# L3DR: 3D-aware LiDAR Diffusion and Rectification

## Supplementary Material

## A. Technical Appendices and Supplementary Material

### A.1. Related Work on Non-RV 3D Diffusion

3D based methods directly diffuse point coordinates on 3D or compressed 3D space instead of RV image space. Specifically, they have seen applications on object point clouds [22, 24, 42, 50, 53], mesh representation [12, 21, 55], and neural implicit fields [5, 9, 43]. However, due to the natural preference for local geometry rather than global structure, 3D diffusion has seen twists when applied to LiDAR point clouds. UltraLiDAR [47] circumvents the issue by encoding point clouds into a VQ-VAE feature map in BEV view. Such representation empowers accurate local geometry but disregards global self-occlusion, resulting in inaccurate projection relationships. LiDiff [31] approaches point cloud completion by 3D diffusion on the point level, but fails on direct generation. While these methods share similarities with our residual regression network, we highlight that they are not designed to recover local geometric properties including the LiDAR projection structure, therefore being unsuitable for the 3D-aware LiDAR generation task.

### A.2. Full Proof of Theorem 1

Without loss of generality (WLOG), we choose DDIM [39] as the analyze target, as it is of general form and has been widely applied. We then continue to derive a constant bound for the image gradient in DDIM-sampled images, and show that such bound for a 3D network is much looser.

**Assumption 1.** *Let $\epsilon_\theta : \mathbb{R}^{u\times v} \times \mathbb{R} \to \mathbb{R}^n$ denote the noise prediction network used in DDIM, where $\epsilon_\theta(x_t, t)$ predicts noise at time step $t$. We assume $\epsilon_\theta$ is spatially Lipschitz continuous:*

$$\|\nabla \epsilon_\theta(x, t)\| \le L_{\theta,t}\|\nabla x\|, \quad \forall x \in \mathbb{R}^{u\times v}. \tag{7}$$

*Where $L_{\theta,t}$ is a finite constant. In practice, the network $\epsilon_\theta$ is often implemented using convolutions, attentions and element-wise operations, which lead to such spatial smoothness.*

According to Song et al. [39], DDIM update rule is given by:

$$x_{t-1} = \sqrt{\alpha_{t-1}} \cdot \hat{x}_0 + \sqrt{1-\alpha_{t-1}} \cdot \epsilon_\theta(x_t, t). \tag{8}$$

$$\hat{x}_0 = \frac{x_t - \sqrt{1-\alpha_t} \cdot \epsilon_\theta(x_t, t)}{\sqrt{\alpha_t}}. \tag{9}$$

Where $\alpha_t$ are constants determined by a diffusing schedule. Each sampling step can be shorthanded as a function $x_{t-1} = f_t(x_t)$, and the final output is:

$$x_0 = f_1 \circ f_2 \circ \cdots \circ f_T(x_T), \tag{10}$$

where $x_T \sim \mathcal{N}(0, I)$ is the initial noise.

**Theorem 1.** *Under the above assumptions, the output image $x_0$ generated by DDIM is locally Lipschitz continuous with respect to the input noise $x_T$. Moreover, the spatial gradient of $x_0$ is bounded:*

$$\|\nabla x_0\| \le L \quad \text{for some constant } L \in \mathbb{R}$$

*Proof.* Each function $f_t$ is spatially Lipschitz continuous due to the spatial Lipschitz property of $\epsilon_\theta$. Scalar operations in Eq.8-9 is also Lipschitz with constant $L_{\epsilon,t}$ due to the finite input parameters and scalar operations. Then their composition in Eq. 10 is also spatially Lipschitz with constant: $L = \prod_{t=1}^{T} L_{\theta,t} L_{\epsilon,t}$.

