Towards Generating Realistic 3D Semantic Training Data for Autonomous Driving

Lucas Nunes   Rodrigo Marcuzzi   Jens Behley   Cyrill Stachniss

Abstract—Semantic scene understanding is crucial for robotics and computer vision applications. In autonomous driving, 3D semantic segmentation plays an important role for enabling safe navigation. Despite significant advances in the field, the complexity of collecting and annotating 3D data is a bottleneck in this developments. To overcome that data annotation limitation, synthetic simulated data has been used to generate annotated data on demand. There is still, however, a domain gap between real and simulated data. More recently, diffusion models have been in the spotlight, enabling close-to-real data synthesis. Those generative models have been recently applied to the 3D data domain for generating scene-scale data with semantic annotations. Still, those methods either rely on image projection or decoupled models trained with different resolutions in a coarse-to-fine manner. Such intermediary representations impact the generated data quality due to errors added in those transformations. In this work, we propose a novel approach able to generate 3D semantic scene-scale data without relying on any projection or decoupled trained multi-resolution models, achieving more realistic semantic scene data generation compared to previous state-of-the-art methods. Besides improving 3D semantic scene-scale data synthesis, we thoroughly evaluate the use of the synthetic scene samples as labeled data to train a semantic segmentation network. In our experiments, we show that using the synthetic annotated data generated by our method as training data together with the real semantic segmentation labels, leads to an improvement in the semantic segmentation model performance. Our results show the potential of generated scene-scale point clouds to generate more training data to extend existing datasets, reducing the data annotation effort. Our code is available at https://github.com/PRBonn/3DiSS.

---

1 INTRODUCTION

SEMANTIC perception is a key element in most robotics and computer vision applications. For example, in autonomous driving, the capability of semantically interpret the scene is crucial for the safe navigation of the vehicle. Semantic segmentation is key for scene understanding and has been intensively studied in recent years [34], [51], [65]. Advances in this field have been made possible also due to the availability of semantically annotated data [8], [25]. However, the availability of such fine-grained annotated data for 3D scene-scale data is scarcer [1], [3], [11] compared to the image domain. Data collection and annotation of 3D data is more challenging than for images, often hindering the scalability of 3D scene-scale labeled data.

Synthetic simulated data has been used to alleviate the labeling effort [10], [46], [55], but there is typically a domain gap between simulated and real data. This often impedes the large-scale use of simulated annotated data as labels for real-world applications [5], [56], [61]. More recently, generative models have been in the spotlight given the recent advances in denoising diffusion probability models (DDPMs) [9], [15], [42], [45] and its realistic results. Due to the proximity between synthetic and real data that DDPMs enable, recent works have studied the use of synthetic data from those generative models as labeled data [12], [48], [52], [53]. Still, those studies focus mainly on image data and on coarse downstream tasks, such as classification.

Synthetic data generation has been studied for 3D data [29], [57], [58] given the advances in the image domain [9], [15], [42],

[45]. For 3D scene-scale data the topic is also of great interest, even though the works in this field are more limited [22], [23], [26], [44]. Some works [22], [26] rely on a discrete formulation of the diffusion process, treating the scene as a fixed voxel grid, and predicting the semantic label for each voxel. The memory cost of such methods is relatively high since empty voxels in the grid are also computed, limiting the resolution of the generated scene. Other methods rely on intermediary data representations from the scene, such as image projections of the 3D point cloud [23], or training multiple models separately at different scene resolutions, modeling the generation process as a multi-resolution conditional distribution [44]. Despite these promising results, those methods rely on simpler or intermediate coarser representations of the real-world data to simplify the 3D scene-scale data generation. Training the DDPM in such intermediate representations can limit its generation capabilities due to the loss of information when creating those representations. In this work, we propose training the diffusion model without relying on image projections or intermediate coarser scene generation. In our experiments, we show that by training the DDPM directly on the 3D data and at the target resolution, we were able to achieve more realistic scene generation compared to previous state-of-the-art methods.

Alongside the proposed model, we extend the analysis from previous works in this field. We evaluate the use of the synthetic data as additional labels for training a 3D semantic segmentation network, following the aforementioned works that used generated data as training data in the image domain [12], [48], [52], [53]. We compare the use of synthetic scenes against real data as training labels, and evaluate its use to extend the training set. We train the model with both real and synthetic scenes and assess the impact of synthetic data in real-world applications. Besides, we compare the gaps between generated and real scenes, providing insights on the current challenges in scene-scale 3D data generation.

- *L. Nunes is with the Center for Robotics, University of Bonn, Germany, and the RWTH Aachen, Germany. E-Mail: lucas.nunes@igg.uni-bonn.de*
- *R. Marcuzzi, and J. Behley are with the Center for Robotics, University of Bonn, Germany. E-mails: {rodrigo.marcuzzi, jens.behley}@igg.uni-bonn.de*
- *C. Stachniss is with the Center for Robotics, University of Bonn, Germany, and the Lamarr Institute for Machine Learning and Artificial Intelligence, Germany. E-Mail: cyrill.stachniss@igg.uni-bonn.de*

In sum, our main contributions are the following:

- We propose a new method for generating 3D scene-scale semantic data without relying on multi-resolution representations or projections of the 3D data.
- Our method generates more realistic scenes and this data is better suited for real-world tasks compared to previous state-of-the-art methods.
- We show that our generated data improves the performance on semantic segmentation when using both real and the synthetic data generated by our method.
- We compare both real and generated data distributions and identify current gaps between synthetic and real data to be addressed in future work.

2 RELATED WORK

Denoising diffusion probabilistic models (DDPMs) recently have been of great interest due to their high-quality image generation results [9], [15], [45], [64]. Such methods model the data generation as a denoising process, starting from Gaussian noise and iteratively predicting and removing noise from it, slowly arriving at a novel sample from the target distribution. Given the close-to-real generation of such models, recent studies focus on extending those methods to enable conditional generation, allowing the synthesis of images resembling a specific input [16], [35], [60], often called conditioning. However, the DDPM image generation is relatively slow compared to other generative methods, e.g., Generative Adversarial Networks (GANs) [19], [40], [59] and Variational Auto-Encoders (VAEs) [17], [21], [54]. Therefore, recent works target reducing the DDPMs inference time while maintaining the quality of the results, e.g., via distillation [33], [47] or by analytically solving multiple denoising steps [18], [28].

Diffusion models for 3D data have been recently studied due to the advances in the image domain. Most of the methods in the field focus on single object generation [30], [57], [58], [63]. Those methods can be divided into two categories, point [30], [63] and latent diffusion methods [57], [58]. The former operates directly over the point's 3D coordinates, learning how to generate the target point cloud from the starting random point coordinates. The latter follows the latent diffusion model first proposed in the image domain [2], [45], where first, a VAE is trained to encode the point cloud into a latent space. Then, the DDPM is trained to generate novel samples from the learned latent space, decoding it to a new sample. More recent works extend such methods to operate in real-world LiDAR data. Some works [36], [43], [66] use image projections from LiDAR scans to train a DDPM to generate new samples of such image projections to then unproject it to the point cloud. Other works [38], [49], [62] use a local point diffusion formulation [38] to enable scene-scale point diffusion, achieving LiDAR scene completion with DDPMs.

Semantic scene-scale diffusion models target generating large-scale 3D scenes with semantic labels and have been studied in recent works [22], [23], [26], [44]. Lee et al. [23] tackle this problem by projecting the target semantic 3D data into a triplane image representation. DDPM image models are then used to generate new triplane samples, which are then unprojected to the corresponding 3D scene. However, this data projection loses information, limiting the details of the generated scene. Ren et al. [44] achieve scene-scale data generation using a hierarchical approach to model the coarse-to-fine nature of the scene. The generation process is formulated as a conditional distribution between different scene resolutions, training independently a VAE and a DDPM for each pre-defined resolution. However, this formulation can lead to errors during the generation process since the finer scene generation is unable to recover from mistakes done at coarser stages. Distinctly, Lee et al. [22] propose using a discrete diffusion formulation to generate semantic scene-scale 3D data. The scene is represented as a fixed 3D grid, and the DDPM is trained to generate the semantic labels for each cell in the grid, including an *unoccupied* class. Liu et al. [26] extend this formulation by also leveraging a hierarchical multi-resolution approach, enabling a more detailed generation. Yet, such discrete formulation limits the scene resolution due to memory constraints since the diffusion process is computed for the whole 3D grid, even though most of the cells are empty.

In contrast to previous works, we propose a 3D latent diffusion model, which does not rely on data projections or decoupled multi-resolution models, but uses a single sparse 3D VAE model to learn the target data distribution. By using a sparse 3D VAE, we train the model with the target data while avoiding the increasing memory usage. Besides, we supervise the model to prune the scene within each decoder upsampling layers, learning the hierarchical scene structure with a single model, achieving more realistic generation.

