U4D: Uncertainty-Aware 4D World Modeling from LiDAR Sequences

Supplementary Material

Table of Contents

A. Additional Implementation Details

In this section, we provide comprehensive implementation details to facilitate reproducibility and enable understanding of our experimental setup. We elaborate on the training pipeline, network configurations, data preprocessing, and evaluation protocols adopted throughout our experiments.

A.1. Training Configurations

In this section, we provide detailed training configurations to facilitate the reproducibility of U4D.

Data Preprocessing. As described in the main paper, we first convert raw LiDAR point clouds into range-image representations [5, 6, 12], which serve as the input format for our diffusion-based generative framework. Given a 3D point (x_i, y_i, z_i), its projected pixel coordinate (u_i, v_i) in a range image of resolution $H \times W$ is computed as:

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - \arctan(y_i, x_i)\pi^{-1}\right] W \\ \left[1 - (\arcsin(z_i d_i^{-1}) - f_{\text{down}}) f^{-1}\right] H \end{pmatrix}, \quad (1)$$

where $d_i = \sqrt{x_i^2 + y_i^2 + z_i^2}$ denotes the radial distance of the point, and $f = f_{\text{up}} - f_{\text{down}}$ is the vertical field of view (FOV) spanned by the LiDAR sensor. The parameters f_{up} and f_{down} correspond to the upper and lower elevation angles defined by the sensor. For range-image construction, we adopt resolutions $(H, W) = (32, 1024)$ for nuScenes [3] and $(64, 1024)$ for SemanticKITTI [1]. The vertical FOV settings $(f_{\text{down}}, f_{\text{up}})$ follow the respective sensor configurations of each dataset: $(-30°, 10°)$ for nuScenes [3] and $(-25°, 3°)$ for SemanticKITTI [1].

Each resulting range image $\mathbf{X} \in \mathbb{R}^{H \times W \times 2}$ contains two channels – depth and intensity[1]. To stabilize diffusion training, we apply depth compression and channel-level normalization before feeding data to the model. For the depth channel, we employ a logarithmic compression [13]:

$$\mathbf{X}_d^{\text{norm}} = \frac{\log_2(\mathbf{X}_d + 1)}{\log_2(d_{\max} + 1)}, \quad (2)$$

where $d_{\max}$ is the maximum measurable LiDAR range, set to 80.0 meters in all experiments. This compression mitigates the large dynamic range of raw depth value and enhances the stability of diffusion noise prediction, particularly in distant sparse regions. Then, all pixel values are linearly scaled into the range $[-1, 1]$ as:

$$\mathbf{X}^{\text{input}} = 2 \cdot \mathbf{X}^{\text{norm}} - 1. \quad (3)$$

This standardization step ensures consistent input statistics across datasets and improves training convergence.

Network Architectures. Following R2DM [13], we adopt a 4-layer Efficient U-Net [17] as the backbone of our diffusion network, with intermediate feature dimensions of 64, 128, 256, and 512. Each layer contains three Mixture of Spatio-Temporal (MoST) blocks, and the spatial resolution is downsampled by a factor of two along both the vertical and horizontal dimensions in the last three layers. In a MoST block, the input features are denoted as $\mathbf{F}_i \in \mathbb{R}^{C_i \times L \times H_i \times W_i}$, where C_i is the channel dimension, L is the temporal length, and (H_i, W_i) are the spatial resolutions at the i-th layer. The spatial branch processes $\mathbf{F}_i$ using a $1 \times 3 \times 3$ convolution to encode intra-frame geometric structures, while the temporal branch applies a $3 \times 1 \times 1$ convolution to capture inter-frame temporal dependencies. The outputs of the two branches are fused and added back to the input through a residual connection, ensuring stable training and efficient integration of spatial and temporal cues throughout the diffusion process.

Selection of Uncertainty Regions. To determine the uncertainty regions, we employ a pretrained RangeNet++ [12]

[1] For the nuScenes [3] dataset, the raw intensity values lies in $[0, 255]$ and are normalized to $[0, 1]$ by dividing by 255.0.

semantic segmentation model to estimate the per-pixel class probability distribution. For each point, we compute its Shannon Entropy [18] as:

$$H(\mathbf{p}) = \sum_{c=1}^{C} D(c \mid \mathbf{p}) \log D(c \mid \mathbf{p}) \,, \tag{4}$$

where $D(c \mid \mathbf{p})$ denotes the predicted probability of class c for point $\mathbf{p}$, and C is the total number of semantic classes. Points with higher entropy represent regions with greater semantic ambiguity, typically corresponding to object boundaries, distant structures, or sparsely scanned areas. To ensure sparse yet consistent uncertainty-mask coverage across the dataset, we retain the top-20% highest-entropy points for nuScenes [3] and the top-5% for SemanticKITTI [1]. These ratios are chosen to provide sufficient supervisory signal for ambiguous regions while preserving stable coverage across frames and sequences.

