Scaling Diffusion Models to Real-World 3D LiDAR Scene Completion

Lucas Nunes, Rodrigo Marcuzzi, Benedikt Mersch, Jens Behley, Cyrill Stachniss

1Center for Robotics, University of Bonn  2Lamarr Institute for Machine Learning and Artificial Intelligence
{firstname.lastname}@igg.uni-bonn.de

Figure 1. Starting from a single input scan \( P \), we add Gaussian noise to each point, defining the noisy input \( P^T \). Then, we use our trained noise predictor \( \epsilon_\theta \) to denoise \( P^T \) iteratively until arriving at \( P^0 \), yielding a completed representation of the 3D scene.

Abstract

Computer vision techniques play a central role in the perception stack of autonomous vehicles. Such methods are employed to perceive the vehicle surroundings given sensor data. 3D LiDAR sensors are commonly used to collect sparse 3D point clouds from the scene. However, compared to human perception, such systems struggle to deduce the unseen parts of the scene given those sparse point clouds. In this matter, the scene completion task aims at predicting the gaps in the LiDAR measurements to achieve a more complete scene representation. Given the promising results of recent diffusion models as generative models for images, we propose extending them to achieve scene completion from a single 3D LiDAR scan. Previous works used diffusion models over range images extracted from LiDAR data, directly applying image-based diffusion methods. Distinctly, we propose to directly operate on the points, reformulating the noising and denoising diffusion process such that it can efficiently work at scene scale. Together with our approach, we propose a regularization loss to stabilize the noise predicted during the denoising process. Our experimental evaluation shows that our method can complete the scene given a single LiDAR scan as input, producing a scene with more details compared to state-of-the-art scene completion methods. We believe that our proposed diffusion process formulation can support further research in diffusion models applied to scene-scale point cloud data.

1. Introduction

Perception systems are a crucial component of self-driving cars, enabling them to understand their surroundings and safely navigate through it. Such systems rely on the data collected by the sensors installed on the vehicle to perceive the environment but fail to deduce areas only partially observable by the sensor. For a human it is comparatively rather simple to infer the complete scene from the scene context. Especially in autonomous driving, LiDAR sensors are employed to collect 3D information of the vehicle surroundings to enable safe navigation. Despite the accuracy of those sensors, collected point clouds are sparse, with large gaps between the data points measured by the sensor beams. Being able to complete the measured scene can add valuable information to perception systems, helping to improve different tasks such as object detection [42], localization [40] or navigation [29].

Scene completion tries to infer the missing parts of a scene, providing a dense and more complete scene representation. Given the LiDAR data sparsity, having a way to fill the gaps of non-observed regions is helpful to enlarge the incomplete data measured by the sensor. Previously, this task was tackled using paired RGB images and LiDAR point clouds by inferring depth maps from an RGB image supervised by the LiDAR depth measurements [8, 21, 22, 44]. Other approaches [15, 25, 40] employ signed distance fields (SDF) where the scene is represented as a voxel grid where each voxel stores its distance to the closest surface in the point cloud. Such methods approx-
imagine the scene by a surface representation, losing details usually present in real-world data since these approaches are limited to the voxel resolution. As an extension to this task, semantic scene completion has emerged [15, 31, 32], where the goal is to infer an occupancy voxel grid with a semantic label associated to each voxel. However, those methods require large amounts of labeled data and operate at a predefined fixed voxel grid resolution. More recently, denoising diffusion probabilistic models (DDPM) were employed in the context of self-driving cars [14, 26, 50] relying on image representations of the LiDAR data, such as range images [26, 50] or a discrete diffusion process formulation, inferring the occupancy on a predefined voxel grid [14].