Reducing labeling effort has been the focus of many recent works given the challenge of scaling data annotation [4], [10], [14], [55]. To alleviate this problem, some works [10], [46], [55] propose using simulated training examples as labeled data. However, such simulated data come with a domain gap compared to the real-world data [5], [56], [61], often impeding the large-scale use of simulated training data. Other works [4], [14], [37], [39] try to alleviate this problem using self-supervised pre-training methods to learn a representation from unlabeled data, that allows to later fine-tuning the pre-trained model to the target downstream tasks. More recently, due to the realistic data synthesis achieved by generative methods, some works [12], [48], [52], [53] have studied the use of generated samples as training data in the image domain. Especially for the 3D data domain, the reduction of annotation requirements can largely impact the field due to the more complex task of data annotation compared to images. Previous semantic scene-scale diffusion methods [22], [23], [26], [44] have motivated their works arguing about the capability of generating semantically annotated scenes on demand. However, none of the previous works have studied the impact of using the generated scenes as training data. Besides achieving more realistic scene generation, we evaluate the use of generated semantic scenes as training data and show its potential to alleviate the burden of data annotation.

In this work, we propose a semantic scene-scale diffusion method that can generate high-resolution scenes without intermediary projections and using a single VAE model. We train a 3D sparse VAE to learn the target scene distribution while supervising it to prune unoccupied voxels at each decoder upsampling layer. By doing so, the network learns to model the hierarchical nature of the scene with a single resolution VAE while avoiding the increasing memory usage. Our experiments show that our method achieves more realistic scene generation compared to previous state-of-the-art methods using a single sparse 3D VAE. Additionally, we assess the use of our generated scenes as training data for semantic segmentation. We show that by using our generated scenes together with real data, we improved the semantic segmentation model performance. Finally, we perform an experiment to identify the gaps between real and generated data, providing insights into the challenges to bridge this gap in future work.
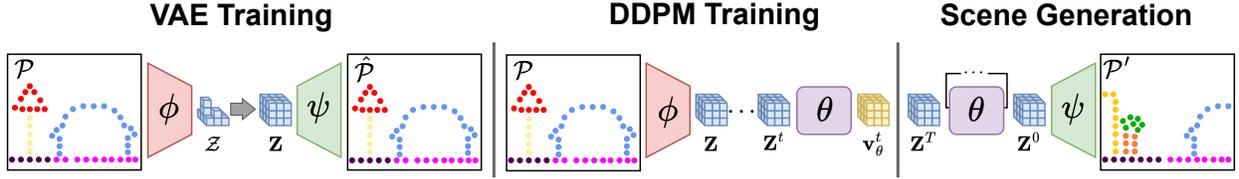
Fig. 1. Our scene generation pipeline: First we train a VAE with the dense scenes $\mathcal{P}$ to reconstruct it as $\hat{\mathcal{P}}$ and to learn the sparse $\mathcal{Z}$ and dense $\boldsymbol{Z}$ latent spaces. Next, the DDPM θ is trained over the latent $\boldsymbol{Z}$ sampling a random step t to compute the noisy latent $\boldsymbol{Z}^t$, training the model θ to predict $\mathbf{v}_\theta^t$, following the **v**-parameterization formulation [47]. Finally, novel scenes are generated by sampling random noise $\boldsymbol{Z}^T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and denoising it with θ over T times, arriving to $\boldsymbol{Z}^0 = \boldsymbol{Z}_\theta$, decoding it with the VAE decoder to get to the generated scene $\mathcal{P}' = \psi(\boldsymbol{Z}_\theta)$.

3 OUR APPROACH

In this section, we describe our approach to generate semantic scene-scale data. We propose a DDPM [9] model to generate semantic scene-scale data without relying on intermediary image projections and without independent multi-resolution models training. We first train a VAE to encode the semantic scene into a descriptive latent space. Then, we train a DDPM to learn the latent space of the VAE to generate new samples. Using the VAE decoder to turn the sampled latent into the semantic scene. Fig. 1 depicts an overview of our pipeline for scene-scale generation.

3.1 Semantic Scene VAE

DDPMs are known for being computationally demanding. A common approach to simplify the generation process is to train an autoencoder to learn a descriptive latent representation of the target data and then train the DDPM to generate novel samples from the learned latent space [45]. Similarly, we first train a VAE with the semantic scene data, and then train the DDPM over the learned latent space, simplifying the semantic scene generation.

Let us define a voxelized 3D point cloud $\mathcal{P}$ with N voxels with coordinates $\mathcal{P}_\mathcal{C} = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N\}$, $\boldsymbol{c}_n \in \mathbb{R}^3$ within a fixed range over $[x, y, z]$, and features $\mathcal{P}_\mathcal{F} = \{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\}$, $\boldsymbol{f}_n \in \mathbb{R}^4$ as $\boldsymbol{f}_n = (x, y, z, s)$ where $s \in \{1, \ldots, C\}$ is the semantic label for that voxel. We want to train a VAE with an encoder ϕ and a decoder ψ, optimizing both such that $\psi(\phi(\mathcal{P})) = \hat{\mathcal{P}} \approx \mathcal{P}$.

Ren et al. [44] modeled this training as R independent sparse 3D VAE models trained at different resolutions of $\mathcal{P}$ as $\hat{\mathcal{P}}_r = \psi_r(\phi_r(\mathcal{P}_r))$. Then, R independent DDPMs θ_r are trained to hierarchically generate a novel scene $\mathcal{P}'$ by conditioning the finer resolutions' scene generation to previous coarser stages as $\mathcal{P}'_r = \psi_r(\theta_{r-1}(\mathcal{N}(\mathbf{0}, \boldsymbol{I}), \mathcal{P}'_{r-1}))$, until the final scene resolution. The motivation behind this formulation is to model the coarse-to-fine scene data nature and to simplify the scene-scale data generation. However, we argue that this coarse-to-fine data aspect can be modeled by a single encoder-decoder network in the sequential downsampling and upsampling layers. Furthermore, this multi-resolution formulation can lead to incremental errors since the R models are trained independently. Therefore, we train a single 3D VAE model at the original resolution of $\mathcal{P}$.

The encoder ϕ receives the voxelized point cloud $\mathcal{P}$ as input and encodes it via consecutive downsampling convolutions into the latent representation $\mathcal{Z}$ with coordinates $\mathcal{Z}_\mathcal{C} = \{\boldsymbol{c}_1^{\boldsymbol{z}}, \ldots, \boldsymbol{c}_M^{\boldsymbol{z}}\}$ with $\boldsymbol{c}_m^{\boldsymbol{z}} \in \mathbb{R}^3$ and features $\mathcal{Z}_\mathcal{F} = \{\boldsymbol{f}_1^{\boldsymbol{z}}, \ldots, \boldsymbol{f}_M^{\boldsymbol{z}}\}$ with $\boldsymbol{f}_m^{\boldsymbol{z}} \in \mathbb{R}^{d_z}$, where $M < N$, and d_z being the latent feature dimension. The latent coordinates $\mathcal{Z}_\mathcal{C}$ are a lower resolution version of the input voxel coordinates $\mathcal{P}_\mathcal{C}$ after the encoder sparse convolutional layers. During the DDPM training, we want to generate novel scenes from scratch, not being constrained by a prior scene shape. Therefore, we reshape the latent representation $\mathcal{Z}$ into a dense latent grid
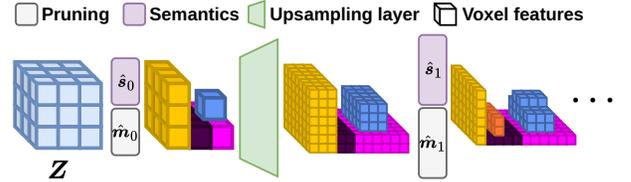

Fig. 2. Diagram of the pruning process. Before each lth upsampling layer the model predicts the semantics $\hat{\boldsymbol{s}}_l$ and the pruning masks $\hat{\boldsymbol{m}}_l$ and prune the unoccupied voxels, starting from the dense latent $\boldsymbol{Z}$.

$\boldsymbol{Z}$. Given that the voxelized point cloud $\mathcal{P}$ has a fixed range over the axes and a fixed resolution of $0.1\,\text{m}$, the dense latent grid dimensions $\boldsymbol{Z} \in \mathbb{R}^{H \times W \times D \times d_z}$ are known. We initialize $\boldsymbol{Z}$ with zeros and set the occupied cells as $\boldsymbol{Z}(\boldsymbol{c}) = \mathcal{Z}(\boldsymbol{c}) \ \forall \boldsymbol{c} \in \boldsymbol{Z}_\mathcal{C} \cap \mathcal{Z}_\mathcal{C}$.

The goal of the decoder ψ is to reconstruct $\mathcal{P}$ from the dense latent grid $\boldsymbol{Z}$. To decode and upsample $\boldsymbol{Z}$ would require many computational resources due to the exponential increase of the grid size within each upsampling. We overcome this issue by predicting a pruning mask $\hat{\boldsymbol{m}}_l \in [0, 1]^{H_l \times W_l \times D_l}$ and per-voxel semantic class $\hat{\boldsymbol{s}}_l \in \mathbb{R}^{H_l \times W_l \times D_l \times C}$ with a pruning layer and a linear layer before l^{th} upsampling convolutional layer from ψ, as shown in Fig. 2. That way, we upsample the latent grid $\boldsymbol{Z}$ back to the original input resolution, pruning the unoccupied voxels while also predicting the semantics of each occupied voxel. At the same time, the supervision of the pruning masks $\hat{\boldsymbol{m}}_l$ optimizes the network towards learning the scene layout at each lth upsampling convolutional layer. We then compute $\boldsymbol{Z} = \phi(\mathcal{P})$, $\hat{\mathcal{P}} = \psi(\boldsymbol{Z})$, the predicted pruning masks $\hat{\mathcal{M}} = \{\hat{\boldsymbol{m}}_1, \ldots, \hat{\boldsymbol{m}}_L\}$, and semantics $\hat{\mathcal{S}} = \{\hat{\boldsymbol{s}}_1, \ldots, \hat{\boldsymbol{s}}_L\}$ from each lth upsampling layer.

