CRISP: Fixing Flying Pixels in Latent LiDAR Generation via Diffusion Decoding

Andrea Ceron[1,2] **Michael Schmidt**[2] **Alvaro Marcos-Ramiro**[2]

Sebastian Schmidt[1,2] **Benjamin Busam**[1]

[1]**TU München** [2]**BMW AG**

{andrea.ceron, sebastian95.schmidt, b.busam}@tum.de

{Michael.SE.Schmidt, Alvaro.Marcos-Ramiro}@bmw.de

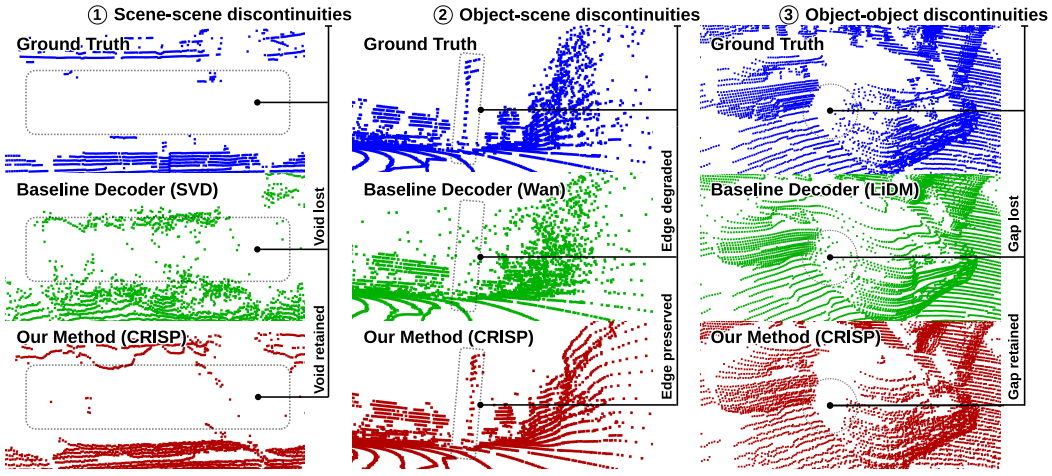

Figure 1: Qualitative comparison across three frozen latent backbones. Blue denotes the ground-truth point cloud, green the inherited VAE decoder, and red our method (CRISP). The three columns show one failure per backbone: SVD, Wan2.1, and LiDM. Across all cases, CRISP suppresses flying pixels and boundary-bridging artifacts that persist in the baseline reconstructions, yielding cleaner geometry and sharper depth contours.

Abstract

Latent LiDAR pipelines suffer from flying pixels: convolutional VAEs blur sharp radial depth discontinuities, yielding edge depths that back-project to points floating between surfaces. We identify this as a major, directly correctable decoder bottleneck and introduce **CRISP**: a pixel-space diffusion decoder with a backbone-agnostic latent adapter, DiT-based denoiser, and support mask predictor. CRISP replaces video-VAE and LiDAR-native decoders alike while keeping the encoder and latent generator fixed. Across KITTI-360, SemanticKITTI, and nuScenes, replacing only the decoder reduces FSVD/FPVD by 50.4% on average across frozen backbones; for generic video VAEs, the reductions reach 71%/74%. On the LiDAR-native LiDM backbone, FRID drops by 71%, with the largest gains at depth discontinuities. In a pretrained LiDM world model, the same zero-shot replacement improves FSVD by 15.5%, narrowing the sim-to-real gap. Code and checkpoints will be released on the Project Page.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

LiDAR provides a geometric backbone for autonomous driving, underpinning scene perception, simulation, and safety-relevant planning [54, 61, 69]. As these systems scale, synthetic LiDAR has become essential for coverage expansion and rare-scenario generation [20], and improvements in reconstruction fidelity translate directly to more trustworthy simulation for the safety-relevant perception and planning pipelines that depend on it. Modern pipelines address this by compressing LiDAR into a compact latent and generating within that space [43, 42, 17, 64], many of which decode through convolutional VAE decoders. These decoders fall into two families: general-purpose video VAEs [3, 53] repurposed for LiDAR [64], and LiDAR-native autoencoders [42] that, like range-image diffusion models [70, 38], use circular convolutions to respect the sensor's 360° azimuthal sweep.

The consequence is a concrete and recurring artifact: *flying pixels* [59, 31]. Convolutional VAE decoders smooth across the sharp depth contours that define LiDAR geometry, emitting interpolated depth at object edges: a leakage problem so recognized that LiDM resorts to flat convolutional kernels specifically to limit it [42]. Even this LiDAR-native design, with 1×4 kernels and horizontal-only resampling in its outermost curve-wise blocks and circular padding that closes the azimuthal seam, still convolves along each scan row and thus across radial depth discontinuities. When the blurred range map is back-projected to 3D, those interpolated boundary pixels become floating points anchored to neither surface [59], clustering at object boundaries and thin structures, and propagating into downstream perception [31].

Pixel-Perfect Depth [59] provides a useful decoder-level diagnosis for geometric prediction: latent decoding can corrupt sharp structure even when the latent representation is informative. We ask whether this bottleneck becomes even more consequential for LiDAR, where range maps are sparse, anisotropic, and subject to explicit missing returns, making decoder oversmoothing especially damaging after back-projection [35]. The LiDAR generation literature has meanwhile advanced through better latents, conditioning, and pretrained or external priors [42, 57, 46, 64], leaving one question that this paper answers directly: *is the decoder itself a bottleneck, and can replacing it alone improve latent LiDAR reconstruction?*

We introduce **CRISP**, a plug-in diffusion decoder that directly addresses this question by replacing the decoder alone, producing crisp, geometrically faithful LiDAR reconstructions from any frozen latent backbone while preserving sharp depth boundaries and suppressing the flying-pixel artifacts that convolutional decoders introduce at depth discontinuities.

Our contributions are:

- We identify convolutional VAE decoders as a major geometry bottleneck causing oversmooth LiDAR maps with flying pixels at depth discontinuities, and show that this bottleneck is correctable: replacing the decoder alone recovers the boundary geometry that the frozen latent already carries.
- **CRISP** adapts pixel-space diffusion decoding [59] to the sparse, support-incomplete range-map regime LiDAR requires. It includes a backbone-agnostic latent adapter, a DiT denoiser with early fusion for geometry-first denoising, and a support mask predictor for sparse valid-return modeling.
- The gains are significant. Replacing only the decoder reduces FSVD/FPVD by 71%/74% for generic video VAEs and FRID by 71% for the LiDAR-native backbone under frozen encoders, with edge F-score improving 107% on average; inside a frozen LiDM [42] world model, the same zero-shot swap improves FSVD by 15.5% and FRID by 4.0%, showing the bottleneck is actionable without retraining.

2 Related Work

LiDAR synthesis has progressed from sensor-level simulation to learned range-view and latent generative models. Early work improved realism with real-world assets, ray casting, and learned sensor deviations [34], while deep generative models synthesized scans as structured 2D point maps [4]. Subsequent methods exploited spinning-LiDAR image geometry more directly: LiDARGen uses stochastic denoising for equirectangular LiDAR [70], UltraLiDAR learns compact discrete representations for completion and generation [58], and R2DM adapts DDPMs to range/reflectance

images [38]. RangeLDM then moves this line into latent diffusion through range-view VAEs for efficient, realistic generation [17]. This motivates our setting: modern LiDAR generators increasingly depend on compressed range-map latents, yet leave the decoder largely unexamined and fixed.

Most recent LiDAR generation work improves latent modeling, conditioning, or the use of pretrained priors rather than the decoder itself. LiDM [42] established a latent diffusion pipeline built on a convolutional VAE for LiDAR range maps. SG-LDM [57] adds latent semantic alignment for semantic-to-LiDAR synthesis while keeping a standard 2D diffusion/autoencoding setup. R3DPA [46] improves range-image generation through RGB-pretrained priors and 3D representation alignment while retaining the range-VAE paradigm. UniDriveDreamer [64] adapts a general-purpose pretrained VAE to LiDAR data through LiDAR-specific fine-tuning and cross-modal latent anchoring for joint RGB-LiDAR generation. Across these lines, the main advances come from *better latents, conditioning, and priors,* not from replacing the convolutional decoder. Circular padding, as in LiDM and R2DM, resolves the azimuthal seam but not the *radial* discontinuities that cause flying pixels: replacing only the LiDM decoder still lowers edge CD by 48.8% and FRID by 71.4% (Tables 2, 1).

A complementary line tackles geometry after or beyond decoding. L3DR [31] attributes artifacts such as depth bleeding and wavy surfaces to range-view generation and corrects them with a post-hoc 3D residual rectifier, leaving the latent diffusion/decoding stack in place, while LidarDM [71] shifts the problem to 4D world generation and physics-based raycasting, prioritizing controllability and temporal coherence rather than decoder redesign. Both are complementary, but they *do not isolate the decoder as a bottleneck*; our focus is precisely that stage, replacing the convolutional decoder itself, inherited or LiDAR-native, with a plug-in pixel-space diffusion decoder tailored to sparse LiDAR range maps.

LiDAR makes the decoder bottleneck strictly harder than in monocular depth: range maps are sparse, anisotropic, and subject to explicit missing returns [35], so decoder-induced smoothing is more damaging, not less. Pixel-Perfect Depth [59] shows that VAE compression degrades edge sharpness even when reconstructing *ground-truth* depth, confirming latent decoding as a geometric bottleneck independent of the generative model. It is our closest architectural relative and also denoises in pixel space with cascaded diffusion transformers [39], but it conditions on a persistent clean RGB image rather than a frozen latent, predicts dense depth without a support branch, and outputs a single depth map rather than depth composed with a validity mask; ablating the first two differences degrades CRISP measurably (Table 14).

PiD [32] and ϵ-VAE [65] replace convolutional decoders with diffusion decoders for natural images, where adding structure or reconstructing stochastically is acceptable; LiDAR instead requires per-frame faithfulness, so CRISP regresses depth against ground truth on valid pixels rather than sampling freely. RAE [66] places the bottleneck at the other end: it replaces the VAE encoder with a frozen representation encoder, which requires a new decoder and a retrained generator. CRISP places it at the decoder, so it keeps both encoder and latent generator frozen, which enables a zero-shot decoder swap within a frozen LiDM world model (Table 3).

CRISP is designed around the decoder failure mode as it manifests in LiDAR, where the consequences after back-projection are more severe; full numbers for all three contrasts, additional coverage of architecture-level structural responses, downstream applications of decoder quality, and the pixel-space diffusion lineage are in Appendix A.

3 Method

Convolutional VAE decoders, whether repurposed video decoders or LiDAR-native, smooth over the sharp support discontinuities that define LiDAR geometry, generating flying pixels at object boundaries that persist independent of latent quality. To address this directly, CRISP introduces a two-branch pixel-space reconstruction module tailored to LiDAR geometry, even when keeping the pretrained encoder frozen. Given an input LiDAR range map, the encoder produces a latent representation shared by both branches (Fig. 2). A *latent adapter* (Sec. 3.2) converts the encoder output into conditioning tokens for the denoising transformer. The first reconstruction branch is a *DiT-powered pixel diffusion decoder* (Secs. 3.3, 3.5) that reconstructs a dense depth map $\hat{y}_d$ by denoising a Gaussian noise sample conditioned on the adapted latent tokens [45]. Conditioned on both z and $\hat{y}_d$, a *support mask predictor* (Secs. 3.4, 3.6) then identifies which pixels contain valid LiDAR returns. The final sparse LiDAR output is $\hat{y} = \hat{m} \odot \hat{y}_d - (1 - \hat{m})$.

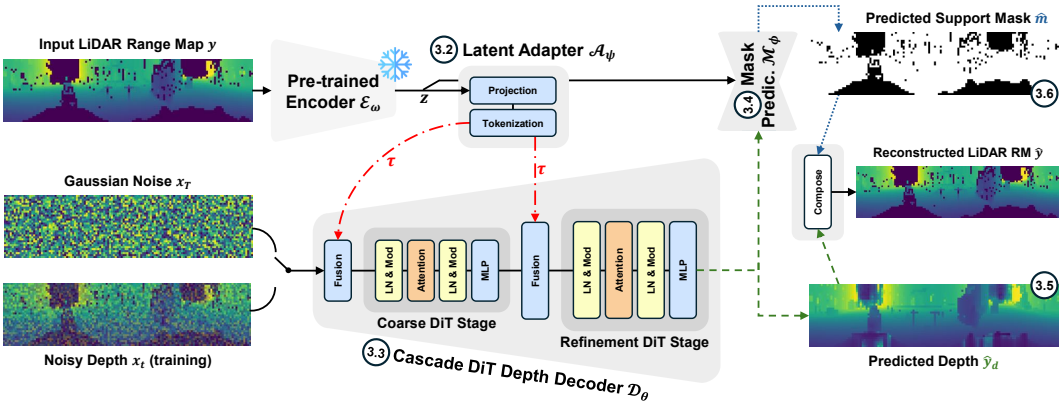

Figure 2: Architecture of CRISP. *Left:* the frozen encoder $\mathcal{E}_\omega$ maps the input range map y to latent z. *Top:* the latent adapter $\mathcal{A}_\psi$ projects and tokenizes z into conditioning tokens τ; subsequently, the support mask predictor $\mathcal{M}_\phi$ takes z and $\hat{y}_d$ to produce the binary mask $\hat{m}$. *Centre:* the cascade DiT decoder $\mathcal{D}_\theta$ denoises x_t on τ at two fusion points (red dashed arrows), yielding the dense depth $\hat{y}_d$. *Right:* $\hat{m}$ and $\hat{y}_d$ are composed as $\hat{y} = \hat{m} \odot \hat{y}_d - (1 - \hat{m})$ to produce the final LiDAR range map.

3.1 Encoding Setup

Let $y \in \mathbb{R}^{H \times W}$ be a LiDAR depth image in cylindrical range-map form [55] and $m \in \{0,1\}^{H \times W}$ the corresponding binary mask, where $m_{ij} = 1$ indicates a valid LiDAR return at pixel (i,j). Following prior range-map LiDAR work [42, 38, 60, 70, 22], we operate in a log-normalised depth representation (Appendix C.1, Eqs. 18-19), mapping $\tilde{y} \in [-1,1]^{H \times W}$ to allocate more resolution to near-range structure [55]. A pretrained encoder $\mathcal{E}_\omega$ maps $\tilde{y}$ to a latent tensor $z = \mathcal{E}_\omega(\tilde{y})$ [21, 17].

3.2 Latent Adapter

CRISP supports heterogeneous latent backbones, including LiDAR-native and general-purpose video VAEs (e.g. LiDM [42], SVD [3], or Wan2.1 [53]). Since these encoders produce latents with different channel counts and spatial resolutions, the adapter $\mathcal{A}_\psi$ maps them to a common token sequence:

$$\tau = \mathcal{A}_\psi(z) = \mathbf{W}_s \, \mathrm{Flat}(\mathbf{W}_c \cdot z) \,, \tag{1}$$

where $\mathbf{W}_c$ is a pointwise channel projection (applied independently at each spatial location) and $\mathbf{W}_s$ reduces the flattened spatial token count to the DiT conditioning length. This separates the decoder from the latent geometry of any specific backbone.

3.3 DiT-Powered Pixel Diffusion Decoder

The denoiser $\mathcal{D}_\theta$ is a Diffusion Transformer (DiT) operating directly in LiDAR pixel space [39, 59]. We use a standard Gaussian forward process and velocity parameterization [16, 48, 45], where α_t and σ_t are the signal and noise coefficients of the diffusion schedule ($\alpha_t^2 + \sigma_t^2 = 1$):

$$x_t = \alpha_t \tilde{y} + \sigma_t \epsilon, \qquad v^\star \coloneqq \alpha_t \epsilon - \sigma_t \tilde{y}, \qquad \hat{v} \coloneqq \mathcal{D}_\theta(x_t, \tau, t), \tag{2}$$

with $\epsilon \sim \mathcal{N}(0, I)$, $v^\star$ the ground-truth velocity target and $\hat{v}$ the predicted velocity. The dense depth estimate $\hat{y}_d$ is recovered as

$$\hat{y}_d = \alpha_t x_t - \sigma_t \hat{v}. \tag{3}$$

Early and mid-level latent fusion. The key architectural change is how latent information enters the denoiser. Unlike monocular depth diffusion, where a guidance image provides clean structure throughout the network [59], our decoder starts from pure Gaussian noise. We therefore inject latent tokens both immediately after patch embedding and again at the network midpoint:

$$h_0 = \mathrm{Fuse}_{\mathrm{early}}(\mathrm{Tok}(x_t), \tau), \qquad h_{K/2} \leftarrow \mathrm{Fuse}_{\mathrm{mid}}\left(\mathrm{Block}_{1:K/2}(h_0, t), \tau\right), \tag{4}$$

after which the remaining DiT blocks predict $\hat{v}$ (Eq. 2). Each fusion module concatenates range-map tokens and latent tokens and projects them back to the hidden dimension. This early conditioning is critical because the denoiser must recover global scene layout before refining local geometry [59]. Full architectural specifications are given in Appendix B.1.