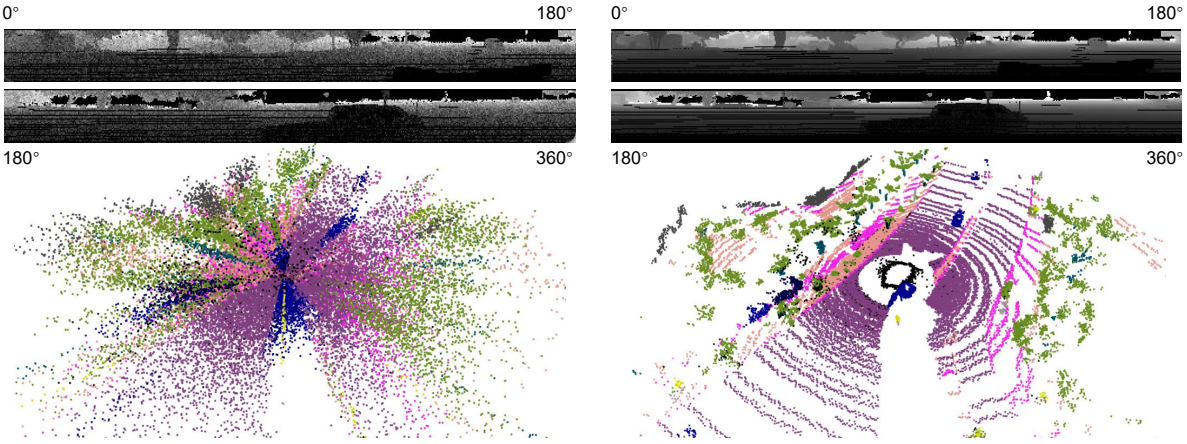
□

### A.3. Qualitative intuition for Theorem 1.

To qualitatively visualize the differences in noise tolerance between 2D range view images and 3D point clouds, We add exactly the same Gaussian noises to the same point cloud, which is displayed in both RV and 3D in Figure 8. Adding large noises ($\sigma = 5$m) does not change the perceivable semantics in images, but dramatically shuffles the point cloud to a cloud of mess, making it beyond human understanding. To preserve the spatial structure after adding noise, the noise has to be very tiny ($\sigma = 0.2$m). While the noise is still clearly visible on 3D according to the wiggling points, such small noise is unpercievable in 2D. Therefore, we conclude that 2D is much more tolerant and robust to large noises but ignores small noises, while 3D is more sensitive and intolerable to large noises but is better at processing small noises. Therefore, 2D RV is better at generating layouts from scratch using diffusion models, while 3D networks are better at rectifying local geometry of the diffusion-generated point clouds.

### A.4. Additional Experiment Setups

**Training details.** For the first-stage diffusion model LiDM, we keep the default set of parameters from the original paper for SemanticKITTI, which is conducted with $(64, 1024)$ RV image size which is shrunk into $(16, 128)$ size for latent diffusion with 16 channels. Unlike the original paper, we also place $(64, 1024)$ for nuScenes to avoid




(a) $\sigma = 5.0m$ (b) $\sigma = 0.2m$

Figure 8. **Intuitive validation for Theorem 1**, *i.e.,* 2D and 3D models have inherently different sensitivity to disturbances. (a) Adding large noises ($\sigma = 5$m) does not hinder human understanding of image contents on RV, yet completely sets the point cloud into a mess. (b) Adding small noises ($\sigma = 0.2$m) adds visible noises in the point cloud while retaining its spatial structure, but this noise scale is almost non-percievable in RV.

divergence, and disable the logarithmic scaling of depth values. The LiDM model is trained with AdamW optimizer under $1 \times 10^{(-6)}$ learning rate for up to 150 epochs. For the second stage RRN model, we follow the standard Pointcept training protocol without task-specific hyperparameter tuning. All RRN models are trained for a fixed 50 epochs using AdamW with OneCycleLR (max LR 0.002, weight decay 0.005), while certain modules in PTV3 have 0.0002 learning rate according to default configuration. No separate validation set or early stopping is used; instead, we observe stable convergence across datasets and report last-epoch performance. This fixed protocol ensures fair and reproducible comparisons across datasets and settings.

**Licenses.** The datasets we use are: 1) KITTI which uses CC BY-NC-SA 3.0, 2) KITTI-360 which uses CC-BY-NC-SA-3.0, 3) nuScenes which uses custom CC BY-NC-SA 4.0 with exceptions for startups and research and separate commercial licenses, 4) Waymo Open Dataset which uses a custom non-commercial licence.