Training Hyperparameters. We employ a two-stage training pipeline for U4D using the PyTorch [15] framework. The first stage, *uncertainty-region modeling*, is trained for 1,000,000 steps, whereas the second stage, *uncertainty-conditioned completion*, is trained for 500,000 steps. Both stages use a batch size of 8 with a sequence length of 6. We use the AdamW optimizer [11] with a learning rate of 1×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.99$, and $\epsilon = 1 \times 10^{-8}$. The learning rate follows a cosine annealing schedule with a warm-up period of the first 10,000 steps [10]. To further stabilize optimization, we apply an exponential moving average (EMA) with a decay rate of 0.995, updated every 10 training steps. The diffusion process adopts continuous timesteps with a cosine noise schedule, consistent with R2DM [13]. During inference, we use 256 sampling steps to ensure a fair comparison with R2DM [13]. All experiments are conducted on a server equipped with four NVIDIA RTX 4090 GPUs under mixed-precision (FP16) training.

A.2. Evaluation Configurations

This section summarizes the evaluation configurations and metrics used to assess the quality of LiDAR scene generation from three perspectives: [1]*Geometric and Spatial Fidelity*, [2]*Temporal Coherence*, and [3]*Downstream Utility*.

Geometric and Spatial Fidelity. This set of metrics evaluates how accurately the generated LiDAR scenes capture the geometry and spatial structure of real-world environments. We adopt four measures:

• *Fréchet Range Distance (FRD).* FRD quantitatively evaluates the generation quality in the range image domain, which provides a structured 2D representation of LiDAR point clouds. Given the real set $\mathcal{R}$ and the generated set $\mathcal{G}$, their corresponding range images are processed using a RangeNet++ [12] model pretrained for semantic segmentation on the real dataset. The intermediate feature activations extracted from the backbone are denoted as $\mathcal{F}_r$ and $\mathcal{F}_g$, representing the feature distributions of real and generated scenes, respectively. The FRD is computed as:

$$\texttt{FRD}(\mathcal{R}, \mathcal{G}) = \|\mu_g - \mu_r\|_2^2 + \texttt{Tr}\left(\Sigma_g + \Sigma_r - 2(\Sigma_g \Sigma_r)^{\frac{1}{2}}\right) , \tag{5}$$

where μ_r and μ_g denote the mean feature embeddings of $\mathcal{F}_r$ and $\mathcal{F}_g$, Σ_r and Σ_g represent their corresponding covariance matrices, and $\texttt{Tr}(\cdot)$ denotes the matrix trace. A lower FRD value indicates that the generated samples more closely match real LiDAR scenes in the learned feature space, implying higher fidelity and semantic consistency.

• *Fréchet Point Distance (FPD).* While FRD operates in the range image domain, FPD assesses the generation quality directly in the 3D point cloud space, offering a complementary geometric perspective. Following [7, 9, 13], we employ a PointNet [16] model pretrained on the ShapeNet dataset [4] for 16-class object classification to extract high-level geometric features from both the real and generated point clouds, resulting in feature sets $\mathcal{F}_r^p$ and $\mathcal{F}_g^p$. Analogous to FRD, the FPD is formulated as:

$$\texttt{FPD}(\mathcal{R}, \mathcal{G}) = \|\mu_g^p - \mu_r^p\|_2^2 + \texttt{Tr}\left(\Sigma_g^p + \Sigma_r^p - 2(\Sigma_g^p \Sigma_r^p)^{\frac{1}{2}}\right) , \tag{6}$$

where μ_r^p and μ_g^p denote the mean feature embeddings of $\mathcal{F}_r^p$ and $\mathcal{F}_g^p$, and Σ_r^p and Σ_g^p are their corresponding covariance matrices. A smaller FPD score suggests that the generated point distributions closely resemble those of the real-world LiDAR data in the latent geometric space, capturing fine-grained structural details beyond surface-level similarities.

• *Jensen-Shannon Divergence (JSD).* JSD measures the similarity between the spatial occupancy distributions of real and generated LiDAR scenes from the bird's-eye-view (BEV) perspective. For each sample, we compute a 2D occupancy histogram projected onto the BEV plane, resulting in two probability distributions, denoted as P (real) and Q (generated). The JSD is then defined as:

$$\texttt{JSD}(P \| Q) = \frac{\texttt{KL}(P \| M)}{2} + \frac{\texttt{KL}(Q \| M)}{2} \,, \tag{7}$$

where $M = (P + Q)/2$ and $\texttt{KL}(\cdot \| \cdot)$ denotes the Kullback-Leibler divergence. A lower JSD value indicates that the generated BEV occupancy maps better approximate the global spatial distribution of real scenes.