3.4 Support Mask Predictor

LiDAR range maps contain two coupled signals: the returned range values and the binary support indicating where the sensor received a valid return. We follow the standard ray-drop formulation (Appendix A), but use the support predictor as a *latent-conditioned branch of the plug-in decoder*. Unlike standalone ray-drop heads, $\mathcal{M}_\phi$ operates directly on the frozen encoder latent z and predicts the valid-return support used to sparsify the decoded range map.

The branch upsamples z to the range-map resolution and outputs a logit field r_ϕ, whose sigmoid defines a Bernoulli probability map:

$$r_\phi = \mathcal{M}_\phi(z, \hat{y}_d), \qquad \pi_\phi = \sigma(r_\phi), \qquad q_\phi(\hat{m} \mid z, \hat{y}_d) = \prod_{i,j} \mathrm{Bernoulli}(\hat{m}_{ij};\, \pi_{\phi,ij})\,. \tag{5}$$

During training, support losses are applied to r_ϕ and π_ϕ against the ground-truth mask m. At inference, the Bernoulli field yields a binary support mask that sparsifies the dense depth:

$$\hat{m} \sim q_\phi(\hat{m} \mid z, \hat{y}_d), \qquad \hat{y} = \hat{m} \odot \hat{y}_d - (1 - \hat{m}). \tag{6}$$

Confident regions ($\pi_{\phi,ij} \approx 1$ or ≈ 0) are effectively deterministic, while ambiguous structures such as vegetation and thin objects can produce stochastic ray-drop variation across samples. The U-Net architecture and dual-branch conditioning are detailed in Appendix B.2.

3.5 Geometry-Aware Diffusion Objective

Both branches share one structure: a *region objective* on valid pixels, *spatial regularisers* of first and second differential order, and *optimisation machinery* that shapes training without modelling a separate failure mode (per-term failure mode and provenance in Appendix B.5).

The depth branch minimizes the following composite objective:

$$\mathcal{L}_{\mathrm{depth}} = \mathcal{L}_{\mathrm{vel}} + \mathcal{L}_{\mathrm{der}} + \lambda_{\mathrm{grad}}\mathcal{L}_{\mathrm{grad}} + \mathcal{L}_{\mathrm{vel,raw}} + \lambda_{x0}\mathcal{L}_{x0} \qquad \lambda_{\mathrm{grad}}, \lambda_{x0} \geq 0. \tag{7}$$

The region objective $\mathcal{L}_{\mathrm{vel}}$ is a boundary-aware velocity regression on valid pixels (importance-weighted). The spatial regularisers are $\mathcal{L}_{\mathrm{der}}$, which penalises finite-difference residuals along both spatial axes, and $\mathcal{L}_{\mathrm{grad}}$, a multi-scale first-order gradient loss on $\hat{y}_d$ from LiDM [42]. The optimisation machinery is $\mathcal{L}_{\mathrm{vel,raw}}$, the unweighted counterpart of $\mathcal{L}_{\mathrm{vel}}$ providing a stable baseline signal, and $\mathcal{L}_{x0}$, a direct ℓ_1 penalty on $\hat{y}_d$ concentrated at low noise levels. Definitions are in Appendix B.3; weights are in Table 13.

Valid-only structure-aware importance weighting. LiDAR errors are not equally costly: mistakes at depth discontinuities back-project to large 3D deviations, while errors in flat regions are comparatively harmless. A naïve Sobel or Laplacian on a zero-filled sparse map produces spurious responses at valid/void borders; we avoid this via validity-normalised filtering [23, 7, 51]:

$$\bar{y} = \frac{G_\sigma * (m \odot \tilde{y})}{G_\sigma * m + \epsilon}, \tag{8}$$

where $*$ denotes convolution, $\odot$ element-wise multiplication, and G_σ a Gaussian kernel (5×5 kernel, σ=2.0). We compute $h = |\tilde{y} - \bar{y}|$ and map it to a bounded importance weight:

$$w = m \odot \Big(1 + (\lambda_{\max} - 1)\,\mathrm{Norm}(h)\Big), \quad \lambda_{\max} = 1.5. \tag{9}$$

Flat valid regions receive weight 1; pixels near true depth discontinuities receive weight up to $\lambda_{\max}$. The boundary-aware loss $\mathcal{L}_{\mathrm{vel}}$ reuses Eq. 13 with m replaced by w (Appendix B.3).

3.6 Support Mask Objective

The mask branch minimizes:

$$\mathcal{L}_{\mathrm{mask}} = \lambda_{\mathrm{focal}}\mathcal{L}_{\mathrm{focal}} + \lambda_{\mathrm{dice}}\mathcal{L}_{\mathrm{dice}} + \lambda_{\mathrm{edge}}\mathcal{L}_{\mathrm{edge}} + \lambda_{\mathrm{lap}}\mathcal{L}_{\mathrm{lap}} + \lambda_{\mathrm{hard}}\mathcal{L}_{\mathrm{hard}} + \lambda_{\mathrm{ds}}\mathcal{L}_{\mathrm{ds}}, \qquad \lambda_* \geq 0. \tag{10}$$

Valid LiDAR support is a multi-scale prediction problem: the mask must capture coarse global structure, sharp foreground-background boundaries, thin valid returns, and isolated single-pixel

ray-drop holes. No single loss handles all four failure modes. As the region objective, focal BCE [29] addresses class imbalance and focuses the gradient on hard boundary and ray-drop regions, and soft Dice [37] optimizes overlap directly, independent of pixel counts. As spatial regularisers, a first-order edge term enforces boundary alignment [18], while a second-order Laplacian term [33, 68] captures isolated ray-drop events invisible to first-order penalties. As optimisation machinery, contextual hard-pixel mining [47] up-weights spatially fragile regions by target-neighborhood structure, and deep supervision [26, 40] at two intermediate resolutions ensures the gradient signal reaches shallow layers. Per-term failure modes and provenance are in Table 10, weights in Table 7. Per-branch ablations identify a seven-term reduced objective that drops the overlapping regulariser of each branch and the mask machinery: removing $\mathcal{L}_{\text{grad}}$ raises depth val L1 by only 2.9% on nuScenes and 0.1% on KITTI-family (Table 6), and Focal + Dice + Laplacian retains 96.7% of the six-term IoU gain over focal-only on LiDM/KITTI-family (Table 9).

3.7 Full Objective and Encoder Adaptation

The default CRISP freezes the encoder and trains only the plug-in decoder branches, and optimizes:

$$\mathcal{L} = \mathcal{L}_{\text{depth}} + \lambda_{\text{mask}}\mathcal{L}_{\text{mask}}, \qquad \lambda_{\text{mask}} \geq 0. \tag{11}$$

Weights are set once per training stage by equalising initial term magnitudes, a static simplification of GradNorm [6], and reused unchanged for every backbone and beam count. We further evaluate encoder-adapted SVD and Wan2.1 variants [3, 53], excluding LiDM since its encoder is LiDAR-native [42]. In the baseline setting, the original VAE architecture is adapted to single-channel LiDAR using the UniDriveDreamer [64] objective, without modification (Appendix C.3). In the CRISP setting, we instead unlock the encoder and jointly train it with the CRISP decoder, adding a KL regulariser to keep the encoder posterior close to the prior $p(z) = \mathcal{N}(0, I)$:

$$\mathcal{L} = \mathcal{L}_{\text{depth}} + \lambda_{\text{mask}}\mathcal{L}_{\text{mask}} + \lambda_{\text{KL}}\text{KL}(q_\omega(z \mid \tilde{y}) \,\|\, p(z)), \qquad \lambda_{\text{mask}}, \lambda_{\text{KL}} \geq 0 \tag{12}$$

4 Experimental Setup

Datasets and evaluation regimes. We evaluate on three standard automotive LiDAR benchmarks: KITTI-360 [28], SemanticKITTI [2], and nuScenes [5]. Following LiDM [42], KITTI-360 and SemanticKITTI are grouped as the KITTI family; nuScenes provides a larger-scale benchmark with lower vertical LiDAR resolution.

Latent backbones and comparison settings. We evaluate CRISP with three latent VAE backbones: LiDM, a LiDAR-native autoencoder with circular padding, evaluated on the KITTI-family setting [42]; and SVD and Wan2.1, two general-purpose video VAEs evaluated on nuScenes [3, 53]. We include Wan2.1 as a realistic stress test for inherited video-VAE decoders used in recent driving world models [64], not as an artificially weak baseline. Because SVD and Wan expect RGB input, we study two regimes for them: *frozen replacement* ($\times$**3**), where the encoder is fixed and the depth channel is replicated three times to match the pretrained (Appendix C.2); and *encoder adaptation* ($\times$**1**), where encoders are either independently or jointly finetuned with CRISP for LiDAR, and the first/last VAE layers are made single-channel (Appendix C.3). For LiDM, whose encoder natively accepts single-channel depth, we additionally evaluate *plug-and-play world-model decoding*, replacing the decoder inside a pretrained LiDM world model while keeping generated latents and latent dynamics unchanged.

Metrics. We evaluate perception realism, distributional fidelity, and paired 3D reconstruction. For perception quality, we report LiDM's Fréchet-style metrics: FRID, FSVD, and FPVD [42]. Analogous to FID [15], these compare real and reconstructed LiDAR in feature spaces: range-image features for FRID, sparse-volume features for FSVD, and point-volume features for FPVD. For distributional fidelity, we report Jensen-Shannon Divergence (JSD), Earth Mover's Distance (EMD), and Minimum Matching Distance (MMD), point-cloud generation metrics that compare occupancy statistics and nearest-set matching [1]. For paired geometry, we report Chamfer Distance (CD) and F-score@0.5m after back-projection to 3D [8, 50]. CD and EMD measure point-set alignment, while F-score summarizes precision and recall under a fixed distance threshold.

Table 1: Statistical and perceptual reconstruction results. *base.* uses the original decoder; *ours* replaces it with CRISP under the same encoder setting. Best values within each backbone pair are bold. Mean over five runs; parenthesized digits give std at mean precision (e.g. $0.123\,(45) = 0.123 \pm 0.045$; (00) denotes std <0.001 at that precision). FRID requires 64-beam range-image features and is reported only for the KITTI-family (LiDM) setting. ↓/↑: lower/higher is better. **Same protocol for all tables.**

Setting	Dec.	Statistical			Perceptual		
		JSD↓	EMD↓	MMD↓ $\times 10^{-4}$	FSVD↓	FPVD↓	FRID↓
Frozen backbones							
SVD $\times 3$	base.	0.18961 (00)	0.25550 (00)	2.6946 (44)	156.487 (23)	153.599 (20)	–
	ours	**0.07145 (02)**	**0.07653 (01)**	**2.2505 (20)**	**40.244 (99)**	**36.847 (75)**	–
Wan $\times 3$	base.	0.16373 (00)	0.11840 (00)	2.2505 (00)	100.917 (00)	113.381 (00)	–
	ours	**0.07326 (02)**	**0.08871 (03)**	**1.7193 (41)**	**32.534 (40)**	**31.065 (29)**	–
LiDM	base.	0.04975 (00)	0.53762 (00)	0.7721 (00)	20.204 (00)	**16.154 (00)**	2.2351 (00)
	ours	**0.04902 (01)**	**0.51924 (24)**	**0.6964 (57)**	**16.693 (49)**	17.034 (30)	**0.6389 (19)**
Encoder-adapted backbones							
SVD $\times 1$	base.	0.15660 (04)	0.19354 (08)	2.538 (10)	**40.813 (64)**	**41.775 (55)**	–
	ours	**0.08316 (06)**	**0.13930 (03)**	**2.0734 (92)**	64.86 (19)	52.83 (15)	–
Wan $\times 1$	base.	**0.06962 (00)**	**0.04700 (00)**	1.3645 (00)	24.029 (00)	23.681 (00)	–
	ours	0.08244 (00)	0.14736 (00)	**1.1682 (00)**	**8.290 (21)**	**7.112 (18)**	–

Table 2: Geometric breakdown into whole-cloud, edge, and smooth sub-clouds. CD↓ and F-score@0.5m↑ are reported for each region.

Setting	Dec.	Whole		Edge		Smooth	
		CD↓	F↑	CD↓	F↑	CD↓	F↑
Frozen backbones							
SVD $\times 3$	base.	1.33107 (00)	0.67804 (00)	8.691 (00)	0.0952 (00)	1.18399 (03)	0.73018 (00)
	ours	**0.63967 (17)**	**0.80465 (02)**	**7.364 (00)**	**0.2383 (00)**	**0.53709 (09)**	**0.82158 (02)**
Wan $\times 3$	base.	0.77027 (00)	0.73240 (00)	**5.645 (00)**	0.1038 (00)	0.59749 (00)	0.79303 (00)
	ours	**0.46453 (09)**	**0.82551 (01)**	5.861 (05)	**0.2848 (02)**	**0.39544 (06)**	**0.84166 (01)**
LiDM	base.	0.16026 (00)	0.93935 (00)	7.913 (00)	0.3781 (00)	0.12358 (00)	0.94477 (00)
	ours	**0.14245 (27)**	**0.94323 (01)**	**4.052 (10)**	**0.4792 (02)**	**0.11032 (17)**	**0.94901 (01)**
Encoder-adapted backbones							
SVD $\times 1$	base.	0.81622 (40)	0.72676 (07)	8.248 (03)	0.1151 (00)	0.67092 (38)	0.75422 (07)
	ours	**0.66981 (30)**	**0.81795 (05)**	**6.009 (01)**	**0.2082 (00)**	**0.56524 (15)**	**0.83863 (05)**
Wan $\times 1$	base.	0.30994 (00)	0.86691 (00)	6.150 (00)	0.2226 (00)	**0.24295 (00)**	0.88618 (00)
	ours	**0.29017 (01)**	**0.87341 (01)**	**2.485 (01)**	**0.4456 (01)**	0.24362 (03)	**0.89043 (01)**

Boundary-focused geometric analysis. Since CRISP targets decoder artifacts at depth discontinuities, we report CD and F-score not only on the full reconstructed point cloud but also on two ground-truth-defined subsets. Following the valid-only structure signal used for boundary-aware supervision in Sec. 3.5, we identify edge pixels by thresholding local depth discontinuities computed over valid LiDAR returns (by value 0.1). The *edge* subset contains valid pixels above this threshold, while the *smooth* subset contains the remaining valid pixels. Invalid/empty transitions are ignored, so missing returns are not counted as edges. Each subset is back-projected separately before evaluation. Full training and evaluation setup is provided in Appendix D. Asset licenses are listed in Appendix F.

5 Results

Because no single metric captures boundary geometry, we emphasize trends consistent across perceptual, paired-3D, and edge/smooth metrics, and report distributional regressions explicitly.

Frozen backbones: the decoder is the bottleneck. Replacing only the inherited decoder yields large improvements in the main perceptual and geometric metrics (Tables 1 and 2), with the only

Table 3: Plug-and-play decoding inside a frozen LiDM world model: statistical, perceptual and geometric quality. The latent generator and sampled latents are fixed; only the decoder changes. Values with a decimal point inside std parentheses are written explicitly at the displayed precision.

Decoder	Statistical			Perceptual		
	JSD↓	EMD↓	MMD↓ $\times 10^{-4}$	FSVD↓	FPVD↓	FRID↓
baseline	**0.21181 (87)**	2.453 (13)	3.980 (44)	37.70 (52)	28.85 (34)	137.81 (4.79)
ours	0.21436 (83)	**2.420 (13)**	**3.849 (64)**	**31.871 (34)**	**28.11 (18)**	**132.36 (4.59)**

Decoder	Whole Geometry		Edge Geometry		Smooth Geometry	
	CD↓	F↑	CD↓	F↑	CD↓	F↑
baseline	16.21 (23)	0.4484 (13)	96.41 (1.33)	**0.0486 (17)**	15.80 (22)	0.4489 (13)
ours	**15.85 (23)**	**0.4510 (13)**	**79.55 (1.09)**	0.0389 (14)	**15.53 (23)**	**0.4514 (13)**

perceptual exception being a small FPVD regression on the LiDM backbone. Averaged over the three frozen settings, FSVD decreases by 53% and FPVD by 48%; for the two general-purpose video VAEs, where the modality mismatch is greatest, these averages reach 71% and 74%, reflecting how severely a decoder trained on dense natural images fails when confronted with sparse, anisotropic LiDAR geometry (Figs. 5, 7). On the LiDAR-native LiDM backbone, FRID drops from 2.24 to 0.64 (71%) showing that even the LiDAR-native decoder suppresses structure already present in the latent (Fig. 9). Whole-cloud CD decreases by 34% on average (46% for generic VAEs, 11% for LiDM), the smaller LiDM gain consistent with a LiDAR-native encoder near its reconstruction ceiling, and the improvement scaling with modality gap confirms the decoder as a controllable bottleneck.