3.1.1 Pruning Loss

The pruning masks $\hat{\mathcal{M}}$ aim at predicting unoccupied voxels and remove them to reduce memory consumption when upsampling the dense latent $\boldsymbol{Z}$, while learning the scene layout at each lth upsampling layer. The goal is to learn the coarse-to-fine nature of the scene within a single decoder, previously done with multiple independent VAE models [44]. Therefore, we supervise the prediction of the pruning masks $\hat{\mathcal{M}}$ with the target masks $\mathcal{M}$, which are computed from the downsampling of the original scene $\mathcal{P}$. We then train the model with the binary cross-entropy (BCE) and dice losses following recent mask-based segmentation approaches [6], [31], [32] computed between the prediction $\hat{\boldsymbol{m}}_l$ and the target $\boldsymbol{m}_l$:

$$\mathcal{L}_{\text{prune}} = \sum_{l=1}^{L} \lambda_l \left(\mathcal{L}_{\text{bce}}^l(\hat{\boldsymbol{m}}_l, \boldsymbol{m}_l) + \mathcal{L}_{\text{dice}}^l(\hat{\boldsymbol{m}}_l, \boldsymbol{m}_l) \right), \quad (1)$$

where λ_l is the weight for pruning loss at the lth upsampling layer. Both losses complement each other, while the BCE loss optimizes the individual voxels learned features, the dice loss targets the scene layout, optimizing for the masks IoU. In that way, we optimize the decoder ψ to learn to prune individual voxels while also learning the whole scene layout at each lth layer.

3.1.2 Semantic Loss

We supervise the model to predict a semantic class for the voxels not pruned by the predicted pruning masks $\hat{\mathcal{M}}$. At each lth upsampling layer, we compute the cross-entropy loss between the set of semantic predictions $\hat{s}_l$ and the target semantics s_l computed from the downsampling of the original scene input features $\mathcal{P}_{\mathcal{F}}$. Given the class imbalance intrinsic to scene-scale data, first we supervise the VAE with the weighted cross-entropy loss:

$$\mathcal{L}_{\text{sem}} = -\sum_{l=1}^{L} \frac{1}{N^l} \sum_{n=1}^{N^l} w_{s_n^l} \log \frac{\exp \hat{s}_{n,s_n^l}^l}{\sum_{c=1}^{C} \exp \hat{s}_{n,c}^l}, \qquad (2)$$

with $w_{s_n^l}$ as the weight of the target class at the nth voxel from the prediction $\hat{s}_l$. Then, for the second half of the training we train the model with the unweighted loss, i.e., $w_{s_n} = 1$. This unweighted loss training allows the VAE to learn the class imbalance from the training data to properly account for the scene data distribution.

3.1.3 Latent Loss

With the pruning and semantic losses, we optimize the model to reconstruct structural and semantic information from the scene. However, for the DDPM to generate meaningful scenes, the learned latent space has to be representative and continuous. Therefore, we regularize encoder ϕ during training using the KL divergence to approximate the encoder latent distribution $q_\phi(\mathcal{P})$ to a target distribution, which is set to $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ as:

$$\mathcal{L}_{\text{latent}} = \mathbb{KL}(q_\phi(\mathcal{P}) \,||\, \mathcal{N}(\mathbf{0}, \boldsymbol{I})), \qquad (3)$$

where $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ is a zero-centered isotropic normal distribution. With the VAE latent regularization we optimize the latent space towards a continuous representative distribution. This facilitates the training of the DDPM and the generation of new samples from the latent distribution q_ϕ, which later can be decoded by ψ.

3.1.4 VAE training

Finally, the VAE is trained to reconstruct the original scene while regularizing the learned latent space. The final loss is a weighted combination of all three losses as:

$$\mathcal{L}_{\text{VAE}} = \lambda_{\text{prune}} \mathcal{L}_{\text{prune}} + \lambda_{\text{sem}} \mathcal{L}_{\text{sem}} + \lambda_{\text{latent}} \mathcal{L}_{\text{latent}}. \qquad (4)$$

Besides, we do a refinement stage similar to previous approaches [38], [44]. After the VAE is trained, we repeat the training adding noise to the latent $\boldsymbol{Z}$, optimizing only the decoder ψ to reconstruct the scene $\mathcal{P}$ from the noisy latent. With Eq. (4), by learning the pruning at each lth layer, we achieved the coarse-to-fine modeling from the recent multi-resolution VAE approach [44] within a single model. In our experiments, we show that we were able to generate more realistic scenes compared to multi-resolution approaches, avoiding the problem from independent model training where the finer models inherit mistakes from coarser stages. In addition to that, our method allows for more efficient scene generation, reducing inference time and memory requirements by modeling the problem with a single model. We report in the supplementary material experiments regarding inference time and memory requirements, comparing our method with hierarchical scene generation baselines.

3.2 Semantic Scene Latent Diffusion

Given the training data distribution p and the learned VAE latent distribution q_ϕ, we want to train a diffusion model θ to generate a novel dense latent from Gaussian noise as $\boldsymbol{Z}_\theta = \theta(\mathcal{N}(\mathbf{0}, \boldsymbol{I}))$, such that $\boldsymbol{Z}_\theta \sim q_\phi$. Then, the VAE decoder ψ reconstructs the scene from the generated latent $\psi(\boldsymbol{Z}_\theta) = \mathcal{P}'$, where $\mathcal{P}' \sim p$ is a novel scene reconstructed from the generated latent $\boldsymbol{Z}_\theta$.

3.2.1 Diffusion Training

Denoising diffusion probabilistic models (DDPM) formulate data generation as a stochastic denoising process. A fixed number of T diffusion steps is defined, and the training data is corrupted by iteratively sampling Gaussian noise and adding it to the data over the T steps. Then, the model is trained to predict the noise added at each one of the T steps, iteratively removing the noise until arriving back to the uncorrupted sample. During inference, Gaussian noise is sampled as the corrupted data at step T, and the model is used to denoise it, generating novel samples.

Latent diffusion training uses the trained VAE to encode a sample $\mathcal{P}$ from the target semantic scene data distribution p into $\boldsymbol{Z} = \phi(\mathcal{P})$. Then, the DDPM training process operates directly on the VAE encoder dense latent grid $\boldsymbol{Z}$. At each training iteration, a random step t is uniformly sampled from the T diffusion steps. Gaussian noise $\boldsymbol{\epsilon}$ is sampled, and the corrupted data $\boldsymbol{Z}^t$ at step t is computed given pre-defined noise factors $\beta_1, \ldots, \beta_T$, with $\alpha_t = 1 - \beta_t$, and the cumulative product $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$ as:

$$\boldsymbol{Z}^t = \sqrt{\bar{\alpha}_t} \boldsymbol{Z}^0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, \, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}). \qquad (5)$$

From the corrupted latent $\boldsymbol{Z}^t$, the model θ is trained to predict the added noise $\boldsymbol{\epsilon}$. However, predicting only $\boldsymbol{\epsilon}$ leads to slow convergence since the target distribution is only implicitly learned. Salimans et al. [47] propose predicting $\mathbf{v}$ for faster convergence, defined in terms of $\boldsymbol{\epsilon}$, $\boldsymbol{Z}^0$ and t. Thus, the model θ is optimized to learn the noise and target data distributions, with $\mathbf{v}^t$ computed as:

$$\mathbf{v}^t = \sqrt{\bar{\alpha}_t} \boldsymbol{\epsilon} - \sqrt{1 - \bar{\alpha}_t} \boldsymbol{Z}^0, \qquad (6)$$

from which $\boldsymbol{\epsilon}$ can be computed from $\mathbf{v}^t$ and $\boldsymbol{Z}^t$ as:

$$\boldsymbol{\epsilon} = \sqrt{\bar{\alpha}_t} \mathbf{v}^t + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{Z}^t, \qquad (7)$$

changing the model target without affecting the inference process. Therefore, the model θ is trained to predict $\mathbf{v}_\theta^t = \theta(\boldsymbol{Z}^t, t)$:

$$\mathcal{L}_{\text{diff}} = \lambda_{\text{SNR}(t)} \left\| \mathbf{v}^t - \mathbf{v}_\theta^t \right\|, \qquad (8)$$

explicitly learning both the $\boldsymbol{\epsilon}$ and $\boldsymbol{Z}^0$ data distributions, where $\lambda_{\text{SNR}(t)}$ is the weight w.r.t. the signal-to-noise ratio (SNR) at the sampled diffusion step t, following the Min-SNR-γ weighting strategy [13] to balance the training at the different diffusion steps.

Latent diffusion inference starts from random noise as the corrupted data as $\boldsymbol{Z}^T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. From $\boldsymbol{Z}^T$ the model θ iteractively predicts $\mathbf{v}_\theta^t$ at each step t. From Eq. (7), the predicted noise $\boldsymbol{\epsilon}_\theta^t$ added at step t is computed from the prediction $\mathbf{v}_\theta^t$, iteratively removing noise from $\boldsymbol{Z}^T$ as:

$$\boldsymbol{Z}^{t-1} = \boldsymbol{Z}^t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta^t + \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}} \beta_t \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \quad (9)$$

until arriving at the uncorrupted data sample $\boldsymbol{Z}^0 = \boldsymbol{Z}_\theta \sim q_\phi(\boldsymbol{Z})$ from the VAE latent space. Then, we decode the generated latent $\boldsymbol{Z}_\theta$ with the VAE decoder ψ, generating a novel semantic scene $\mathcal{P}'$ from the target scene data distribution $\psi(\boldsymbol{Z}_\theta) = \mathcal{P}' \sim p$.

