PointDiT: Pixel-Space Diffusion for Monocular Geometry Estimation

Haofei Xu [1 2 3] **Rundi Wu** [1] **Philipp Henzler** [1] **Nikolai Kalischek** [1] **Michael Oechsle** [1] **Fabian Manhardt** [1]
Marc Pollefeys [2 4] **Andreas Geiger** [3 5] **Federico Tombari** [1 6] **Michael Niemeyer** [1]

https://haofeixu.github.io/pointdit

Abstract

State-of-the-art single-image 3D reconstruction methods often rely on complex hybrid architectures and loss functions, or compress geometry into latent spaces in order to leverage pre-trained latent diffusion models. In this work, we show that such architectural overhead and intricate loss formulations are unnecessary. We introduce a minimalist pixel-space Diffusion Transformer, built on a plain ViT, that operates directly on raw 3D point map patches and is conditioned on image tokens from a pre-trained DINOv3. Unlike existing latent diffusion approaches, we train our diffusion backbone entirely from scratch, eliminating the need for point map tokenizers. Despite its simplicity, our approach surpasses complex latent-based diffusion models while remaining significantly simpler than hybrid alternatives. Notably, it produces sharper geometric structure and is more robust in highly ambiguous regions, such as transparent objects.

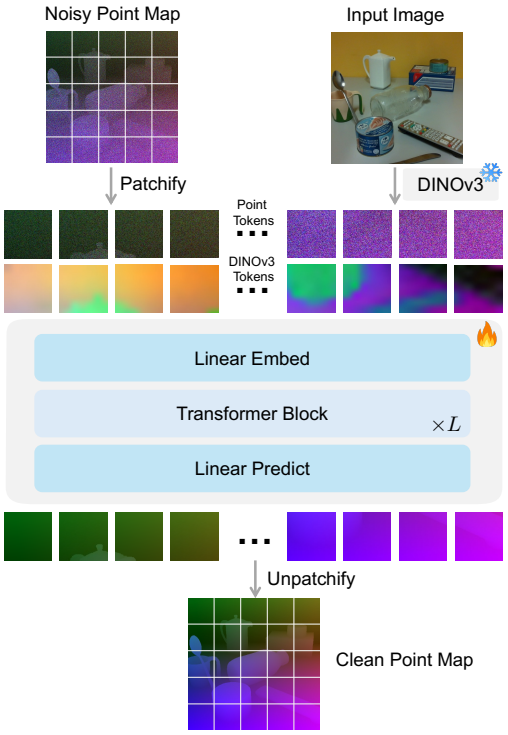


Figure 1. **PointDiT.** A minimalist pixel-space Diffusion Transformer operating directly on raw point map patches, conditioned on image tokens from a pre-trained DINOv3. The 3D point map ($H \times W \times 3$) is visualized as an RGB image, with color encoding the spatial (X, Y, Z) coordinates.

1. Introduction

Monocular geometry estimation is a fundamental building block of 3D scene understanding, bridging 2D visual inputs and 3D spatial structure. In this work, we focus on predicting dense 3D point maps from single RGB images (Wang et al., 2025b; Piccinelli et al., 2025). Unlike depth maps, which capture only scalar distance and require camera intrinsics to recover 3D structure, point maps represent scene geometry directly in the camera coordinate system, enabling immediate 3D reconstruction without knowing the camera's calibration. However, mapping a single 2D image to a dense 3D representation remains ill-posed, owing to the inherent scale and depth ambiguities of perspective projection.

Existing approaches to this challenge fall broadly into two categories. The first comprises deterministic regression models (Yang et al., 2024; Bochkovskii et al., 2025; Piccinelli et al., 2025). These methods often rely on complex hybrid architectures (Wang et al., 2025b;c;a; Lin et al., 2026) that combine Vision Transformers (ViT) (Dosovitskiy, 2020) with convolutions (Ranftl et al., 2021), and require intricate loss functions (Wang et al., 2025b) to regularize training. Moreover, because of the task's inherent ambiguity, deterministic regressors tend to predict the mean of the output

[1]Google [2]ETH Zurich [3]University of Tübingen, Tübingen AI Center [4]Microsoft [5]KE:SAI [6]Technical University of Munich. Correspondence to: Haofei Xu <haofei.xu@inf.ethz.ch>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.

distribution, often yielding over-smoothed geometry that lacks high-frequency detail, particularly in complex scene regions (Figure 2b).

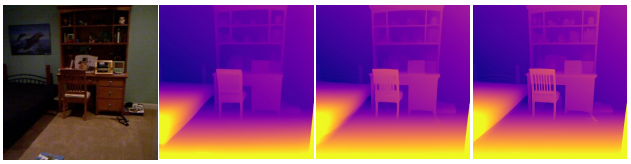
The second category seeks to resolve this ambiguity with Latent Diffusion Models (LDMs) (Rombach et al., 2022), such as GeometryCrafter (Xu et al., 2025b). Although these methods exploit generative priors, they require compressing point maps into a latent space via a Variational Autoencoder (VAE) (Kingma & Welling, 2014). Building an expressive latent space for geometric data (*e.g.*, point clouds) is non-trivial: their unbounded range and the relative scarcity of geometric data can limit the autoencoder's reconstruction quality and out-of-distribution generalization. In addition, constructing such a space often demands sophisticated tokenizer designs (Xu et al., 2025b). As a result, these methods frequently lose information during encoding and decoding, struggling to reconstruct fine geometric structures accurately (Figure 2a). Furthermore, a fundamental trade-off exists between VAE reconstruction fidelity and diffusion generation capabilities due to their conflicting optimization objectives (Black Forest Labs, 2025), which inherently bounds the potential of latent diffusion models for geometric tasks.

In this work, we show that such architectural overhead and intricate loss formulations are unnecessary. Inspired by JiT (Li & He, 2026), we introduce a minimalist pixel-space diffusion framework that trains directly on the raw point map space. This lets us exploit the probabilistic nature of diffusion to model ambiguous regions, without the signal degradation introduced by VAEs. Our architecture is simple by design: a plain Vision Transformer (ViT) operating on point map patches. A key element of our training recipe is the x-prediction objective, *i.e.*, predicting the clean point map directly, rather than the v-prediction target commonly used in flow matching (Salimans & Ho, 2022; Xu et al., 2025a). Extending the findings of JiT (Li & He, 2026) beyond image generation, we show that this objective is highly effective for geometric data and yields substantially better point map estimation.

To guide geometry prediction, our diffusion model is conditioned on the input RGB image. Although it already works well with naive linear patchification (Dosovitskiy, 2020), we find that injecting strong priors substantially improves performance. Specifically, we adopt DINOv3 (Siméoni et al., 2025) as a robust general-purpose feature extractor. Conditioning our plain ViT backbone on DINOv3 tokens bridges powerful priors from representation learning with diffusion training (Yu et al., 2025; Zheng et al., 2025).

We show that this streamlined approach surpasses complex latent-based diffusion models (Xu et al., 2025b) while remaining significantly simpler than hybrid deterministic alternatives (Wang et al., 2025b; Lin et al., 2026). Our model further excels at generating sharp geometric boundaries and resolving depth in highly ambiguous scenarios, such as transparent objects. PointDiT achieves highly competitive results with just one-step diffusion, and its structural details improve further with additional sampling steps. Beyond this specific task, our results suggest that pixel-space diffusion extends naturally beyond natural images to structured geometric signals such as 3D point maps, pointing toward a simpler paradigm for future 3D and 4D generation models.

(a) **Point VAE reconstruction.** Even without diffusion, the VAE-reconstructed point cloud exhibits substantial noise, which fundamentally limits the quality attainable by latent diffusion models.



(b) **Structural details.** PointDiT recovers intricate, thin structures such as the chair more faithfully than GeometryCrafter (latent diffusion) and MoGe-2 (deterministic regression). Here we visualize the z-depth from the predicted 3D point maps.

Figure 2. **Comparison with latent diffusion and regression.** The two dominant paradigms each have an inherent limitation: (a) the VAE in latent diffusion models introduces reconstruction noise that caps the attainable quality, while (b) deterministic regression over-smooths fine geometric structures. PointDiT avoids both.