Geometric breakdown: boundary evidence. The sub-cloud breakdown is the sharpest test of the decoder-bottleneck hypothesis. Edge F-score improves by an average of 107% across all five settings versus 9% for the whole cloud; a roughly 12× contrast that localizes the failure mode: convolutional oversmoothing is not a diffuse reconstruction error but is concentrated at depth discontinuities, precisely where blurred transitions back-project to flying pixels. Edge Chamfer Distance decreases in four of five settings, with the largest reductions for LiDM and Wan ×1, confirming that CRISP improves boundary geometry rather than merely increasing local point density. Smooth-region Chamfer Distance decreases by 23% on average, with only a negligible Wan ×1 exception, confirming that the diffusion decoder does not trade flat-surface accuracy for boundary sharpness. The breakdown thus validates the hypothesis at a spatial granularity unavailable from whole-cloud metrics alone: prior decoders fail selectively at boundaries, and CRISP corrects that.

Encoder-adapted backbones. Wan ×1 shows compounding gains: FSVD −65%, FPVD −70%, edge CD −60%, edge F-score +100%, achieving the best perceptual scores of any setting in the study, because encoder adaptation and decoder replacement address complementary failure modes: the adapted encoder produces a LiDAR-appropriate latent that CRISP can decode without the modality-mismatch artifacts of the convolutional baseline (Fig. 8). SVD ×1 improves on all geometric metrics (whole-cloud CD −18%, edge F-score +81%, Fig. 6), but FSVD and FPVD regress. We attribute this to reconstruction overfitting: without a perceptual or distributional training signal, CRISP on SVD ×1 minimizes per-sample L1/L2 error at the cost of feature-space realism. Consistently, frozen SVD ×3 already outperforms adapted SVD ×1 on both perceptual metrics. Wan ×1 JSD and EMD similarly regress: we hypothesize that the co-adapted baseline implicitly calibrated its global valid-return density end-to-end, whereas CRISP's support predictor, trained with per-pixel classification losses, does not constrain total point count: a small density miscalibration is sufficient to inflate EMD and JSD without degrading the per-point geometry that FSVD and FPVD reflect.

Plug-and-play world-model decoding. Table 3 evaluates a zero-shot decoder swap inside a pretrained LiDM world model: the latent generator is frozen, and only the decoder is changed. This is the most demanding test: the world model's latent dynamics were optimised together with the original decoder, yet FSVD still decreases by 15.5%, FRID by 4.0%, whole-cloud CD by 2.2%, and edge CD by 17.5%. JSD increases marginally, and edge F-score decreases by 20%, consistent with the latent dynamics having implicitly compensated for the baseline decoder's oversmoothing: removing that compensation exposes a residual distribution mismatch that fine-tuning the world model alongside

Table 4: Decoder capacity and cost (inherited → CRISP). Params/FLOPs: batch 1, 5 Euler steps, native range-map resolution per setting. Decoder ms/frame: 8×H200, `torch.compile`, batch 25, 5 Euler steps, as Table 17.

Backbone	Decoder params		GFLOPs		Decoder ms/frame
	Inh. → CRISP	Ratio	Inh. → CRISP	Ratio	(Inh. → CRISP, ratio)
SVD ×3	63.6M → 516.0M	8.1×	376.6 → 1290.7	3.4×	1.49 → 3.92 (2.63×)
Wan ×3	73.3M → 516.1M	7.0×	352.6 → 1291.4	3.7×	9.71 → 3.84 (0.40×)
LiDM	8.6M → 516.0M	60.0×	119.0 → 2717.2	22.8×	1.39 → 23.06 (16.6×)

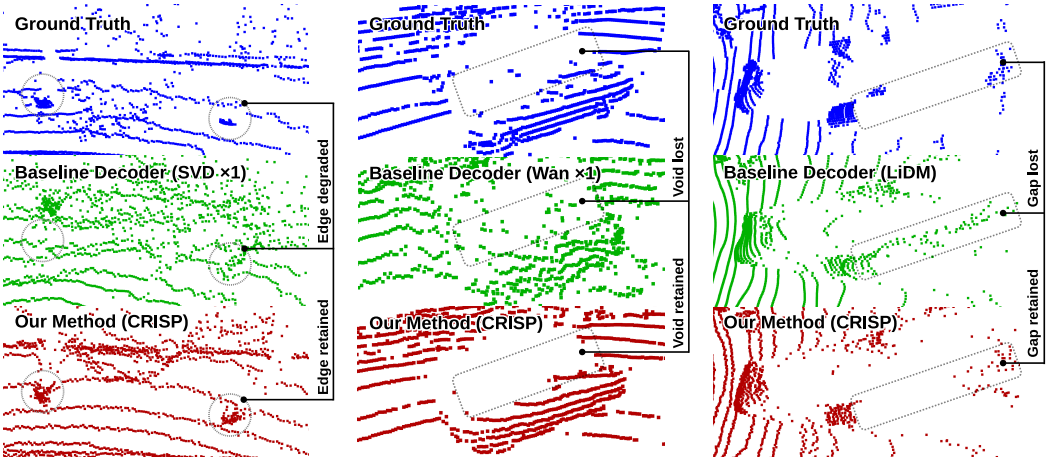

Figure 3: Qualitative LiDAR reconstruction results (colors as in Fig. 1). Columns show, left to right, failure modes CRISP addresses: lost narrow objects, incorrectly filled voids, and flying pixels.

CRISP would likely close. The primary geometry improvements nonetheless demonstrate that the decoder bottleneck is actionable in a deployed world model without any retraining (see Fig. 10).

Inference cost. Table 17 reports end-to-end throughput. CRISP on Wan2.1 is faster end-to-end than the original pipeline (86 fps vs. 57 fps) because its decoder (3.84 ms/frame) is 2.5× faster than the inherited video VAE decoder. For SVD and LiDM, CRISP's 5-step diffusion decoder runs in 3.9 and 23 ms/frame respectively, which can be further reduced via distillation [45] (see Appendix E, Fig. 4). Thus, despite CRISP's architectural complexity, system cost depends on the inherited backbone: on Wan ×3, it is faster and substantially more accurate than the original decoder.

Decoder capacity. CRISP (≈516M parameters) is 8.1×, 7.0× and 60.0× larger than the SVD, Wan and LiDM decoders it replaces (Table 4). A larger decoder could improve reconstruction through added capacity alone, independent of pixel-space diffusion decoding. Two controls show that this extra capacity does not by itself explain the gain. At fixed capacity, the step sweep alone moves SVD ×3 FSVD from 94.2 at 2 steps to 40.3 at 5 (Fig. 4); and the gain runs opposite to the added compute, as LiDM receives the most (60× parameters, 22.8× FLOPs) yet improves least, though partly because its baseline is already closest to its reconstruction ceiling (Appendix E).

Qualitative analysis. Figure 3 illustrates three distinct decoder failure modes, each suppressed by CRISP: boundary flying pixels (SVD ×1), spurious void infilling (Wan ×1), and depth-gap bridging at foreground-background transitions (LiDM). These correspond directly to the edge and smooth sub-cloud gains in Table 2. Extended per-setting comparisons are in Appendix G.

Ablations. Appendix E isolates each design choice (sampling steps, early fusion, the mask branch, the loss stack) on SVD ×3, with a per-term leave-one-out analysis (Appendices B.3–B.4) and cross-backbone replication on LiDM (Appendix E). Early latent fusion is necessary for decoding from pure noise: removing it increases whole-cloud CD by 15% and FSVD by 21% (Table 14). Modeling support with a dedicated mask branch, rather than diffusing over all pixels, improves edge F-score

Table 5: Per-component attribution (SVD $\times 3$, nuScenes, converged): share of the *Inherited→CRISP* gain from the decoder swap alone (*w/o mask*) vs. the mask branch added on top. *Inherited* from Tables 1–2; *w/o mask* and *CRISP* from Table 14.

Metric	Inherited	w/o mask	CRISP	Decoder swap	Mask branch
Whole CD ↓	1.331	0.660	0.639	97%	3%
FSVD ↓	156.5	42.6	40.3	98%	2%
JSD ↓	0.190	0.109	0.071	68%	32%
Edge F ↑	0.0952	0.193	0.238	68%	32%

by 23% and JSD by 35%, confirming that sparse LiDAR support should be predicted explicitly. The mask ablation further shows the loss stack improves IoU from 0.758 to 0.844, while adding latent conditioning to depth-only input raises converged IoU from 0.630 to 0.788 (Table 15). Thus, adapter fusion, explicit support branch, and mask supervision each addresses a distinct failure mode rather than merely increasing capacity. The no-mask ablation controls for point-density changes: simultaneous edge-F and edge-CD gains show better boundary placement, not density alone.

Per-component attribution. The decoder swap accounts for 68–98% of the SVD $\times 3$ gain and the mask branch for the rest, mainly on JSD and edge F-score (Table 5). The *w/o mask* proxy is not a clean module ablation and beats CRISP only on FPVD (35.993 vs. 36.782).

Limitations. CRISP inherits the range-map assumptions shared by most latent LiDAR pipelines [42, 38, 70]: it predicts a single depth channel and does not model remission. Although CRISP suppresses boundary-bridging artifacts and flying pixels, its diffusion-based denoising can still leave residual high-frequency noise in the decoded range maps, especially on smooth surfaces. Since CRISP decodes frames independently, these small artifacts may also appear as thin-structure flicker in multi-frame outputs, a limitation shared with single-frame range-view decoders. The decoder takes 3.9 ms/frame for SVD and 23 ms/frame for LiDM (Table 17), so latency remains a limitation for high-throughput downstream perception pipelines. End-to-end, CRISP is faster on Wan $\times 3$ (57→86 fps) but slower on LiDM (369→41 fps); distillation [45] is future work.

Faithfulness to the latent. A generative decoder could invent boundary geometry the latent does not support. Four measurements indicate that CRISP reads this geometry from the latent instead: edge F-score improves in all five settings and edge CD in four (Table 2); cross-seed edge F-score std is ≤ 0.0002 (Appendix D); removing the latent input from the mask predictor drops its IoU from 0.788 to 0.630, and removing early latent fusion raises CD from 0.985 to 1.134 (Tables 15, 14); and ungated diffusion over all pixels raises JSD from 0.071 to 0.109 (Table 14). Two residual limits remain: predictor precision is 0.862 (Table 15), and in the world model edge F-score falls while edge CD improves (Table 3), which indicates under-prediction rather than fabrication.

6 Conclusions

We showed that convolutional VAE decoders, both repurposed and LiDAR-native, are a concrete, correctable bottleneck in latent LiDAR pipelines. Replacing the decoder alone with CRISP recovers sharper geometry from frozen latents, improving edge F-score by 107% on average and reducing edge CD by 17.5% in a zero-shot swap inside a frozen LiDM world model, without retraining the latent generator. CRISP therefore offers a plug-in upgrade path for range-map latent LiDAR systems: the generative model can remain unchanged while decoded geometry becomes cleaner and less prone to flying-pixel artifacts. *Future work* includes remission prediction with multi-channel diffusion, density-aware support calibration, and temporal conditioning for flicker-free rollouts.

Acknowledgments and Disclosure of Funding

Funding in direct support of this work: salary, compute and research budget provided by BMW AG. Additional revenues related to this work: A. Ceron is employed by BMW AG as an industrial PhD candidate with TU München; M. Schmidt and A. Marcos-Ramiro are full-time employees of BMW AG; S. Schmidt became a full-time employee of BMW AG during this work.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pages 40–49. PMLR, 2018.

[2] 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 *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9297–9307, 2019.

[3] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[4] Lucas Caccia, Herke Van Hoof, Aaron Courville, and Joelle Pineau. Deep generative modeling of lidar data. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5034–5040. IEEE, 2019.

[5] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[6] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pages 794–803. PMLR, 2018.

[7] Abdelrahman Eldesokey, Michael Felsberg, Karl Holmquist, and Michael Persson. Uncertainty-aware cnns for depth completion: Uncertainty from beginning to end. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12014–12023, 2020.

[8] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[9] Tuo Feng, Wenguan Wang, and Yi Yang. A survey of world models for autonomous driving. *arXiv preprint arXiv:2501.11260*, 2025.

[10] Shenyuan Gao, Jiazhi Yang, Li Chen, Kashyap Chitta, Yihang Qiu, Andreas Geiger, Jun Zhang, and Hongyang Li. Vista: A generalizable driving world model with high fidelity and versatile controllability. *Advances in Neural Information Processing Systems*, 37:91560–91596, 2024.

[11] Thibault Groueix. ChamferDistancePytorch. `https://github.com/ThibaultGROUEIX/ChamferDistancePytorch`, 2017. MIT License.

[12] Hamed Haghighi, Amir Samadi, Mehrdad Dianati, Valentina Donzella, and Kurt Debattista. Taming transformers for realistic lidar point cloud generation. *arXiv preprint arXiv:2404.05505*, 2024.

[13] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[14] Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-key normalization for transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4246–4253, 2020.

[15] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[17] Qianjiang Hu, Zhimin Zhang, and Wei Hu. Rangeldm: Fast realistic lidar point cloud generation. In *European Conference on Computer Vision*, pages 115–135. Springer, 2024.

[18] Yu-Kai Huang, Tsung-Han Wu, Yueh-Cheng Liu, and Winston H Hsu. Indoor depth completion with boundary consistency and self-attention. In *Proceedings of the IEEE/CVF international conference on computer vision workshops*, pages 0–0, 2019.

[19] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9492–9502, 2024.

[20] Jongsuk Kim, Jaeyoung Lee, Gyojin Han, Dong-Jae Lee, Minki Jeong, and Junmo Kim. Synad: Enhancing real-world end-to-end autonomous driving models through synthetic data integration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 25197–25206, 2025.

[21] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[22] Ellington Kirby, Mickael Chen, Renaud Marlet, and Nermin Samet. Logen: Toward lidar object generation by point diffusion. *arXiv preprint arXiv:2412.07385*, 2024.

[23] Hans Knutsson and C-F Westin. Normalized and differential convolution. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 515–523. IEEE, 1993.

[24] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 228–240, 2023.

[25] Lingdong Kong, Wesley Yang, Jianbiao Mei, Youquan Liu, Ao Liang, Dekai Zhu, Dongyue Lu, Wei Yin, Xiaotao Hu, Mingkai Jia, et al. 3d and 4d world modeling: A survey. *arXiv preprint arXiv:2509.07996*, 2025.

[26] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial intelligence and statistics*, pages 562–570. Pmlr, 2015.

[27] Ao Liang, Lingdong Kong, Tianyi Yan, Hongsi Liu, Wesley Yang, Ziqi Huang, Wei Yin, Jialong Zuo, Yixuan Hu, Dekai Zhu, et al. Worldlens: Full-spectrum evaluations of driving world models in real world. *arXiv preprint arXiv:2512.10958*, 2025.

[28] 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, 2022.

[29] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[30] Jiuming Liu, Zheng Huang, Mengmeng Liu, Tianchen Deng, Francesco Nex, Hao Cheng, and Hesheng Wang. Topolidm: Topology-aware lidar diffusion models for interpretable and realistic lidar point cloud generation. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8180–8186. IEEE, 2025.

[31] Quan Liu, Xiaoqin Zhang, Ling Shao, and Shijian Lu. L3dr: 3d-aware lidar diffusion and rectification. *arXiv preprint arXiv:2602.19064*, 2026.

[32] Yifan Lu, Qi Wu, Jay Zhangjie Wu, Zian Wang, Huan Ling, Sanja Fidler, and Xuanchi Ren. Pid: Fast and high-resolution latent decoding with pixel diffusion. *arXiv preprint arXiv:2605.23902*, 2026.

[33] Fangchang Ma, Guilherme Venturelli Cavalheiro, and Sertac Karaman. Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera. In *2019 international conference on robotics and automation (ICRA)*, pages 3288–3295. IEEE, 2019.

[34] Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. Lidarsim: Realistic lidar simulation by leveraging the real world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11167–11176, 2020.

[35] Richard Marcus and Marc Stamminger. Physically based neural lidar resimulation. *arXiv preprint arXiv:2507.12489*, 2025.

[36] Andrea Matteazzi, Pascal Colling, Michael Arnold, and Dietmar Tutsch. Adverse weather conditions augmentation of lidar scenes with latent diffusion models. *arXiv preprint arXiv:2501.01761*, 2025.

[37] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. Ieee, 2016.

[38] Kazuto Nakashima and Ryo Kurazume. Lidar data synthesis with denoising diffusion probabilistic models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14724–14731. IEEE, 2024.