• *Maximum Mean Discrepancy (MMD).* MMD also evaluates distributional similarity in the BEV domain, but from a kernel-based perspective. Given the same occupancy histograms P and Q, MMD is defined as:

$$\texttt{MMD}(P, Q) = \|\mathbb{E}_P\left[\phi(x)\right] - \mathbb{E}_Q\left[\phi(y)\right]\|_{\mathcal{H}}^2 . \tag{8}$$

where $\phi(\cdot)$ denotes the feature mapping to a reproducing kernel Hilbert space (RKHS). A smaller MMD value reflects a higher degree of alignment between the real and generated distributions, complementing JSD by providing a non-parametric statistical measure.

Temporal Coherence. Temporal coherence measures how consistently the generated LiDAR scenes evolve over time, ensuring smooth object motion and realistic scene dynamics. Following [7], we adopt two metrics:

• *Temporal Transformation Consistency Error (TTCE).* TTCE evaluates the temporal coherence of generated LiDAR sequences by comparing frame-to-frame transformations against ground truth. We first apply the Iterative Closest Point (ICP) algorithm [2] to consecutive generated frames to estimate the rigid transformation $\mathbf{T}_t^p = [\mathbf{R}_t^p \mid \mathbf{t}_t^p]$, where $\mathbf{R}_t^p$ and $\mathbf{t}_t^p$ denote rotation and translation, respectively. Let $\mathbf{T}_t^g = [\mathbf{R}_t^g \mid \mathbf{t}_t^g]$ be the ground-truth transformation. TTCE is computed as:

$$\texttt{TTCErot} = \frac{1}{T-1}\sum_{t=1}^{T-1} \|\mathbf{R}_t^p (\mathbf{R}_t^g)^\top - \mathbf{I}\|_F \,, \tag{9}$$

$$\texttt{TTCEtrans} = \frac{1}{T-1}\sum_{t=1}^{T-1} \|\mathbf{t}_t^p - \mathbf{t}_t^p\|_2 \,, \tag{10}$$

where T is the number of frames and $\|\cdot\|_F$ denotes the Frobenius norm. Lower TTCE values indicate better alignment with the ground-truth transformations, reflecting higher temporal consistency and more realistic motion in the generated LiDAR sequences.

• *Chamfer Temporal Consistency (CTC).* CTC measures temporal smoothness at the geometric level by computing the Chamfer Distance (CD) between consecutive frames after aligning them using ground-truth transformations. Let $\mathcal{P}_t$ and $\mathcal{P}_{t+1}$ be generated point clouds at frames t and $t+1$. We align $\mathcal{P}_{t+1}$ to frame t via:

$$\hat{\mathcal{P}}_{t+1} = (\mathbf{R}_t^g)^{-1}(\mathcal{P}_{t+1} - \mathbf{t}_t^g) \,. \tag{11}$$

The Chamfer Distance between $\mathcal{P}_t$ and $\hat{\mathcal{P}}_{t+1}$ is:

$$\begin{aligned}\texttt{CD}(\mathcal{P}_t, \hat{\mathcal{P}}_{t+1}) = &\frac{1}{|\mathcal{P}_t|}\sum_{x\in\mathcal{P}_t} \min_{y\in\hat{\mathcal{P}}_{t+1}} \|x-y\|_2^2 + \\ &\frac{1}{|\hat{\mathcal{P}}_{t+1}|}\sum_{y\in\hat{\mathcal{P}}_{t+1}} \min_{x\in\mathcal{P}_t} \|y-x\|_2^2 \,.\end{aligned} \tag{12}$$

CTC is then averaged across all consecutive frame pairs:

$$\texttt{CTC} = \frac{1}{T-1}\sum_{t=1}^{T-1} \texttt{CD}(\mathcal{P}_t, \hat{\mathcal{P}}_{t+1}) \,. \tag{13}$$

Lower CTC values indicate smoother frame-to-frame transitions, reflecting stronger temporal coherence in the generated 4D LiDAR sequences.

Table A. **Comparison of state-of-the-art LiDAR scene generation methods** on the *KITTI-360* [8] dataset. Metrics marked with ↓ indicate that lower values are better. The **MMD** scores are reported in units of 10^{-4}. The **best** and second-best scores are highlighted in **bold** and underline, respectively.