2. Related Work

Latent Diffusion Models. Latent Diffusion Models (LDMs) (Rombach et al., 2022) have become the dominant paradigm for high-resolution image synthesis, decoupling the modeling of semantic content from perceptual detail by operating in a compressed latent space. Following this success, recent works adapt LDMs to geometric tasks (Ke et al., 2024; He et al., 2024; Szymanowicz et al., 2025; Hu et al., 2025). For instance, GeometryCrafter (Xu et al., 2025b) and generative depth estimators (Ke et al., 2024) use Variational Autoencoders (VAEs) to encode geometric maps into latent tokens. Although this reduces computational cost, the compression is fundamentally lossy: constructing a tokenizer that preserves the high-frequency precision required for 3D geometry is non-trivial, and standard image VAEs often smooth away fine structural details or introduce artifacts during decoding (Figure 2a). In contrast, our approach bypasses the latent space entirely (Li & He, 2026).

By avoiding this architectural overhead and the associated information loss, we recover substantially sharper geometric boundaries (Figure 2b).

Pixel-Space Diffusion Models. Early diffusion models operate directly in pixel space (Ho et al., 2020). Latent Diffusion Models (LDMs) subsequently shift generation into the latent space, and the Diffusion Transformer (DiT) (Peebles & Xie, 2023) replaces the conventional U-Net backbone with a Vision Transformer (ViT), achieving state-of-the-art class-conditional image generation. More recently, JiT (Li & He, 2026) shows that a ViT can be trained directly in pixel space, using direct data prediction (x-prediction) to cope with the high dimensionality of pixel space. While these advances have focused primarily on 2D image synthesis, we extend this minimalist pixel-space philosophy to dense 3D geometry estimation. By treating 3D point maps as multi-channel images and training a plain ViT backbone from scratch, we show that pixel-space diffusion is not only computationally feasible for 3D geometry but also superior to complex alternatives in reconstructing sharp details and resolving ambiguities.

Representation Learning and Generative Models. A recent line of work connects representation learning with generative models. REPA (Yu et al., 2025) and VA-VAE (Yao et al., 2025) observe that pre-trained vision encoders can dramatically improve generative diffusion models by regularizing their latent space. RAE (Zheng et al., 2025) replaces the VAE in latent diffusion models with pre-trained representation autoencoders (*e.g.*, DINOv2). At a high level, PointDiT shares this spirit, connecting DINOv3 with diffusion models. However, there are several key differences. First, RAE must be trained in two stages (reconstruction decoder and diffusion), whereas PointDiT is end-to-end. Second, RAE uses v-prediction, which requires scaling up the Transformer width, whereas PointDiT uses x-prediction, allowing us to train a smaller variant, PointDiT-B. Third, RAE requires 50 sampling steps for image synthesis, whereas PointDiT can perform one-step or few-step generation.

Monocular Depth Estimation. Estimating dense geometry from a single image is a longstanding problem in computer vision. Traditional discriminative approaches cast this as a regression task, using Convolutional Neural Networks (CNNs) (Eigen et al., 2014) or, more recently, Transformers such as DPT (Ranftl et al., 2021) and Depth Anything (Yang et al., 2024) to predict scalar depth maps. However, depth maps are 2.5D representations that require known camera intrinsics to be lifted into 3D, which are often unavailable in unconstrained settings. Generative approaches such as Marigold (Ke et al., 2024) instead repurpose pre-trained image diffusion models (*e.g.*, Stable Diffusion) for depth estimation. Although these methods exploit strong generative priors, they remain fundamentally limited by the quality of VAEs. Moreover, there is often a trade-off between VAE reconstruction and diffusion generation, and balancing the two requires additional effort (Black Forest Labs, 2025). More recently, PPD (Xu et al., 2025a) applies pixel-space diffusion to monocular depth estimation. However, PPD still uses the v-prediction target, which performs worse than x-prediction in our controlled comparisons (Table 3(a)).

Monocular Point Map Estimation. To overcome the limitations of scalar depth, point map estimation predicts dense 3D coordinates xyz directly in the camera coordinate system, enabling holistic 3D reconstruction without intrinsic calibration. The current state of the art is dominated by deterministic feed-forward models such as MoGe (Wang et al., 2025b;c). These methods typically employ complex hybrid architectures that fuse ViTs with convolutional layers and rely on intricate loss functions to enforce geometric consistency. Despite their efficacy, deterministic regressors suffer from the inherent ambiguity of monocular projection, tending to predict the mean of the distribution. This often yields over-smoothed geometry, particularly in regions of high uncertainty or transparency (Figure 5b). We address this by casting point map estimation as a probabilistic generation task, allowing our model to capture sharp, high-frequency detail that deterministic baselines fail to resolve.

3. Approach

We address dense point map prediction from a single RGB image. Formally, given an input image $\mathbf{c} \in \mathbb{R}^{H\times W\times 3}$, our goal is to estimate the corresponding point map $\mathbf{x} \in \mathbb{R}^{H\times W\times 3}$, in which each pixel encodes its 3D spatial (X, Y, Z) coordinates. To model the inherent ambiguities of this single-image setting, we propose a flow matching framework parameterized by a Vision Transformer (ViT) (Dosovitskiy, 2020; Peebles & Xie, 2023). Our method learns to transport a simple Gaussian noise distribution to the data distribution of point maps, conditioned on the input image.

3.1. Point Map Generation with Flow Matching

Flow Matching. We adopt the flow matching formulation to model point map generation from a single image. Flow matching learns an Ordinary Differential Equation (ODE) that continuously transforms a prior noise distribution p_0 into the data distribution p_1.

Let $\mathbf{z}_t$ denote the state at time $t \in [0, 1]$, defined by a linear interpolation between a noise sample $\boldsymbol{\epsilon} \sim p_0 = \mathcal{N}(\mathbf{0}, \mathbf{I})$ and a ground-truth data sample $\mathbf{x} \sim p_1$:

$$\mathbf{z}_t = t \cdot \mathbf{x} + (1 - t) \cdot \boldsymbol{\epsilon}. \tag{1}$$

In this formulation, $t = 0$ corresponds to pure noise ($\mathbf{z}_0 = \boldsymbol{\epsilon}$) and $t = 1$ to the clean data ($\mathbf{z}_1 = \mathbf{x}$). The vector field

$\mathbf{v}_t(\mathbf{z}_t)$ that generates this probability path is given by the time derivative of the state:

$$\mathbf{v}_t = \frac{d\mathbf{z}_t}{dt} = \mathbf{x} - \boldsymbol{\epsilon}. \tag{2}$$

This linear path induces a constant velocity for each sample pair $(\mathbf{x}, \boldsymbol{\epsilon})$, ensuring straight-line transport between the noise and data distributions.

Image Conditioned Flow Matching. We extend this framework to model the conditional distribution $p(\mathbf{x}|\mathbf{c})$, where $\mathbf{c}$ is the input RGB image and $\mathbf{x}$ is the target dense point map. Specifically, we learn a conditional vector field $\mathbf{v}_\theta(\mathbf{z}_t, t|\mathbf{c})$ that predicts the target velocity defined in Equation (2). By conditioning on $\mathbf{c}$, the model exploits the image's spatial context to resolve geometric ambiguity, steering the flow toward the target point map.

Point Map Normalization. Unlike standard RGB images bounded within $[0, 1]$, dense point maps exhibit varying coordinate ranges depending on the scene domain (*e.g.*, indoor *vs.* outdoor scenes). In our flow matching formulation, the training target relies on the interpolation $\mathbf{z}_t = t\mathbf{x} + (1-t)\boldsymbol{\epsilon}$, where the noise $\boldsymbol{\epsilon}$ follows a fixed standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. If the scale of the point data $\mathbf{x}$ far exceeds that of the noise, the data signal dominates the interpolation path even at near-zero time steps. This prevents the noise from destroying the data structure, destabilizing diffusion training. To mitigate this, we standardize the point maps before training. For each point map, we compute the centroid $\boldsymbol{\mu}$ and a scalar scale factor s, defined as the mean Euclidean distance of the points from the centroid. The normalized data $\tilde{\mathbf{x}}$ is given by:

$$\tilde{\mathbf{x}} = \frac{\mathbf{x} - \boldsymbol{\mu}}{s}. \tag{3}$$

This normalization brings the data to a scale comparable to the noise prior, facilitating stable flow matching. Our model is trained in this normalized space, and thus its point map predictions are affine-invariant, *i.e.*, recovered up to an unknown scale and shift.