[39] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[40] Xuebin Qin, Zichen Zhang, Chenyang Huang, Chao Gao, Masood Dehghan, and Martin Jagersand. Basnet: Boundary-aware salient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7479–7489, 2019.

[41] Wentao Qu, Guofeng Mei, Yang Wu, Yongshun Gong, Xiaoshui Huang, and Liang Xiao. A self-conditioned representation guided diffusion model for realistic text-to-lidar scene generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9434–9444, 2026.

[42] Haoxi Ran, Vitor Guizilini, and Yue Wang. Towards realistic scene generation with lidar diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14738–14748, 2024.

[43] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[44] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[45] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

[46] Nicolas Sereyjol-Garros, Ellington Kirby, Victor Besnier, and Nermin Samet. Leveraging 3d representation alignment and rgb pretrained priors for lidar scene generation. *arXiv preprint arXiv:2601.07692*, 2026.

[47] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 761–769, 2016.

[48] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[49] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[50] Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3405–3414, 2019.

[51] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *2017 international conference on 3D Vision (3DV)*, pages 11–20. IEEE, 2017.

[52] Suchetan G Uppur, Hemant Kumar, and Vaibhav Kumar. Range-edit: Semantic mask guided outdoor lidar scene editing. *arXiv preprint arXiv:2511.17269*, 2025.

[53] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[54] Song Wang, Lingdong Kong, Xiaolu Liu, Hao Shi, Wentong Li, Jianke Zhu, and Steven CH Hoi. Forging spatial intelligence: A roadmap of multi-modal data pre-training for autonomous systems. *arXiv preprint arXiv:2512.24385*, 2025.

[55] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1887–1893. IEEE, 2018.

[56] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[57] Zhengkang Xiang, Zizhao Li, Amir Khodabandeh, and Kourosh Khoshelham. Sg-ldm: Semantic-guided lidar generation via latent-aligned diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 24965–24976, 2025.

[58] Yuwen Xiong, Wei-Chiu Ma, Jingkang Wang, and Raquel Urtasun. Learning compact representations for lidar completion and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1074–1083, 2023.

[59] Gangwei Xu, Haotong Lin, Hongcheng Luo, Xianqi Wang, Jingfeng Yao, Lianghui Zhu, Yuechuan Pu, Cheng Chi, Haiyang Sun, Bing Wang, et al. Pixel-perfect depth with semantics-prompted diffusion transformers. *arXiv preprint arXiv:2510.07316*, 2025.

[60] Xiang Xu, Ao Liang, Youquan Liu, Linfeng Li, Lingdong Kong, Ziwei Liu, and Qingshan Liu. U4d: Uncertainty-aware 4d world modeling from lidar sequences. *arXiv preprint arXiv:2512.02982*, 2025.

[61] Yanzhao Yang, Jian Wang, Xinyu Guo, Xinyu Yang, and Wei Qin. Methods for improving point cloud authenticity in lidar simulation for autonomous driving: a review. *IEEE Access*, 13: 4562–4580, 2025.

[62] Rongxiang Zeng and Yongqi Dong. Latent world models for automated driving: A unified taxonomy, evaluation framework, and open challenges. *arXiv preprint arXiv:2603.09086*, 2026.

[63] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[64] Guosheng Zhao, Yaozeng Wang, Xiaofeng Wang, Zheng Zhu, Tingdong Yu, Guan Huang, Yongchen Zai, Ji Jiao, Changliang Xue, Xiaole Wang, et al. Unidrivedreamer: A single-stage multimodal world model for autonomous driving. *arXiv preprint arXiv:2602.02002*, 2026.

[65] Long Zhao, Sanghyun Woo, Ziyu Wan, Yandong Li, Han Zhang, Boqing Gong, Hartwig Adam, Xuhui Jia, and Ting Liu. Epsilon-vae: Denoising as visual decoding. *arXiv preprint arXiv:2410.04081*, 2024.

[66] Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *International Conference on Learning Representations*, volume 2026, pages 35791–35820, 2026.

[67] Zehan Zheng, Fan Lu, Weiyi Xue, Guang Chen, and Changjun Jiang. Lidar4d: Dynamic neural fields for novel space-time view lidar synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5145–5154, 2024.

[68] Yiqi Zhong, Cho-Ying Wu, Suya You, and Ulrich Neumann. Deep rgb-d canonical correlation analysis for sparse depth completion. *Advances in Neural Information Processing Systems*, 32, 2019.

[69] Jingmeng Zhou. A review of lidar sensor technologies for perception in automated driving. *Academic Journal of Science and Technology*, 3(3):255–261, 2022.

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

[71] Vlas Zyrianov, Henry Che, Zhijian Liu, and Shenlong Wang. Lidardm: Generative lidar simulation in a generated world. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6055–6062. IEEE, 2025.

A Extended Related Work

Decoder taxonomy mechanism. LiDM [42] and R2DM [38], in the LiDARGen lineage [70], use circular convolutions that wrap around the sensor's 360° azimuthal sweep, whereas general-purpose video VAEs such as SVD [3] and Wan2.1 [53] inherit ordinary image/video padding. Circular padding resolves the azimuthal seam, but any convolutional decoder, regardless of padding, still blends across the *radial* depth discontinuities that cause flying pixels (Sec. 2).

Architecture-level structural responses. Beyond the post-hoc rectifier of L3DR and the raycasting approach of LidarDM (discussed in Sec. 2), several methods address range-view geometric fidelity through representation and architecture choices. TopoLiDM introduces graph-structured latents with topological regularisation to preserve global scene connectivity [30]. LidarGRIT explicitly separates range and ray-drop decoding with a vector-quantized transformer, treating valid-return prediction as a distinct problem from depth synthesis [12]. RangeFormer identifies projection conflicts and shape deformation as core range-view limitations, motivating a redesigned tokenisation strategy [24]. T2LDM targets over-smoothing at the representation level: a self-conditioned representation guidance term aligns the denoiser's intermediate features with real-data representations during training, decoupled at inference, to recover sharper geometric detail without changing the decoder architecture itself [41]. These works improve geometric fidelity through latent representation or tokenisation; none isolates the decoder as a separable stage.

Downstream applications and world modeling. Decoder fidelity is consequential in applied settings beyond standalone generation. Adverse-weather augmentation translates clean LiDAR scans to rain, fog, and snow conditions for detector robustness training [36], and semantic mask-guided editing synthesises controlled scene variations [52]; in both cases, flying pixels in the decoded output corrupt the resulting training data. U4D further extends world modeling to uncertainty-aware 4D LiDAR prediction [60]. A growing body of world-model surveys treats latent multimodal representations as the core substrate for autonomous driving systems [9, 25, 62, 27], making decoder quality increasingly consequential as generation is embedded in closed-loop evaluation and planning pipelines.

Support prediction in LiDAR generation. Explicit support prediction is a standard component of LiDAR generation. Prior work models ray-dropping as learned binary classification with U-Net-style architectures [44, 34, 71], and support prediction has also appeared in latent LiDAR pipelines [67].

Pixel-space diffusion for depth. The pixel-space diffusion lineage relevant to CRISP begins with Marigold, which showed that pretrained diffusion priors can be repurposed for dense monocular depth estimation through finetuning on synthetic data [19]. This established that generative models carry geometric priors useful beyond image synthesis, but the decoder in that setting operates on a dense, support-complete depth map and the 3D consequences of decoder-induced smoothing were not the focus. Pixel-Perfect Depth carries this further with a cascade of diffusion transformers [39] conditioned on a clean RGB image, showing that even ground-truth-quality depth is degraded by VAE decoding [59]; CRISP adapts this cascade-DiT idea to a regime with no clean guidance image and genuine missing returns (Sec. 2).

Concretely, CRISP differs from Pixel-Perfect Depth on three axes: removing CRISP's early latent injection, which replaces Pixel-Perfect Depth's persistent RGB conditioning, costs whole-cloud CD $0.985 \rightarrow 1.134$ and FSVD $98.6 \rightarrow 119.6$; diffusing over all pixels instead of gating through CRISP's predicted support mask (needed because LiDAR range maps, unlike dense monocular depth, have genuine missing returns) costs JSD $0.071 \rightarrow 0.109$ and edge F-score $0.238 \rightarrow 0.193$ (both Table 14); and unlike Pixel-Perfect Depth's single continuous depth output, CRISP composes a depth map with a predicted validity mask, $\hat{y} = \hat{m} \odot \hat{y}_d - (1 - \hat{m})$. Two recent lines replace convolutional decoders with diffusion decoders for natural images rather than depth: PiD unifies latent decoding and super-resolution into a single diffusion pass, distilled to a handful of steps for speed [32], and ϵ-VAE reframes VAE decoding itself as iterative denoising [65]; both are optimized for perceptual realism rather than per-pixel faithfulness to the latent (Sec. 2). LiDAR range maps, by contrast, are geometric measurements feeding safety-relevant perception, where per-frame faithfulness rather than distributional typicality is required, which is why CRISP's depth branch is a validity-masked regression against ground truth rather than a free sampler. RAE moves in the opposite direction, replacing the VAE encoder with a frozen pretrained representation encoder (e.g. DINO, SigLIP, or MAE) paired with a trained decoder, and argues that the VAE's low-dimensional, weakly semantic latent limits diffusion transformers [66]. CRISP instead keeps the encoder and the pretrained latent generator entirely frozen, which is what allows a plug-in decoder swap inside an already-trained LiDM world model (Table 3).

B Architectural Details

B.1 DiT Decoder

Model scale. The DiT decoder $\mathcal{D}_\theta$ has $K{=}24$ transformer blocks, hidden dimension $C_d{=}1024$, and 16 attention heads. Each block uses an MLP with ratio 4.0 and tanh-approximated GELU activation [13]. Attention uses QK normalisation for training stability [14]. All weight matrices are Xavier-initialised; adaLN modulation projection weights are zero-initialised so every block is an identity map at the start of training [39].

Coarse-to-fine cascade. The noisy input x_t is tokenised into $p{\times}p = 16{\times}16$ patches, giving $T_0 = \frac{H}{16} \cdot \frac{W}{16}$ tokens. Blocks 0-$(K/2{-}1)$ process these at the coarse scale with 2D RoPE at resolution $(H/16, W/16)$ [49]. At the mid-fusion step, the T_0 token sequence is pixel-shuffled to $4T_0 = \frac{H}{8} \cdot \frac{W}{8}$ tokens (a reshape-einsum operation equivalent to spatial unfolding), and blocks $K/2$-$(K{-}1)$ refine at this finer scale with RoPE at $(H/8, W/8)$. The coarse-first progression lets global layout be established at low token count before local geometry is refined at higher resolution, which matters because the decoder starts from pure Gaussian noise with no auxiliary observation.

Latent fusion asymmetry. Adapter tokens τ are L2-normalised before both injections. Early fusion is applied to the token stream *before* block 0 processes its input, using a lightweight two-layer MLP ($2C_d \rightarrow C_d \rightarrow C_d$, SiLU): the global layout signal must enter cheaply so it is available from the first block. Mid-level fusion (block $K/2{-}1$) uses a wider three-layer MLP ($2C_d \rightarrow 4C_d \rightarrow 4C_d \rightarrow 4C_d$): this projection simultaneously fuses geometry-aware context and provides the spatial token counts required for the pixel-shuffle upsample, so additional capacity is justified.

B.2 Support Mask Predictor

$\mathcal{M}_\phi$ is a U-Net with three encoder levels and base channel width $C_m{=}128$ ($C_m{\rightarrow}2C_m{\rightarrow}4C_m$) [44]. Standard components (skip connections, average-pool downsampling, bilinear upsampling, GroupNorm [56], Xavier init) follow established practice.

Dual-branch conditioning. A latent branch processes z through a two-layer convolutional block (Conv-GroupNorm-GELU $\times 2$) and upsamples it bilinearly to the range-map resolution. A separate depth-hint branch maps the single decoded depth channel to $C_m/2{=}64$ features through a shallow convolutional block. The two branches are concatenated ($C_m + C_m/2$) and fused before the U-Net encoder stack, giving the predictor access to both semantic layout (from the latent) and high-frequency geometric structure (from the depth hint).

B.3 Geometry-Aware Diffusion Objective

Valid-region velocity loss. The base supervision signal is a plain validity-masked regression on the velocity target ($\hat{v}$, $v^\star$ as in Eq. 2):

$$\mathcal{L}_{\text{vel,raw}} = \Big(\sum\nolimits_{ij} m_{ij}\, \ell(\hat{v}_{ij},\, v^\star_{ij})\Big) \Big/ \Big(\sum\nolimits_{ij} m_{ij}\Big), \tag{13}$$

where ℓ is ℓ_2 until validation depth metrics plateau, then ℓ_1 for fine-grained convergence. Every valid pixel contributes equally.

Directional finite-difference losses. Range maps are anisotropic: horizontal neighbors lie along the same LiDAR beam sweep, while vertical neighbors correspond to different elevation beams at the same azimuth. We therefore regularize local structure along both axes. First-order terms preserve local slopes, while second-order terms suppress high-frequency waviness introduced by denoising. Letting $\delta_x v_{ij} = v_{i,j+1} - v_{i,j}$, $\delta_{xx} v_{ij} = v_{i,j+1} - 2v_{i,j} + v_{i,j-1}$, and δ_y, δ_{yy} the analogous vertical operators, all four terms share the form:

$$\mathcal{L}_{d\alpha} = \frac{1}{|\Omega_\alpha|} \sum_{\Omega_\alpha} \big|\delta_\alpha \hat{v}_{ij} - \delta_\alpha v^\star_{ij}\big|, \quad \alpha \in \{x,\, xx,\, y,\, yy\}, \tag{14}$$

where Ω_α contains pixels whose neighbors involved in δ_α all satisfy $m = 1$. The combined loss is:

$$\mathcal{L}_{\text{der}} = \lambda_{dx}\mathcal{L}_{dx} + \lambda_{d2x}\mathcal{L}_{d2x} + \lambda_{dy}\mathcal{L}_{dy} + \lambda_{d2y}\mathcal{L}_{d2y}, \qquad \lambda_{dx}, \lambda_{d2x}, \lambda_{dy}, \lambda_{d2y} \geq 0. \tag{15}$$

Low-noise reconstruction term. As a fine-tuning signal, $\mathcal{L}_{x0}$ penalizes the denoised estimate $\hat{y}_d$ against the target range map, with a linear timestep weight that emphasizes low-noise steps:

$$\mathcal{L}_{x0} = \frac{1}{\sum_{ij} m_{ij}} \sum_{ij} \Big(1 - \tfrac{t}{T}\Big)\, m_{ij}\, \big|\hat{y}_{d,ij} - \tilde{y}_{ij}\big|. \tag{16}$$

Boundary-aware velocity loss (weighted form). The boundary-aware velocity loss $\mathcal{L}_{\text{vel}}$ referenced in the main text replaces m with the importance weight w from Eq. 9:

$$\mathcal{L}_{\text{vel}} = \Big(\sum\nolimits_{ij} w_{ij}\, \ell(\hat{v}_{ij},\, v^\star_{ij})\Big) \Big/ \Big(\sum\nolimits_{ij} w_{ij}\Big). \tag{17}$$

The structure signal uses G_σ with kernel 5×5, σ=2.0, and cap $\lambda_{\max}$=1.5. Loss weights for all frozen and encoder-adapted training stages are listed in Table 13.

Depth-branch leave-one-out. Table 6 reports validation L1 (↓) when each depth term is removed (for $\mathcal{L}_{\text{vel}}$, its importance weighting is set uniform, $w \to m$) from an otherwise-matched reference run, on SVD $\times$3/nuScenes and, separately, on the frozen LiDAR-native LiDM backbone/KITTI-family. Every surviving term is load-bearing on at least one dataset, and the two disagree about which: the ℓ_1 pixel penalty $\mathcal{L}_{x0}$ matters on nuScenes but far less on KITTI-family, while the finite-difference term $\mathcal{L}_{\text{der}}$ shows the reverse, consistent with sharper radial discontinuities at 64 beams. Dropping either would silently specialise the objective to one beam count. $\mathcal{L}_{\text{grad}}$ is the only term small on both datasets and is the one dropped from the four-term reduced depth objective (Table 10).

Table 6: Depth-branch leave-one-out: validation L1 (↓) when each term is removed from the reference objective (for $\mathcal{L}_{\text{vel}}$, its importance weighting is set uniform), on SVD $\times$3/nuScenes and on the frozen LiDM backbone/KITTI-family; percentages are the change against the matched reference.

Removed term	nuScenes val L1	KITTI-family val L1
– (reference)	0.06052	0.0218291
Importance weighting ($w \to m$ in $\mathcal{L}_{\text{vel}}$)	0.07578 (+25.2%)	0.0231271 (+5.9%)
ℓ_1 pixel penalty ($\mathcal{L}_{x0}$)	0.06540 (+8.1%)	0.0218987 (+0.3%)
Finite differences ($\mathcal{L}_{\text{der}}$)	0.06291 (+3.9%)	0.0235116 (+7.7%)
Multi-scale gradient ($\mathcal{L}_{\text{grad}}$)	0.06228 (+2.9%)	0.0218484 (+0.1%)

Table 7: Mask loss sub-term weights (Eq. 10).