Method	Venue	FRD ↓	FPD ↓	JSD ↓	MMD ↓
LiDARGen [22]	ECCV'22	579.39	90.29	0.07	7.39
R2DM [13]	ICRA'24	153.73	6.24	**0.03**	1.91
U4D	**Ours**	**142.53**	**5.94**	**0.03**	**1.84**

Downstream Utility. To evaluate the practical usefulness of generated LiDAR sequences for real-world perception tasks, we consider two downstream metrics: semantic segmentation performance and model calibration.

• *Mean Intersection-over-Union (mIoU).* mIoU is a standard evaluation metric for semantic segmentation that quantifies the overlap between predicted and ground-truth regions over all classes. It is computed as:

$$\texttt{mIoU} = \frac{1}{|\mathbb{C}|}\sum_{c\in\mathbb{C}} \frac{\texttt{TP}_c}{\texttt{TP}_c + \texttt{FP}_c + \texttt{FN}_c} \,, \tag{14}$$

where $\mathbb{C}$ denotes the set of semantic classes, and $\texttt{TP}_c$, $\texttt{FP}_c$, and $\texttt{FN}_c$ represent the number of true positives, false positives, and false negatives for class c, respectively. Higher mIoU values indicate more accurate segmentation and better utilization of generated LiDAR scenes in downstream perception tasks.

• *Expected Calibration Error (ECE).* ECE is an important metric that evaluates the calibration of a perception model, *i.e.*, how well the predicted confidence aligns with the actual accuracy. It is defined as:

$$\texttt{ECE} = \frac{1}{M}\sum_{m=1}^{M} \frac{|B_m|}{N} |\texttt{acc}(B_m) - \texttt{conf}(B_m)| \,, \tag{15}$$

where M is the number of confidence bins, N is the total number of samples, and $|B_m|$ is the number of samples falling into the m-th bin. $\texttt{acc}(B_m)$ and $\texttt{conf}(B_m)$ denote the empirical accuracy and average confidence of bin B_m, respectively. Lower ECE values indicate better calibration, meaning the model's predicted probabilities closely match the true likelihood of correctness, which is particularly important for evaluating uncertainty-aware modeling.

B. Additional Quantitative Results

In this section, we present additional quantitative results to further demonstrate the effectiveness of the U4D design.

B.1. Benchmark on KITTI-360

To further evaluate the effectiveness and generalization ability of U4D, we conduct additional benchmarking experiments on the widely used KITTI-360 dataset [8]. Following the same training configuration as SemanticKITTI [1],

Table B. **Ablation study on uncertainty region selection** on the *nuScenes* [3] dataset. Metrics marked with ↓ indicate that lower values are better. The **MMD** scores are reported in units of 10^{-4}. The **best** scores are highlighted in **bold**.

#	Threshold	FRD ↓	FPD ↓	JSD ↓	MMD ↓
(1)	Score-based	531.65	21.53	0.06	0.69
(2)	Top-10%	242.23	13.53	0.04	0.55
(3)	Top-15%	231.06	13.02	0.04	0.51
(4)	Top-20%	**223.96**	**12.90**	**0.03**	0.53
(5)	Top-25%	227.53	12.94	**0.03**	**0.50**

Table C. **Ablation study on different segmentors for selecting uncertainty region** on the *nuScenes* [3] dataset. Metrics marked with ↓ indicate that lower values are better. The **MMD** scores are reported in units of 10^{-4}. The **best** scores are highlighted in **bold**.

#	Segmentor	FRD ↓	FPD ↓	JSD ↓	MMD ↓
(1)	RangeNet++ [12]	**223.96**	12.90	**0.03**	0.53
(2)	FRNet [20]	227.03	**12.31**	**0.03**	**0.52**
(3)	Cylinder3D [21]	235.31	12.48	0.04	**0.52**
(4)	SPVCNN [19]	232.56	13.01	**0.03**	0.54

we train and evaluate our model under identical settings to ensure a fair comparison. As reported in Tab. A, U4D consistently outperforms existing LiDAR generation methods. In particular, our method achieves lower distribution discrepancies and improved geometric fidelity, indicating that the proposed spatio-temporal modeling effectively captures both structural and temporal characteristics of LiDAR data.

B.2. Uncertainty-Region Selection

To maintain a consistent number of uncertainty points across different scenes, we select the top-K high-entropy points to form uncertainty regions. In Tab. B, we conduct a series of ablation studies on the selection of the top-K points. First, we directly select uncertainty points based on entropy scores (row 1). This strategy leads to poor performance, which we attribute to the large distribution gaps across scenes. As a result, the number of remaining points varies significantly, making the first-stage generation difficult to learn. To this end, we instead select the top-K high-entropy points (rows 2-5), which ensures a consistent number of uncertain regions. As K increases, the performance gradually improves, since more structural layout information is preserved to guide the full-scene generation. However, when K becomes sufficiently large, the performance begins to saturate or even degrade, suggesting that introducing too many uncertainty points may reduce the effectiveness of the guidance.