**Metric specifications.** Our performance metrics measure generation quality from two different perspectives: perceptual and distributional. The extended FIDs include Fréchet Sparse Volume Distance (FSVD) and Frećhet Point Voxel Distance (FPVD), which are calculated with the average bottleneck features of pre-trained MinkowskiNet [6] and SPVCNN [41], respectively. The JSD and MMD represent global distributional similarity in BEV space, where the point clouds are voxelized with $0.5m$ voxel size and accumulated along the z-axis into a 2D bin-count map. JSD is then defined as the Jensen-Shannon Divergence between the average BEV map distribution of two sets of point clouds. MMD represents the average of the minimum distance from generated BEV maps to ground-truth BEV maps.

**Training cost of RRN.** With a batch size of 60, the RRN training time is 2 hours on a single RTX 4090. In comparison, LiDM training on 4 RTX 4090s costs 36 hours for 150 epochs. Therefore, RRN is much more training-efficient than LiDAR RV diffusion.

### A.5. Additional Experiments

**Ablation on nuScenes.** We ablate various components of the L3DR framework in nuScenes in Table 5. We calculated metrics with voxelized GT instead of original GT for fast evaluation, which results in slightly different figures. Specifically, applying MSE Loss deteriorates all metrics, which is also consistent with the conclusion on KITTI. The best overall performance again falls on SPUNet with Welsch loss and Segmentation input, with 9.8 FSVD, 9.7 FPVD, $9.5 \times 10^{-2}$ JSD, and $20.9 \times 10^{-5}$ MMD. On the other hand, adding semantic maps for PTV3 does not provide such consistent performance boosts, making it a suboptimal choice. Additionally, substituting the 3D RRN with a 2D UNet again deteriorates all performance metrics. We conclude that SPUNET and Welsch Loss are the best setup for nuScenes.

**Ablation on Waymo.** We ablate various components of the L3DR framework in Waymo Open Dataset in Table 6. We calculated metrics with voxelized GT instead of original GT for fast evaluation, which results in slightly different figures. Specifically, applying MSE Loss deteriorates

| Configurations | | | Perceptual | | Statistical | |
|---|---|---|---|---|---|---|
| Backbone | Loss | RRN Sem. Input | FSVD↓ | FPVD↓ | JSD×$10^{(-2)}$ ↓ | MMD×$10^{(-5)}$ ↓ |
| / (baseline) | / | - | 11.7 | 13.1 | 11.6 | 21.5 |
| SPUNet | Welsch | - | 10.6 | 12.1 | **6.7** | **16.6** |
| SPUNet | MSE | - | 27.0 | 25.7 | 10.1 | 22.3 |
| SPUNet | Welsch | ✓ | <u>9.8</u> | **9.7** | <u>9.5</u> | <u>20.9</u> |
| PTV3 | Welsch | - | **9.3** | <u>9.8</u> | 9.7 | 22.1 |
| PTV3 | MSE | - | 48.9 | 50.9 | 10.7 | 26.1 |
| PTV3 | Welsch | ✓ | 10.2 | 10.6 | 9.6 | 21.9 |

Table 5. **Ablation experiment** on nuScenes, including RRN backbone structure, loss function and semantic-map input to RRN.

| Configurations | | | Perceptual | | Statistical | |
|---|---|---|---|---|---|---|
| Backbone | Loss | RRN Sem. Input | FSVD↓ | FPVD↓ | JSD×$10^{(-2)}$ ↓ | MMD×$10^{(-5)}$ ↓ |
| / (baseline) | / | - | 11.8 | 12.4 | 9.5 | 13.4 |
| SPUNet | Welsch | - | <u>9.8</u> | **9.9** | 7.9 | 13.4 |
| SPUNet | MSE | - | 24.8 | 27.3 | 8.7 | **11.7** |
| SPUNet | Welsch | ✓ | **9.6** | 11.6 | 7.8 | <u>13.0</u> |
| PTV3 | Welsch | - | 10.3 | <u>11.4</u> | <u>7.6</u> | 13.6 |
| PTV3 | MSE | - | 25.6 | 27.8 | 9.2 | 15.3 |
| PTV3 | Welsch | ✓ | 11.4 | 14.5 | **7.5** | <u>13.0</u> |

Table 6. **Ablation experiment** on Waymo, including RRN backbone structure, loss function, and semantic-map input to RRN.