Conditioned latent diffusion uses the diffusion model θ to generate samples conditioned to a target input data. In this context, we condition the semantic scene generation to an input LiDAR scan to generate a dense and semantically annotated scene. Following the classifier-free guidance [16], we train the diffusion model θ, conditioning it to a LiDAR point

cloud $\mathcal{C} = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_P\}$ with $\boldsymbol{c}_p \in \mathbb{R}^3$ being the points coordinates. For the conditioned training, the prediction $\mathbf{v}_\theta^t$ in Eq. (8) then becomes $\mathbf{v}_\theta^t = \theta(\boldsymbol{Z}^t, \tilde{\mathcal{C}}, t)$, with $\tilde{\mathcal{C}}$ having a probability ρ of being the null token $\emptyset$ or the LiDAR point cloud $\mathcal{C}$ otherwise, i.e., switch between unconditioned and conditioned generation. We refer to the supplementary material for further details.

4 EXPERIMENTAL EVALUATION

Datasets. To generate our training data dense scenes, we use the SemanticKITTI dataset [1], [11], a 3D LiDAR semantic segmentation dataset in the context of autonomous driving. The dataset provides sequences of 3D LiDAR scans with semantic labels. To generate the dense scenes, we use an off-the-shelf SLAM [41] method to compute scan poses throughout the whole sequence. With the labeled scans and the corresponding poses, we aggregate the scan sequences with the labels, generating a voxel map of the scene with 0.1 m resolution. We also use the labels to remove moving objects from the scene to avoid aggregating them and generating large artifacts from the moving objects trajectories. During training, we query a pose from one of the individual LiDAR scans and crop the corresponding dense scene from the map at the queried pose within a fixed range at $[x, y, z]$ between $[-25.6, -25.6, -2.2]$m and $[25.6, 25.6, 4.2]$m. For the semantic segmentation experiments, we use the same dataset, sequence 00 from KITTI-360 dataset [24], and collected our own data with an Ouster LiDAR OS-1 with 128 beams. We use LiDAR scans from KITTI-360 and our collected data as conditioning to generate the semantic dense scene from it, and to later use it as further labeled data to train a semantic segmentation model.

Training. As our VAE model, we use a 3D Sparse UNet [7]. We train our VAE model for 50 epochs, using the Adam optimizer [20] with a learning rate of 10^{-4}, multiplying it by 0.9 every 5 epochs. For the first 25 epochs, we use the weighted cross-entropy loss with the class weights provided by the SemanticKITTI dataset [1], [11]. For the last 25 epochs, we use the unweighted cross-entropy. For the losses weights we use $\lambda_1 = 1.0$, $\lambda_2 = 1.0$, $\lambda_3 = 2.0$, $\lambda_4 = 3.0$ for the individual pruning layers losses in Eq. (1), with $\lambda_{\text{prune}} = 1.0$, $\lambda_{\text{sem}} = 1.0$ and $\lambda_{\text{latent}} = 0.002$ in Eq. (4). We trained the VAE with 6 NVIDIA A40 GPUs. We use the same training scheme for the VAE refinement training.

For the DDPM, we use a 3D UNet model to learn the dense latent $\boldsymbol{Z}$ and we train the model for 150 epochs, using the AdamW optimizer [27] with a learning rate of $2 \cdot 10^{-4}$, multiplying it by 0.8 every 50 epochs. The DDPM is trained with $\beta_T = 0.015$ and $\beta_0 = 10^{-4}$, with $T = 1000$, linearly interpolating it to get the noise factors $\beta_1, \ldots, \beta_{T-1}$ in between. For the Min-SNR-γ DDPM loss weighting, we use $\gamma = 5$. We trained the DDPM with 8 NVIDIA A40 GPUs. For the DDPM conditioned generation, we set the classifier-free guidance probability $\rho = 0.1$ and conditioning weight to 2.0. In the supplementary material, we provide further details regarding the VAE and DDPM models.

Baselines. All baselines and our method are trained on the same set of dense scans. We compare our method with three baselines: SemCity [23], PDD [26] and XCube [44]. All baselines are trained using their official implementations and default configurations. SemCity [23] and PDD [26] are limited by the voxel resolution, with a maximum resolution of 0.2 m. For those baselines we downsample the training data to 0.2 m resolution. XCube [44] and our method can generate scenes up to 0.1 m voxel

	Eval. resolution [m]	MMD $\downarrow$
PDD [26]	0.2	0.207
SemCity [23]	0.2	0.239
XCube [44]	0.2	0.160
Ours	0.2	**0.146**
XCube [44]	0.1	0.101
Ours	0.1	**0.073**

TABLE 1. Maximum mean discrepancy (MMD) between real and synthetic data.

	Eval. resolution [m]	mIoU [%] $\uparrow$
Validation set	0.2	55.59
PDD [26]	0.2	36.53
SemCity [23]	0.2	40.12
XCube [44]	0.2	40.02
Ours	0.2	**50.83**
Validation set	0.1	61.08
XCube [44]	0.1	27.24
Ours	0.1	**53.09**

TABLE 2. mIoU over synthetic data with semantic segmentation model, trained on real data.

resolution. Therefore, we compare all methods at 0.2 m resolution, downsampling XCube [44] and our generated scenes, and compare our method with XCube [44] at 0.1 m resolution.

Semantic segmentation. For the semantic segmentation training, we used the pipeline from our previous work [37] to train a 3D Sparse UNet. We train the model for 15 epochs in all experiments with SGD optimizer with a learning rate of 0.24 with a cosine annealing learning rate scheduler. For the experiments in Sec. 4.1, we generate $8,000$ samples from each method and compute the metrics over those samples. For Sec. 4.2 and Sec. 4.3, we generate the same amount of samples as SemanticKITTI dataset [1], [11], i.e., $19,130$ samples, to evaluate the performance with different subsets between real and generated samples. Also, since we remove the moving objects during the creation of the sequences maps, the occurrence of some classes decreases drastically, impacting the IoU over those classes. Therefore, in the mIoU computation, we ignore the classes that are mainly moving in the dataset, i.e., *bicycle*, *motorcycle*, *person*, *bicyclist* and *motorcyclist*.