In this work, we propose a diffusion scheme for 3D data operating at point level and at scene scale. We exploit the generative properties of DDPMs to infer the unseen regions of a scene measured by a 3D LiDAR sensor, achieving scene completion from a single point cloud as illustrated in Fig. 1. We reformulate the (de)noising scheme used in DDPMs by adding noise locally to each point without scaling the input data to the noise range, allowing the model to learn detailed structural information of the scene. Furthermore, we propose a regularization to stabilize the DDPMs during training, approximating the predicted noise distribution closer to the real data. We compare our method with different scene completion approaches and conduct extensive experiments to validate our proposed scene-scale 3D diffusion scheme. In summary, our key contributions are:

- We propose a novel scene-scale diffusion scheme for 3D sensor data that operates at the point level.
- We propose a regularization that approximates the predicted noise to the expected noise distribution.
- Our method can generate more fine-grained details compared to previous methods.
- Our approach achieves competitive performance in scene completion compared to previous diffusion and non-diffusion methods.

2. Related Work

Scene completion aims at inferring missing 3D scene information given an incomplete sensor measurement. This inference of unseen information can be helpful for perception tasks [42], localization [40] or navigation [29]. Some works [8, 21, 22, 44] tackled this task by jointly extracting information from paired RBG images and LiDAR point clouds, predicting a depth map from an RGB image supervised by the LiDAR data. Differently, other methods [15, 40] approach the problem by optimizing a signed distance field (SDF) given only the LiDAR measurements, representing the scene as a voxel grid where each voxel stores its distance to the closest surface in the scene. However, such methods are bound to the voxel resolution and lose details in the scene due to the discretization by voxels.

Distinctly, our approach works directly on the points and exploits the generative properties of DDPMs to complete the unseen data without relying on a voxel grid representation.

Semantic scene completion has been of great interest more recently due to the availability of large datasets with semantic labels [2–4, 7, 9, 16, 39]. This task extends the scene completion task by predicting a semantic label for each occupied voxel [15, 31, 32]. However, those methods are also tightly bound to the voxel grid resolution, which usually has a low resolution due to memory limitations. Besides operating at point level, given the recent research effort for DDPMs, our method could also later be extended to predict a semantic class for each generated point.

Denoising diffusion probabilistic models have gained attention due to their high-quality results in image generation [6, 11, 27, 28, 30, 33, 48, 49]. Besides that, conditioned diffusion models gained even more relevance due to the possibility of generating data towards an input condition [1, 10, 46]. The drawback of DDPMs is usually the time needed during the denoising process. For that reason, many efforts have been put to achieve a faster generation, e.g., by doing a distillation of the denoising model [23, 34] or by analytically approximating the denoising steps solution to reduce the amount of steps needed [12, 17, 18, 37].

Diffusion models for 3D data have been investigated due to their promising performance in the image domain. Such methods [19, 20, 35, 36, 43, 45, 47] are focused on single object shapes, achieving novel object shape generation or completion. Few works [14, 26, 50] target real-world data generation. Some works [26, 50] rely on projecting the 3D data to an image-based representation such as range images, such that the methods proposed in the image domain can be directly applied. For such approaches, the 3D scene cannot be completed since when reprojecting the image to the 3D world, some regions do not have any information due to occlusions in the projected point cloud. Lee et al. [14] achieves scene-scale 3D data generation using a discrete diffusion model formulation and a fixed voxel grid representation of the environment. The model is then used to infer for each voxel whether it is occupied, and a semantic label is predicted. Different from previous works, our method operates directly at point level and does not rely on a grid representation or projection to the image domain.

Given the recent advances in DDPMs for data generation, we propose a formulation of the denoising diffusion process that works at point level, achieving competitive performance in scene-scale diffusion scene completion. Our formulation enables the use of DDPMs to generate scene scale, real-world-like data without relying on any discretization or projection of the LiDAR data.
3. Approach

We propose using DDPMs to achieve scene completion from a single 3D LiDAR scan as input. First, we reformulate the DDPMs [19, 20, 47] to work at scene scale. Instead of normalizing the input point cloud, we add and predict the noise locally for each point. During the denoising process, we condition the noise prediction with the input scan such that the final scene retains the structural information from the input scan while inferring the missing parts. In this formulation, the initial point cloud is a noisy version of the input scan and the networks task is to denoise it to get the complete scene as depicted in Fig. 1. Next, we provide the needed background on diffusion models and describe the individual components of our approach.