B.3. Segmentor for Uncertainty Regions

We adopt RangeNet++ [12] as the default segmentation model to estimate uncertainty maps. To evaluate the robustness of U4D to the choice of segmentation model for uncertainty region estimation, we experiment with several representative segmentors, including the range-view-based RangeNet++ [12] and FRNet [20], the sparse-voxel-based Cylinder3D [21], and the multi-view-fusion-based SPVCNN [19]. The results are summarized in Tab. C, U4D consistently produces high-quality LiDAR scenes regardless of the segmentation model used for uncertainty estimation. This observation indicates that U4D is not tied to a specific segmentation model and demonstrates strong robustness and generalization across different segmentors.

C. Additional Qualitative Results

In Fig. A, we present qualitative comparisons between U4D and a recent state-of-the-art LiDAR sequence generator [14], together with the corresponding reference sequences. U4D exhibits notably improved geometric fidelity and temporal coherence. It better preserves fine-grained structures that are often blurred or missing in prior methods, and in distant or low-density regions it reconstructs plausible planar surfaces with correct depth gradients. This robustness benefits from the proposed uncertainty-region modeling, which directs generation capacity toward hard-to-reconstruct areas. For dynamic objects, U4D yields smoother inter-frame transitions and more consistent object shapes and trajectories. The MoST block plays a key role here by enhancing temporal activations in intermediate layers while preserving spatial details elsewhere, enabling a more balanced spatio-temporal representation.

Beyond reconstructing observed sequences, we also explore U4D's capability for future frame prediction. As shown in Fig. B, given only the first frame, U4D can generate plausible future LiDAR observations that exhibit coherent scene evolution and realistic motion patterns. This highlights the model's ability not only to replicate existing sequences but also to forecast future dynamics in a physically consistent manner.

D. Broad Impact & Limitations

In this section, we discuss the broader impact of U4D and outline its potential limitations to provide a balanced and transparent assessment of our work.

D.1. Broader Impact

U4D contributes to the development of safer and more scalable autonomous driving systems by enabling high-fidelity LiDAR scene generation at both spatial and temporal levels. Its capability to synthesize realistic 4D LiDAR sequences can substantially reduce the cost of data collection and annotation, particularly for safety-critical or rare scenarios such as adverse weather, long-tail object categories, and hazardous corner cases. This can accelerate the training and benchmarking of perception models, facilitate re-

search in uncertainty estimation, and alleviate the heavy dependency on real-world data collection, which often poses privacy, safety, and logistical challenges.

Moreover, U4D can support simulation platforms and digital-twin systems by offering controllable, diverse, and uncertainty-aware scene generation. These synthetic environments can improve reproducibility, broaden research access, and lower the entry barrier for institutions with limited resources, thereby fostering a more inclusive and equitable autonomous driving research ecosystem.

Nonetheless, as with other generative frameworks, misuse is possible. Synthetic LiDAR data could, in principle, be used to create deceptive or manipulated sensor recordings. We therefore encourage responsible and transparent use of generative models and recommend deploying proper verification and auditing mechanisms to mitigate unintended or malicious misuse.

D.2. Potential Limitations

Despite the strong performance of U4D, several limitations remain and point toward directions for future improvement.

Scene Diversity and Rare Case Modeling. The generative capability of U4D is inherently tied to the data distribution it is trained on. While it performs robustly on common driving scenes, it may struggle to accurately reproduce extremely rare events or highly complex environments that are sparsely represented in the training set. Capturing such tail scenarios may require more diverse datasets or the integration of additional priors.

Computational Cost. Although U4D adopts an efficient architecture, generating long-range temporally consistent 4D LiDAR sequences remains computationally expensive. The two-stage diffusion process requires considerable GPU resources for training, and real-time generation is still challenging for large-scale or on-vehicle deployment.

Limited Generation Horizon. While U4D excels at modeling short-term temporal dynamics, its performance degrades when generating sequences longer than approximately 10 frames. This limitation mainly arises from the accumulation of stochastic errors during iterative denoising, which becomes more pronounced in the range-image space where generation is performed in a pixel-level manner. Small inconsistencies in the latent features can be amplified during decoding back into point clouds, leading to geometric drift, motion inconsistency, or structural artifacts over longer horizons. Developing more stable latent representations, stronger temporal constraints, or hierarchical generation strategies could alleviate this limitation.