Sky Processing. To accommodate the effectively infinite depth of the sky in outdoor scenes, we exclude sky points when computing the normalization statistics $\boldsymbol{\mu}$ and s, and then, in the resulting normalized frame, project them onto a virtual sphere of fixed radius 3 (corresponding to 3σ of the standard normal noise prior). Since this radius is only a synthetic proxy for the true depth, we down-weight sky pixels in the training loss rather than masking them out entirely (Section 3.3). Retaining a small but nonzero weight keeps the sky supervised as a distant background, so its geometry does not become arbitrary in the absence of supervision, while preventing its pseudo-depth values from dominating the optimization. This, in turn, enables stable joint training across heterogeneous indoor and outdoor datasets. At inference, we discard predicted 3D points whose norm exceeds 2.9, a small margin below the sky sphere of radius 3.

3.2. Architecture

We implement F_θ as a Vision Transformer (ViT), which serves as our pixel-space Diffusion Transformer. The network takes the noisy point map $\mathbf{z}_t$, the current time step t, and the conditioning image $\mathbf{c}$ as input. Crucially, unlike previous flow matching models that typically predict the velocity, our network is trained to predict the clean point map. Inspired by JiT (Li & He, 2026) for 2D image generation, we show that this clean data prediction target is likewise crucial for 3D point map data. Figure 1 shows an overview of our architecture.

Point Map Patchification. The noisy point map $\mathbf{z}_t \in \mathbb{R}^{H\times W\times 3}$ has the same spatial resolution as the input image, so applying a ViT directly at the pixel level would be prohibitively expensive. We therefore patchify it, partitioning the map into a regular grid of non-overlapping $p \times p$ patches to reduce the ViT sequence length from pixels to patches. This yields $N = (H/p) \times (W/p)$ patches, each flattened into a vector of size $3p^2$. These vectors are then mapped to the embedding dimension D by a learnable linear projection ϕ, giving the point map tokens $\mathbf{T}_z = \phi(\text{Patchify}(\mathbf{z}_t)) \in \mathbb{R}^{N\times D}$.

Image Conditioning. The conditioning image $\mathbf{c}$ provides structural guidance for generation. Since $\mathbf{c}$ is clean, unlike the noisy $\mathbf{z}_t$, we can exploit powerful pre-trained representations to encode it. Although a standard learnable linear patch embedding (Dosovitskiy, 2020) already works well, we find that extracting patch tokens with a frozen DINOv3 encoder (Siméoni et al., 2025) leads to better performance. Unlike RAE (Zheng et al., 2025), which uses only the last layer, we find it beneficial to combine DINOv3 features from multiple layers. In particular, we use four uniformly spaced intermediate layers, following the layer selection of the DPT (Ranftl et al., 2021) head. Unlike DPT, which relies on sophisticated convolutions to fuse these features, we simply concatenate the tokens along the channel dimension, capturing a rich feature hierarchy that ranges from low-level details to high-level abstractions. This yields a composite image representation $\mathbf{T}_c \in \mathbb{R}^{N\times 4D}$, where D is the per-layer feature dimension. To ensure spatial alignment, we use the same patch size $p = 16$ and embedding dimension D for both the point map and DINOv3 branches.

Image and Point Map Fusion. Given the spatial alignment between $\mathbf{T}_c$ and $\mathbf{T}_z$, we fuse the two modalities by channel-wise concatenation, forming the input $\mathbf{T}_{\text{in}} =$

Concat$(\mathbf{T}_c, \mathbf{T}_z) \in \mathbb{R}^{N\times 5D}$ to the Diffusion Transformer. A linear layer projects this from $5D$ to the embedding dimension D. The sequence is then processed by a stack of Transformer blocks (Dosovitskiy, 2020; Li & He, 2026), each comprising multi-head self-attention and an MLP.

Clean Point Map Prediction. The Diffusion Transformer outputs a sequence of refined tokens $\mathbf{T}_{\text{out}} \in \mathbb{R}^{N\times D}$. To recover the dense point map, we apply a linear prediction head that projects each token from D back to the flattened patch dimension $3p^2$, yielding patch vectors in $\mathbb{R}^{N\times 3p^2}$. An unpatchification operation then rearranges these vectors into the original spatial grid $(H/p) \times (W/p) \times p \times p \times 3$ and permutes the dimensions to form the full-resolution tensor $\hat{\mathbf{x}} \in \mathbb{R}^{H\times W\times 3}$. This tensor is the model's estimate of the clean point map at the current step.

3.3. Training

Noise Schedule. To sample the time step $t \in [0, 1]$ during training, we use a logit-normal distribution (Esser et al., 2024), following JiT (Li & He, 2026). Specifically, we draw $z \sim \mathcal{N}(\mu, \sigma^2)$ with $\mu = -0.8$ and $\sigma = 0.8$, and map it to the time domain through the sigmoid function, $t = \text{sigmoid}(z)$.

In our point map generation task, we observe that the sigmoid only asymptotically approaches its boundaries, so the model is never trained on the exact pure-noise state ($t = 0$). This creates a train-test discrepancy, since inference always starts at $t = 0$, and the model may then struggle to initiate the flow trajectory from the prior (Lin et al., 2024). To resolve this, we adopt a rectified sampling strategy: with probability $p_{\text{zero}} = 0.1$, we override the logit-normal sample and set $t = 0$ explicitly. This calibrates the model to the pure-noise distribution it encounters at the start of inference.

Velocity Loss. Although our network F_θ is parameterized to predict the clean point map $\hat{\mathbf{x}}$, we optimize it in velocity space (v-loss), following JiT (Li & He, 2026). In our experiments, this performs slightly better than computing the loss directly on $\hat{\mathbf{x}}$ (x-loss). During training, we construct the noisy input $\mathbf{z}_t$ (Equation (1)) and obtain the network prediction $\hat{\mathbf{x}} = F_\theta(\mathbf{z}_t, t, \mathbf{c})$. We then convert $\hat{\mathbf{x}}$ into an estimated velocity $\hat{\mathbf{v}}_t$ by rearranging the interpolation path:

$$\hat{\mathbf{v}}_t(\mathbf{z}_t, t) = \frac{\hat{\mathbf{x}} - \mathbf{z}_t}{1 - t}. \tag{4}$$

To ensure numerical stability as $t \to 1$, we clip the denominator $(1 - t)$ to a minimum threshold $\delta = 0.05$.

We minimize the Mean Squared Error (MSE) between this estimated velocity $\hat{\mathbf{v}}_t$ and the constant ground-truth velocity target $\mathbf{u}_t = \mathbf{x} - \boldsymbol{\epsilon}$:

$$\mathcal{L}_{\text{fm}} = \mathbb{E}_{\mathbf{x},t,\boldsymbol{\epsilon}}\left[\frac{1}{M}\sum_{i=1}^{M} w_i \left\|\hat{\mathbf{v}}_{t,i} - (\mathbf{x}_i - \boldsymbol{\epsilon}_i)\right\|_2^2\right], \tag{5}$$

where i indexes the M pixels and w_i is a per-pixel weight that down-weights sky pixels ($w_i = 0.01$ for sky pixels and $w_i = 1$ otherwise), as motivated in Section 3.1.

Relative Point Loss. The flow matching loss alone already yields an effective model. Nonetheless, because our model predicts the clean output directly in the original data space, without a VAE, it is straightforward to impose auxiliary losses on this output when necessary. To show this flexibility, we add a relative point loss. Point maps span a high dynamic range: distant points have large coordinate norms that dominate standard error metrics, leaving nearby details under-weighted. We therefore normalize each per-pixel error by the magnitude of the target point, which emphasizes the reconstruction of local detail:

$$\mathcal{L}_{\text{rel}} = \mathbb{E}_{\mathbf{x},t,\boldsymbol{\epsilon}}\left[\frac{1}{M}\sum_{i=1}^{M} w_i \frac{\|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_1}{\|\mathbf{x}_i\|_2 + \xi}\right], \tag{6}$$

where w_i is the same per-pixel sky weight as in Equation (5) and ξ is a small stability constant.