| $\nu$ | Perceptual | | Statistical | |
|---|---|---|---|---|
| | FSVD↓ | FPVD↓ | JSD×$10^{(-2)}$ ↓ | MMD×$10^{(-5)}$ ↓ |
| / (baseline) | 18.3 | 15.3 | 7.1 | 16.2 |
| 0.01 | 18.1 | 14.9 | 6.9 | **16.2** |
| 0.05 | 17.9 | 14.3 | 6.9 | <u>16.3</u> |
| 0.1 | 17.9 | <u>13.4</u> | 7.0 | <u>16.3</u> |
| 0.5 | <u>17.7</u> | 13.9 | <u>6.8</u> | 17.2 |
| 1.0 | **16.0** | **13.3** | **6.7** | 17.5 |

Table 7. **Parameter tuning experiment of** $\nu$ with PTV3 on SemanticKITTI.

most performance metrics compared to baseline, where the FSVD and FPVD all doubled for SPUNet and PTV3. The best overall performance attributes to SPUNET with Welsch loss, with 9.8 FSVD, 9.9 FPVD, $7.9 \times 10^{-2}$ JSD, and $13.4 \times 10^{-5}$ MMD. Additionally, using a 2D Image Unet as RRN again brings considerable degradation. We conclude that SPUNET and Welsch Loss are the best setup for Waymo.

**Parameter choice.** The choice of the width parameter $\nu$ of the Welsch Loss is discussed in Table 7. We observe contrary trends between the first 3 metrics and the last metric, where FSVD, FPVD and JSD improve with larger $\nu$ while MMD deteriorate in contrast. This trend reveals an intriguing property of the metrics: MMD is a local metric accumulated from individual point cloud pairs, which favors an average shape to match all possible point clouds. In comparison, FID and JSD are global metrics calculated over all point clouds, which favors perceptual realism and coherence. Therefore, we interpret that larger $\nu$ means individual point clouds are highly rectified and further from the average shape, which results in higher MMD, while larger $\nu$ offers better realism which is preferable for perceptual FID scores. Therefore, we choose $\nu = 0.5$ for good FID and JSD performance, while keeping an acceptable performance on MMD.

**Compatibility with reconstruction methods.** Table 8 shows RRN experiments on very limited reconstruction data of GS-LiDAR on KITTI-360. Since GS-LiDAR reconstruction fails on 3 out of 10 scenes due to VRAM OOM, the experiments were conducted over 7 scenes only (all training data has been augmented with z-axis rotation, flipping, and scaling, the network was trained for 1000 epochs, and the width parameter is set to 0.2 due to shrinked artifact sizes). Specifically, we trained two RRNs by using the GS-LiDAR validation frames (4/50 frames in a scene) and

| RRN Training Data | Test Data | C-D ↓ | F-score ↑ | RMSE ↓ | MedAE ↓ | LPIPS ↓ | SSIM ↑ | PSNR ↑ |
|---|---|---|---|---|---|---|---|---|
| Val (3 frames × 7 scenes) | Val (1 frame × 7 scenes) | 0.0557 | 0.966 | 1.64 | 0.00647 | 0.0233 | 0.978 | 33.9 |
| None (GS-LiDAR original) | Val (1 frame × 7 scenes) | 0.0560 | 0.963 | 1.64 | 0.00832 | 0.0231 | 0.978 | 34.0 |
| Train (46 frames × 7 scenes) | Val (4 frames × 7 scenes) | 0.0473 | 0.966 | 2.46 | 0.0171 | 0.0651 | 0.776 | 30.3 |
| None (GS-LiDAR original) | Val (4 frames × 7 scenes) | 0.0480 | 0.961 | 2.21 | 0.0156 | 0.0518 | 0.818 | 31.2 |

Table 8. RRN performance on GS-LiDAR reconstructed LiDAR data under different training and test data configurations.

| | CD↓ | EMD↓ | EdgeRMSE↓ | SV↑ |
|---|---|---|---|---|
| LiDM | 0.94 | 0.57 | 2.95 | 0.028 |
| Ours | 0.88 | 0.56 | 2.85 | 0.032 |

Table 9. Point-wise metrics on semantic-conditioned KITTI. *The SV of GT is the highest among all with value* 0.051, so higher SV is better.

the GS-LiDAR training frames (46/50 frames in a scene), respectively. While training RRN on the validation frames, we use the first 3 validation frames of each scene for training and the last validation frame for testing. While training RRN on the training frames, we test the trained model on all 4 validation frames of the scene. We can see that the validation-frame trained RRN performs slightly better than the original GS-LiDAR reconstructed frames across most performance metrics, while the training-frame trained RRN performs worse on multiple metrics besides Chamfer Distance and F-Score, confirming that reconstruction artifacts are only present in the validation frames. These results suggest that RRN could be incorporated as a promising addition to the reconstruction methods like GS-LiDAR. Experiments on larger-scale reconstruction data counts as future work.