Term	λ
Focal BCE ($\gamma=2$, $\alpha=0.4$)	1.0
Soft Dice	1.0
Edge (1st-order)	0.5
Laplacian (2nd-order)	1.5
Hard-pixel mining	0.5
Deep supervision	0.5

B.4 Mask loss weights.

The mask objective (Eq. 10) combines six terms with the following weights:

Mask-branch leave-one-out and minimal subset. Table 8 reports the paired change in IoU when each term is removed from the full six-term objective against a matched, same-seed reference (SVD $\times$3/nuScenes; the frozen LiDM backbone/KITTI-family, reference IoU 0.9888). Focal BCE and soft Dice separate the two datasets: load-bearing on nuScenes, but next to unresolvable on KITTI-family, where support prediction on the LiDAR-native backbone is already near ceiling. Table 9 instead trains only a subset of terms (LiDM backbone/KITTI-family, 10 epochs, seed 23, batch 16; weights Focal 1.0, Dice 1.0, Laplacian 1.5, the rest zeroed). Three terms, Focal + Dice + Laplacian, recover 96.7% of the full objective's IoU gain over focal-only supervision, and outperform the Focal + Dice + Edge combination an additive sweep would select instead (cf. Table 15): an ordering artifact, since Edge enters first in that cumulative ablation and absorbs the shared boundary term. The minimal mask objective is therefore Focal + Dice + Laplacian, three of six (Sec. 3.6).

Table 8: Mask-branch leave-one-out: paired change in IoU vs. a matched, same-seed full-objective reference when each term is removed.

Removed term	nuScenes ΔIoU	KITTI-family ΔIoU
Laplacian	-0.00245	-0.00110
Soft Dice	-0.00206	-0.00010
Focal BCE	-0.00172	-0.00010
Hard-pixel mining	-0.00054	≈ 0
Deep supervision	-0.00034	≈ 0
Edge	-0.00033	-0.00020

Table 9: Mask-branch minimal subset (LiDM backbone, KITTI-family, 10 epochs): converged IoU when zeroing all but the listed terms, and the resulting share of the full objective's IoU gain over focal-only supervision retained.

Mask objective	IoU	Gain retained
Focal only	0.97418	–
Focal + Dice + Edge	0.98761	91.9%
Focal + Dice + Laplacian	0.98831	96.7%
Full six-term	0.98879	100%

B.5 Objective Design: Provenance, Balancing, and Compute Cost

Table 10 lists, for every term in Eqs. 7 and 10, the failure mode it is admitted for, the literature precedent for the operator, its weight (Stage F2, Table 13), and whether it survives in the reduced objective (Tables 6-9).

Term count. In the frozen setting the objective comprises eleven loss terms: five in $\mathcal{L}_{\mathrm{depth}}$ and six in $\mathcal{L}_{\mathrm{mask}}$; encoder-adapted variants add the KL term of Eq. 12. A count of fourteen arises only if $\mathcal{L}_{\mathrm{der}}$ (Eq. 15) is unpacked into its four $(\mathrm{axis}, \mathrm{order})$ components, $\mathcal{L}_{dx}$, $\mathcal{L}_{d2x}$, $\mathcal{L}_{dy}$, $\mathcal{L}_{d2y}$ (Eq. 14),

Table 10: Loss-term justification: failure mode, operator provenance, weight, and reduced-objective status for every term in $\mathcal{L}_{\mathrm{depth}}$ (Eq. 7) and $\mathcal{L}_{\mathrm{mask}}$ (Eq. 10).

Term	Failure mode	Provenance	Weight	Reduced
Depth branch (Eq. 7)				
$\mathcal{L}_{\mathrm{vel}}$	region: boundary-aware velocity regression on valid pixels	–	1.0	yes
$\mathcal{L}_{\mathrm{vel,raw}}$	stable unweighted baseline; normalises $\mathcal{L}_{\mathrm{vel}}$'s per-edge gradient share	structural, no operator citation	1.0	yes
$\mathcal{L}_{\mathrm{der}}$	anisotropic per-ring radial bias invisible per-pixel	per-axis derivative matching (cf. LiDM gradient loss [42])	Table 13	yes
$\mathcal{L}_{x0}$	low-noise accuracy of $\hat{y}_d$ (fine-tuning signal)	timestep-weighted diffusion loss [45, 48]	0.5	yes
$\mathcal{L}_{\mathrm{grad}}$	low-frequency range-gradient residual	LiDM multi-scale gradient loss [42]	0.5	no
Mask branch (Eq. 10)				
$\mathcal{L}_{\mathrm{focal}}$	region: valid/void class imbalance	focal loss for dense detection [29]	1.0	yes
$\mathcal{L}_{\mathrm{dice}}$	scarce thin/sparse valid-return recall	overlap-based segmentation loss [37]	1.0	yes
$\mathcal{L}_{\mathrm{lap}}$	isolated single-pixel ray-drop speckle	depth-completion smoothness [33, 68]	1.5	yes
$\mathcal{L}_{\mathrm{edge}}$	boundary misalignment (first-order instance of $\mathcal{L}_{\mathrm{lap}}$'s prior)	boundary-consistency loss [18]	0.5	no
$\mathcal{L}_{\mathrm{hard}}$	under-trained near-boundary pixels	online hard-example mining [47]	0.5	no
$\mathcal{L}_{\mathrm{ds}}$	weak gradient signal at shallow layers	deep supervision [26, 40]	0.5	no

and each counted separately. They are computed separately because the range map is a 2D grid with two physically distinct axes (horizontal beam sweep, vertical elevation) and because first- and second-order differences capture different structure (slope vs. curvature), not because each is an independently motivated objective.

Balancing procedure. Per-term weights (Table 13) are set once per training stage: every term is scaled to an equal initial magnitude and then lightly adjusted with a simplified, static form of GradNorm [6] rather than its full dynamic procedure. The same resulting weight vector is used for every backbone (SVD, Wan2.1, LiDM) and both beam counts (32-beam nuScenes, 64-beam KITTI-family), with no per-setting retuning. Apart from $\mathcal{L}_{x0}$, $\mathcal{L}_{\mathrm{grad}}$ and the mask branch, which are switched on at Stage F2, the derivative-loss weights are the only ones that change within a run, up-weighted 8-17$\times$ between Stage F1 and Stage F2 (Table 13); this did not destabilise training, with all twenty unsmoothed validation checkpoints on the matched reference run improving monotonically. The one resulting sensitivity, gradient clipping on 64-beam KITTI-family, is discussed in Appendix D.

Compute and memory cost. Table 11 measures training step time and peak memory as loss terms are added, holding the network, data, and checkpoints fixed so only the objective changes (one H200, BF16, SVD $\times$3/nuScenes, 24-block DiT, 25-frame clip; median of 100 steps after 50 warm-up steps, five interleaved runs per configuration; between-run spread 1.09-2.18 ms). The measurable cost is the second branch (the mask predictor's forward, loss, and backward pass), not the number of terms: going from the minimal two-branch configuration ($\mathcal{L}_{\mathrm{vel}}$, $\mathcal{L}_{\mathrm{vel,raw}}$, $\mathcal{L}_{\mathrm{focal}}$) to the full eleven-term objective adds 7.81 ms, 3.0% of a step, at effectively identical peak memory (34.397 vs. 34.394 GiB), because the added terms are elementwise or finite-difference operations on single-channel maps with no parameters and no extra forward pass. Individual terms cost at or below the between-run spread, so we do not report them separately. The seven-term reduced objective (Tables 6-9) is motivated by overlap between terms, not by this already-small compute: it saves only 5.36 ms, 2% of a step, versus the full objective. At inference no loss is computed, but the mask branch itself still runs: 295.49 ms per clip vs. 252.46 ms with the branch stubbed out, i.e. 43.03 ms, or 14.6% of decoder latency (batch 1, 25-frame clip, 5 Euler steps).

C Additional Experimental Details

C.1 Data Preparation

KITTI family (KITTI-360, SemanticKITTI). Raw 64-beam point clouds are projected onto 64×1024 spherical range images following the LiDM projection protocol (vertical FoV $[-25^\circ, +3^\circ]$,

Table 11: Training step time and peak memory as loss terms are added, network/data/checkpoints held fixed (one H200, BF16, SVD $\times 3$/nuScenes, median of 100 steps).

Objective	Branches	Terms	ms/step↓	Peak GiB↓
Depth only ($\mathcal{L}_{\text{vel}}+\mathcal{L}_{\text{vel,raw}}$)	Depth	2	165.11	19.878
$+\,\mathcal{L}_{\text{focal}}$	Depth + Mask	3	263.23	34.397
Reduced set	Depth + Mask	7	265.68	34.393
Full objective	Depth + Mask	11	271.04	34.394

azimuthal range 360°). Valid depths lie in $[1.0,\ 56.0]$ m; each depth value d is log-normalised and mapped to $[-1, 1]$:

$$\hat{d} = \frac{2\,\log_2(d+1)}{\delta} - 1, \qquad \delta = 5.84, \tag{18}$$

with empty bins assigned $\hat{d} = -1$. The training split provided by LiDM contains 4,376 25-frame LiDAR clips. We apply the LiDM augmentation protocol 5× (identity, vertical flip, horizontal flip, point dropout, and horizontal shift), yielding 21,880 augmented 25-frame clips per epoch. Although clips define the sampling unit, all frames are encoded and decoded independently. The validation split is used as provided; no test-set frames are used for any model-selection decision.

nuScenes. Each scan is projected onto a 32×1024 beam-indexed range image (one row per laser ring), with valid depths in $[2.0,\ 45.0]$ m, log-normalised to $[-1, 1]$:

$$\hat{d} = \frac{2\big(\log_2(d+1) - \log_2(d_{\min}+1)\big)}{\log_2(d_{\max}+1) - \log_2(d_{\min}+1)} - 1, \tag{19}$$

with $d_{\min} = 2.0$ m and $d_{\max} = 45.0$ m; empty bins are set to -1. We follow the VISTA [10] data partition: the official VISTA *training* set is split into a training portion of 19,088 25-frame LiDAR clips and a held-out validation subset; the VISTA test set is reserved exclusively for evaluation. Although clips provide the sampling unit, all frames are encoded and decoded independently, without temporal compression.

C.2 Frozen Backbones.

For the frozen-encoder ($\times 3$) variants, neither the SVD nor the Wan2.1 VAE encoder receives any parameter updates. The single-channel range image is replicated across all three input channels before encoding ($\hat{d}_{\text{rgb}} = \hat{d} \cdot \mathbf{1}_3$). For the inherited-decoder baseline, the original VAE decoder outputs a three-channel reconstruction, which we convert back to a single-channel range map by averaging the three decoded channels. For CRISP, the original decoder is replaced by the CRISP DiT decoder, which directly predicts a single-channel range map from the same frozen latent. No other modifications are made to the backbone.

C.3 Encoder Adaptation.

Both the SVD and Wan2.1 VAEs are pretrained on 3-channel RGB video. For the *adapted baseline* setting, we follow the encoder-adaptation pipeline proposed in UniDriveDreamer [64] without modification: the VAE is converted to single-channel LiDAR input/output by weight surgery and then fine-tuned with the same reconstruction objective. This baseline answers whether CRISP remains useful after the inherited VAE itself has been adapted to LiDAR.

Both adapted baseline VAEs use the UniDriveDreamer-style objective:

$$\mathcal{L}_{\text{enc}} = \lambda_{\text{recon}}\mathcal{L}_1 + \lambda_{\text{kl}}\mathcal{L}_{\text{KL}} + \lambda_{\text{lpips}}\mathcal{L}_{\text{LPIPS}},$$

where LPIPS uses a frozen VGG backbone [63]. Since VGG-LPIPS expects three-channel input, we replicate the single-channel normalised LiDAR range map across RGB channels before computing LPIPS. This objective is used only to produce the adapted VAE baselines; CRISP-adapted variants instead train the encoder jointly with the CRISP depth and mask objectives plus the KL regulariser in Eq. 12.

SVD $\times$1. The SVD VAE is evaluated in single-frame mode: although its video decoder contains temporal-mixing modules, we pass `timesteps=1`, so these modules reduce to per-frame operations and do not introduce temporal-compression artifacts. No layer-wise freeze tiers are needed: all parameters are trained at a uniform learning rate $\eta_{\text{vae}} = 5 \times 10^{-5}$. Before any gradient updates, the 3-channel I/O weights are averaged into 1-channel initialisations: `encoder.conv_in` ($[128, 3, 3, 3] \rightarrow [128, 1, 3, 3]$, averaged over the input-channel axis) and `decoder.conv_out` ($[3, 128, 3, 3] \rightarrow [1, 128, 3, 3]$, averaged over the output-channel axis); biases and temporal-mixing parameters are kept from the pretrained checkpoint. This preserves pretrained filter magnitudes while yielding a valid 1-channel starting point. Loss weights: $(\lambda_{\text{recon}}, \lambda_{\text{kl}}, \lambda_{\text{lpips}}) = (1.0,\ 10^{-5},\ 0.3)$.

Wan2.1 $\times$1. Wan2.1 introduces four purely-temporal convolutions (`encoder.downsamples.{5,8}.time_conv`, `decoder.upsamples.{3,7}.time_conv`) that implement $\times 4$ temporal compression. Updating these layers (even at extremely low learning rates) causes the model to collapse within a handful of iterations. We therefore *freeze all four temporal convolutions entirely* and apply a two-tier learning rate: the four I/O convolutions (`encoder.conv1`, `encoder.head`, `decoder.conv1`, `decoder.head`) and the two latent projection layers (`conv1`, `conv2`), which undergo the largest domain shift, are trained at $\eta_{\text{vae}} = 5 \times 10^{-5}$; all remaining interior ResBlocks at $0.1\,\eta_{\text{vae}}$. Loss weights: $(\lambda_{\text{recon}}, \lambda_{\text{kl}}, \lambda_{\text{lpips}}) = (1.0,\ 10^{-2},\ 0.3)$. The larger λ_{kl} compared to SVD reflects the temporal architecture's greater tolerance for KL regularisation after the I/O surgery.

C.4 Temporal Compression Bypass.

Wan2.1 applies $\times 4$ temporal compression, under which only the first frame in the encoded clip is reconstructed best, while subsequent frames progressively degrade. Table 12 quantifies this for Wan $\times 3$ and Wan $\times 1$ over 25 frames: at $t = 0$, CD is 0.770 / 0.309 for $\times 3$ / $\times 1$; by $t = 5$ it reaches 0.806 / 0.377 (ΔCD $= +0.036$ / $+0.068$). To eliminate this artifact in the main comparison, we evaluate both the inherited Wan decoder and CRISP independently on each frame, feeding 25 separate single-frame latents rather than a single temporally compressed video tensor.

Table 12: Per-frame CD (↓) and F-score@0.5m (↑) for Wan $\times 3$ (left) and Wan $\times 1$ (right) under joint temporal encoding. Frame $t = 0$ is systematically the best; the degradation motivates per-frame inference. **Bold** = best per column.

	Wan $\times 3$		Wan $\times 1$	
t	CD↓	F-score@0.5m↑	CD↓	F-score@0.5m↑
0	**0.770**	**0.732**	**0.309**	**0.867**
1	0.792	0.718	0.336	0.862
2	0.805	0.712	0.350	0.859
3	0.794	0.721	0.333	0.862
4	0.804	0.716	0.351	0.858
5	0.806	0.710	0.377	0.854
⋮			⋮	
Mean	0.795	0.717	0.353	0.857

D Training Protocol

The training protocol differs between frozen-encoder ($\times 3$) and encoder-adapted ($\times 1$) variants.

Frozen-encoder variants ($\times$3): two stages. Frozen-encoder variants use two stages. In Stage F1, the encoder is fixed and only the DiT decoder is trained with an L_2 velocity objective; the mask predictor is absent. In Stage F2, the encoder remains fixed, the objective switches from L_2 to L_1, gradient and derivative losses are up-weighted, and the support mask predictor is introduced and trained jointly with the DiT. We transition from Stage F1 to Stage F2 after $\approx$12 h on 8$\times$A100 GPUs, when validation reconstruction loss plateaus.

Encoder-adapted variants ($\times$1): three stages. Encoder-adapted variants use three stages. In Stage A1, the encoder and DiT decoder are trained jointly with an L_2 velocity objective and KL

regularisation; the mask predictor is absent. In Stage A2, the mask predictor is added while the encoder remains trainable and the depth branch still uses the L_2 objective. This lets the mask branch adapt to the latent space without prematurely dominating the initial encoder/DiT training. In Stage A3, the encoder is frozen, the objective switches from L_2 to L_1, gradient and derivative losses are up-weighted, and the DiT and mask predictor are jointly fine-tuned on the fixed latent manifold.