Total Loss. The final optimization objective is the weighted sum:

$$\mathcal{L} = \mathcal{L}_{\text{fm}} + \lambda\mathcal{L}_{\text{rel}}, \tag{7}$$

where $\lambda = 0.1$ is the loss weight. We train the full model end-to-end. Unlike existing methods (*e.g.*, MoGe (Wang et al., 2025b)) that typically rely on several regularization losses, our training is driven primarily by the flow matching loss, with only a single lightweight auxiliary term.

3.4. Inference

During inference, we recover $\mathbf{x}$ from pure noise $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, conditioned on the input image $\mathbf{c}$, by solving the ODE $d\mathbf{z}_t = \mathbf{v}_\theta(\mathbf{z}_t, t \mid \mathbf{c})\, dt$ from $t = 0$ to $t = 1$. We use a standard Euler solver with step size Δt. At each step t, we predict the clean data $\hat{\mathbf{x}}$, derive the velocity $\hat{\mathbf{v}}_t$, and update the state:

$$\mathbf{z}_{t+\Delta t} \leftarrow \mathbf{z}_t + \Delta t \cdot \hat{\mathbf{v}}_t. \tag{8}$$

This iterative process transports the sample along the learned linear trajectory to reconstruct the final point map.

Surprisingly, our model can perform single-step inference with competitive results, while additional steps further improve quality using the same model (Figure 3). We attribute this to the per-pixel alignment between the predicted point map and the conditioning image: each output location is

largely determined by its corresponding image feature, making the transport from noise to the target geometry nearly a direct mapping that is accurate even in one step, with additional steps mainly refining details. We further observe that our model can serve as a deterministic estimator at inference time, by initializing from all zeros instead of random noise (Table 2). A similar behavior was reported for diffusion-based depth estimation (Garcia et al., 2025), likely because the model learns to be robust to the input noise, or even to constant zeros.

4. Experiments

4.1. Datasets

We adopt a two-stage training strategy for efficient training. The model is first pre-trained at 256×256 resolution and then fine-tuned at 512×512, using only synthetic data throughout. For 256×256 pre-training, we use SceneNet-RGBD (McCormac et al., 2017), which provides approximately 5.36 million photorealistic RGB-D samples. The 512×512 fine-tuning stage uses a high-fidelity mixture of 11 synthetic datasets: Hypersim (Roberts et al., 2021), VKITTI2 (Cabon et al., 2020), UrbanSyn (Gómez et al., 2025), Synscapes (Wrenninge & Unger, 2018), TartanAir (Wang et al., 2020), OmniWorldGame (Zhou et al., 2025), EDEN (Lê et al., 2021), IRS (Wang et al., 2019), Dynamic Replica (Karaev et al., 2023), MVSSynth (Huang et al., 2018), and TartanGround (Wang et al., 2025d), totaling approximately 6.22 million samples. As all of these datasets are RGB-D, we convert their raw depth maps into point maps using the provided camera intrinsics. More dataset details are provided in the appendix (Table 4).

We train exclusively on synthetic data for two reasons. 1) Geometric precision: synthetic environments provide "pixel-perfect" ground-truth point maps, which are essential for learning high-quality, dense 3D distributions. 2) Domain agnosticism: because our architecture models the underlying geometry (point maps) rather than low-level image textures, the synthetic-to-real appearance gap matters less for our task. To further bridge the gap between synthetic and real-world distributions, we incorporate frozen features from a pre-trained DINOv3 backbone. These self-supervised representations provide robust, domain-invariant visual cues that allow our model to focus on geometric reconstruction while generalizing to natural images.

4.2. Implementation Details

Model Configurations. We implement three scale variants of our architecture: PointDiT-B (Base), PointDiT-L (Large), and PointDiT-H (Huge), following the configurations of JiT (Li & He, 2026). For the visual backbone, we use a frozen DINOv3 encoder to extract patch-level embeddings, scaling its capacity with each variant (*e.g.*, ViT-L features for PointDiT-L). Apart from this frozen encoder, all Transformer layers and prediction heads are trained from scratch. We use the same patch size of 16 for all variants.

Table 1. **Comparisons**. Average results on 7 real-world evaluation datasets with 3,444 samples. The image resolution is 512×512. Rel^{p} and δ_1^{p} are point map metrics, while Rel^{d} and δ_1^{d} are depth map metrics. BF1 measures boundary sharpness. PointDiT-H attains the best depth accuracy (Rel^{d} and δ_1^{d}) and PointDiT the sharpest boundaries (BF1) among all methods, while being far more efficient than the latent diffusion model GeometryCrafter (72 *vs.* 1,178 ms for single-step inference). Even a single sampling step already surpasses all prior methods on BF1, and additional steps further sharpen boundaries at a modest cost.

Method	$\text{Rel}^{\text{p}} \downarrow$	$\delta_1^{\text{p}} \uparrow$	$\text{Rel}^{\text{d}} \downarrow$	$\delta_1^{\text{d}} \uparrow$	BF1 ↑	Param (M)	Time (ms)
GeometryCrafter	5.45	96.75	3.52	97.84	4.64	1,937	1,178
PPD	5.54	96.59	3.88	97.78	9.28	804	402
Depth Pro	5.71	96.71	3.84	97.63	9.41	952	68
UniDepthV2	4.45	97.35	2.86	98.52	6.94	354	26
DA3	4.77	96.63	3.22	97.81	6.33	1,356	82
MoGe	**4.21**	97.45	3.10	98.01	5.61	314	34
MoGe-2	4.53	97.46	2.90	98.45	7.40	326	24
PointDiT-B (1 step)	5.84	96.71	3.70	97.84	8.18	223	31
PointDiT-B (2 steps)	5.81	96.77	3.64	97.86	8.88	223	47
PointDiT-B (3 steps)	5.83	96.79	3.64	97.86	9.09	223	63
PointDiT-B (4 steps)	5.85	96.80	3.64	97.86	9.16	223	79
PointDiT-L (1 step)	4.90	97.42	3.15	98.22	9.56	771	65
PointDiT-L (2 steps)	4.84	97.52	3.09	98.24	10.11	771	87
PointDiT-L (3 steps)	4.85	97.54	3.09	98.25	10.36	771	109
PointDiT-L (4 steps)	4.85	97.55	3.09	98.25	**10.50**	771	131
PointDiT-H (1 step)	4.45	97.93	2.81	98.51	9.79	1,807	72
PointDiT-H (2 steps)	4.38	97.99	**2.75**	**98.54**	10.31	1,807	116
PointDiT-H (3 steps)	4.39	98.01	**2.75**	**98.54**	10.44	1,807	160
PointDiT-H (4 steps)	4.40	**98.02**	**2.75**	**98.54**	10.49	1,807	204

Training Schedule. Our training curriculum consists of large-scale pre-training followed by high-resolution fine-tuning. We use the AdamW optimizer (Loshchilov & Hutter, 2019), with a learning rate schedule and hyperparameters consistent with JiT (Li & He, 2026). More implementation details are provided in the appendix (Section A.3).

All variants are pre-trained at 256×256 for 30 epochs (including a 5-epoch warmup) and then fine-tuned at 512×512, scaling the number of GPUs with resolution and model capacity. Interestingly, we observe that larger models converge faster and require fewer fine-tuning epochs. We report the detailed per-variant training cost (GPU count and wall-clock time) in the appendix (Table 6).

4.3. Evaluation Setup and Metrics

To assess the zero-shot generalization of our model, we evaluate on seven commonly used real-world datasets: DIODE (Vasiljevic et al., 2019), KITTI (Geiger et al., 2012), NYUv2 (Silberman et al., 2012), ETH3D (Schöps et al., 2017), HAMMER (Jung et al., 2022), iBims-1 (Koch et al.,

Table 2. **Single-step feed-forward inference**. Single-step results of PointDiT-H from random noise (three seeds) or an all-zeros input. Performance is nearly invariant to the noise, with all-zeros matching or slightly exceeding stochastic sampling, indicating the model learns to be robust to different noise realizations.

