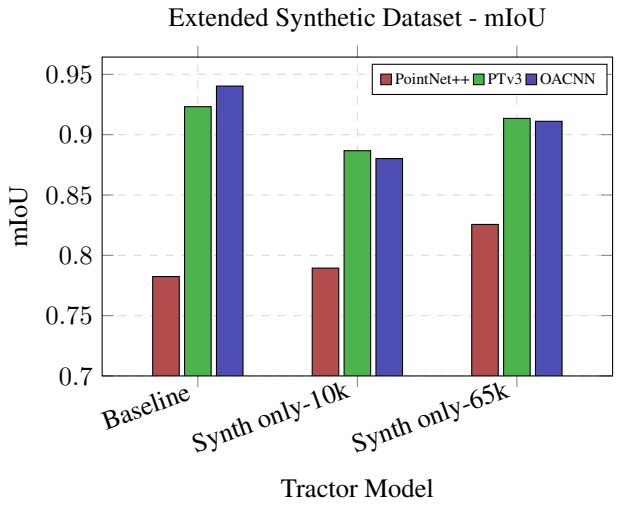


Figure 9. Figure shows the comparison between real only trained baseline models and synth only 10k models, from Table 1, with synth only trained models using 65k synthetic point clouds.

5.4. Qualitative Analysis

Using the models trained in Section 4.2 and 4.4, all seen in Figure 9, combined point clouds can be used to find the places where they differ, that are not apparent from IoU and mIoU numbers. The first example of where they differ is in models under represented in the real training data. This is seen on Figure 10a, where the trailer is completely misclassified as a combine harvester. On Figure 10b the model trained on the largest synthetic dataset handles the trailer perfectly and classifies it as the correct "other" class in the majority of the points. However, we find that the the synthetic models fail segmenting when tall grass is present, as shown in the supplementary materials.

6. Discussion

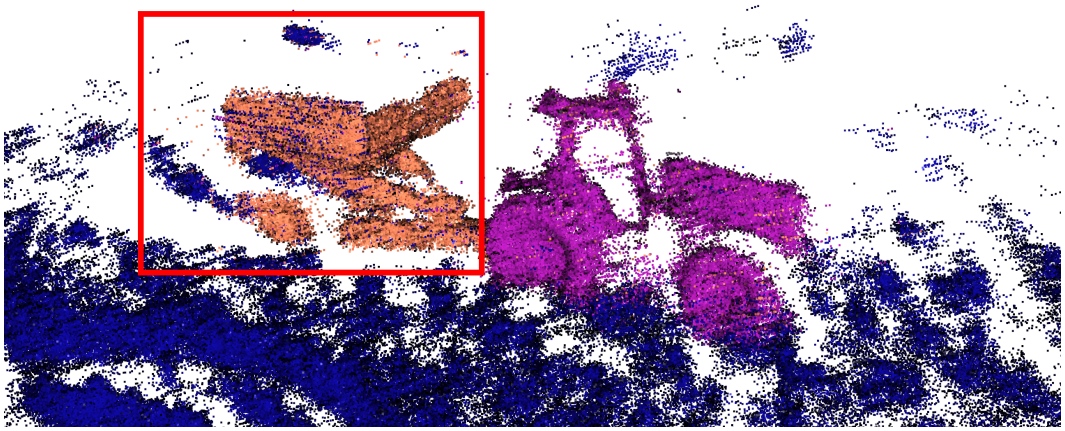
Throughout the experiments in this paper, we have not focused on the performance of the individual segmentation models, but rather on determining the influence of synthetically generated data on the performance of point cloud segmentation models. As a result, the performance in our tests could potentially be improved by modifying and fine-tuning the hyperparameters. One of the tests used to gauge the influence of synthetic data on the segmentation models, was the synthetic only test. This test showed that training only on synthetic data, could potentially be a feasible solution in domains where data is hard to acquire. Especially with larger synthetic datasets, as seen in Section 5.3, where OACNN and PTv3, trained on 65k point clouds, compared similarly to the baseline models which were trained on real data. The qualitative analysis revealed that a trailer in the testing dataset was misclassified as a combine with the real-only OACNN baseline. This was not the case with the OACNN model trained on 65k synthetic point clouds. The big difference between the training datasets of these two models, apart from the size, is the distribution of points per mesh/model. In the synthetically generated dataset, the trailer is much more common than in the real dataset. This, along with the enlarged training dataset, could be the reason for the increased performance. Additionally, the qualitative analysis also revealed that the real data helps with the classification of the tall grass in the background of the point clouds, as shown in the supplementary materials. This is most likely due to the similarity of the testing- and training dataset as they were captured on the same field, which would explain why the baseline outperformed the synthetic-only model in the area with tall grass.