Runtime and compute. Each CRISP training run takes approximately 36 h on 8×A100-80GB GPUs: about 24 h for the DiT-only stage and an additional 12 h after introducing the support mask branch. This corresponds to roughly 288 GPU-hours per full CRISP run. Runtime varies slightly across backbones and excludes baseline VAE adaptation and ablations.

Table 13: Training stages and loss weights. Frozen-encoder variants (×3) use Stages F1-F2; encoder-adapted variants (×1) use Stages A1-A3. "–" denotes an inactive branch or loss.

Loss / setting	Frozen encoder (×3)		Encoder-adapted (×1)		
	F1	F2	A1	A2	A3
Encoder trainable	no	no	yes	yes	no
Mask branch	–	yes	–	yes	yes
Velocity loss	L_2	L_1	L_2	L_2	L_1
λ_{x0}	0.0	0.5	0.0	0.0	0.5
λ_{grad}	0.0	0.5	0.0	0.0	0.5
λ_{dx}	0.25	2.0	0.25	0.25	2.0
λ_{d2x}	0.25	2.0	0.25	0.25	2.0
λ_{dy}	0.12	2.0	0.12	0.12	2.0
λ_{d2y}	0.12	2.0	0.12	0.12	2.0
λ_{KL}	–	–	5×10^{-6}	5×10^{-6}	–
λ_{mask}	–	1.2	–	1.2	1.2

Importance weights emphasise geometrically complex regions using the valid-only local structure signal from Eqs. 8-9, with a 5×5 Gaussian kernel, $\sigma = 2.0$, and cap $\lambda_{\max} = 1.5$. During Stage A2, mask gradients are scaled by 0.1 when back-propagated through the shared encoder, preventing mask-classification gradients from corrupting the depth latent representation.

Optimizer and Regularization. All models use AdamW. Learning rates: $\eta_{\text{DiT}} = \eta_{\text{mask}} = 10^{-4}$. For frozen-encoder (×3) variants, $\eta_{\text{enc}} = 0$ throughout. For encoder-adapted (×1) variants, $\eta_{\text{enc}} = 5\times10^{-5}$ during Stages A1-A2, with the per-layer scaling described in Appendix C.3, and $\eta_{\text{enc}} = 0$ in Stage A3. A linear warm-up over $10{,}000$ steps is applied from 10^{-6} to each target value.

Gradient norms are clipped at 1.0. Without clipping, sharp depth discontinuities in the 64-beam KITTI scans drive the gradient norm from ≈ 2 to ≈ 42 in a single step; no similar instability was observed on nuScenes.

Dropout rates: $p_{\text{mlp}} = 0.1$, $p_{\text{attn}} = 0.05$, $p_{\text{proj}} = 0.1$.

EMA shadow weights are maintained at decay 0.9999 for the DiT decoder and encoder, and at decay 0.999 for the mask predictor. The faster mask EMA allows the shadow weights to track the rapidly evolving dynamics when the mask branch is introduced in Stage F2 or Stage A2.

Training uses 8×A100-80 GB GPUs, DDP, and BF16 mixed precision. Each GPU processes one 25-frame clip, but the model treats the frames as independent single-frame samples: no temporal attention, recurrent state, or temporal loss is used. Maximum training duration is 100 epochs.

Evaluation Protocol. All point clouds are reconstructed in 3D Euclidean space at each dataset's native metric scale (metres). The F-score threshold of 0.5 m is therefore identical in absolute terms for both KITTI and nuScenes. Chamfer Distance and F-score are computed using the `ChamferDistancePytorch` library [11].

Results report mean ± standard deviation over **5 inference runs** that differ only in the stochastic seed used for the diffusion sampling and Bernoulli support sampling. No retraining is performed between runs; each run initialises from the same final-stage checkpoint, namely Stage F2 for frozen-encoder variants and Stage A3 for encoder-adapted variants.

Fréchet metric protocol. All Fréchet-style metrics (FRID, FSVD, FPVD) use every unique frame in the test split, with no subsampling: nuScenes provides 33,021 real and 33,021 generated frames after deduplicating overlapping 25-frame clips (the same frame count used in the step ablation, Fig. 4), and the KITTI-family provides 5,100 real and 5,100 generated frames. Each generated sample is the reconstruction paired with one real test frame, and means and covariances are computed over the complete feature sets. Feature extractors follow LiDM/LiDARGen: MinkowskiNet for FSVD, SPVCNN for FPVD, and RangeNet++ for FRID; the FRID checkpoint exists only for 64-beam range images, hence FRID is reported only for the KITTI-family (LiDM) setting.

Cross-seed stability. Across the five reconstruction settings of Tables 1–2, seed-to-seed standard deviation on edge F-score is at most 0.0002 absolute (under 0.1% relative) over the five runs that differ only in the diffusion and Bernoulli support-sampling seeds: sampling noise contributes almost nothing when the conditioning latent is fixed. In the plug-and-play world-model setting (Table 3), where latents are themselves generated rather than encoded from data, the spread rises to 0.0014 for CRISP, but the deterministic baseline decoder shows a comparable 0.0017 in the same setting: the increase comes from the sampled latents, not from CRISP's own sampling.

E Ablations

The design ablations below use the SVD $\times 3$ setting on nuScenes, the most demanding frozen configuration due to the largest modality gap between a generic video VAE and sparse LiDAR geometry. This setting makes decoder failures most visible because the modality gap is largest; gains on LiDAR-native backbones may be smaller, and the LiDM replication at the end of this appendix (Table 16) confirms that the same design choices transfer to the LiDAR-native LiDM backbone.

Diffusion Sampling Steps. We ablate the number of Euler sampling steps over $n = 33{,}021$ test-split LiDAR frames; results are shown in Fig. 4. The ablation reveals a Pareto trade-off: CD and perceptual quality pull in opposite directions across the step count, and the optimal operating point must balance both.

Geometric quality (top-left): CD is lowest at 2 steps (CD$= 0.499$) and degrades toward 4 steps (CD$= 0.656$) before recovering at 5 (CD$= 0.639$). The 2-step minimum reflects a known property of few-step diffusion: fewer denoising steps leave the output closer to the diffusion prior mean, which minimizes per-sample ℓ_2-aligned metrics such as CD at the cost of perceptual sharpness. The non-monotone behaviour between 4 and 5 steps coincides with the sharpest perceptual improvement, suggesting the model reorganises its output distribution at this transition rather than simply refining it. F-score@0.5m decreases gradually with more steps (from 0.825 at 2 steps to 0.804 at 6), consistent with a small support miscalibration introduced as the denoiser sharpens boundary geometry.

Perceptual quality (top-right): FSVD and FPVD decrease monotonically, from $94.2/117.3$ at 2 steps to $30.1/26.5$ at 6. The largest single drop occurs between 4 and 5 steps ($61.2 \rightarrow 40.3$ FSVD, $55.4 \rightarrow 36.9$ FPVD), confirming that 2-step outputs, despite their favourable CD, are perceptually poor and not representative of the real LiDAR distribution.

Statistical divergences (bottom-left): MMD increases and plateaus after 4 steps, indicating a mild trade-off: additional denoising improves JSD/EMD and perceptual realism, but slightly worsens this nearest-set distributional metric.

Inference time (bottom-right): encode+decode time grows approximately linearly, from 315 s at 2 steps to 761 s at 6 steps on 8×H200 (without `torch.compile`; compiled throughput is in Table 17).

We choose *5 Euler steps* as the operating point: it sits past the perceptual phase transition (clearing most of the FSVD/FPVD gap), avoids the geometric regression at 4 steps, and adds only 30 s over 4 steps. The residual 6-step improvement is marginal and not worth the additional latency.

DiT decoder design. Table 14 reports two independent comparisons on SVD $\times 3$.

Early fusion (upper group, GT mask, 35 epochs). Removing early latent fusion degrades all metrics substantially: whole-cloud CD rises from 0.985 to 1.134 (+15%), FSVD from 98.6 to 119.6 (+21%), and edge CD from 7.244 to 8.030. Because the decoder starts from pure Gaussian noise with no auxiliary clean observation, the global scene layout must be established in the earliest transformer

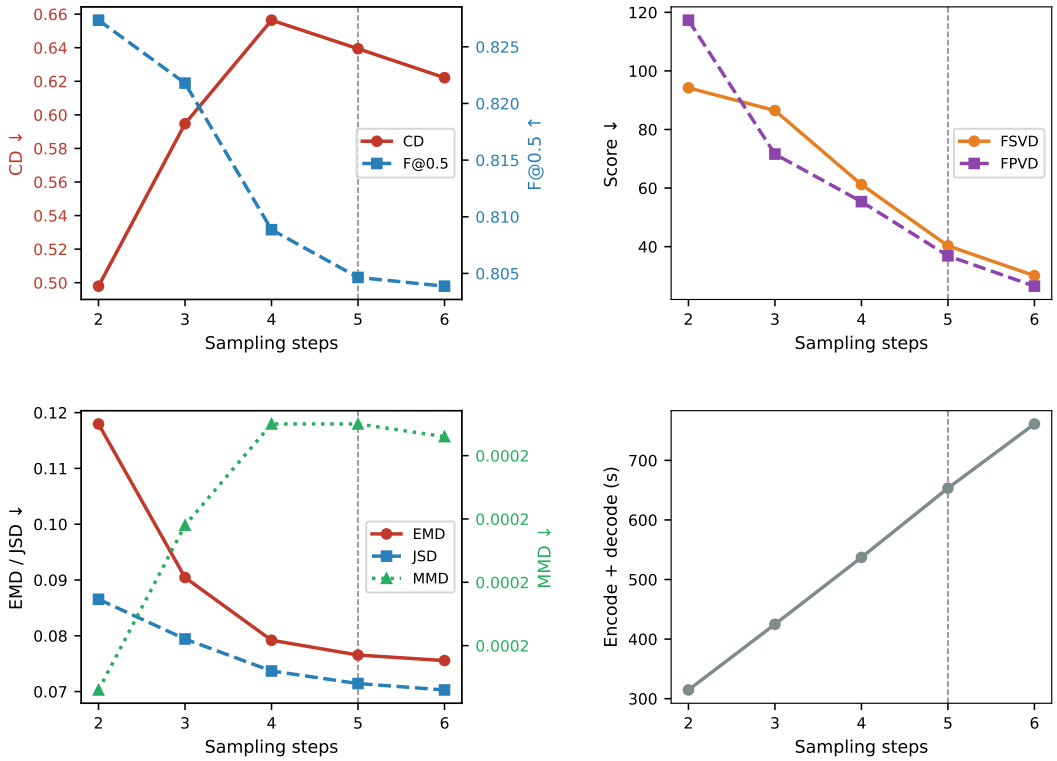

Figure 4: Step ablation on nuScenes with the SVD $\times 3$ backbone ($n = 33{,}021$ LiDAR frames). Each panel plots a different metric group as a function of Euler sampling steps; the dashed vertical line marks the adopted 5-step setting. CD and perceptual metrics exhibit opposing trends, motivating a deliberate operating-point choice rather than optimising either metric alone.

blocks; without early injection of latent tokens, the network can only infer layout from mid-level context, arriving too late to avoid locally incoherent geometry.

Explicit support prediction (lower group, converged). Adding the mask branch reduces JSD by 35% ($0.109 \rightarrow 0.071$), reduces EMD by 31%, and improves edge F-score by 23% ($0.193 \rightarrow 0.238$). The intuition is that depth values and binary valid returns are statistically incompatible signals: a single diffusion process that must simultaneously model continuous depth and binary support learns a compromised representation for both. The mask branch separates these two prediction problems, allowing the DiT to focus entirely on depth geometry. FPVD is the one metric where the no-mask variant edges out CRISP (35.993 vs 36.782); FPVD is a volume-occupancy feature metric less sensitive to boundary precision than edge F-score or JSD, and the marginal gap is within the noise range observed across runs.

Mask predictor design. Table 15 ablates the mask predictor along two axes. The upper group holds the input fixed (depth + latent) and adds one loss term at a time, each trained for 10 epochs; the lower group holds the loss set fixed (full) and varies the input, both trained to convergence. The two groups use different training durations and are not directly comparable in absolute values.

Loss composition (upper group). The largest single IoU gain is from adding Dice loss to Focal alone ($0.758 \rightarrow 0.827$, $+0.069$). This is expected: Focal BCE addresses per-pixel class imbalance but optimises a surrogate that does not directly measure overlap; Dice optimises overlap directly and is inherently recall-sensitive, recovering the many thin and sparse valid-return structures that Focal under-weights. Edge and Laplacian terms each add a further $+0.011$ and $+0.003$ respectively, capturing first-order boundary alignment and isolated ray-drop events that first-order gradients cannot detect. Hard-pixel mining adds $+0.004$ by up-weighting spatially fragile regions missed by the spatial-average losses. The full loss configuration with deep supervision (Stage 6) achieves IoU 0.844, marginally below Stage 5's 0.845; this gap is within single-run noise, but the full stack is

Table 14: DiT decoder ablation (SVD $\times 3$, nuScenes). *Upper*: GT support mask applied, 35 epochs (3-run mean); early-fusion comparison. *Lower*: converged end-to-end; mask-formulation comparison. Best within each group is bold.

Variant	Statistical			Perceptual		
	JSD↓	EMD↓	MMD↓ $\times 10^{-4}$	FSVD↓	FPVD↓	FRID↓
GT mask, 35 epochs						
w/o early fusion	0.120	0.241	3.101	119.557	100.839	-
w/ early fusion	**0.104**	**0.196**	**2.739**	**98.649**	**81.651**	-
Full system, converged						
Loss everywhere (no mask branch)	0.109	0.110	2.371	42.556	**35.993**	-
CRISP (ours)	**0.071**	**0.076**	**2.251**	**40.302**	36.782	-

Decoder	Whole Geometry		Edge Geometry		Smooth Geometry	
	CD↓	F↑	CD↓	F↑	CD↓	F↑
GT mask, 35 epochs						
w/o early fusion	1.134	0.753	8.030	0.192	0.956	0.776
w/ early fusion	**0.985**	**0.757**	**7.244**	**0.199**	**0.825**	**0.780**
Full system, converged						
Loss everywhere (no mask branch)	0.660	0.797	8.083	0.193	**0.537**	0.819
CRISP (ours)	**0.639**	**0.804**	**7.364**	**0.238**	**0.537**	**0.821**

retained because deep supervision ensures gradient signal reaches shallow encoder layers, improving robustness on thin-structure and high-azimuth-rate scenes beyond what the final IoU reflects.

Input conditioning (lower group). Removing the latent branch and conditioning on depth alone drops converged IoU from 0.788 to 0.630 (-20%) and F1 from 0.881 to 0.773. The depth hint provides high-frequency geometric cues, but cannot distinguish a true geometric void (sky, open road) from a sensor shadow or a sparse-return region caused by surface properties. The encoder latent carries semantic layout that resolves this ambiguity, enabling the predictor to correctly assign near-zero return probability to open voids and near-one probability to solid surfaces regardless of local depth texture.

Table 15: Mask predictor ablation (SVD $\times 3$, nuScenes). *Upper*: cumulative loss composition, depth + latent input, 10 ep. each. *Lower*: input conditioning, full loss set, converged. † Stage 6 is the deployed configuration; the marginal gap vs. Stage 5 is within single-run noise but deep supervision is retained for robustness on thin-structure regions.

Config	IoU↑	Prec.↑	Rec.↑	F1↑
Loss composition (depth + latent, 10 ep.)				
Focal only	0.758	0.860	0.865	0.862
+ Dice	0.827	0.864	0.951	0.906
+ Edge loss	0.838	0.871	0.957	0.912
+ Laplacian	0.841	0.869	**0.963**	0.914
+ Pixel mining	**0.845**	0.874	0.962	0.916
+ Deep sup. (full)†	0.844	**0.874**	0.961	**0.917**
Input conditioning (full losses, converged)				
Depth only	0.630	0.717	0.839	0.773
Depth + latent (ours)	**0.788**	**0.862**	**0.901**	**0.881**

Cross-backbone replication (LiDM). The ablations above all use the frozen SVD $\times 3$ backbone. Table 16 reruns the two central design choices, latent-conditioning mechanism and Euler step count, on the frozen LiDAR-native LiDM backbone instead (24 blocks, 5 steps, 20 epochs unless varied; matched seed, data, schedule). The same choices win on LiDM: early+mid concatenation beats cross-attention and register tokens (on SVD $\times 3$ we instead ablated early fusion, Table 14), and 5 Euler steps gives the lowest val L1, matching the operating point chosen on SVD $\times 3$ (Fig. 4). The mask-loss study behaves differently: LiDM support IoU starts at 0.9888, and no single term moves it

by more than 0.0011 across a six-way leave-one-out (Table 8), so there is little left for the objective to recover once the LiDAR-native backbone is already near ceiling. The same pattern appears on the depth side, where removing importance weighting costs 25.2% val L1 on nuScenes against only 5.9% on LiDM/KITTI-family (Table 6). One confound remains: the LiDAR-native runs are also the 64-beam runs, so latent match and return density are entangled.