E. Public Resources Used

In this section, we acknowledge the use of the following public resources, during the course of this work.

E.1. Public Codebase Used

We acknowledge the use of the following public codebase, during the course of this work:

- MMEngine[2] ..... Apache License 2.0
- MMCV[3] ..... Apache License 2.0
- MMDetection[4] ..... Apache License 2.0
- MMDetection3D[5] ..... Apache License 2.0
- OpenPCDet[6] ..... Apache License 2.0

E.2. Public Datasets Used

We acknowledge the use of the following public datasets, during the course of this work:

- nuScenes[7] ..... CC BY-NC-SA 4.0
- SemanticKITTI[8] ..... CC BY-NC-SA 4.0

E.3. Public Implementations Used

We acknowledge the use of the following implementations, during the course of this work:

- pytorch[9] ..... BSD License
- nuscenes-devkit[10] ..... Apache License 2.0
- r2dm[11] ..... MIT License
- lidarcrafter[12] ..... MIT License
- Open3D[13] ..... MIT License
- torchsparse[14] ..... MIT License
- LiMoE[15] ..... Apache License 2.0
- Vista[16] ..... Apache License 2.0

[2] https://github.com/open-mmlab/mmengine.
[3] https://github.com/open-mmlab/mmcv.
[4] https://github.com/open-mmlab/mmdetection.
[5] https://github.com/open-mmlab/mmdetection3d.
[6] https://github.com/open-mmlab/OpenPCDet.
[7] https://www.nuscenes.org/nuscenes.
[8] https://semantic-kitti.org.
[9] https://github.com/pytorch/pytorch.
[10] https://github.com/nutonomy/nuscenes-devkit.
[11] https://github.com/kazuto1011/r2dm.
[12] https://github.com/worldbench/lidarcrafter.
[13] https://github.com/isl-org/Open3D.
[14] https://github.com/mit-han-lab/torchsparse.
[15] https://github.com/Xiangxu-0103/LiMoE.
[16] https://github.com/OpenDriveLab/Vista.