4.1 Generated Scene Realism

As the first experiment, we want to assess the quality of the generated scenes by evaluating how close they are to the real data. To do so, we first train a semantic segmentation model following the training protocol used in our previous work [37] with the real semantic scenes created through the scans aggregation from SemanticKITTI dataset [1], [11]. Then, we input both real and synthetic scenes to the trained semantic segmentation network to compute their latent features, evaluating them with the maximum mean discrepancy (MMD). Besides, we follow the evaluation done by Liu et al. [26], using the trained semantic segmentation model to compute the mIoU over the generated scenes comparing it with the SemanticKITTI [1], [11] validation set results. Given that the model was trained on real data, we expect higher mIoU over the generated scenes if they are similar to the real data in this evaluation. We do both evaluations at two resolutions, 0.1 m, and
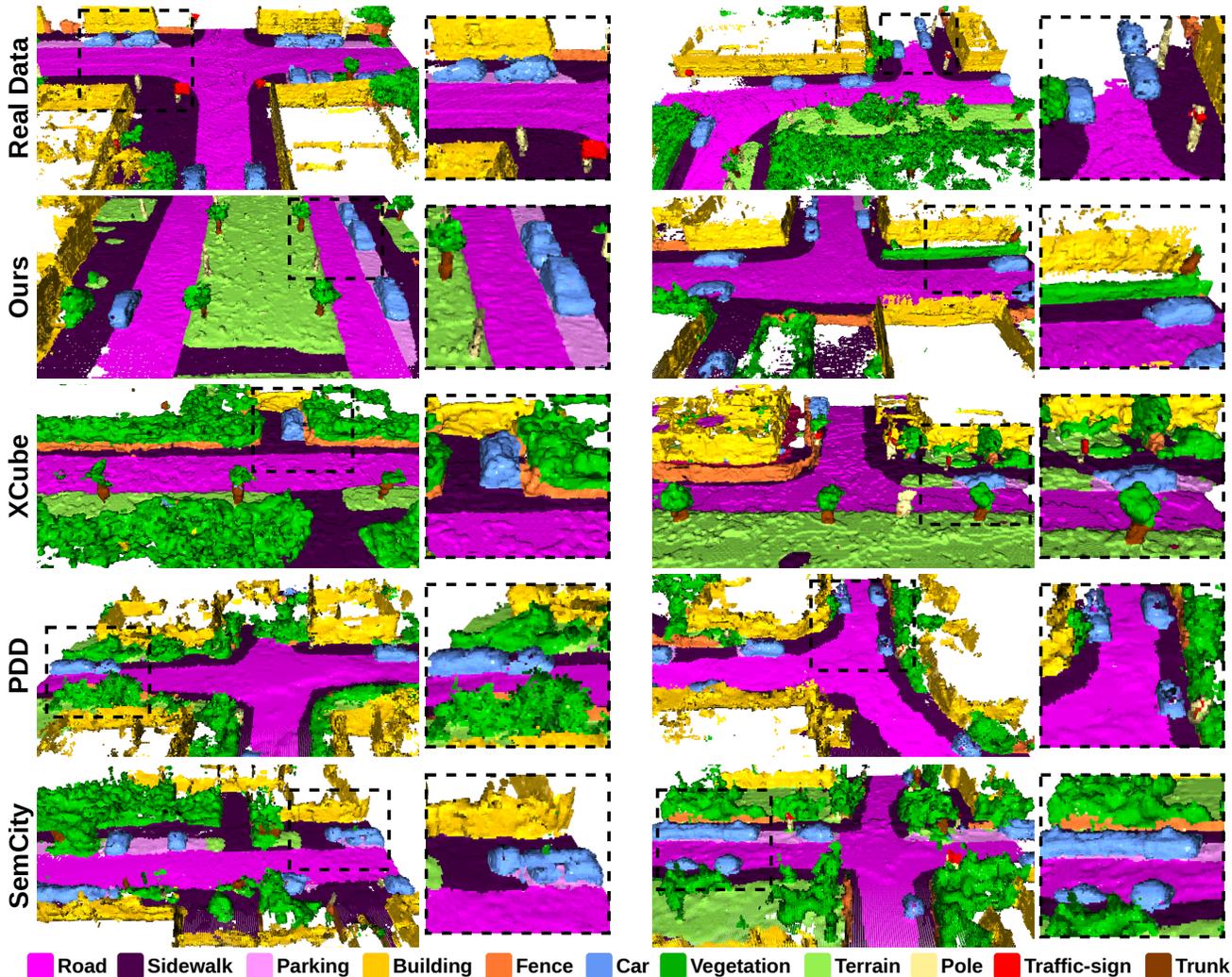
Fig. 3. Comparison of real and unconditioned generated scenes from different methods. PDD and SemCity scenes are limited to $0.2\,\mathrm{m}$ resolution. The baselines scenes present rounder and too smooth shapes. Our method can generate more fine-grained details, closer to real data.
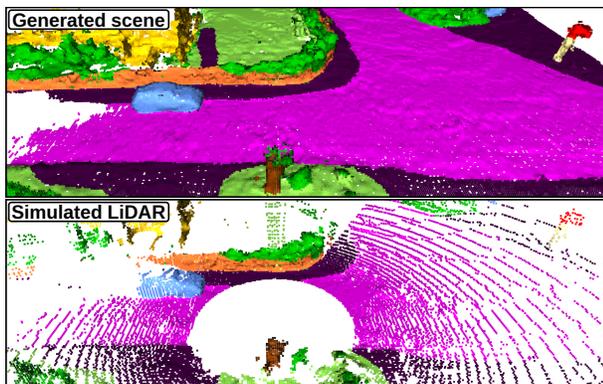

Fig. 4. Simulated LiDAR point clouds from a dense generated scene.

$0.2\,\mathrm{m}$, since the PDD [26] and SemCity [23] can only generate scenes up to $0.2\,\mathrm{m}$ resolution.

Tab. 1 shows the MMD metric between the generated and real scenes from SemanticKITTI dataset [1], [11]. At $0.2\,\mathrm{m}$ resolution our approach and XCube [44] surpass previous methods, showing the advantages of using latent diffusion and operating directly on the 3D data. At $0.1\,\mathrm{m}$ resolution, our method and XCube [44] performance improves, still our method achieves the best performance. Tab. 2 depicts the results when computing the mIoU

over the generated data with the semantic segmentation model trained on SemanticKITTI dataset [1], [11]. At lower resolution the baselines' performance are similar while our method outperforms them, achieving a mIoU close to the real data performance. At higher resolution, XCube [44] performance drops, while our method performance increases compared to the performance at $0.2\,\mathrm{m}$ resolution, showing that our method is able to generate scenes which are closer to the real data compared to the baselines.

Fig. 3 shows scene examples from the real data compared with random scenes generated by our method and the baselines, where SemCity [23] and PDD [26] generation is limited to $0.2\,\mathrm{m}$ resolution. In those examples, all baselines are able to learn the overall structure in the scene and generate reasonable scenarios. However, while the baselines generate smoother and rounder shapes, our approach is able to generated more fine-grained details closer to the real data. By avoiding intermediary representations and decoupled trained VAEs, our model can generate scenes with more details. These results show that our method generates closer-to-real scenes compared to the baselines. In the supplementary material we present additional experiments on the Waymo dataset [50], and also experiments regarding efficiency comparing our method with the hierarchical-models baseline XCube [44].

| | mIoU [%] ↑ | | | | |
| | Real/Synth. [%] | | | | |
Synth. source	10/90	25/75	50/50	90/10	100/0
Real only	45.58	50.48	53.73	55.48	**55.59**
XCube [44]	49.28	52.12	55.04	55.75	**55.59**
Ours	**55.54**	**57.44**	**57.15**	**56.86**	**55.59**

TABLE 3. Semantic segmentation performance for varying percentages of real data available to train the network, and complementing it with synthetic data to achieve 100% size of the full training set trained with the LiDAR scans simulated from the densely generated scenes.

| | mIoU [%] ↑ | | | | |
| | Real/Synth. [%] | | | | |
Synth. source	10/90	25/75	50/50	90/10	100/0
Real only	56.22	59.40	60.60	60.73	**61.08**
XCube [44]	53.75	58.79	60.20	60.98	**61.08**
Ours	**61.73**	**63.91**	**64.06**	**63.28**	**61.08**

TABLE 4. Semantic segmentation performance for varying percentages of real data available to train the network, and complementing it with synthetic data to achieve 100% size of the full training set we train and evaluate with the densely generated point clouds.
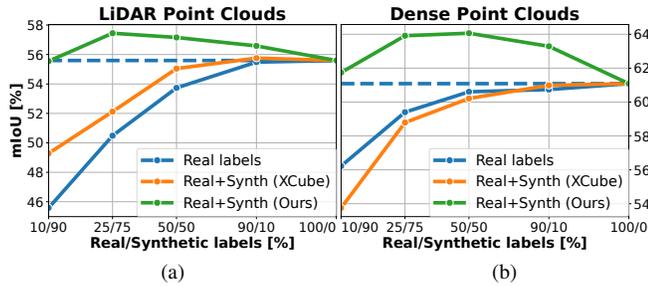

Fig. 5. Semantic segmentation model performance trained with the different percentages of real data complemented with synthetic data from our model and XCube [44]. (a) Model trained with LiDAR scans simulated from the dense scenes. (b) Model trained with dense scenes.

4.2 Generated Labels as Training Data

In this section, we aim at assessing the usability of the generated scenes as training data for downstream perception tasks. We define subsets of training data varying the percentages of real data from SemanticKITTI dataset [1], [11] available for training. For each subset, we complement the real data with synthetic data such that the total amount of samples is the same as the full real data training set, i.e., 100% of data. Then, we train the semantic segmentation network with both, the subsets of real data without any synthetic data, and the subsets complemented with generated data. We compare the results with generated data from our method and XCube [44], the only two methods able to generate scenes of 0.1 m resolution, computing the mIoU over the SemanticKITTI [1], [11] validation set. Besides evaluating it over the dense scenes, we also aim at evaluating the impact using sparse LiDAR data. Therefore, we simulate a LiDAR point cloud from each dense scene from the real and generated scenes, as shown in Fig. 4, replicating the experiments also with the simulated LiDAR scans.

Tab. 3 shows the results of the semantic segmentation network trained with the simulated LiDAR scans with varying percentages of real data complemented with generated data. As shown, the use of synthetic data improves the network performance whenever using scenes generated by XCube or by our method, converging to the same performance as the amount of real data gets closer to 100%. Still, the network trained with our synthetic data overall outperforms the network trained with the scenes generated by this baseline, as seen also in Fig. 5a. For dense point clouds in Tab. 4, the network trained with synthetic data generated by XCube achieves worse performance than the network trained only with real data. For such dense point clouds, the differences between the real and generated scene distributions have a higher impact on the network performance due to the amount of points in each generated sample. These results highlight the gap between the scenes generated by XCube and the real scenes. In contrast,

when training with a subset of real data and with our generated scenes, the model achieved better performance than the network trained only with the full real training set. At first, this may sound counterintuitive due to the results presented previously in Tab. 2, where our generated scenes presented a performance gap compared to the real data even though performing better than the baselines. Still, such improvement can be explained by the variability added to the training set when using generated data. The real scenes are collected sequentially, with few changes between consecutive point clouds, while the randomly generated scenes will be different from each other, adding variability to the training set. This increase in variability improves the network performance as long as enough real data is also seen by the model during training, as shown in Fig. 5b. These results show that our method can generate scenes better suited to be used for downstream tasks compared to the baseline, and also show the potential of using generated samples as training data.

Even though the experiments reported in the main paper are performed only on data from SemanticKITTI [1], [11], we argue that our approach is not limited to this dataset, as the annotated data generation is done over dense scenes, not limiting it to sensor- or dataset-specific characteristics. To support this argument, we refer to the supplementary material where we report experiments and ablations using the Waymo dataset.