7. Conclusion

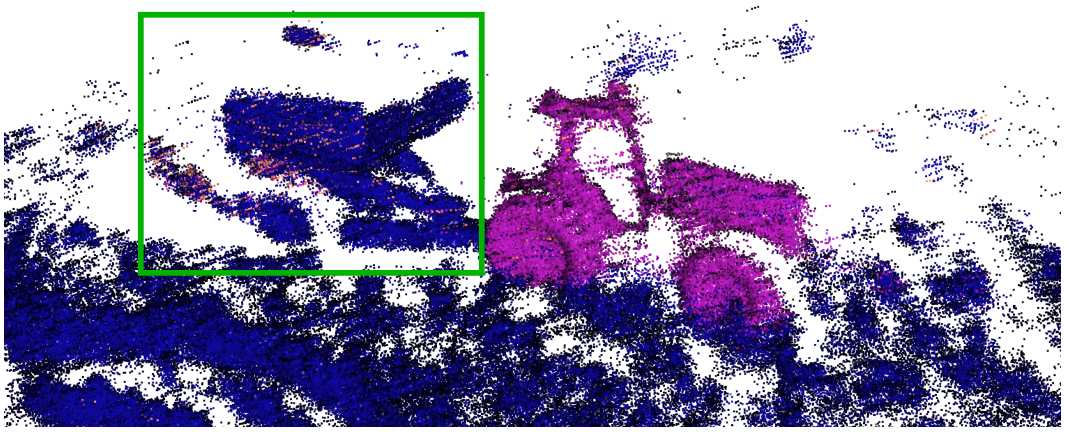
In this paper we have proposed and evaluated a novel pipeline to efficiently train point cloud segmentation models in scenarios with limited real data. The pipeline leverages GOF, a state of the art technique for mesh extraction, to obtain meshes that accurately represent target vehicles in high detail.

A simulation environment, utilizing the high quality meshes, was developed to efficiently generate LiDAR datasets suitable for semantic segmentation. Secondly, a semi-automatic annotation technique was developed, to annotate the data from the real LiDAR.

Three models were tested, namely: Point Transformer V3 [33], Omni-Adaptive Sparse CNN [19], and Pointnet++ [21]. Multiple tests were carried out for the different models exploring different ways that synthetic data could be used to train point cloud segmentation models. The synthetic only test, seen in Section 5.1, shows the potential of training models purely on synthetic data, with OACNN and PTv3



(a) **OACNN Baseline model point predictions**. Notice how the trailer is incorrectly predicted to be a combine harvester, as highlighted by the red ■ bounding box.



(b) **OACNN Synth only-65k model point predictions**. Notice how the trailer is correctly predicted to be of the "other", as highlighted by the green ■ bounding box.

Figure 10. We compare the point predictions of a OACNN network trained in the baseline configuration versus trained with the Synth only-65K configuration. The scene depicts a tractor and a trailer. The point colors indicate the predicted class: ■ other, ■ tractor, and ■ combine harvester

achieving $+88\%$ mIoU. Additionally, the extended dataset test, seen in Section 5.3, showed that expanding the dataset significantly improved the mIoU for all the models, with OACNN and PTv3 now surpassing $+91\%$ in mIoU, almost comparable to the baseline trained on real data. Secondly it was shown that the model is able to generalize well to unseen tractor models when trained only on synthetic data, with a mean performance drop in IoU for unseen tractors of 3.65 percentage points from Table 2. Thirdly, qualitative analysis showed that in some cases the models trained on synthetic data had more desirable predictions, which is also a strong argument for synthetic data, as this is presumably caused by the perfect annotations that are acquired when using the proposed pipeline. Finally, with the results gathered from all the experiments, it is shown that using the proposed novel pipeline for synthetic data acquisition, is a viable solution when gathering data for training point cloud segmentation models in uncommon domains. While this paper focused on a single domain, there is strong evidence which suggests that other domains would benefit from using a similar data generation pipeline. Moving forward, it would be interesting to see the effects of this method on more common domains, such as semanticKITTI [3], to evaluate its effect on well-known datasets and gain insights into the drawbacks and benefits.