3.1. Denoising diffusion probabilistic models

Denoising diffusion probabilistic models [6, 11, 27] formulate the data generation as an iterative denoising process. Commonly, the model starts from Gaussian noise [6, 11, 27] and iteratively removes noise from the input until it converges to the target output (e.g., images [6, 11, 27, 28, 30, 33, 48, 49] or shapes [19, 20, 35, 36, 43, 45, 47]). This can be achieved by defining a forward diffusion process where noise is iteratively added $T$ times to the target data. Then, the model is trained to predict the noise added at each step $t$. By predicting the noise at each step $t$ and removing it, the denoised sample should be closer to the target training data.

The diffusion process as formulated by Ho et al. [11] can be generally written as follows. Given a sample $x^0 \sim q(x)$ from a target data distribution, the diffusion process adds noise to $x^0$ over $T$ steps, resulting in $x^1, \ldots, x^T$, where $q(x^T) \approx N(0, I)$, where $N(0, I)$ is a normal distribution with mean 0 and the identity matrix $I$ as diagonal covariance. This diffusion process is parameterized by a sequence of defined noise factors $\beta_1, \ldots, \beta_T$, where iteratively at each step $t$, Gaussian noise is sampled and added to $x^{t-1}$ given $\beta_t$. This can be simplified to sample $x^t$ from $x^0$, without computing the intermediary steps $x^1, \ldots, x^{t-1}$. To do so, Ho et al. [11] define $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, and $x^t$ can be sampled as:

$$x^t = \sqrt{\bar{\alpha}_t} x^0 + \sqrt{1 - \bar{\alpha}_t} \epsilon,$$

where $\epsilon \sim N(0, I)$. Note that when $T$ is large enough $q(x^T) \approx N(0, I)$, since $\bar{\alpha}_T$ gets closer to zero.

The denoising process aims to undo the $T$ noising steps by predicting the noise $\epsilon$ added at each step $t$ [11]. Given an initial $x^T$, we want to reverse the diffusion process and get to $x^0$. The reverse diffusion step can be written as:

$$x^{t-1} = x^t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x^t, t) + \frac{1 - \bar{\alpha}_t - 1}{1 - \bar{\alpha}_t} \beta_t N(0, I),$$

where $\epsilon_\theta(x^t, t)$ is the noise predicted from $x^t$ at step $t$.

This generation can also be guided by a condition $c$. This conditional generation can either stem from a pre-trained encoder [6] or from classifier-free guidance [10], where the encoder is trained together with the noise predictor. In our case, we use the classifier-free guidance since it does not require a pre-trained encoder. With the classifier-free guidance, the model is trained to learn the conditional and unconditional noise distribution. In this case, at each training step the model has a probability $p$ of predicting the unconditional noise distribution, where the conditioning is set to a null token, i.e., $c = \emptyset$.

The training process optimizes the denoising model to predict the noise $\epsilon$ added at step $t$ to a given input. Given an input $x^0$ and a condition $c$, a random step $t \in [0, T]$ is sampled, and $x^t$ is sampled from Eq. (1) with a Gaussian noise $\epsilon$. Then, from $x^t$, $c$ and $t$, the model computes the noise prediction, supervising it with an $L_2$ loss:

$$\mathcal{L}(x^t, \hat{c}, c, t) = \|\epsilon - \epsilon_\theta(x^t, \hat{c}, t)\|^2,$$

with $\hat{c} \sim B[p]$ where $B$ is a Bernoulli distribution with outcomes $\{0, c\}$ with probability $p$ that $0$ occurs.