4.3 Synthetic Training Set Extension

In this section, we extend the experiment done in Sec. 4.2. Given the real data training set, we would not expect to reduce the amount of real data available but rather enlarge this dataset to achieve even better performance in a downstream task. Therefore, in this experiment, we use the scenes generated by our method to enlarge the real data training set, evaluating the impact on the semantic segmentation model performance. We train the model with all the available real training data while progressively adding synthetic data and evaluating the performance on the SemanticKITTI [1], [11] validation set. Given the results from previous sections, in this experiment we only evaluate the model using our generated data. As in Sec. 4.2, we perform the experiment twice, over the dense and the simulated LiDAR point clouds.

Tab. 5 shows the mIoU from the models trained with the full real training set and adding different percentages of the synthetic scenes as additional data for both dense scenes and the LiDAR simulated point clouds. As shown, by using the synthetic scenes generated by our model, we were able to improve the semantic segmentation model for both dense and LiDAR point clouds. As the amount of synthetic data increases, the model performance also increases, saturating at 75% additional synthetic labels, as seen in Fig. 6. When training with 100% of the generated data,

Additional synth. data [%]	mIoU [%] ↑	
	Dense	LiDAR
0	61.08	55.59
10	62.42	55.96
25	62.47	56.70
50	62.97	56.88
75	**64.14**	**57.77**
100	64.07	56.53

TABLE 5. Semantic segmentation results when training the model with the full real data training set and adding additional synthetic data, both with dense point clouds and LiDAR scans simulated from the dense scenes.
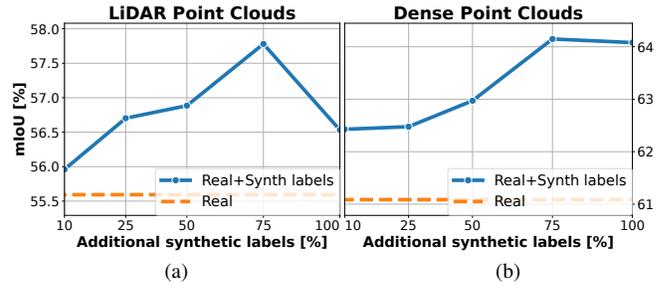

Fig. 6. Semantic segmentation model performance when trained with the full real training set and adding different amount of additional synthetic data from our model. (a) Model trained with LiDAR scans simulated from the dense scenes. (b) Model trained with dense scenes.

the model performance starts to degrade, although still achieving better performance than the model trained only with real data. This can be explained by the fact that, even though better than previous state-of-the-methods, our generated samples still have differences compared to the real data, as shown in Tab. 2. Therefore, by increasing the amount of synthetic data, the semantic segmentation model starts to learn more from the synthetic data distribution, and consequently decreasing its performance on the real data.

Tab. 6 and Tab. 7 show the class IoU when training only with the full real training set compared adding 75% synthetic data to the real training set, which achieved the best performance. In both cases, for LiDAR scans and for dense point clouds, the training with the synthetic data improves the performance in almost all the classes. These results show the potential of synthetic data in helping to improve the performance in the target downstream task. Still, due to a distribution gap between real and synthetic data discussed in Sec. 4.1, there is space for improvement.

4.4 Conditional DDPM as Data Annotator

Besides extending the training set with randomly generated samples as evaluated in Sec. 4.3, one useful application of DDPMs is to use conditional generation to annotate data. Target scenes could be recorded with a LiDAR sensor and the annotated scene could be generated conditioned to the collected point clouds. In this case, the exhaustive manual data annotation can be replaced with a simpler data curation, selecting only the most realistic scenes generated by the DDPM. Therefore, in this section, we train the DDPM with the dense SemanticKITTI [1], [11] scenes conditioned to their corresponding LiDAR scans from the dataset. With this conditional DDPM, we generate novel scenes conditioned to sequence 00 from KITTI-360 [24] and to our data collected with an Ouster LiDAR OS-1 with 128 beams. Examples of conditional generated scenes are shown in Fig. 7. Next, we curate those conditional generated scenes by manually selecting scenes that look more realistic until arriving at a total of 4, 782 point clouds, corresponding to approximately 25% of the size of the SemanticKITTI [1], [11] training set. In Tab. 8, we provide information regarding the amount of scenes selected during the curation process. Finally, we train the semantic segmentation model with the SemanticKITTI [1], [11] training set together with the curated generated scenes to evaluate the use of conditioned DDPM as data annotator.

Tab. 9 shows the performance of the semantic segmentation network trained only with real data and with the 25% additional synthetic data from unconditional and conditional curated generated scenes, also comparing with the best-performing model from

Tab. 5 trained with additional 75% synthetic data. In this table, we notice that the network trained with the curated data achieves better performance than both uncurated training sets. Even though using fewer samples, the model trained with only 25% curated generated data achieves better performance than the model trained with additional 75% randomly generated data. Those results show that the data curation process further improves the impact of the generated scenes on the network performance. This improvement shows the potential of using conditioned DDPMs as data annotator, where the exhaustive manual data annotation could be replaced by a simpler data curation process. Such an application could help to generate training data from specific scenarios, aiding the data annotation scalability.

4.5 Generated and Real Data Gaps

In previous sections, we performed experiments to investigate the potential of generated data to augment the real training set, improving the performance of the models in the target downstream task. In this section, we aim at evaluating the opposite, comparing the data generated by our method with real data to identify the gaps between both. For that, we first compute the class distributions from the real and generated data. Then, we use the semantic segmentation network trained on real data to compute the IoU for each class in the generated and real scenes to identify the current gaps between synthetic and real data.

Fig. 8 shows the class distribution of SemanticKITTI [1], [11] training set and of the scenes generated by our method which was trained with SemanticKITTI [1], [11]. As can be seen, the generated scenes class distribution follows the overall distribution present in the dataset used to train the DDPM. Even though the generated scenes may increase the variability of the data compared to the real dataset, as discussed in Sec. 4.2, the amount of data per class remains close to the training data distribution. Tab. 10 shows the IoU per class of the network trained with SemanticKITTI [1], [11] training set and with the synthetic scenes generated by our approach evaluated with the validation set, as well as the difference between both results. When comparing both Fig. 8 and Tab. 10, we notice that the classes with high frequency in the synthetic scenes in Fig. 8, e.g., vegetation, road, sidewalk, fence, are also the classes that achieve higher IoU in Tab. 10. For those classes, the performance is comparable or even better compared to the evaluation on the real data validation set. At the same time, the less occurring classes, e.g., truck, other-vehicle (OV), pole, traffic-sign, are the classes that perform worse in Tab. 10, and for which there is a bigger performance gap compared with the evaluation on the real validation set. The main gap between the generated and

	IoU [%] ↑											
	car	truck	OV	road	park.	sidewalk	build.	fence	veg.	trunk	terrain	pole	sign
Real only	94.25	48.60	23.99	90.36	33.34	67.83	86.74	37.29	85.56	**44.29**	65.77	**56.27**	**43.84**
Real + 75% Synth.	**94.67**	**59.81**	**28.45**	**91.28**	**42.02**	**70.14**	**87.59**	**42.36**	**86.14**	40.57	**67.38**	**56.27**	42.17

TABLE 6. LiDAR simulated data class-wise IoU evaluated on the real data validation set comparing the network trained only with the full real training set with the training with the full training set with additional 75% synthetic data generated with our method. OV refers to other-vehicle.

	IoU [%] ↑											
	car	truck	OV	road	park.	sidewalk	build.	fence	veg.	trunk	terrain	pole	sign
Real only	93.77	42.24	**49.00**	92.16	46.52	69.24	89.28	41.08	88.07	51.88	65.81	61.12	**58.36**
Real + 75% Synth.	**94.01**	**70.49**	48.27	**92.59**	**50.04**	**73.11**	**90.00**	**44.30**	**88.29**	**52.26**	**66.96**	**62.40**	57.51