Method	$\text{Rel}^{\text{p}} \downarrow$	$\delta_1^{\text{p}} \uparrow$	$\text{Rel}^{\text{d}} \downarrow$	$\delta_1^{\text{d}} \uparrow$	BF1 ↑
rand noise (seed 1)	4.454	97.928	2.815	98.505	9.772
rand noise (seed 2)	4.452	97.938	2.811	98.513	9.778
rand noise (seed 3)	4.454	97.921	2.812	98.513	9.772
all zeros (no rand)	4.446	97.934	2.806	98.508	9.792

2018), and Booster (Zama Ramirez et al., 2022). These span diverse environments, from indoor rooms to complex outdoor driving scenes. More details are provided in the appendix (Table 5). Consistent with our training, we evaluate at both 256×256 and 512×512 resolutions. Given the heterogeneous aspect ratios and resolutions of the original test sets, we adopt a standardized preprocessing pipeline: each input image is rescaled so that its shorter side matches the target resolution (256 or 512 pixels) and then center-cropped to the square input required by the model. For a fair comparison, we benchmark against several state-of-the-art baselines, evaluating their publicly available pre-trained weights under the same preprocessing and cropping protocol.

Our model predicts affine-invariant point maps, from which affine-invariant depth maps are obtained by extracting the z-component of each point. For evaluation, we follow the alignment procedure of MoGe (Wang et al., 2025b), determining the optimal scale and shift by solving a least-squares problem that minimizes the discrepancy between the prediction and the ground truth. We assess prediction quality in both the point map and depth domains using standard metrics (Wang et al., 2025b):

- **Accuracy** (δ_1): the percentage of pixels for which the ratio between prediction and ground truth is below 1.25.
- **Relative absolute error** (Rel): $\frac{1}{N}\sum \frac{|y-\hat{y}|}{y}$, the scale-normalized error.
- **Geometric edge fidelity** (BF1): a local boundary metric, following Depth Pro (Bochkovskii et al., 2025), that assesses the recovery of fine structural details and sharp depth discontinuities.

We report metrics in both domains, using Rel^{p} and δ_1^{p} for point maps and Rel^{d} and δ_1^{d} for depth.

4.4. Evaluation Results

Main Comparisons. Table 1 reports average results over the seven evaluation datasets at 512×512. Our largest model, PointDiT-H, achieves the best depth accuracy (Rel^{d} and δ_1^{d}) and the best point map δ_1^{p}, while PointDiT achieves the highest boundary sharpness (BF1) among all methods.

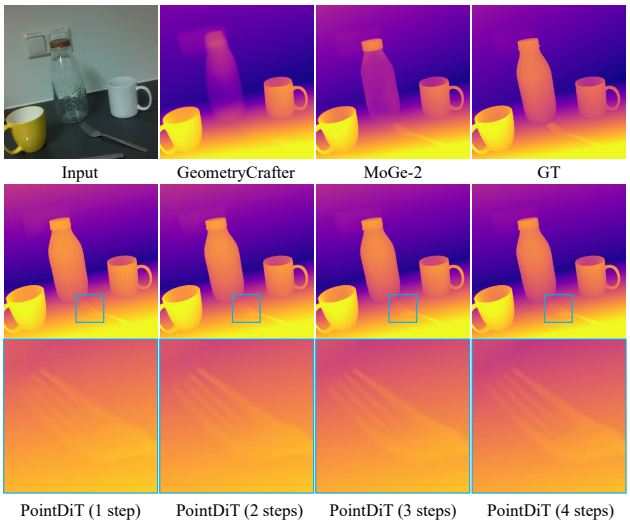


Figure 3. **Different diffusion sampling steps.** Our single-step diffusion already significantly outperforms prior works, and increasing the sampling steps further enhances reconstruction details (see the zoomed-in region).

The improvement is most pronounced on BF1: PointDiT raises boundary sharpness from 9.41 (the best baseline) to 10.50, reflecting markedly sharper geometry (Figure 4). On Rel^{p}, MoGe remains slightly ahead (4.21 *vs.* 4.40), yet PointDiT is more accurate on every depth metric. PointDiT-L attains comparable boundary quality at lower cost, and our smallest variant, PointDiT-B, stays competitive with fewer parameters. Compared with PPD (Xu et al., 2025a), the most closely related pixel-space diffusion model, PointDiT is substantially better across all metrics and runs faster thanks to its fewer sampling steps. As PPD predicts only monocular depth, we compute its point map metrics by recovering camera intrinsics with MoGe-2 (Wang et al., 2025c), following the official PPD repository.

Sharper Local Structures. Figure 4 shows qualitative comparisons with previous methods. PointDiT produces noticeably sharper local structures while preserving high-quality overall geometry. Unlike the latent-diffusion method GeometryCrafter, our approach avoids the lossy VAE compression that degrades fine detail, particularly at object boundaries (Figure 2a). By removing the VAE entirely, PointDiT recovers substantially more accurate local structures, as reflected in the BF1 metric (Table 1).

Single-Step Feed-Forward Inference. Although our model is trained with flow matching, we find that it can perform single-step feed-forward inference, as also observed for diffusion-based monocular depth estimation (Garcia et al., 2025). In Table 2, we study the model's sensitivity to noise sampling in single-step inference, and find it highly robust across stochastic initializations. Across different random noise seeds, performance fluctuations are negligible, with

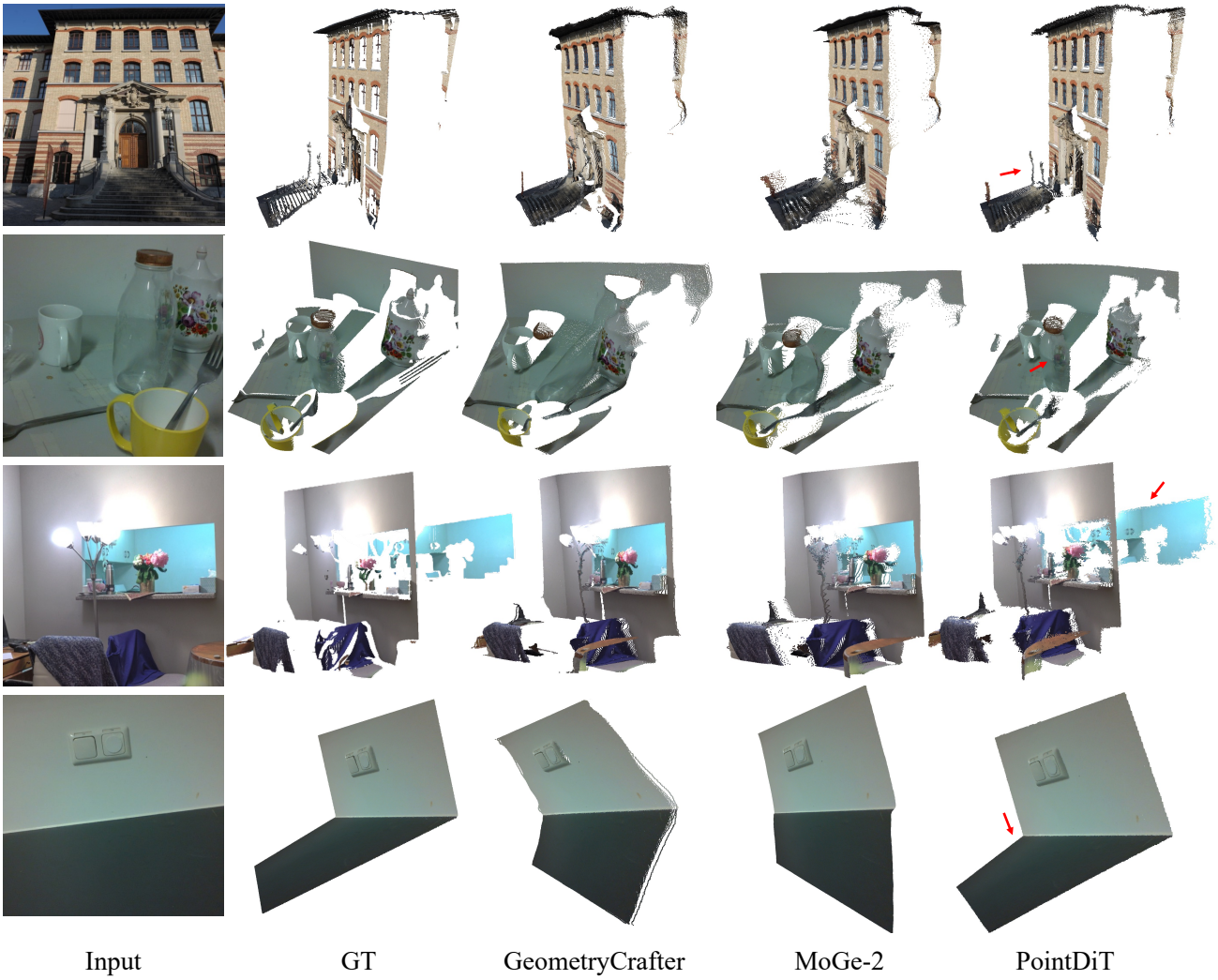


Figure 4. **Point map comparisons.** Our PointDiT is significantly better in terms of reconstructing thin structures (1st row), transparent objects (2nd rows), and maintaining a more accurate relative scale across the global scene (3rd and 4th rows). We show additional depth comparisons in the appendix (Figure 7).