The inference starts from an initial $x^0 \sim N(0, I)$ and iteratively denoises it to get $x^0$. For the classifier-free guidance [10], we predict the conditional and unconditional noise distribution and compute the final predicted noise as:

$$\epsilon_\theta(x^t, c, t) = \epsilon_\theta(x^t, \emptyset, t) + s[\epsilon_\theta(x^t, c, t) - \epsilon_\theta(x^t, \emptyset, t)],$$

where $s \in \mathbb{R}$ is a parameter that weights the conditioning to $c$, and $\epsilon_\theta(x^t, \emptyset, t)$ is the unconditional noise prediction.

With Eq. (4) we can compute the noise at any step $t$, from which we can use Eq. (2) to compute $x^{T-1}, \ldots, x^0$, where $x^0$ is a newly generated sample conditioned on $c$.

3.2. Diffusion scene completion

In this work, we use the generative aspect of DDPMs to complete a scene measured in a single scan by a LiDAR sensor. Similarly to shape completion [19, 20, 47], the input is a partial point cloud $\mathcal{P} = \{p_1, \ldots, p_N\}$ where $p \in \mathbb{R}^3$, and the output should be the complete point cloud $\mathcal{P}' = \{p'_1, \ldots, p'_M\}$ where $p' \in \mathbb{R}^3$. In our case, the partial point cloud is a single LiDAR scan from which we want to achieve scene completion. Given a sequence of consecutive LiDAR scans and their poses, we can build a map and sample the complete scene ground truth $\mathcal{G}$ for an individual scan $\mathcal{P}$, where our scene completion $\mathcal{P}'$ should be as close as possible to $\mathcal{G}$.

Given the pair of input scan $\mathcal{P}$ and ground truth $\mathcal{G}$, we can train the DDPM to achieve scene completion. As detailed in Sec. 3.1, we can compute a noisy point cloud $\mathcal{G}'$ at step $t$ from the complete scene $\mathcal{G}$ in a point-wise fashion:

$$p'_m = \sqrt{\bar{\alpha}_t} p_m + \sqrt{1 - \bar{\alpha}_t} \epsilon, \forall p_m \in \mathcal{G},$$

with $\mathcal{G}' = \{p'_1, \ldots, p'_M\}$. 

In our case, we want to retrieve the complete scene \( \tilde{G} \) from \( G^T \). However, \( G^T \) retains little information from \( G \) due to the \( T \) diffusion steps. Therefore, we condition the generation with the scan \( P \) such that its structure guides the point cloud generation. From Eq. (4), the point-wise classifier-free noise prediction at step \( t \) can be written as:

\[
e_\theta(G^t, P, t) = e_\theta(G^t, \emptyset, t) + s e_\theta(G^t, P, t) - e_\theta(G^t, \emptyset, t).
\]

For training, at each iteration we select a random step \( t \in [0, T] \) and compute \( G^t \) from \( G \) given Gaussian noise \( \epsilon \sim \mathcal{N}(0, I) \). Then, we use the model to predict the noise from \( G^t \) conditioned to the LiDAR scan \( P \) or a null token \( \emptyset \) given a probability \( p \) as in Eq. (3), supervising with the loss:

\[
L_{\text{diff}}(G^t, \hat{c}, t) = || e - e_\theta(G^t, \hat{c}, t) ||^2,
\]

where as in Eq. (3), \( \hat{c} \sim \mathcal{B}(p) \) with \( \mathcal{B} \) as a Bernoulli distribution with outcomes \( \{\emptyset, P\} \) with probability \( p \) that \( \hat{c} \) occurs.

During inference, as detailed in Sec. 3.1, we can generate a scene conditioned to a LiDAR scan \( P \), by denoising from \( G^T \) to \( G^0 \) which is the predicted completion \( P^* \).

3.3. Local point denoising

The formulation detailed in Sec. 3.2 is usually used for shape completion [20, 47]. Even though achieving promising results for shape completion, this formulation may not directly work at the scene scale. For single object shapes, the data is either normalized or within a small range close to a Gaussian distribution with mean \( \mu = 0 \) and standard deviation \( \Sigma = I \). For scene scale, the LiDAR data has a much larger scale, and the data range differs depending on the point cloud axis. Therefore, the input data distribution is far from a Gaussian distribution \( \mathcal{N}(0, I) \), and if we normalize the data, we lose many details in the scene due to compressing it into a much smaller range as illustrated in Fig. 2.