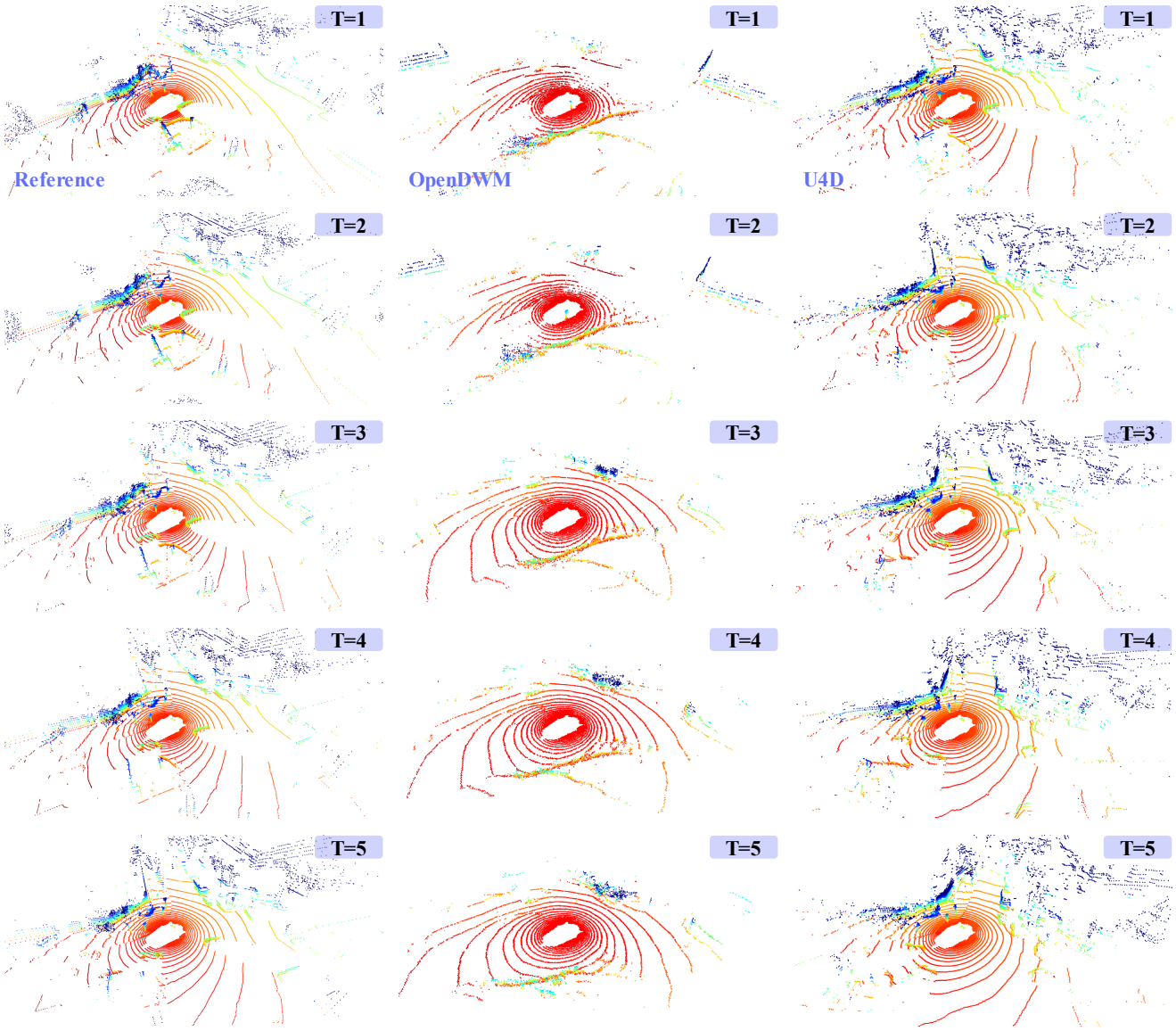

Figure A. Qualitative results of **sequence point cloud generation** on the *nuScenes* dataset [3]. U4D preserves both geometric fidelity and temporal consistency, producing sequences most similar to the reference. It reliably reconstructs distant, sparse regions and captures dynamic objects across frames, maintaining coherent structure and motion. Frames are shown in temporal order from top to bottom. The colors are rendered based on the height information of the point cloud. Best viewed in zoom.