TABLE 7. Dense scenes class-wise IoU evaluated on the real data validation set comparing the network trained only with the full real training set with the training with the full training set with additional 75% synthetic data generated with our method. OV refers to other-vehicle.
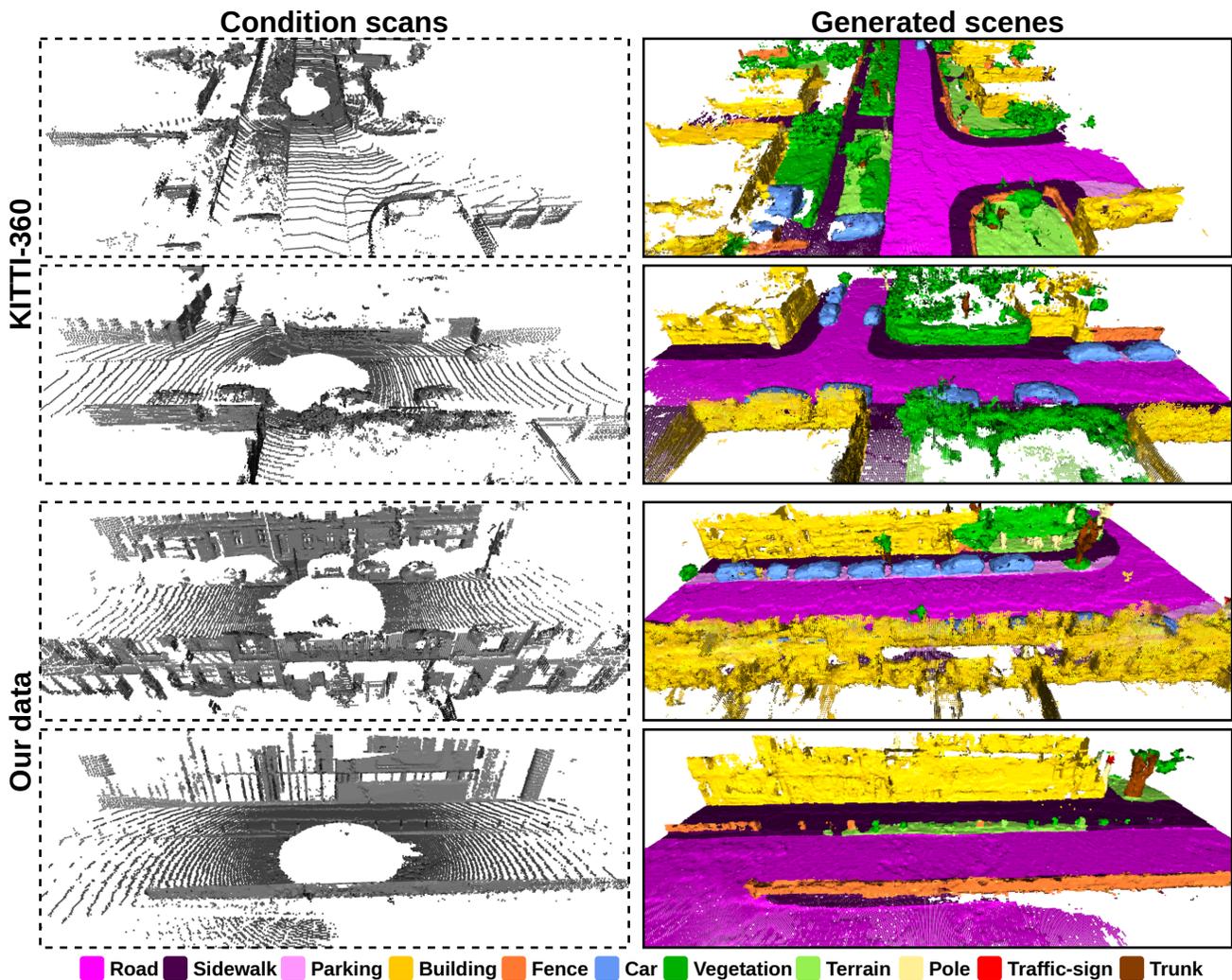
Fig. 7. Generated scenes from our approach conditioned to KITTI-360 [24] scans and to our data collected with an Ouster LiDAR.

Source	#point clouds	#curated scenes
KITTI-360 [24]	11,518	3,939
Our data	5,276	843
Total	16,794	4,782

TABLE 8. Number of point clouds in sequence 00 from KITTI-360 and in our collected dataset and number of conditional generated scenes selected during the curation process.

Data source		mIoU [%] ↑	
Real [%]	Synth. [%]	Dense	LiDAR
100	0	61.08	55.59
100	25	62.47	56.70
100	75	64.14	57.77
100	$25^\dagger$	**64.47**	**58.11**

TABLE 9. Semantic segmentation trained with all the real SemanticKITTI training set together with unconditional randomly generated scenes and with curated conditional generated scenes. † refers to the curated conditional generated scenes.

real data distributions therefore comes from those less occurring classes. This analysis shows a direct relation between the amount of data samples per class and its generation quality. Hence, the DDPM training would have to be balanced to account for the semantic classes imbalance present in scene-scale data to bridge this distribution gap in those underrepresented classes. Even though we account for this class imbalance in the VAE training, during the DDPM training this is not straightforward since the model is trained over the VAE latent, where there is no direct map between the semantic classes and the latent features. Therefore, this class imbalance is not addressed in this article. Future works should address this class imbalance in the DDPM training, which would improve the generation quality of underrepresented classes, increasing the generated scene quality.

5 CONCLUSION

In this article, we propose a denoising diffusion probabilistic model able to achieve state-of-the-art semantic scene generation without relying on image projections or multi-resolution decoupled autoencoders trainings. We follow recent latent diffusion methods and mask-based 3D segmentation approaches and train the model to learn the scene coarse-to-fine data nature within a single VAE decoder upsampling layers. Our proposed method achieves closer-to-real scene generation compared to previous state-of-the-art methods by avoiding intermediary representations and learning the scene data distribution at the original resolution. Besides, we performed several experiments to assess the use of unconditional and conditional generated scenes as training data for real-world perception tasks. In our experiments, we showed that by using the semantic scenes generated by our method together with the real data, we improve the model performance in the semantic segmentation task. Our results show the potential of using DDPMs to enlarge the training set from unconditional random samples and LiDAR conditioned generated scenes. In our experiments, we also evaluated the relation between the semantic classes distribution between real and generated scenes and the generation quality of each class. We show that the class imbalance inherent from the dataset directly impacts the class generation quality. From
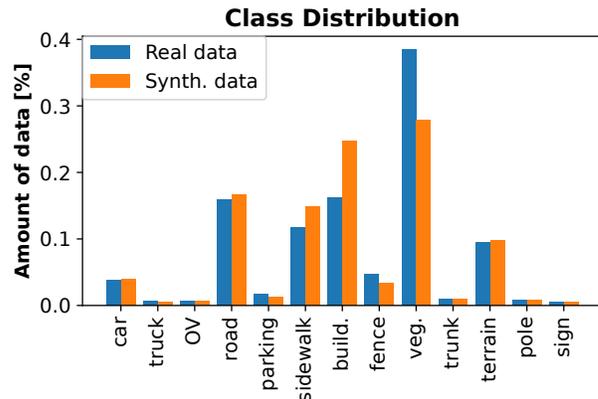

Fig. 8. Class distribution over the real dataset compared with the synthetic data class distribution. (OV refers to other-vehicle class).

our analysis, addressing this class imbalance during the DDPM training could help bridge the gap between real and generated scenes distributions. More realistic scenes achieved by addressing this class imbalance can enable the large-scale use of generated scenes as training data, extending the already available labeled data with random scenes and scenes conditioned to target scenarios, improving the performance on real-world perception tasks.

Limitations. Despite the promising results, our training data generation is still affected by the removal of dynamic objects in the scene, leading to the removal of examples from potentially underrepresented classes, such as pedestrians, motorcyclists, and bicyclists. Besides, our evaluation mainly relies on the generation of annotated data from SemanticKITTI dataset. However, we expect that these findings are not limited to this dataset, but can also be extended to other datasets as suggested by the experiments on Waymo dataset reported in the supplementary material.

ACKNOWLEDGMENTS

REFERENCES

[1] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences," in *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019. 1, 5, 6, 7, 8

[2] A. Blattmann, R. Rombach, K. Oktay, J. Müller, and B. Ommer, "Retrieval-augmented diffusion models," in *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2022. 2

[3] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A Multimodal Dataset for Autonomous Driving," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[4] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021. 2

[5] S. Chen, H. Xu, R. Li, G. Liu, C. Fu, and S. Liu, "SIRA-PCR: Sim-to-Real Adaptation for 3D Point Cloud Registration," in *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2023. 1, 2

	IoU [%] ↑												
	car	truck	OV	road	park.	sidewalk	build.	fence	veg.	trunk	terrain	pole	sign
Validation scenes	93.77	42.24	49.00	92.16	46.52	69.24	89.28	41.08	88.07	51.88	65.81	61.12	58.36
Generated scenes	92.29	10.42	24.10	92.48	33.74	77.34	89.75	47.73	81.81	42.99	61.72	43.55	28.79
Performance gap	-1.48	-31.82	-24.90	+0.32	-12.78	+1.64	+0.47	+6.65	-6.26	-8.89	-8.89	-17.57	-29.57

TABLE 10. Class-wise IoU evaluated on real data validation set and synthetic data generated by our method with semantic segmentation model trained on real data, and the difference in performance between both real and synthetic data evaluation.

[6] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-attention mask transformer for universal image segmentation," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[7] C. Choy, J. Gwak, and S. Savarese, "4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 5

[8] M. Cordts, S. M. Omran, Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[9] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," in *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2021. 1, 2, 3

[10] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *Proc. of the Conf. on Robot Learning (CoRL)*, 2017. 1, 2

[11] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012. 1, 5, 6, 7, 8

[12] S. Geng, C.-Y. Hsieh, V. Ramanujan, M. Wallingford, C.-L. Li, P. W. Koh, and R. Krishna, "The unmet promise of synthetic training images: Using retrieved real images performs better," *arXiv preprint*, vol. arxiv:2406.05184, 2024. 1, 2

[13] T. Hang, S. Gu, C. Li, J. Bao, D. Chen, H. Hu, X. Geng, and B. Guo, "Efficient diffusion training via min-snr weighting strategy," in *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2023. 4

[14] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum Contrast for Unsupervised Visual Representation Learning," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[15] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2020. 1, 2

[16] J. Ho and T. Salimans, "Classifier-free diffusion guidance," in *Proc. of the NeurIPS Workshop on Deep Generative Models and Downstream Applications*, 2021. 2, 4

[17] M. Huang, Z. Mao, Z. Chen, and Y. Zhang, "Towards Accurate Image Coding: Improved Autoregressive Image Generation With Dynamic Vector Quantization," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[18] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," in *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2022. 2

[19] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and Improving the Image Quality of StyleGAN," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[20] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2015. 5