To overcome this problem, we reformulate the diffusion process as a point-wise local problem. Instead of sampling \( x^\alpha \) as a mixed distribution between \( \epsilon \sim \mathcal{N}(0, I) \) and \( x^\alpha \) as in Eq. (1), we formulate the diffusion process as a noise offset added locally to each point \( p_m \in \tilde{G} \). In this case, from Eq. (1), we set \( x_0 = 0 \) and add \( x^\alpha \) to \( p_m \):

\[
p_m^t = p_m + (\sqrt{\alpha_t} \epsilon + \sqrt{1 - \alpha_t} \epsilon), \quad (8)
\]

\[
= p_m + \sqrt{1 - \alpha_t} \epsilon, \quad (9)
\]

With this formulation, the noise \( \epsilon \) is a random offset scaled w.r.t. the step \( t \) added to each point \( p_m \) in \( \tilde{G} \). The model needs to predict the noise at each step \( t \), slowly moving the noisy points towards the target scene \( \tilde{G} \) conditioned to the LiDAR scan \( P \), still operating in the original scale.

During inference, due to this local diffusion formulation, \( G^T \) cannot be approximated by a Gaussian distribution. Instead, we can generate \( \tilde{G}^T \) from the LiDAR scan \( P \). Besides, to complete the LiDAR scan, we need more points than the input scan. Therefore, given a single LiDAR scan \( P \), we increase its size by concatenating its points \( K \) times to get \( P^* = \{p_1^*, \ldots, p_K^*\} \), where \( M = K N \). Then, we sample a Gaussian noise for each point \( p_m^* \in P^* \) and compute the initial noisy point cloud \( \tilde{G}^T \) from \( P^* \) with Eq. (9). Finally, we calculate the \( T \) denoising steps by predicting the noise at step \( t \) from Eq. (4), and denoising it with Eq. (2) to get the complete scan \( \tilde{P}^* = P^0 \).

Note that, as long as \( P^T \) is “noisy enough” to resemble \( G^T \) as seen during training, the generation process is the same independent of using \( P^* \) or the ground truth \( G \) to sample the initial \( x^T \).

3.4. Noise prediction regularization

DDPMs use a leveraged formulation to train the model to predict only the noise added to the data. This formulation has only to optimize an \( L_2 \) loss between the added noise and the model prediction. However, this formulation optimizes the model to precisely predict the noise added to each point, ignoring the overall distribution of the noise sampled.

Given that the added noise \( \epsilon \sim \mathcal{N}(0, I) \), it is reasonable to expect that the prediction \( e_\theta(G^t, P, t) \approx \mathcal{N}(0, I) \). However, the model predicts a peaky distribution far from the expected, as shown in Fig. 3. The predicted noise starts with a mean far from zero and with a large standard deviation. As the denoising starts the mean gets closer to zero but the standard deviation is still far from one. Therefore, we propose a regularization to approximate \( e_\theta(G^t, P, t) \) to \( \mathcal{N}(0, I) \). We compute the mean \( \bar{\epsilon}_\theta \) and the standard deviation \( \epsilon_\theta \) over \( e_\theta(G^t, P, t) \) and calculate the regularization losses:

\[
L_{\text{mean}} = \bar{\epsilon}_\theta^2 \quad \text{and} \quad L_{\text{std}} = (\epsilon_\theta - 1)^2, \quad (10)
\]
Then, we compute it over an MLP to get $F_{\text{model}}$, we compute the closest point between the layer $t$ is the encoder output embedding size. To encode the embedding $= \{ w_{n_i'} \in \mathbb{R}^{d_l} \mid 1 < n_i' < N_l' \}$. Finally, we use one more MLP layer to project $W_l$ to the layer feature dimension $d_l$ and get $W_l'$. Then, we compute $F'_l = W_l' \otimes F_l$ as an element-wise multiplication, which is then feed as the input to layer $l$, as depicted in Fig. 4. As the refinement network, we use the same MinkUNet architecture used for the noise predictor without the conditioning encoder. For more details on the embeddings dimensions, noise predictor and refinement network architectures, we refer to the supplementary material.