**Point-wise metrics for depth bleeding and surface regularity.** We report Chamfer Distance (CD), Earth Mover's Distance (EMD), EdgeRMSE, and Surface Variation (SV) of conditional generation on KITTI in Table 9. EdgeRMSE measures depth bleeding by calculating the MSE of pixels with $\geq 0.5$ GT depth image differential. Surface variation (SV) indicates the average surface regularity, which is calculated with $\min_i (\lambda_i / \sum_j \lambda_j)$ of the covariance matrix in a 50-point patch for each point. These experiments are limited due to extended validation time especially for SV. In general, our rectified point clouds achieve better point-wise metrics, indicating better adherence to GT, lower depth bleeding, and improved surface regularity. We note that GT has the highest SV due to dominating complicated geometry (*e.g.*, sharp corners) over planar regions (see Figure 10). Our method pushes SV towards GT, while baseline generates low-variation curves distinct from GT.

## A.6. Limitations

While our proposed 3D rectification framework significantly improves the geometric fidelity of diffusion-generated LiDAR outputs, several limitations remain. First, the method depends on the initial quality of the range-view depth prediction. Severe RV artifacts that exceed the restoration capacity of the RRN may still leak into the rectified point clouds. Second, although our approach suppresses RV artifacts such as wavy surfaces, round corners, and depth bleeding, our pipeline does not explicitly model such artifacts, which may still result in room for improvement, especially in complex scenes. Finally, the L3DR framework relies on high-quality semantic-conditioned LiDAR diffusion model for training data generation. The general application of the L3DR framework in indoors or objects depends on the availability and quality of conditional diffusion models in these fields.

## A.7. Discussions

**Training cost of RRN.** With a batch size of 60, the RRN training time is 2 hours on a single RTX 4090. In comparison, LiDM training on 4 RTX 4090s costs 36 hours for 150 epochs. Therefore, RRN is much more training-efficient than LiDAR RV diffusion.

**Integraton of two-stage training into one.** We think it is inefficient and damaging to integrate the two-stage training into a single-stage joint training. Compared with point clouds generated by a converged diffusion model, point clouds generated during training have a compound of residual Gaussian noise and RV artifacts. Such noisy point clouds offer little help to the RV artifact regression. They also make RNN training challenging, leading to a degraded RNN model and further degraded generation performance.

**Regression of all error regions rather than only regressing RV artifacts.** We highlight that RV artifacts are errors with consistent patterns across different diffusion models and datasets, while high-bias errors are not even consistent in the same model and same dataset with different random seeds. This is because RV artifacts are caused by inherent property that a 2D model tends to attend to global structures and ignore small errors as discussed in Section 3, which is shared across all models despite nuances in implementation. On the other hand, high-bias errors are introduced by insufficient geometric constraints in large chunks of the semantic map control input, which can have multiple reasonable in-

| Method | Benefits | Limitations |
|---|---|---|
| RV Diffusion (e.g., R2DM, LiDM, ours) | • Lightweight and low-cost generation • Broad condition compatibility • Natural modeling of LiDAR's projective structure | • Single-frame generation Geometry artifacts • Moderate Gen2Real gap)) |
| BEV Diffusion (e.g., UltraLiDAR) | • Supports manipulation of BEV grid • Local spatial control with partial point input | • Poor projection realism • Limited multimodal control • Still single-frame |
| 4D Occupancy Diffusion (e.g., UniScene, OccSora) | • Spatiotemporal coherence • Multimodal output such as image and occupancy • Spatio-temporal consistency | • High computational burden • Occupancy control is unnatural to human users • Hard-hit approximated with volumetric rendering |
| Simulators (e.g., CARLA) | • Accurate projection model • Expanding asset libraries | • Large Sim2Real gap • Difficult to scale up |
| Reconstruction Methods (e.g., LiDiff, NeRF-LiDAR) | • High geometric fidelity • Fast inference (esp. 3DGS) | • Requires per-scene training • Limited layout diversity • Large storage and pre-processing overhead |

Table 10. Comparative overview of existing genres for LiDAR point cloud generation.