References

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022. 3

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anto-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19697–19705, 2023. 3

[3] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences, 2019. 8

[4] Yunlu Chen, Vincent Tao Hu, Efstratios Gavves, Thomas Mensink, Pascal Mettes, Pengwan Yang, and Cees GM Snoek. Pointmixup: Augmentation for point clouds. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 330–345. Springer, 2020. 1, 2

[5] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 3

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017. 2

[7] Magnus Kaufmann Gjerde, Filip Slezák, Joakim Bruslund Haurum, and Thomas B. Moeslund. From neRF to 3DGS: A leap in stereo dataset quality? In *Synthetic Data for Computer Vision Workshop @ CVPR 2024*, 2024. 3

[8] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5354–5363, 2024. 3, 4

[9] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5875–5884, 2021. 3

[10] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 3, 4

[11] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering, 2023. 3

[12] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, pages 2149–2154. Ieee, 2004. 3

[13] 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. 1, 2

[14] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31, 2018. 1, 2

[15] Han Ling, Quansen Sun, Yinghui Sun, Xian Xu, and Xingfeng Li. Adfactory: An effective framework for generalizing optical flow with nerf. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20591–20600, 2024. 3

[16] Jong Won Ma, Thomas Czerniawski, and Fernanda Leite. Semantic segmentation of point clouds of building interiors with deep learning: Augmenting training datasets with synthetic bim-based point clouds. *Automation in construction*, 113:103144, 2020. 1, 2, 4

[17] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934*, 2020. 3

[18] Ouster. Ouster os0: High-precision ultra wide, 2024. Accessed on November 22, 2024. 3

[19] Bohao Peng, Xiaoyang Wu, Li Jiang, Yukang Chen, Hengshuang Zhao, Zhuotao Tian, and Jiaya Jia. Oa-cnns: Omni-adaptive sparse cnns for 3d semantic segmentation, 2024. 1, 2, 7

[20] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017. 2

[21] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017. 1, 2, 7

[22] Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 42(4):1–12, 2023. 3

[23] Sadra Safadoust, Fabio Tosi, Fatma Güney, and Matteo Poggi. Self-evolving depth-supervised 3d gaussian splatting from rendered stereo pairs. In *British Machine Vision Conference (BMVC)*, 2024. 3

[24] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4

[25] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[26] Ayça Takmaz, Jonas Schult, Irem Kaftan, Mertcan Akçay, Bastian Leibe, Robert Sumner, Francis Engelmann, and Siyu Tang. 3d segmentation of humans in point clouds with synthetic data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1292–1304, 2023. 2, 3

[27] Fabio Tosi, Alessio Tonioni, Daniele De Gregorio, and Matteo Poggi. Nerf-supervised deep stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 855–866, 2023. 3

[28] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 2

[29] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1029–1036, 2023. 4, 6

[30] Fei Wang, Yan Zhuang, Hong Gu, and Huosheng Hu. Automatic generation of synthetic lidar point clouds for 3-d data analysis. *IEEE Transactions on Instrumentation and Measurement*, 68(7):2671–2673, 2019. 1, 2

[31] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5):1–12, 2019. 1, 2

[32] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. *Advances in Neural Information Processing Systems*, 35:33330–33342, 2022. 2

[33] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger, 2024. 1, 2, 7

[34] Aoran Xiao, Jiaxing Huang, Dayan Guan, Fangneng Zhan, and Shijian Lu. Synlidar: Learning from synthetic lidar sequential point cloud for semantic segmentation. *CoRR*, abs/2107.05399, 2021. 2, 4

[35] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P Srinivasan, Richard Szeliski, Jonathan T Barron, and Ben Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–9, 2023. 3

[36] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient and compact surface reconstruction in unbounded scenes. *arXiv preprint arXiv:2404.10772*, 2024. 3, 4

[37] 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. 1, 2

[38] Siwei Zhang, Yan Zhang, Qianli Ma, Michael J Black, and Siyu Tang. Place: Proximity learning of articulation and contact in 3d environments. In *2020 International Conference on 3D Vision (3DV)*, pages 642–651. IEEE, 2020. 3

[39] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16259–16268, 2021. 1, 2

Point Cloud Segmentation of Agricultural Vehicles using 3D Gaussian Splatting

Supplementary Material

Epochs	PointNet++	PTv3	OACNN
10	0.7608	0.8549	0.8708
20	0.7844	0.8867	0.8802
30	0.7894	0.8911	0.8917

Table 3. Table displays the mIoU of the three different models when trained for three different epoch amounts.

LR scaling	PointNet++	PTv3	OACNN
0.1	0.6876	0.8936	0.8914
1.0	0.7894	0.8867	0.8802
10.0	0.6932	N/A	0.8968

Table 4. Table displays the mIoU of the three different models when trained on three different learning rate scales. The original implementation learning rates were 2e-3 with batch size 16 for both PointNet++ [21] and OACNN [19] and 5e-3 for PTv3 with batch size 12 [33]. The conducted tests were run with batch size 32 for all models, consequently the learning rates were scaled proportially to this as well.