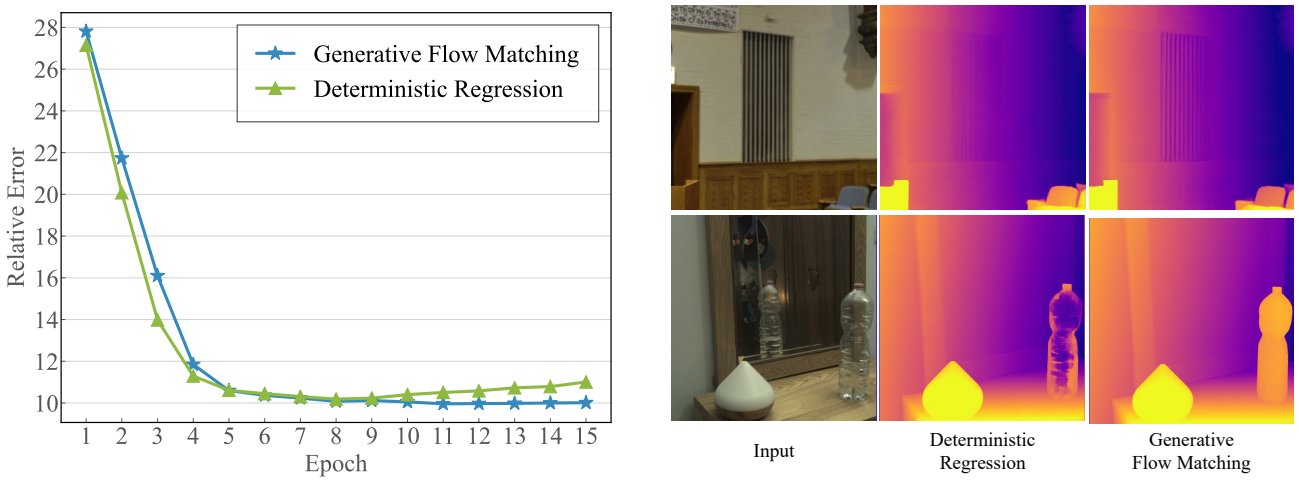
Rel^p and δ_1^p remaining nearly constant. More notably, the model maintains high-fidelity predictions even under deterministic sampling, where the input noise is set to all zeros. This "all-zeros" configuration not only matches but occasionally exceeds stochastic sampling. These results suggest that our model has learned a robust mapping from the DINOv3-encoded image patch tokens to the geometric point map. Remarkably, even with a single step, PointDiT-H already outperforms prior methods (Table 1), at a fraction of the inference cost of latent diffusion models.

Multi-Step Refinement. Thanks to its flow matching formulation, PointDiT can also benefit from additional inference steps using the same model. As shown in Table 1, more steps steadily improve the boundary metric BF1, while Rel and δ_1 remain stable, since a single step already yields high-quality results. Figure 3 shows the corresponding visual refinement. Supporting a variable number of inference steps with one network underscores the flexibility of our approach.

4.5. Ablation and Analysis

In this section, we evaluate the design choices of our model with controlled experiments. By default we train on the 256×256 SceneNet-RGBD dataset and report the average metrics on the seven unseen test sets with single-step inference.

Generative Flow Matching *vs.* Deterministic Regression. To further demonstrate the benefits of the generative flow matching formulation, we train a deterministic regressor by fixing both the time step and the noise to 0, while keeping exactly the same network architecture, training data, and training procedure. Figure 5a shows the validation curves



(a) **Validation curves.** *(b)* **Structural details and transparent objects.**

Figure 5. **Generative flow matching *vs.* deterministic regression.** (a) The deterministic regressor converges faster at first but soon overfits, while the generative model trains stably and reaches lower error. (b) The generative model recovers sharper boundaries, thin structures, and transparent objects than the deterministic regressor. Overall, the generative formulation improves the boundary metric BF1 from 10.90 to 13.92 under this controlled comparison.

of the two models. The deterministic regressor converges faster at first but soon overfits, whereas the generative model remains stable and ultimately reaches lower error. The gain in boundary quality is especially large, with a BF1 of 13.92 (generative) *vs.* 10.90 (deterministic). Figure 5b shows the corresponding visual comparison: the deterministic regressor produces blurry boundaries and fails to recover thin structures and transparent objects, while the generative model reconstructs much sharper boundaries and handles transparent objects more robustly. This controlled experiment confirms that the generative formulation learns sharper geometry and better handles ambiguous regions.

Prediction Target: x-Prediction *vs.* v-Prediction. Key to our method's success is predicting the clean point map (x-prediction) rather than the velocity (v-prediction). Table 3(a) shows that v-prediction fails catastrophically, consistent with the findings of JiT (Li & He, 2026) for image generation. We demonstrate that x-prediction is equally crucial for 3D point map generation.

Noise Schedule. The original JiT for image generation uses a logit-normal noise schedule. However, we find that this alone yields unsatisfactory results for point map generation, where we measure per-point accuracy. Because the logit-normal sampler maps the timestep through a sigmoid, it is nearly impossible to draw an exact 0 during training. As a result, the model rarely sees pure noise, causing a train-test discrepancy that hurts performance (Lin et al., 2024). Shifting the schedule toward high-noise regions with a smaller mean partially alleviates this, but the issue remains (Table 3(b)). We instead randomly set the sampled timestep to exactly 0 with 10% probability, a simple fix that substantially improves quality.

Image Patch Embeddings. In Table 3(c), we compare different patch embedding methods. Even without any pre-trained image backbone (*i.e.*, with plain linear embeddings), our model already achieves decent results. Pre-trained embeddings nonetheless help: comparing the last-layer features of DINOv2 and DINOv3, DINOv3 performs slightly better. Uniformly sampling 4 intermediate layers improves the results further, especially the BF1 metric, indicating the benefit of integrating different levels of abstraction from pre-trained backbones. We additionally evaluate the embeddings of MoGe-2 (Wang et al., 2025c) and DA3 (Lin et al., 2026), which are specifically fine-tuned for monocular geometry estimation. They further improve the accuracy metrics (Rel and δ_1), confirming that our model readily benefits from geometry-aware features. Interestingly, however, DINOv3 still attains the best boundary sharpness (BF1 of 13.47 *vs.* 11.75 for MoGe-2 and 12.58 for DA3). We hypothesize that, because MoGe-2 and DA3 are trained with regression objectives that tend to over-smooth geometry, their representations retain less high-frequency boundary information, despite encoding more accurate global structure. We do not, however, use any of these features in our main results, as our focus is to demonstrate the effectiveness of the pure pixel-space diffusion framework. Since both MoGe-2 and DA3 are themselves fine-tuned from pre-trained DINO features, we deliberately keep the same pre-trained model for a controlled comparison, so that our improvements can be attributed to the framework itself rather than to stronger, task-specific features.

Table 3. **Ablation experiments.** Trained on 256×256 SceneNet RGB-D and averaged over the seven unseen test sets with single-step inference (PointDiT-L). Our default setting is highlighted in gray.