4. Experiments

Datasets. For training our DDPM, we used the SemanticKITTI dataset [2, 9], an autonomous driving benchmark with point-wise annotations over sequences of LiDAR scans collected in an urban environment. To generate the ground truth complete scans, we used the dataset poses to aggregate the scans in the sequence and remove moving objects with the semantic labels, building a map for each sequence. For evaluation, we used the validation set from SemanticKITTI, i.e., sequence 08. Additionally, we used sequence 00 from the KITTI-360 dataset [16] and collected our own data with an Ouster LiDAR OS-1 with 128 beams to further compare the approaches.

For SemanticKITTI and KITTI-360, we used the ground truth poses to build the map, and for our data, we used KISS-ICP [41] to get the scan poses for our sequence. To remove the moving objects from the map in KITTI-360 and our data, we used an off-the-shelf moving object segmentation [24]. To compute the evaluation metrics, for each scan in the sequences, we remove the moving objects using the semantic labels using only the static points as input to the scene completion methods. Then, we evaluate the completed scene by comparing it with the corresponding region in the ground truth map.

Training. We train our model for 20 epochs, using only the training set from SemanticKITTI. As optimizer, we used Adam [13] with a learning rate of $10^{-4}$ decreased by half every 5 epochs, and decay of $10^{-4}$, with batch size equal to 2. For the diffusion parameters, we used $\beta_0 = 3.5 \cdot 10^{-5}$ and $\beta_T = 0.007$, with the number of diffusion steps $T = 1000$, linearly interpolating between $\beta_0$ and $\beta_T$ to define $\beta_1, \ldots, \beta_{T-1}$. We set the noise regularization $r = 5.0$, and the classifier-free probability $p = 0.1$. For the MinkUNet parameters, we set the quantization resolution to 0.05 m. For each input scan, we define the scan range as 50 m and sample 18,000 points with farthest point sampling. For the ground truth, we randomly sample 180,000 points without replacement. For the refinement network we use $k = 6$ as the number of offsets.

Inference. During inference, we use DPMSolver proposed by Lu et al. [17], reducing the number of denoising.
steps $T$ from 1,000 to 50. Besides, we set the classifier-free conditioning weight to $s = 6.0$. To maintain the same amount of points used during training, we again use the scan max range as 50 m and sample 18,000 points with farthest point sampling. Furthermore, as explained in Sec. 3.3, we set $K = 10$ to define the input noisy scan $P^* .$

**Baselines.** We compare our method with different scene completion methods, LMSCNet [32], PVD [47], Make It Dense (MID) [40], and LODE [15]. For all baselines, we used their official code and the provided weights also trained on SemanticKITTI. For PVD, we trained the approach with SemanticKITTI with their default parameters. We also follow their data loading, where the point clouds are normalized before the diffusion process. LMSCNet [32] and LODE [15] are limited to a fixed voxel grid of $51.2 \times 51.2 \times 6.4 \text{m}$. Given that our point cloud generation is done over a scan with a radius of 50 m, we divide the input scan into four quadrants over the $360^\circ$ LiDAR field of view, generating the complete scene over each quadrant and finally gathering them together as the final prediction. All baselines and our method were trained only with SemanticKITTI, and later evaluated on SemanticKITTI, and on KITTI-360 and our data without fine-tuning.