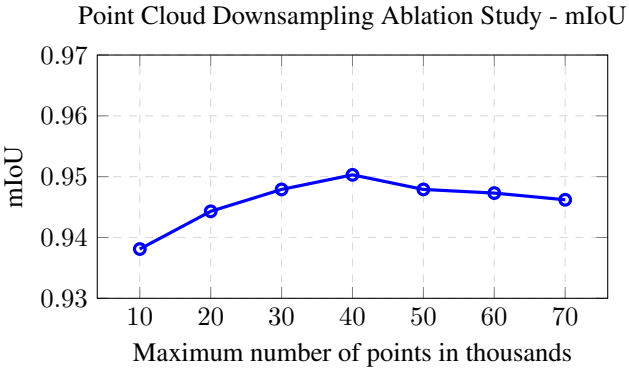


Figure 11. Figure shows the mIoU results for the OACNN model, trained on a real only dataset, with varying amounts of downsampling performed on the input point clouds.

A. Model and Training Tuning

Given the objective of the paper has not been to fine-tune the individual models to obtain the best possible performance, model training efficiency was valued highly when weighing performance against training time for the hyperparameter selection, and for the most part, hyperparameters have been chosen based solely on the model's original paper implementations [19, 21, 33]. Thus, as a last test it also seemed interesting to observe the performance differences when modifying the base hyperparameters. The tests focused on three key hyperparameters: the number of training epochs, the learning rate and the degree of point cloud downsampling.

A.1. Epoch Tuning

The default number of epoch used throughout the test has been 20, thus when varying the number of epochs it was chosen to test {10, 20, 30} epochs. The results can be found in Table 3. As the results suggest, increasing the number of training epochs helps performance quite notably, and from the training process it seems like the models haven't quite converged, and thus could improve even further given additional epochs.

A.2. Learning Rate Tuning

The learning rate is usually the most impactful hyperparameter when changed, in this case a sweep was conducted where the base learning rate used for all other model's tests was scaled by {0.1, 1.0, 10.0}. The results can be seen in Table 4. Interestingly it seems the PTv3 model would benefit from a lowered learning rate, and when scaling it by 10, it would outright crash because of exploding gradients, further suggesting that the PTv3 model should have its learning rate lowered. The results for the OACNN model encourages the opposite, that it should be trained with a higher learning rate.

A.3. Point Cloud Downsampling

To assess the trade-off between computational efficiency and segmentation performance, different point cloud densities were tested. A downsampled point cloud is obtained by performing a uniform random sampling without replacement from the original point cloud. The results can be seen in Figure 11. Mostly due to VRAM usage constaints, point clouds have been downsampled to 30.000 points for PointNet++ and 40.000 points for OACNN and PTv3 for all tests, and as the results display, this downsampling does not hinder the model from sufficiently learning the classwise point cloud representations, unless the point clouds are significantly downsampled.

A.4. Mixed Training Dataset

To establish the effects of synthetic data on the 3D semantic segmentation models, we trained the three different models on a combination of synthetic data and real data, and then compared this to models trained only on real data. This was inspired by Ma et al. [16], Wang et al. [30], and Yue et al. [37]. In total, the training dataset consists of 10,000 point clouds, with a 50/50 split of synthetic and real, as done by

Sample	PointNet++		PTv3		OACNN		Mean
percentage	Real only	Mixed	Real only	Mixed	Real only	Mixed	increase
20%	0.6982	0.7461	0.8369	0.8762	0.8465	0.9247	0.0551
40%	0.7215	0.7942	0.8983	0.9116	0.9191	0.9451	0.0373
60%	0.7406	0.7992	0.9096	0.9301	0.9349	0.9474	0.0305
80%	0.7505	0.7673	0.9065	0.9297	0.9378	0.9498	0.0173
100%	0.7824	0.7894	0.9232	0.9328	0.9403	0.9517	0.0093

Table 5. Table displays the mIoU performance comparison between models trained on real dataset comprised of different amounts of real point clouds, and mixed datasets. The real only column shows baseline models trained on varying amounts of real data. Mean increase displays the average increase seen when using a mixed dataset compared to using only real. The data is presented for each sample percentage and each individual model.