Setting	$\text{Rel}^{\text{p}} \downarrow$	$\delta_1^{\text{p}} \uparrow$	$\text{Rel}^{\text{d}} \downarrow$	$\delta_1^{\text{d}} \uparrow$	BF1 ↑
(a) Prediction target					
v-pred	35.44	30.03	24.07	58.21	0.46
x-pred	**9.29**	**91.18**	**5.54**	**95.08**	**13.47**
(b) Noise schedule (t-shift)					
-0.8	12.19	84.82	7.80	91.34	8.05
-1.2	11.86	85.53	7.46	91.87	**8.11**
-1.6	10.73	88.06	6.74	93.09	7.31
-0.8 & rand zero	**9.68**	**90.54**	**6.00**	**94.64**	7.24
(c) Image patch embedding					
Linear	13.32	82.56	9.64	88.09	9.68
DINOv2 (last layer)	9.80	90.07	5.99	94.34	5.11
DINOv3 (last layer)	9.68	90.54	6.00	94.64	7.24
DINOv3 (4 layers)	9.29	91.18	5.54	95.08	**13.47**
MoGe-2 (4 layers)	8.29	**93.35**	4.93	96.19	11.75
DA3 (4 layers)	**8.26**	93.09	**4.74**	**96.47**	12.58
(d) Training loss					
v-loss	9.29	91.18	5.54	**95.08**	13.47
v-loss & point loss	**9.10**	**91.48**	**5.53**	94.88	**13.92**
(e) Patch size (512×512)					
32	5.35	96.88	3.48	97.78	6.17
16	**5.01**	**97.34**	**3.06**	**98.17**	**10.37**

Training Loss. The ablation results discussed so far use only the flow matching loss (Equation (5)), which is already highly effective at recovering high-quality geometry. Adding the relative loss (Equation (6)), specifically designed for point map data, further improves the results, as shown in Table 3(d).

Patch Size. We further evaluate the impact of patch size on high-resolution images by fine-tuning the 256×256 pre-trained model to 512×512 resolution. To save compute, the 512×512 models in this part are fine-tuned on a 6-dataset subset (Hypersim, VKITTI2, UrbanSyn, Synscapes, TartanAir, and OmniWorldGame; 1.48M samples) rather than the full training set. Comparing patch sizes 16 and 32, we find that 16 yields better overall results and sharper boundaries (Table 3(e)). This is expected, since point map prediction requires pixel-perfect accuracy, and a larger patch size tends to discard more high-frequency detail. Figure 6 shows the corresponding qualitative comparison.

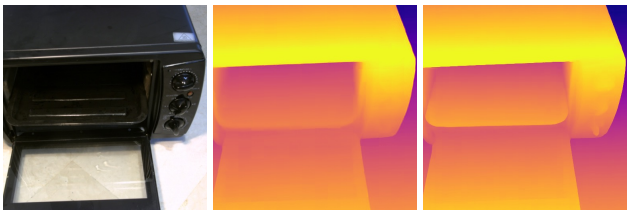


Figure 6. **Effect of patch size.** At 512×512 resolution, a patch size of 16 recovers sharper boundaries and finer local structures than a patch size of 32.

5. Conclusion

We presented a minimalist pixel-space diffusion model for monocular point map prediction that removes the architectural overhead of VAEs and hybrid networks. A plain Vision Transformer trained directly on raw point maps, conditioned on DINOv3 features, already produces accurate geometry with notably sharper boundaries and supports both single-step and multi-step inference. By showing that dense geometry can be modeled effectively in pixel space, we bridge the gap between standard image generation and 3D reconstruction, paving the way for VAE-free, end-to-end 3D and 4D generation that relies solely on direct diffusion to model complex structural distributions.

Limitation. While our framework delivers robust geometric estimation, it is currently trained at fixed resolutions (256×256 and 512×512); mixed-resolution training is a promising direction for generalizing seamlessly across image resolutions. In addition, our model still has room for improvement on outdoor scenes (Table 7) due to the relatively limited scale and diversity of the training data. Further scaling of the training set, particularly with more outdoor data, will likely be key to realizing the model's full potential. Finally, our model currently predicts geometry alone. However, since the backbone is a plain ViT, extending it to jointly output additional modalities, such as RGB appearance, would be straightforward and require minimal architectural changes. The same flexibility makes it natural to explore multi-view generation, alternative 3D representations, and richer conditioning signals (*e.g.*, camera parameters), which we view as exciting avenues for future work.

Acknowledgements. We thank Nando Metzger, Weirong Chen, Felix Wimbauer, Haiwen Huang, Gene Chou, Luca Zanella, Erik Sandström, Keisuke Tateno, Goutam Bhat, Mattia Segu, Vasile Lup, Tobias Fischer, Shaohui Liu and Bingxin Ke for the insightful discussions and support.

Impact Statement

This paper presents work whose goal is to advance the field of machine learning and computer vision by simplifying the paradigm for monocular 3D reconstruction. There are some potential societal consequences of our work, ranging from advancements in robotics to spatial intelligence, none of which we feel must be specifically highlighted here from an ethical standpoint.

References

Black Forest Labs. FLUX.2: Analyzing and enhancing the latent space of FLUX – representation comparison, 2025. URL https://bfl.ai/research/representation-comparison.

Bochkovskii, A., Delaunoy, A., Germain, H., Santos, M., Zhou, Y., Richter, S. R., and Koltun, V. Depth pro: Sharp monocular metric depth in less than a second. In *ICLR*, 2025.

Cabon, Y., Murray, N., and Humenberger, M. Virtual kitti 2. *arXiv preprint arXiv:2001.10773*, 2020.

Dosovitskiy, A. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Eigen, D., Puhrsch, C., and Fergus, R. Depth map prediction from a single image using a multi-scale deep network. *NIPS*, 27, 2014.

Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., et al. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024.

Garcia, G. M., Zeid, K. A., Schmidt, C., De Geus, D., Hermans, A., and Leibe, B. Fine-tuning image-conditional diffusion models is easier than you think. In *WACV*, 2025.

Geiger, A., Lenz, P., and Urtasun, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

Gómez, J. L., Silva, M., Seoane, A., Borrás, A., Noriega, M., Ros, G., Iglesias-Guitian, J. A., and López, A. M. All for one, and one for all: Urbansyn dataset, the third musketeer of synthetic driving scenes. *Neurocomputing*, 637:130038, 2025.

He, J., Li, H., Yin, W., Liang, Y., Li, L., Zhou, K., Zhang, H., Liu, B., and Chen, Y.-C. Lotus: Diffusion-based visual foundation model for high-quality dense prediction. *arXiv preprint arXiv:2409.18124*, 2024.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *NIPS*, 33, 2020.

Hu, W., Gao, X., Li, X., Zhao, S., Cun, X., Zhang, Y., Quan, L., and Shan, Y. Depthcrafter: Generating consistent long depth sequences for open-world videos. In *CVPR*, 2025.

Huang, P.-H., Matzen, K., Kopf, J., Ahuja, N., and Huang, J.-B. Deepmvs: Learning multi-view stereopsis. In *CVPR*, 2018.

Jung, H., Ruhkamp, P., Zhai, G., Brasch, N., Li, Y., Verdie, Y., Song, J., Zhou, Y., Armagan, A., Ilic, S., et al. Is my depth ground-truth good enough? hammer – highly accurate multi-modal dataset for dense 3d scene regression. *arXiv preprint arXiv:2205.04565*, 2022.

Karaev, N., Rocco, I., Graham, B., Neverova, N., Vedaldi, A., and Rupprecht, C. Dynamicstereo: Consistent dynamic depth from stereo videos. In *CVPR*, 2023.

Ke, B., Obukhov, A., Huang, S., Metzger, N., Daudt, R. C., and Schindler, K. Repurposing diffusion-based image generators for monocular depth estimation. In *CVPR*, 2024.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. 2014.

Koch, T., Liebel, L., Fraundorfer, F., and Körner, M. Evaluation of cnn-based single-image depth estimation methods. In *ECCV Workshops*, 2018.

Lê, H.-Â., Mensink, T., Das, P., and Gevers, T. Eden: Multimodal synthetic dataset of enclosed garden scenes. In *WACV*, 2021.

Li, T. and He, K. Back to basics: Let denoising generative models denoise. In *CVPR*, 2026.

Lin, H., Chen, S., Liew, J., Chen, D. Y., Li, Z., Shi, G., Feng, J., and Kang, B. Depth anything 3: Recovering the visual space from any views. In *ICLR*, 2026.

Lin, S., Liu, B., Li, J., and Yang, X. Common diffusion noise schedules and sample steps are flawed. In *WACV*, 2024.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. 2019.

McCormac, J., Handa, A., Leutenegger, S., and Davison, A. J. Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation? In *ICCV*, 2017.

Peebles, W. and Xie, S. Scalable diffusion models with transformers. In *ICCV*, 2023.

Piccinelli, L., Sakaridis, C., Yang, Y.-H., Segu, M., Li, S., Abbeloos, W., and Van Gool, L. Unidepthv2: Universal monocular metric depth estimation made simpler. *TPAMI*, 2025.