### 4.1. Scene reconstruction

In this experiment we evaluate how close is the predicted scan completion from the expected complete scene. To do so, we quantify it with two metrics, the Chamfer distance (CD) and the Jensen-Shannon divergence (JSD). The Chamfer distance evaluates the completion at point level, measuring the level of detail of the generated scene by calculating how far are its points from the expected scene. The JSD compares the point distributions between the generated and the ground truth scene. For the JSD, we follow the evaluation done by Xiong et al. [42], where the scene is first voxelized with a grid resolution of 0.5 m and then projected to a bird’s eye view (BEV) evaluating over this projection.

Tab. 1 shows the results of the scene completion methods on KITTI-360 and our collected data. Due to the poor performance of PVD over KITTI dataset, we do not evaluate it on those datasets. For KITTI-360 we notice the same behavior as in Tab. 1, where our method achieves the best performance in both metrics. When evaluating in our data, the performance of the SDF-based methods improve. This is expected since our data has denser point clouds, which is an advantage for such methods since they rely on the input points to approximate a surface to represent the scene. However, our method still achieves the best performance on the JSD metric and competitive performance on the CD metric. This evaluation shows that our method can still achieve scene completion over different datasets without fine-tuning since its generation is conditioned to the input scan. In Fig. 5 we compare the scene completion generation between the methods. We can see that the diffusion baseline, PVD, fails on generating scene-scale data. SDF-based methods inherit artifacts from the voxelization, while our method, especially after the refinement, can generate a scene closer to the expected, following closely the structural information from the input scan.

### 4.2. Scene occupancy

In this experiment, we assess the scene completion by evaluating the occupancy of the predicted scene compared with the ground truth. To do so, we follow the evaluation pro-
Figure 5. Qualitative results on one scan from KITTI-360. Colors depict point height normalized by the height range of each point cloud.

Table 3. Completion metric where the IoU is computed against the ground truth and prediction grids with different resolutions.

<table>
<thead>
<tr>
<th>Method</th>
<th>0.5</th>
<th>0.2</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMSCNet [32]</td>
<td>32.23</td>
<td>23.05</td>
<td>3.48</td>
</tr>
<tr>
<td>LODE [15]</td>
<td>43.56</td>
<td>47.88</td>
<td>6.06</td>
</tr>
<tr>
<td>MID [40]</td>
<td>45.02</td>
<td>41.01</td>
<td>16.98</td>
</tr>
<tr>
<td>PVD [47]</td>
<td>21.20</td>
<td>7.96</td>
<td>1.44</td>
</tr>
<tr>
<td>Ours</td>
<td>42.49</td>
<td>33.12</td>
<td>11.02</td>
</tr>
<tr>
<td>Ours refined</td>
<td>40.71</td>
<td>38.92</td>
<td>24.75</td>
</tr>
</tbody>
</table>

Table 4. Completion metric where the IoU is computed against the ground truth and prediction grids with different resolutions.

<table>
<thead>
<tr>
<th>Method</th>
<th>KITTI-360 (IoU) [%]</th>
<th>Our data (IoU) [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMSCNet [32]</td>
<td>25.46</td>
<td>21.93</td>
</tr>
<tr>
<td>LODE [15]</td>
<td>42.08</td>
<td>42.99</td>
</tr>
<tr>
<td>MID [40]</td>
<td>44.11</td>
<td>44.47</td>
</tr>
<tr>
<td>Ours</td>
<td>42.22</td>
<td>37.16</td>
</tr>
<tr>
<td>Ours refined</td>
<td>40.82</td>
<td>38.51</td>
</tr>
</tbody>
</table>

Tab. 3 shows the IoU of our method compared to the baselines at the different voxel resolutions. First, the diffusion baseline PVD has the lowest performance overall. This again shows that current state-of-the-art 3D shape completion diffusion methods cannot be directly applied to scene-scale data. At a higher voxel size, our approach stays behind some SDF-based baselines. This is reasonable in this evaluation since SDF methods use a voxel representation to reconstruct the scene. Therefore, its reconstruction is equally distributed over the point cloud and its voxel representation is denser compared to our result voxelized. As we decrease the voxel size, the baselines performance drops. At the low-
Figure 6. Mean and standard deviation of the predicted noise \( \epsilon_0 \) over different regularization weights. In this experiment we use DPM Solver [17] to reduce the denoising steps from 1,000 to 10.