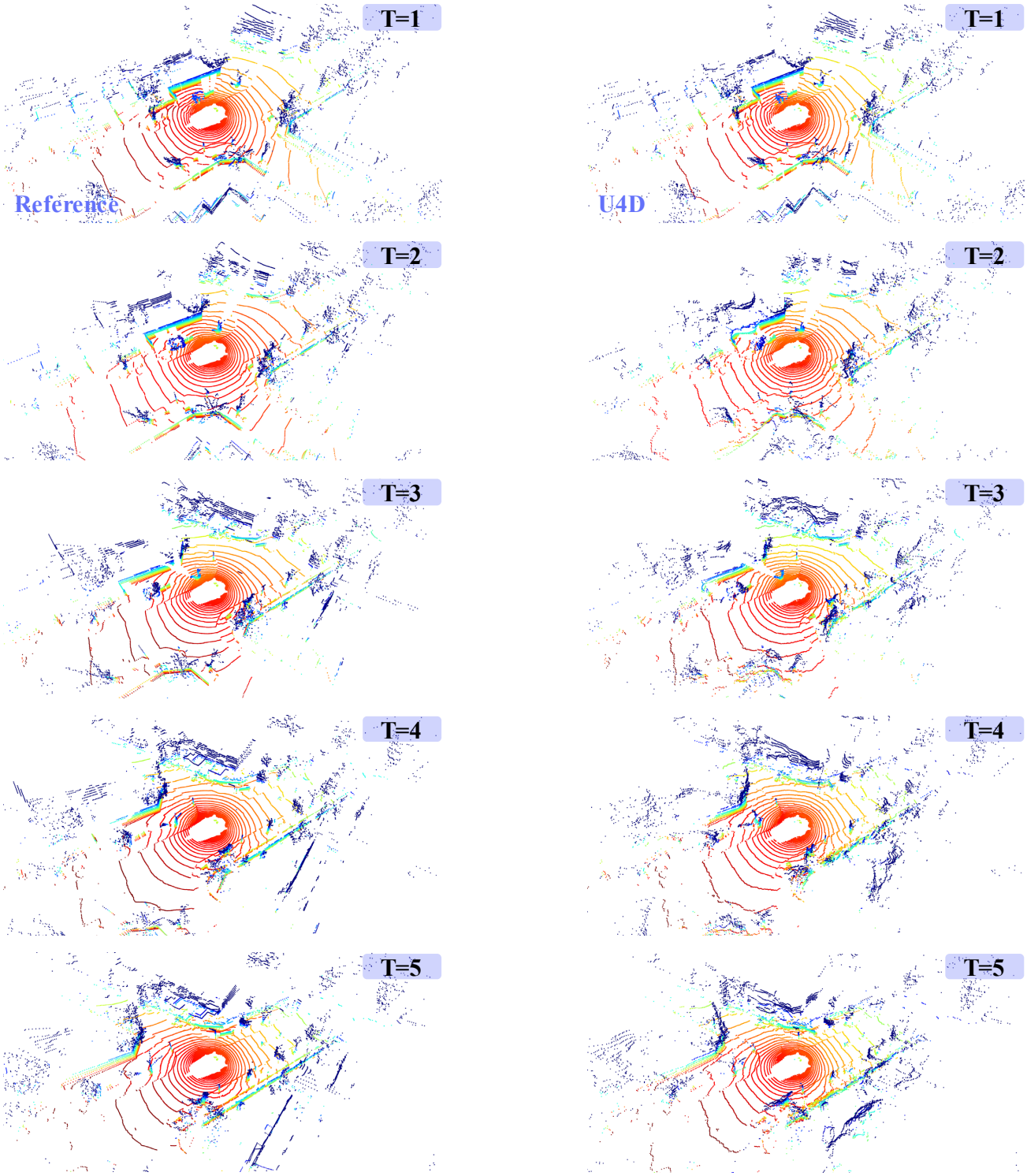

Figure B. **Future scene prediction** on the *nuScenes* dataset [3]. Given only the first frame as input, U4D can predict plausible future LiDAR frames that maintain both geometric fidelity and temporal consistency. The model successfully captures object motion, scene evolution, and structural continuity across time. Frames are shown in temporal order from top to bottom. The colors are rendered based on the height information of the point cloud. Best viewed in zoom.

References

[1] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. In *IEEE/CVF International Conference on Computer Vision*, pages 9297–9307, 2019. 1, 2, 3

[2] Paul J Besl and Neil D McKay. Method for registration of 3-D shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, pages 586–606, 1992. 3

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020. 1, 2, 4, 6, 7

[4] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2

[5] Lingdong Kong, Youquan Liu, Runnan Chen, Yuexin Ma, Xinge Zhu, Yikang Li, Yuenan Hou, Yu Qiao, and Ziwei Liu. Rethinking range view representation for LiDAR segmentation. In *IEEE/CVF International Conference on Computer Vision*, pages 228–240, 2023. 1

[6] Lingdong Kong, Youquan Liu, Xin Li, Runnan Chen, Wenwei Zhang, Jiawei Ren, Liang Pan, Kai Chen, and Ziwei Liu. Robo3D: Towards robust and reliable 3D perception against corruptions. In *IEEE/CVF International Conference on Computer Vision*, pages 19994–20006, 2023. 1

[7] Ao Liang, Youquan Liu, Yu Yang, Dongyue Lu, Linfeng Li, et al. LiDARCrafter: Dynamic 4D world modeling from LiDAR sequences. In *AAAI Conference on Artificial Intelligence*, pages 18406–18414, 2025. 2, 3

[8] Yiyi Liao, Jun Xie, and Andreas Geiger. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2D and 3D. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3292–3310, 2023. 3

[9] Youquan Liu et al. La La LiDAR: Large-scale layout generation from LiDAR data. In *AAAI Conference on Artificial Intelligence*, pages 7377–7385, 2026. 2

[10] Zhao Liu. Super convergence cosine annealing with warm-up learning rate. In *International Conference on Artificial Intelligence, Big Data and Algorithms*, pages 1–7, 2022. 2

[11] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 2

[12] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. RangeNet++: Fast and accurate LiDAR semantic segmentation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4213–4220, 2019. 1, 2, 4

[13] Kazuto Nakashima and Ryo Kurazume. LiDAR data synthesis with denoising diffusion probabilistic models. In *IEEE International Conference on Robotics and Automation*, pages 14724–14731, 2024. 1, 2, 3

[14] Jingcheng Ni, Yuxin Guo, Yichen Liu, Rui Chen, Lewei Lu, and Zehuan Wu. OpenDWM: Open driving world models. https://github.com/SenseTime-FVG/OpenDWM, 2025. 4

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019. 2

[16] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 2

[17] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*, pages 36479–36494, 2022. 1

[18] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. 2

[19] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3D architectures with sparse point-voxel convolution. In *European Conference on Computer Vision*, pages 685–702, 2020. 4

[20] Xiang Xu, Lingdong Kong, Hui Shuai, and Qingshan Liu. FRNet: Frustum-range networks for scalable LiDAR segmentation. *IEEE Transactions on Image Processing*, 34: 2173–2186, 2025. 4

[21] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3D convolution networks for LiDAR segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9939–9948, 2021. 4

[22] Vlas Zyrianov, Xiyue Zhu, and Shenlong Wang. Learning to generate realistic LiDAR point clouds. In *European Conference on Computer Vision*, pages 17–35, 2022. 3