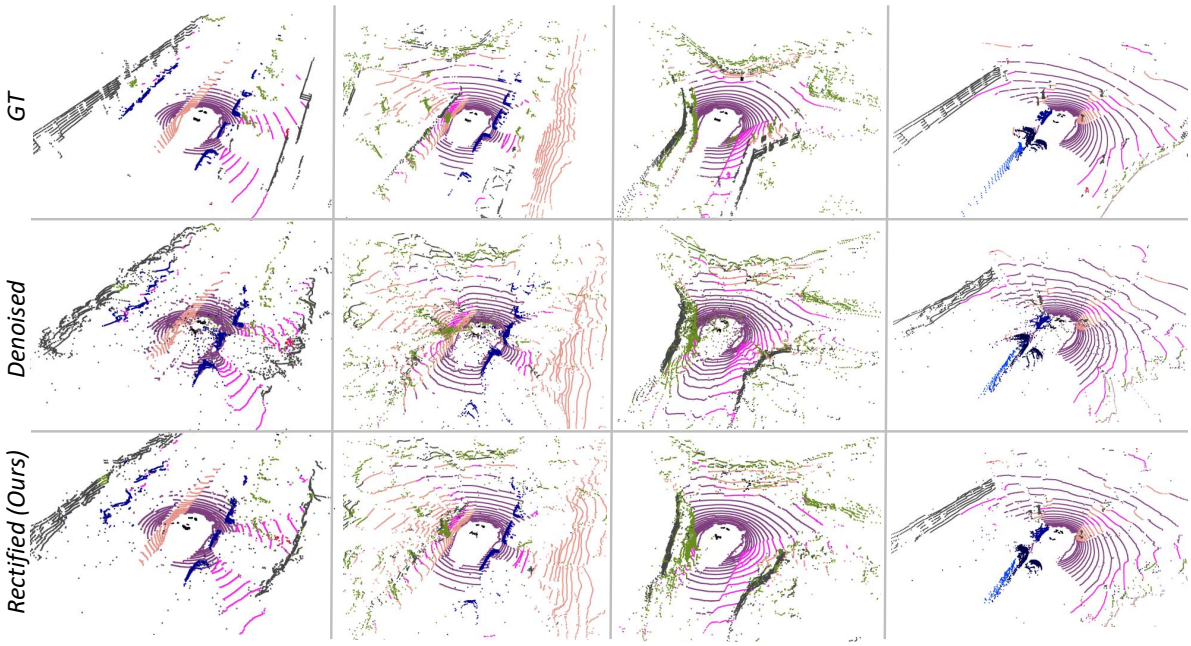



Figure 9. **Visualization** of conditional generation on nuScenes. Our results exhibit better geometry-realism such as sharp borders, flat surfaces, less noise points and reduced depth bleeding.

terpretations and each diffusion run could randomly choose from one of them. Therefore, it shares significantly less general value to regress all errors instead of RV artifacts only, which makes the model specific to that run rather than generally applicable to other models.

**Comparative overview of RV in all LiDAR generation genres.** To contextualize our method, we provide a comparative overview of representative LiDAR generation paradigms in Table 10. Unlike BEV or volumetric (4D occupancy), range-view (RV) LiDAR generation natively handles self-occlusion and projection realism, which are crucial in LiDAR perception. Additionally, image-based conditioning is far more accessible and generalizable than occupancy grids or partial point clouds, thus positioning RV diffusion as a lightweight and highly versatile choice. Further, RV diffusion enjoys a higher level of versatility compared to driving simulators or reconstruction methods. At the other end, RV LiDAR generation does potentially produce geometric artifacts, but this is rather a common issue across different LiDAR generation paradigms. Hence, the conclusion is that each generation paradigm has its unique pros and cons and application scenarios. RV diffusion possesses superior versatility and computational efficiency which is a promising research direction in our humble opinion. We aim to address the key bottleneck of RV diffusion, i.e., geometry realism, and propose an effective solution through