[21] D. Lee, C. Kim, S. Kim, M. Cho, and W. Han, "Autoregressive Image Generation Using Residual Quantization," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[22] J. Lee, W. Im, S. Lee, and S.-E. Yoon, "Diffusion probabilistic models for scene-scale 3d categorical data," *arXiv preprint*, vol. arXiv:2301.00527, 2023. 1, 2

[23] J. Lee, S. Lee, C. Jo, W. Im, J. Seon, and S.-E. Yoon, "Semcity: Semantic scene generation with triplane diffusion," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 2, 5, 6

[24] Y. Liao, J. Xie, and A. Geiger, "KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d," *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 45, no. 3, pp. 3292–3310, 2022. 5, 8, 9, 10

[25] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2014. 1

[26] Y. Liu, X. Li, X. Li, L. Qi, C. Li, and M.-H. Yang, "Pyramid diffusion for fine 3d large scene generation," in *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2024. 1, 2, 5, 6

[27] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2019. 5

[28] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu, "DPM-solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps," in *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2022. 2

[29] S. Luo and W. Hu, "Diffusion probabilistic models for 3d point cloud generation," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1

[30] Z. Lyu, Z. Kong, X. XU, L. Pan, and D. Lin, "A conditional point diffusion-refinement paradigm for 3d point cloud completion," in *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2022. 2

[31] R. Marcuzzi, L. Nunes, L. Wiesmann, J. Behley, and C. Stachniss, "Mask-Based Panoptic LiDAR Segmentation for Autonomous Driving," *IEEE Robotics and Automation Letters (RA-L)*, vol. 8, no. 2, pp. 1141–1148, 2023. 3

[32] R. Marcuzzi, L. Nunes, L. Wiesmann, E. Marks, J. Behley, and C. Stachniss, "Mask4D: End-to-End Mask-Based 4D Panoptic Segmentation for LiDAR Sequences," *IEEE Robotics and Automation Letters (RA-L)*, vol. 8, no. 11, pp. 7487–7494, 2023. 3

[33] C. Meng, R. Rombach, R. Gao, D. Kingma, S. Ermon, J. Ho, and T. Salimans, "On distillation of guided diffusion models," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[34] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet++: Fast and Accurate LiDAR Semantic Segmentation," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019. 1

[35] S. Mo, F. Mu, K. H. Lin, Y. Liu, B. Guan, Y. Li, and B. Zhou, "FreeControl: Training-Free Spatial Control of Any Text-to-Image Diffusion Model with Any Condition," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[36] K. Nakashima and R. Kurazume, "Lidar data synthesis with denoising diffusion probabilistic models," *arXiv preprint*, vol. arXiV:2309.09256, 2023. 2

[37] L. Nunes, R. Marcuzzi, X. Chen, J. Behley, and C. Stachniss, "SegContrast: 3D Point Cloud Feature Representation Learning through Self-supervised Segment Discrimination," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 2, pp. 2116–2123, 2022. 2, 5

[38] L. Nunes, R. Marcuzzi, B. Mersch, J. Behley, and C. Stachniss, "Scaling Diffusion Models to Real-World 3D LiDAR Scene Completion," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2, 4

[39] L. Nunes, L. Wiesmann, R. Marcuzzi, X. Chen, J. Behley, and C. Stachniss, "Temporal Consistent 3D LiDAR Representation Learning for Semantic Perception in Autonomous Driving," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[40] C. Oeldorf and G. Spanakis, "LoGANv2: Conditional Style-Based Logo Generation with Generative Adversarial Networks," in *Proc. of the IEEE Intl. Conf. on Machine Learning And Applications (ICMLA)*, 2019. 2

[41] Y. Pan, X. Zhong, L. Wiesmann, T. Posewsky, J. Behley, and C. Stachniss, "PIN-SLAM: LiDAR SLAM Using a Point-Based Implicit Neural Representation for Achieving Global Map Consistency," *IEEE Trans. on Robotics (TRO)*, vol. 40, pp. 4045–4064, 2024. 5

[42] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2023. 1

[43] H. Ran, V. Guizilini, and Y. Wang, "Towards realistic scene generation with lidar diffusion models," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[44] X. Ren, J. Huang, X. Zeng, K. Museth, S. Fidler, and F. Williams, "Xcube: Large-scale 3d generative modeling using sparse voxel hierar-

chies," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 2, 3, 4, 5, 6, 7

[45] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-Resolution Image Synthesis With Latent Diffusion Models," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 3

[46] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2

[47] T. Salimans and J. Ho, "Progressive distillation for fast sampling of diffusion models," in *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2022. 2, 3, 4

[48] M. B. Sariyildiz, K. Alahari, D. Larlus, and Y. Kalantidis, "Fake it till you make it: Learning transferable representations from synthetic imagenet clones," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 2

[49] Q. Song and W. Bai, "Map imagination like blind humans: Group diffusion model for robotic map generation," *arXiv preprint*, vol. arXiv:2412.16908, 2024. 2

[50] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in Perception for Autonomous Driving: Waymo Open Dataset," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 6

[51] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han, "Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution," in *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020. 1

[52] Y. Tian, L. Fan, K. Chen, D. Katabi, D. Krishnan, and P. Isola, "Learning Vision from Models Rivals Learning Vision from Data," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 2

[53] Y. Tian, L. Fan, P. Isola, H. Chang, and D. Krishnan, "Stablerep: Synthetic images from text-to-image models make strong visual representation learners," in *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2021. 1, 2

[54] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Proc. of the Conf. Neural Information Processing Systems (NIPS)*, 2017, p. 6309–6318. 2

[55] J. Wilson, J. Song, Y. Fu, A. Zhang, A. Capodieci, P. Jayakumar, K. Barton, and M. Ghaffari, "Motionsc: Data set and network for real-time semantic mapping in dynamic environments," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 3, pp. 8439–8446, 2022. 1, 2

[56] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, "SqueezeSegV2: Improved Model Structure and Unsupervised Domain Adaptation for Road-Object Segmentation from a LiDAR Point Cloud," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019. 1, 2

[57] J. Xu, X. Wang, W. Cheng, Y.-P. Cao, Y. Shan, X. Qie, and S. Gao, "Dream3D: Zero-Shot Text-to-3D Synthesis Using 3D Shape Prior and Text-to-Image Diffusion Models," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 2

[58] X. Zeng, A. Vahdat, F. Williams, Z. Gojcic, O. Litany, S. Fidler, and K. Kreis, "Lion: Latent point diffusion models for 3d shape generation," in *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2022. 1, 2

[59] B. Zhang, S. Gu, B. Zhang, J. Bao, D. Chen, F. Wen, Y. Wang, and B. Guo, "StyleSwin: Transformer-Based GAN for High-Resolution Image Generation," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[60] L. Zhang, A. Rao, and M. Agrawala, "Adding conditional control to text-to-image diffusion models," in *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2023. 2

[61] M. Zhang, W. Peng, G. Ding, C. Lei, C. Ji, and Q. Hao, "CTS Sim-To-Real Unsupervised Domain Adaptation on 3D Detection," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2024. 1, 2

[62] S. Zhang, A. Zhao, L. Yang, Z. Li, C. Meng, H. Xu, T. Chen, A. Wei, P. P. GU, and L. Sun, "Distilling diffusion models to efficient 3d lidar scene completion," *arXiv preprint*, vol. arXiv:2412.03515, 2024. 2

[63] L. Zhou, Y. Du, and J. Wu, "3D Shape Generation and Completion Through Point-Voxel Diffusion," in *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021. 2

[64] Y. Zhou, B. Liu, Y. Zhu, X. Yang, C. Chen, and J. Xu, "Shifted Diffusion for Text-to-Image Generation," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[65] X. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, and D. Lin, "Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR Segmentation," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1

[66] V. Zyrianov, X. Zhu, and S. Wang, "Learning to generate realistic lidar point clouds," in *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2022. 2

Lucas Nunes is a PhD student at the Photogrammetry and Robotics Lab at the University of Bonn since 2020. He obtained his MSc degree in Artificial Intelligence and Robotics from University of São Paulo in 2020, with a thesis on depth estimation of occluded regions. His area of interest lies in the area of perception for autonomous vehicles, representation learning for 3D LiDAR data, generative models for 3D scene-scale data.

Rodrigo Marcuzzi is a PhD student at Photogrammetry and Robotics Lab at the University of Bonn since 2020. He obtained his Electrical Engineering Degree from National University of Rosario in 2019. During his studies he spent six months at IMT-Atlantique, France as an internship student. His area of interest lies in the area of 4D panoptic segmentation for autonomous vehicles, semantic occupancy prediction for autonomous vehicles.

Jens Behley received his Ph.D. in computer science in 2014 from the Dept. of Computer Science at the University of Bonn, Germany. Since 2016, he is a postdoctoral researcher at the Photogrammetry & Robotics Lab at the University of Bonn, Germany. He completed his habilitation at the University of Bonn in 2023. His area of interest lies in the area of perception for autonomous vehicles, deep learning for semantic interpretation, and LiDAR-based SLAM.

Cyrill Stachniss is a full professor at the University of Bonn, Germany, and with Lamarr Institute for Machine Learning and AI, Germany. He is the Spokesperson of the DFG Cluster of Excellence PhenoRob at the University of Bonn. His research focuses on probabilistic techniques and learning approaches for mobile robotics, perception, and navigation. Main application areas of his research are agricultural and service robotics and self-driving cars.