Table 16: Design-choice replication on the frozen LiDAR-native LiDM backbone (KITTI-family). Validation L1 (↓); cf. the SVD $\times 3$/nuScenes ablations in Table 14 and Fig. 4.

Design axis	Variant	Val L1 ↓
Latent conditioning	Cross-attention	0.1872
	Register tokens	0.1156
	Early+mid concat (ours)	0.0446
Euler steps	1	0.0643
	3	0.0527
	5 (ours)	0.0446
	10	0.0789

Inference latency. Table 17 reports encoder and decoder throughput on $8\times$ H200 GPUs with `torch.compile`, batch size 25, and 5 diffusion sampling steps. The most notable result is Wan $\times 3$: CRISP's decoder (3.84 ms/frame) is $2.5\times$ faster than the inherited Wan2.1 VAE decoder (9.71 ms/frame), because the original video VAE decoder is a full temporal video-decoding stack designed for high-resolution RGB video: a substantially heavier operation than CRISP's single-channel LiDAR-targeted DiT. The end-to-end CRISP pipeline on Wan $\times 3$ therefore runs at 86 fps, faster than the baseline at 57 fps, demonstrating that decoder replacement can simultaneously improve geometric quality and reduce latency when the inherited decoder is overbuilt for the target modality. For LiDM and SVD, CRISP's 5-step diffusion decoder adds 23.06 ms and 3.92 ms per frame respectively; both remain real-time capable (41 and 215 fps) and the overhead is reducible via distillation [45].

Table 17: End-to-end inference throughput (ms/frame and fps). Encoder cost is shared between baseline and CRISP. Measured on $8\times$ H200, `torch.compile`, batch 25, 5 Euler steps. †CRISP decoder is faster than the inherited Wan2.1 VAE decoder.

Backbone	Encoder (ms/f)	Decoder (ms/f)		End-to-end	
		Base.	CRISP	Base. fps	CRISP fps
LiDM	1.31	1.39	23.06	369	41
SVD $\times 3$	0.74	1.49	3.92	448	215
Wan $\times 3$	7.78	9.71	$3.84^{\dagger}$	57	**86**

Decoder capacity: parameters and FLOPs. Table 4 reports the parameter, FLOP, and latency comparison underlying the capacity controls discussed in the main text (paragraph "Decoder capacity"). CRISP is essentially the same ≈516M network in every row (DiT decoder plus latent adapter ≈502M parameters, support-mask U-Net 13.8M; only the adapter's input width changes per backbone), so the ratios differ only through the size of the inherited decoder being replaced. Params and FLOPs are measured at batch 1, 5 Euler steps, and each setting's native range-map resolution (32×1024 for SVD/Wan on nuScenes, 64×1024 for LiDM on KITTI-family), so the identical network performs $\approx 2\times$ the FLOPs in the LiDM row. On LiDM, the same fixed-capacity step sweep as Fig. 4 gives val L1 0.0446 (5 steps) rising to 0.0643 (1 step) if iterative refinement is removed, i.e. 44% higher error, with no change to parameters, conditioning, or data. One confound: LiDM's baseline decoder is already closest to its reconstruction ceiling, so part of its comparatively small gain (FSVD -17.4%, FPVD 5.4% worse) despite receiving the most extra capacity ($60\times$ params, $22.8\times$ FLOPs) is headroom rather than a capacity effect; SVD $\times 3$ and Wan $\times 3$ receive far less extra capacity (7–8$\times$ params, 3.4–3.7$\times$ FLOPs) and improve the most (FSVD -74.3% and -67.8%).

Table 18: Licenses for datasets, pretrained models, and third-party code used in this work.

Asset	Reference	License
KITTI-360	[28]	CC BY-NC-SA 3.0
SemanticKITTI	[2]	CC BY-NC-SA 4.0
nuScenes	[5]	CC BY-NC-SA 4.0
LiDM	[42]	MIT
SVD VAE	[3]	Stability AI Community License
Wan2.1 VAE	[53]	Apache 2.0
ChamferDistancePytorch	[11]	MIT

F Asset Licenses

G Extended Qualitative Results

Except for the world-model samples in Fig. 10, each figure shows the full-frame point cloud and the corresponding range map (horizontal strip); inset boxes indicate the cropped region shown on the right. Colors follow Fig. 1: blue is ground truth, green is the inherited baseline decoder, and red is CRISP.

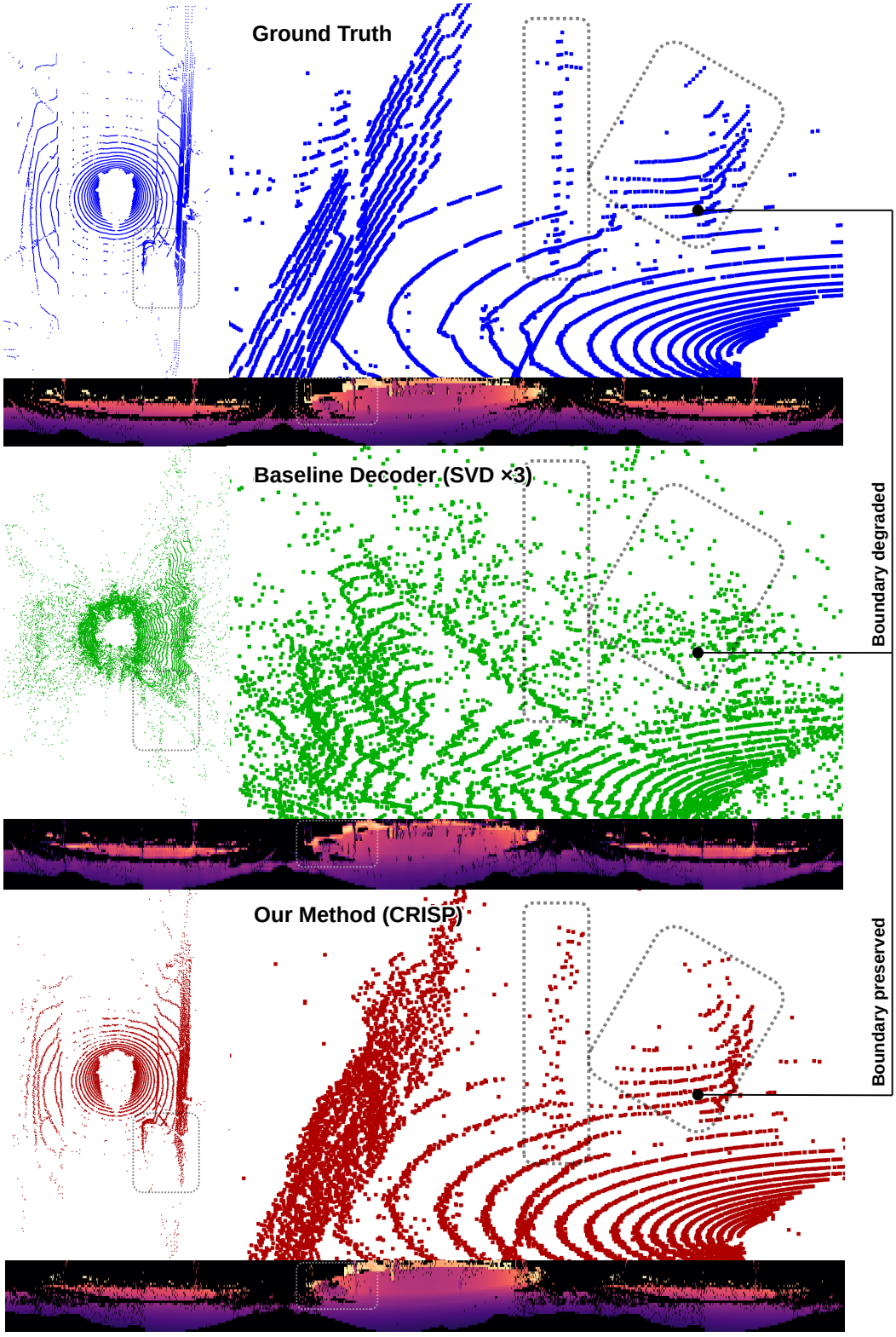

Figure 5: Extended qualitative comparison, SVD $\times 3$ (frozen encoder, nuScenes). The baseline decoder introduces heavy scan-line scatter at depth discontinuities: beam-level structure of a vehicle and pole, visible in ground truth, dissolves into noise clouds in the baseline reconstruction. CRISP restores continuous scan-line geometry and suppresses the boundary-bridging artifacts highlighted by the inset box.

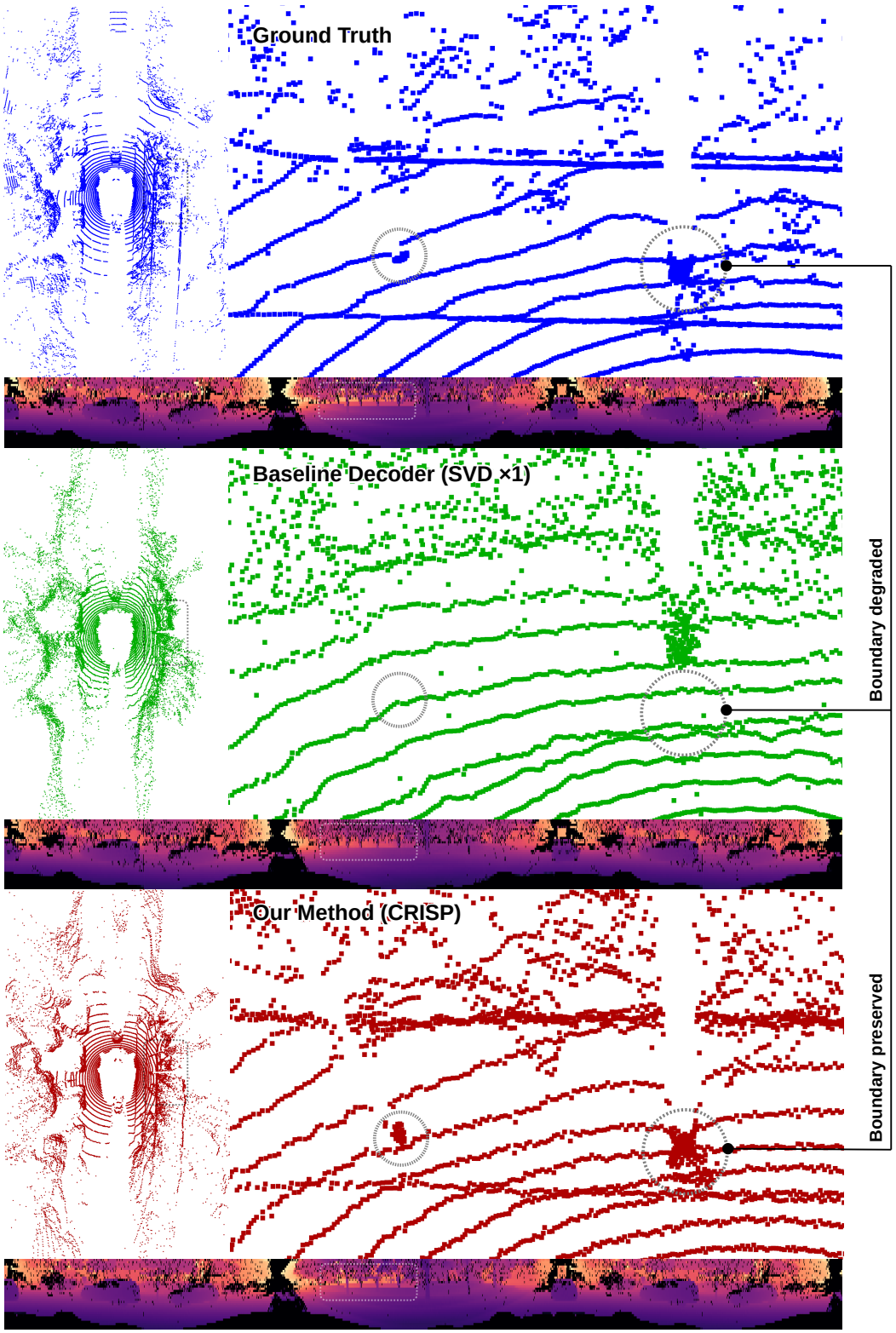

Figure 6: Extended qualitative comparison, SVD $\times 1$ (encoder-adapted, nuScenes). The baseline decoder misplaces a mid-range object cluster (circled tree trunks): points smear outward rather than forming a compact boundary. CRISP localises the cluster accurately and sharpens the depth contour at the foreground-background transition indicated by the inset box.

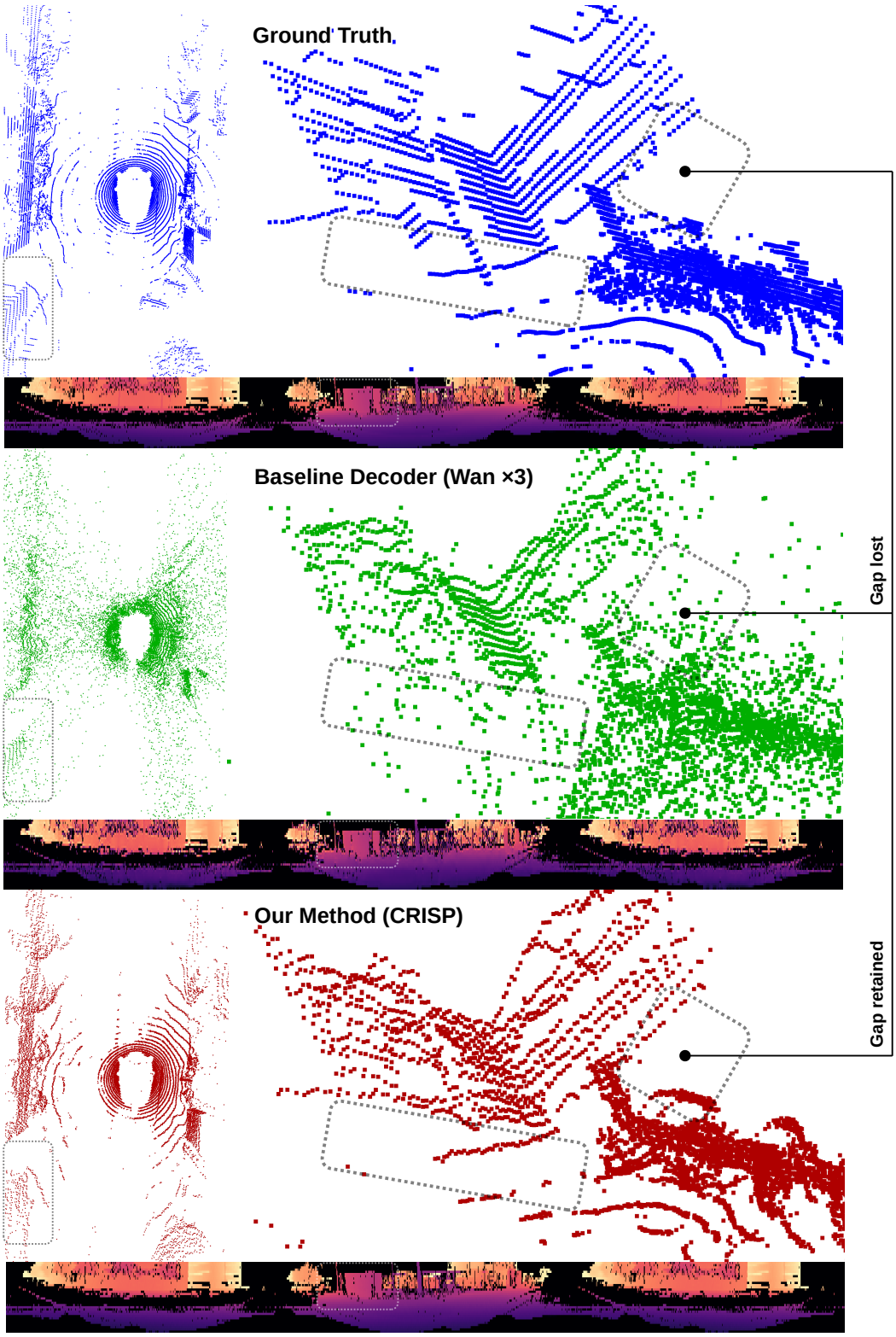

Figure 7: Extended qualitative comparison, Wan $\times 3$ (frozen encoder, nuScenes). The highlighted region contains a foreground-background scene depth gap that the baseline decoder bridges: points fill in between the two surfaces, erasing the geometric discontinuity. CRISP retains the gap as a sharp void, consistent with the ground-truth scan.