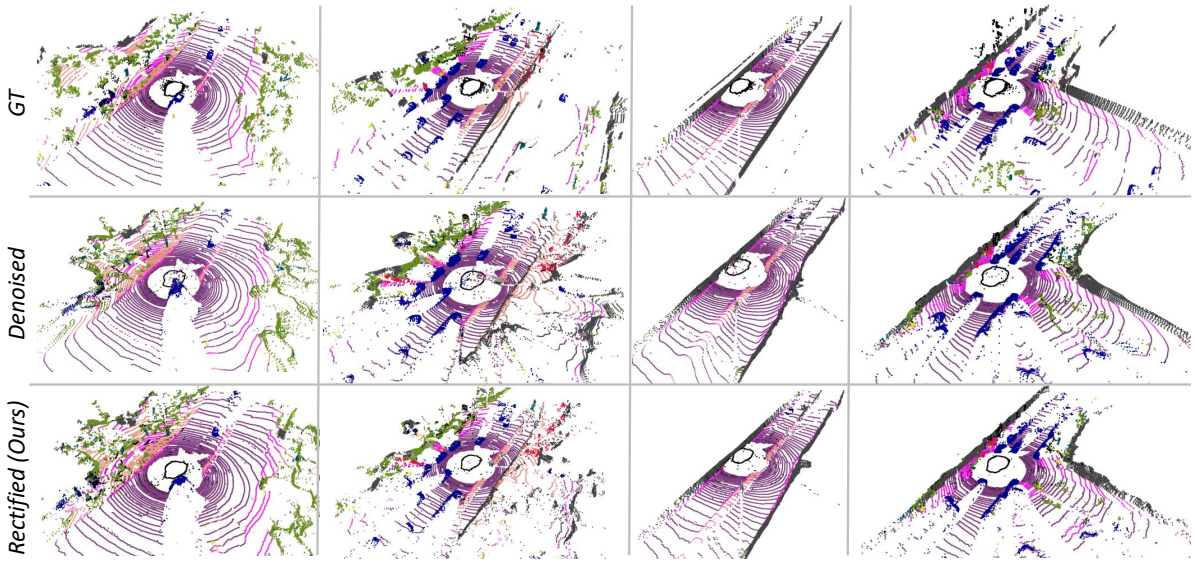



Figure 10. **Visualization** of conditional generation on Waymo Open Dataset. Our results exhibit better geometry-realism with reduced depth bleeding, flattened surfaces, rectified corners and reduced noise points.

3D residual regression. We believe that our work meaningfully advances the line of RV-based LiDAR generation research.

**Directly diffuse in 3D space instead of RV.** Diffusing in 3D from scratch cannot generate authentic global structures [31]. Our preliminary experiments on increasing diffusing steps in 3D also show receding performance as more diffusion steps are involved. We attribute this phenomenon to the noise tolerance capability discussed in Figure 8, *i.e.*, 2D diffusion models are good at reducing global noise instead of local noises or range view (RV) artifacts, while 3D networks are good at handling local geometry instead of global noises.

**Multi-step rectification with RRN.** Empirically, we have tested an alternative version of multi-step RRN where each step are trained to regresses the residual error of the previous step. However, experiments reveal receding performance as more steps are added. We interpret this phenomenon as increasingly harder and misaligned learning objectives hinder the learning of of a consistent strategy. For example, 90% of the error may come from RV artifacts in the first round, the second round may still have 30% unsolved RV artifacts and 60% residual regression error of the first step, and the third round still contains raw RV artifatcs, artifacts unsolved in the first step, and those unsolved in the second step. More steps indicate a more convoluted error pattern which hinders accurate recognition and cancellation of these residuals, which damages the overall performance.

**Difference between RRN and post-processing networks in 3DGS.** We point out that in reconstruction models like NeRF-LiDAR, LiDAR4D, GS-LiDAR, and LiDAR-RT, the post-processing by U-Net serves to regress raydrop possibilities for reconstruction. It is formulated as a binary classification task and trained with BCE loss. Differently, our 3D RRN is a novel design to LiDAR diffusion aiming at removing RV artifacts present in 3D coordinates. It is formulated as a vector regression task and trained with MSE/Welsch loss. Hence, both approaches share little similarity besides using U-Nets for post-processing.