Yue et al. [37]. The synthetic part is composed solely from unique point clouds, while the real part is composed of 20%, 40%, 60%, 80% and 100% of all available unique real point clouds. The real point clouds are over-sampled to match the amount of synthetic point clouds, to avoid biasing the model towards the synthetic data. In total, five different datasets are produced, which individual models are trained upon, each of these datasets also contains the same validation split consisting of 1,200 real point clouds. For evaluation, all models are tested on the test set containing 3,000 real point clouds, outlined in Section 3.2.

The test results evaluating the impact of the mixed dataset training compared to real-only dataset training are presented in Table 5 and Figure 12. Table 5 compares the performance of the models, trained only on real data, to the models which were trained on a combination of real data and synthetic data. Additionally, the mean increase is shown, and it can be seen that the synthetic data improves performance, especially when only a small amount of real data is available. Figure 12 visually illustrates the mIoU performance of the OACNN model, comparing mixed dataset training with real-only dataset training as the number of different real point clouds increases.

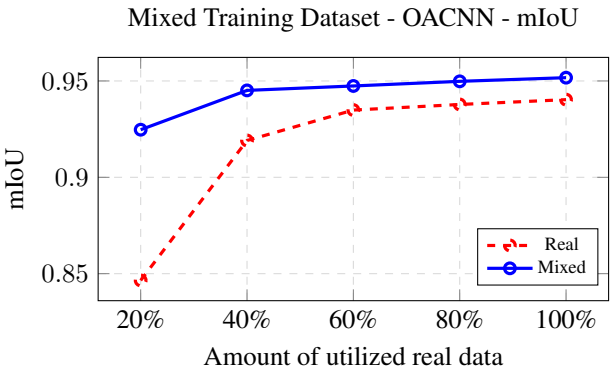
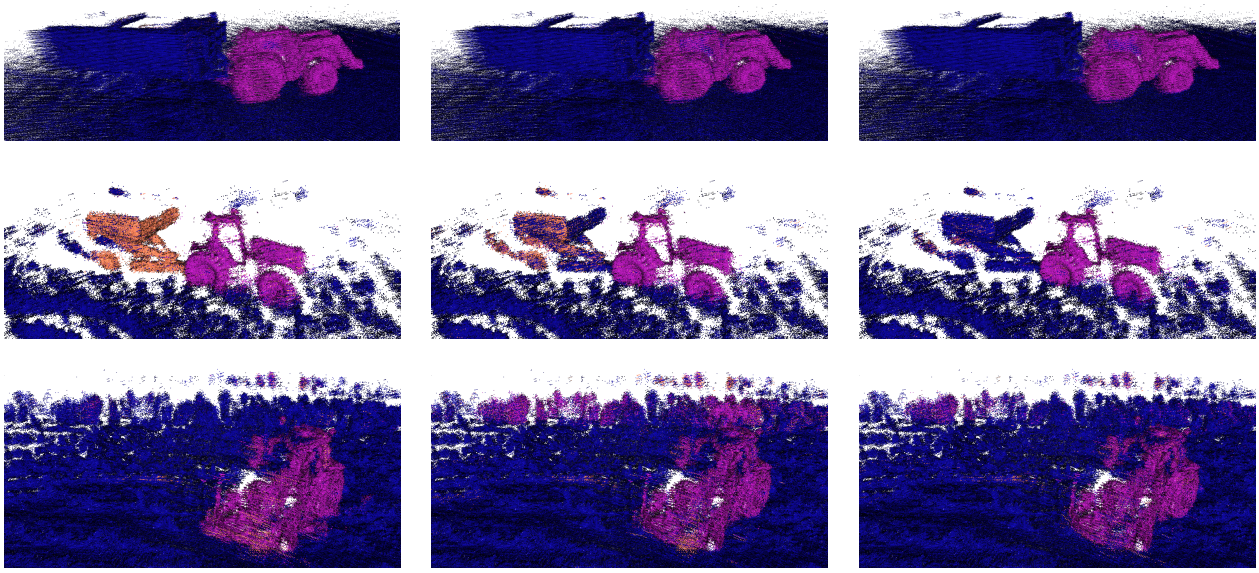


Figure 12. Figure shows the mIoU results for the OACNN model, trained on both a mixed dataset and a real only dataset, consisting of different amounts of different real point clouds.

B. Combined point clouds with predictions

Selection of combined point clouds with predictions from multiple models are shown in 13. Notice how the baseline model can produce better prediction in the presence of tall grass, due to tall grass not being included in the data simulation.



(a) OACNN Baseline (b) OACNN 10k synth (c) OACNN 65k synth

Figure 13. ■ other, ■ tractor, and ■ combine harvester Top row: tractor with large trailer, second row: tractor with small trailer, Third row tractor with tall grass backdrop.