At the highest resolution, our method outperforms the baselines. LM-SCNet [32] and LODE [15] are bound to a voxel resolution of 0.2 m, therefore with a voxel size of 0.1 m their performance drops drastically. Make It Dense [40] was trained with a voxel size of 0.1 m, however, our method still outperforms it at this resolution. This shows the advantage of our approach. Since it is trained at point level, it can produce a more detailed scene, not limited to a fixed grid size.

Due to the poor performance of PVD on SemanticKITTI, we compared our method only with non-diffusion approaches for the other two datasets. In Tab. 4, the same behavior is seen on KITTI-360 and our data. At higher voxel resolution, the SDF baselines have a higher IoU, while with a lower voxel size, our method achieves the best performance. It is also noteworthy that despite of SDF-based having advantage in our data as discussed in Sec. 4.1, our method still achieves the best performance at lower resolution. This suggests that our approach can reconstruct the scene with more details, and it is able to generate data from a different dataset than the one it was trained on, since its generation is guided by the input LiDAR scan.

4.3. Noise regularization

In this section, we evaluate the impact of the proposed noise prediction regularization on the generated scene. We compare the predicted noise distribution with different regularization weights \( r \) in Fig. 6 from the 10 denoising steps in one scan as in Fig. 3. As can be seen, without the regularization, \( r = 0 \), the predicted noise starts far from the expected distribution, with a mean of around −9.0 and a standard deviation of about 526. As we denoise the input, the distribution gets closer to the expected, however, still with a high standard deviation. When we add our proposed regularization, the model already starts predicting a more reasonable noise distribution from the beginning, stabilizing the denoising process. From this evaluation, we noticed that using \( r = 5.0 \) achieved a more stable distribution over the denoising steps. In our supplementary material, we provide also qualitative comparison between the generated point clouds with different regularization weights.

To evaluate how the regularization impacts the data generation, we compare the model performance over a short sequence from the SemanticKITTI validation set. We run the scene completion pipeline every one hundred scans without using the refinement network, evaluating only the regularization influence over the noise predictor. In Tab. 5, we compute the chamfer distance to compare the impact of the regularization over the quality of the generated scene. As we increase the regularization, the generation quality improves. Despite \( r = 3.0 \) achieving a slightly better result in this evaluation, we stick with \( r = 5.0 \) due to the analysis of the noise distribution from Fig. 6, and from the qualitative comparisons provided in the supplementary material.

5. Conclusion

In this paper, we propose a novel point-level denoising diffusion probabilistic model to achieve scene completion using autonomous driving data. We exploit the generative capabilities of DDPMs to generate the missing parts from a single sparse LiDAR scan. We reformatulate the diffusion process as a local problem. We define each point as the origin of the sampled Gaussian noise, learning an iterative denoising process to gradually predict offsets to reconstruct the scene from the input noisy LiDAR scan. This formulation enables the processing of scene-scale 3D data, retaining more details during the denoising process. In our experiments, we compare our method with recent state-of-the-art diffusion and non-diffusion methods. Our results show that our approach produces a more fine-grained completion compared to the baselines and can achieve scene completion on different datasets since its generation is conditioned to the input LiDAR scan. Besides, our proposed diffusion formulation distinguishes from previous state-of-the-art diffusion approaches by enabling the generation of scene-scale 3D data. Furthermore, we believe that our scene-scale diffusion formulation can support further research in the 3D diffusion generation research field.

Limitations. Even though achieving compelling results on scene completion, our method is still not able to generate unconditional data. This limits the data generation capability since it requires an input scan to guide the generation. In our supplementary material, we show examples of the unconditional generation of our approach. For future work, we plan on extending our method to generate unconditional data, creating novel 3D point cloud scenes.

Acknowledgments. This work has partially been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany’s Excellence Strategy, EXC-2070 – 390732324 – PhenoRob.
References