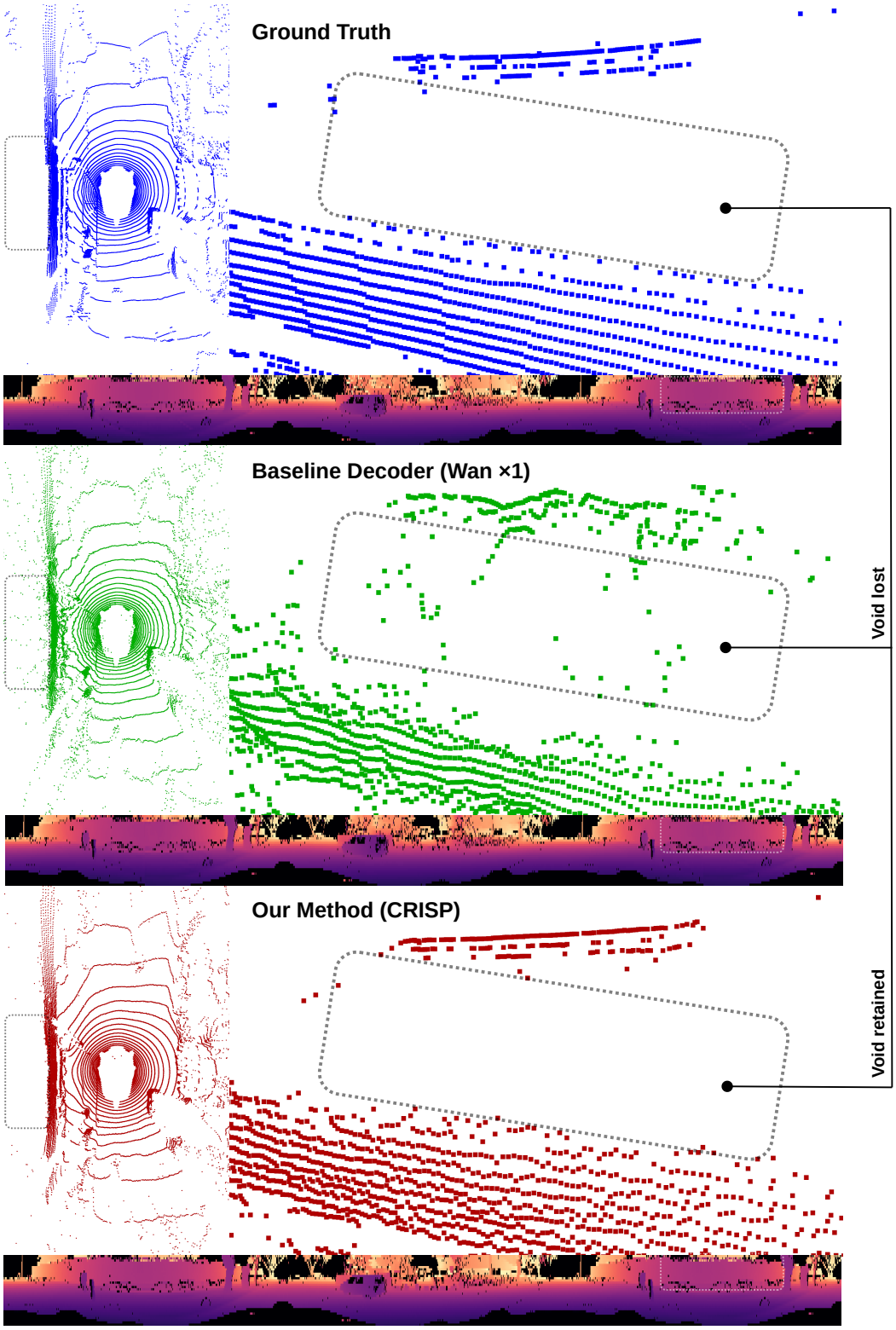

Figure 8: Extended qualitative comparison, Wan $\times 1$ (encoder-adapted, nuScenes). The baseline decoder incorrectly fills the open void region visible in ground truth (inset box): spurious returns appear where no surface exists. CRISP preserves the void, producing a sparse valid-return map that matches the support structure of the real scan.

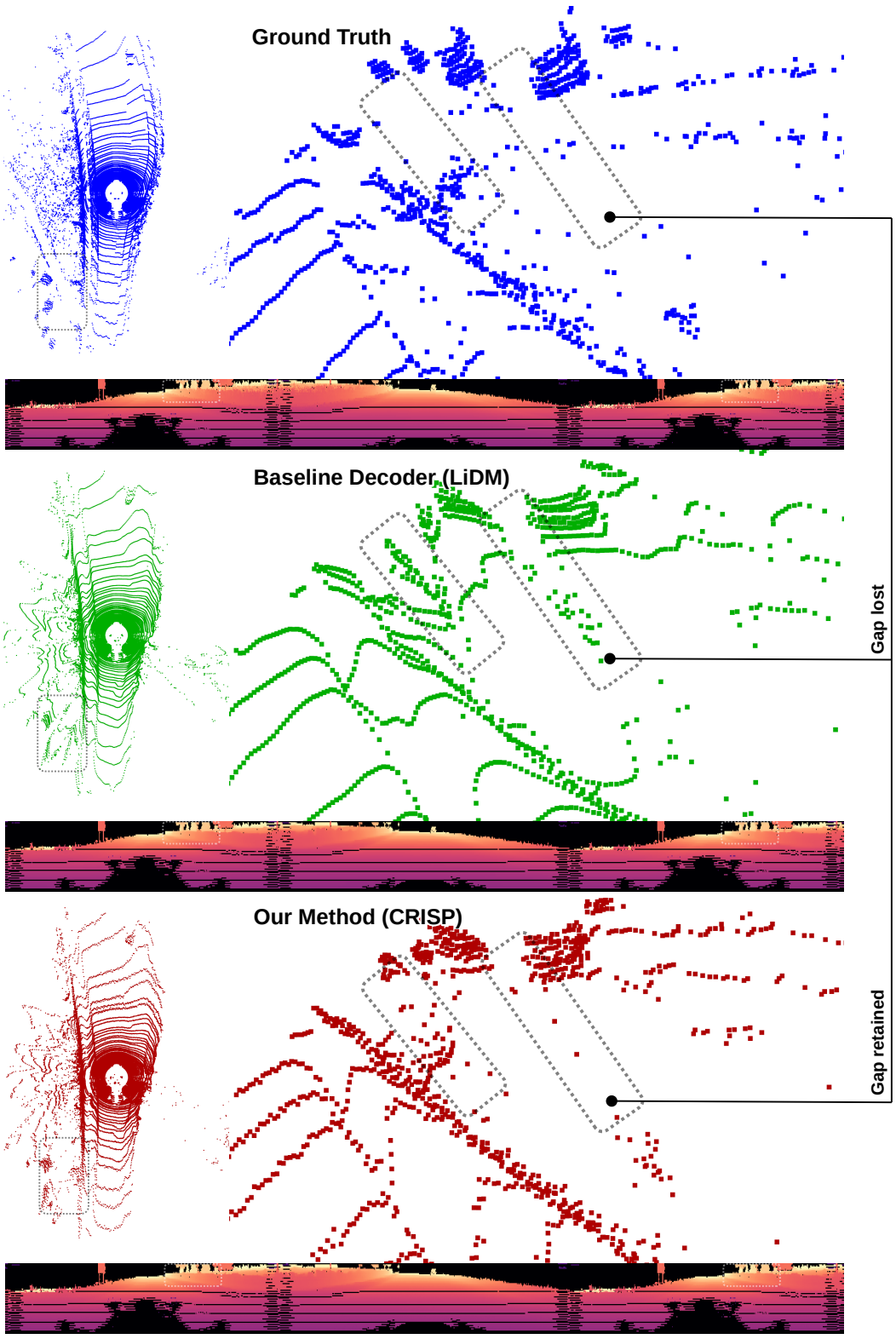

Figure 9: Extended qualitative comparison, LiDM (frozen encoder, KITTI family). The 64-beam KITTI geometry makes foreground-background depth gaps particularly sharp. The baseline decoder bridges the gap in the highlighted region, collapsing two geometrically distinct surfaces. CRISP reproduces the correct depth discontinuity, matching the gap geometry visible in the ground-truth scan.

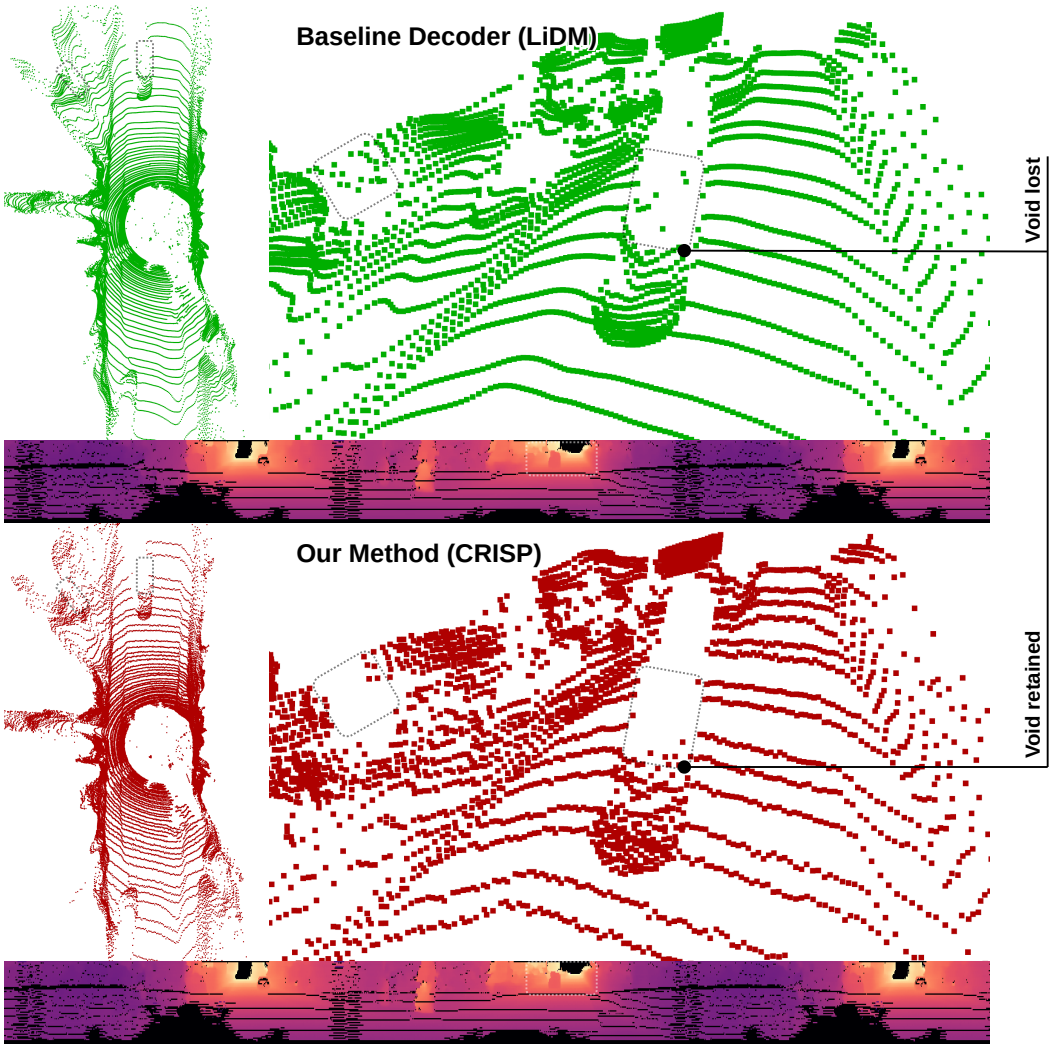

Figure 10: Extended qualitative comparison, LiDM world model (plug-and-play decoding, KITTI family). No ground truth is shown because these are generative samples from the frozen LiDM latent generator; both rows decode the same sampled latent. The baseline decoder introduces grid-aligned blocky artifacts and smears foreground boundaries. CRISP, swapped in without retraining the generator, produces cleaner scan geometry and sharper depth contours from the same latent input.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction claim a plug-in pixel-space LiDAR decoder that works across multiple latent backbones and improves reconstruction, especially near discontinuities. These claims match the method and results sections. The paper also discusses non-uniform behavior across metrics, including perceptual or distributional regressions in some adapted and plug-and-play settings, rather than claiming uniform superiority.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: A dedicated Limitations paragraph at the end of the Results section discusses: (1) operation on a single depth channel without remission or intensity; (2) reliance on range-map assumptions shared by prior latent LiDAR pipelines; (3) residual high-frequency noise on smooth surfaces; (4) possible thin-structure flicker due to independent frame decoding; and (5) inference overhead of 3.9–23 ms/frame for SVD and LiDM, with distillation noted as a mitigation path.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present formal theoretical results such as theorems, lemmas, or proofs. Its equations define the model and training objectives rather than results requiring mathematical proof.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The method section specifies the model forward pass, adapter, diffusion decoder, support mask predictor, and main optimization objectives. The appendix provides architecture details, dataset preprocessing, splits, augmentations, optimizer settings, learning rates, training stages, EMA, dropout, gradient clipping, loss weights, and inference protocol. Main tables report mean and standard deviation over five stochastic inference runs from a fixed checkpoint. The boundary-focused evaluation defines edge pixels by thresholding valid-only local depth discontinuities, with the threshold value reported in the Experimental Setup section and implementation details provided in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Code and checkpoints are not yet publicly released. All evaluation datasets are publicly available under their respective licenses, and the pretrained backbones are publicly available. The paper provides detailed training and evaluation specifications to support reproducibility, and code, training scripts, evaluation scripts, and CRISP checkpoints will be released on the project page (`https://andrea25512.github.io/CRISP/`).

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The Experimental Setup section specifies datasets, split sizes, range-map resolutions, latent backbones, comparison regimes (frozen vs. encoder-adapted vs. plug-and-play), and all reported metrics with their definitions. Appendices C–D provides full training hyperparameters, augmentation protocol, diffusion schedule, and the backbone-specific adapter configurations used for each setting.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All main quantitative tables report mean and standard deviation over five stochastic inference runs that differ only in the random seed used for diffusion sampling and Bernoulli support sampling. No retraining is performed between runs; each run uses the same final-stage checkpoint, namely Stage F2 for frozen-encoder variants and Stage A3 for encoder-adapted variants. The reported variability therefore reflects sampling stochasticity rather than training-seed variability.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The appendix reports the GPU type and memory used for training (8×A100-80GB), mixed-precision/DDP setup, batch structure, optimizer configuration, and approximate CRISP training cost: about 36 h per full run, consisting of roughly 24 h for the DiT-only stage and 12 h after introducing the support mask branch, corresponding to about 288 GPU-hours per run. It also reports inference throughput on 8×H200 GPUs with `torch.compile`, batch size 25, and 5 Euler steps. Baseline VAE adaptation and ablation runtimes are excluded from the reported training-cost estimate.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work is an offline methodological study on LiDAR reconstruction using publicly licensed automotive LiDAR datasets and pretrained models. It does not involve human subjects, deceptive experimentation, or private/sensitive data collection. To the authors' knowledge, the study conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The intended positive impact is improved synthetic LiDAR fidelity for autonomous-driving simulation, which may support safer perception and planning evaluation and reduce reliance on costly real-world collection. Potential negative impacts include misuse of high-fidelity synthetic LiDAR to generate deceptive sensor traces, spoof perception systems, or create misleading safety-audit evidence. These risks are moderated by the domain-specific nature of CRISP, which is a decoder component rather than a standalone broad generative model, but they remain relevant for responsible release and downstream use.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: CRISP is a domain-specific decoder component for automotive LiDAR range maps and is not a broad content-generation model, scraped dataset, or high-risk pretrained foundation model. The anticipated release consists of code and checkpoints trained on publicly licensed automotive LiDAR datasets, so dedicated release safeguards beyond standard licensing, documentation, and responsible-use notes are not applicable.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets and pretrained models used in experiments are publicly available and cited. Full license information for each asset is provided in Appendix F.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [No]

    Justification: The paper introduces new CRISP decoder checkpoints, latent adapter weights, and training/evaluation code, but these assets are not yet publicly released. The release on the project page will include documentation covering training data, intended use, limitations, licenses, dependencies, and reproduction instructions.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing, surveys, manual human-subject studies, or compensated participant labor. Therefore instructions to participants, screenshots, and compensation details are not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work does not involve human subjects or participant studies. Accordingly, IRB approval and participant-risk disclosures are not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

Justification: LLMs are not part of the core method, experiments, or scientific claims in this paper. Under the checklist guidance, writing or editing assistance alone does not require declaration here.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.