Ranftl, R., Bochkovskiy, A., and Koltun, V. Vision transformers for dense prediction. In *ICCV*, 2021.

Roberts, M., Ramapuram, J., Ranjan, A., Kumar, A., Bautista, M. A., Paczan, N., Webb, R., and Susskind, J. M. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *ICCV*, 2021.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

Salimans, T. and Ho, J. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Schöps, T., Schönberger, J. L., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M., and Geiger, A. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *CVPR*, 2017.

Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

Siméoni, O., Vo, H. V., Seitzer, M., Baldassarre, F., Oquab, M., Jose, C., Khalidov, V., Szafraniec, M., Yi, S., Ramamonjisoa, M., et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Szymanowicz, S., Zhang, J. Y., Srinivasan, P., Gao, R., Brussee, A., Holynski, A., Martin-Brualla, R., Barron, J. T., and Henzler, P. Bolt3d: Generating 3d scenes in seconds. In *ICCV*, 2025.

Vasiljevic, I., Kolkin, N., Zhang, S., Luo, R., Wang, H., Dai, F. Z., Daniele, A. F., Mostajabi, M., Basart, S., Walter, M. R., and Shakhnarovich, G. Diode: A dense indoor and outdoor depth dataset. *arXiv preprint arXiv:1908.00463*, 2019.

Wang, J., Chen, M., Karaev, N., Vedaldi, A., Rupprecht, C., and Novotny, D. Vggt: Visual geometry grounded transformer. In *CVPR*, 2025a.

Wang, Q., Zheng, S., Yan, Q., Deng, F., Zhao, K., and Chu, X. Irs: A large synthetic indoor robotics stereo dataset for disparity and surface normal estimation. *arXiv preprint arXiv:1912.09632*, 2019.

Wang, R., Xu, S., Dai, C., Xiang, J., Deng, Y., Tong, X., and Yang, J. Moge: Unlocking accurate monocular geometry estimation for open-domain images with optimal training supervision. In *CVPR*, 2025b.

Wang, R., Xu, S., Dong, Y., Deng, Y., Xiang, J., Lv, Z., Sun, G., Tong, X., and Yang, J. Moge-2: Accurate monocular geometry with metric scale and sharp details. *arXiv preprint arXiv:2507.02546*, 2025c.

Wang, W., Zhu, D., Wang, X., Hu, Y., Qiu, Y., Wang, C., Hu, Y., Kapoor, A., and Scherer, S. Tartanair: A dataset to push the limits of visual slam. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

Wang, W. et al. Tartanground: A large-scale dataset for ground robot perception and navigation. *arXiv preprint arXiv:2505.10696*, 2025d.

Wrenninge, M. and Unger, J. Synscapes: A photorealistic synthetic dataset for street scene parsing. *arXiv preprint arXiv:1810.08705*, 2018.

Xu, G., Lin, H., Luo, H., Wang, X., Yao, J., Zhu, L., Pu, Y., Chi, C., Sun, H., Wang, B., et al. Pixel-perfect depth with semantics-prompted diffusion transformers. In *NeurIPS*, 2025a.

Xu, T.-X., Gao, X., Hu, W., Li, X., Zhang, S.-H., and Shan, Y. Geometrycrafter: Consistent geometry estimation for open-world videos with diffusion priors. In *ICCV*, 2025b.

Yang, L., Kang, B., Huang, Z., Xu, X., Feng, J., and Zhao, H. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, 2024.

Yao, J., Yang, B., and Wang, X. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *CVPR*, 2025.

Yu, S., Kwak, S., Jang, H., Jeong, J., Huang, J., Shin, J., and Xie, S. Representation alignment for generation: Training diffusion transformers is easier than you think. 2025.

Zama Ramirez, P., Tosi, F., Poggi, M., Salti, S., Di Stefano, L., and Mattoccia, S. Open challenges in deep stereo: The booster dataset. In *CVPR*, 2022.

Zheng, B., Ma, N., Tong, S., and Xie, S. Diffusion transformers with representation autoencoders. *arXiv preprint arXiv:2510.11690*, 2025.

Zhou, Y. et al. Omniworld: A multi-domain and multi-modal dataset for 4d world modeling. *arXiv preprint arXiv:2509.12201*, 2025.

Appendix

A. Experimental Details

We train our PointDiT on synthetic datasets that provide dense, accurate ground-truth depth together with known camera intrinsics, and we evaluate zero-shot on unseen real-world benchmarks. From each image we back-project the depth map through the intrinsics into a per-pixel 3D point map, which is the prediction target of the model. Training proceeds in two stages: a 256×256 pre-training stage on a single large indoor dataset, followed by a 512×512 fine-tuning stage on a diverse multi-dataset mixture.

A.1. Training Datasets

Table 4 lists the training data of both stages.

Stage 1 (Pre-Training, 256×256). We first pre-train on SceneNet-RGBD (McCormac et al., 2017), a single large-scale synthetic indoor dataset (≈ 5.36M samples). Its scale and clean indoor geometry let the model cheaply acquire a strong image-to-point prior at low resolution.

Stage 2 (Fine-Tuning, 512×512). We then fine-tune at 512×512 on a mixture of 11 synthetic datasets (≈ 6.22M samples) spanning indoor, outdoor ground-level, and aerial/diverse domains, which adds the outdoor, large-scale, and high-detail geometry absent from Stage 1. We combine these datasets through weighted sampling. Each dataset d is assigned a mixing weight w_d (Table 4, $\sum_d w_d{=}1$); every sample of dataset d receives the per-sample probability w_d/N_d, where N_d is the number of samples in d. After global normalization, the probability that a drawn sample comes from dataset d is therefore exactly w_d, independent of the corpus size N_d. This decouples the effective data mixture from the highly imbalanced raw corpus sizes: *e.g.* TartanGround accounts for 67.1% of all samples but is sampled only 15% of the time, while small high-quality sets such as Synscapes (25k samples) are upsampled to 9%.

A.2. Evaluation Datasets

We evaluate zero-shot on seven real-world depth benchmarks, none of which overlaps the (synthetic) training datasets of either stage; this measures synthetic-to-real generalization. We report the full-set sample counts in Table 5. They span indoor, outdoor, and mixed scenes, including three challenging boundary-focused sets with transparent/specular surfaces and sharp planar discontinuities (HAMMER, iBims-1, Booster), on which we additionally measure boundary sharpness.

A.3. Training Details

Data Augmentation. We apply two groups of augmentations: geometric operations, applied jointly to the image and the point map (with intrinsics updated accordingly), and photometric operations, applied to the image only.

- **Geometric** (image and point map):
  (a) Resize to the working resolution. In Stage 2 this is aspect-ratio-preserving so the image height is 512 (smaller images are upscaled; larger images are downscaled with probability 0.5, otherwise cropped at native resolution); in Stage 1 the image height is set to 256.
  (b) A square random crop to the working resolution (center crop at test time).
  (c) A random horizontal flip ($p{=}0.5$), which also negates the point X-coordinate to keep the geometry consistent with the flipped image.
- **Photometric** (image only): color jitter (brightness/contrast/saturation 0.1, hue 0.05) applied to every sample, plus appearance augmentations each applied independently with probability 0.2: Gaussian blur, autocontrast, histogram equalization, random channel permutation, JPEG compression ($q \in [40, 100]$), and grayscale conversion.

Optimization. Both stages use AdamW ($\beta_1{=}0.9$, $\beta_2{=}0.95$, weight decay 0) in `bf16` mixed precision, the linear learning-rate scaling rule $\text{lr} = \text{blr} \cdot (\text{global batch})/256$, no gradient clipping or accumulation, and two exponential moving averages of the weights (decays 0.9999 and 0.9996); the 0.9999 EMA is used for all evaluations. Stage 1 (256×256) uses blr$=5{\times}10^{-5}$, a 5-epoch linear warmup followed by a constant learning rate, for 30 epochs. Stage 2 (512×512) is initialized from the

Table 4. **Training datasets.** All sources are synthetic and provide dense depth with known camera intrinsics. Weight is the Stage-2 dataset mixing (sampling) probability, applied independently of the corpus size. Stage 1 pre-trains on a single dataset.

Dataset	Domain	#Samples	Weight	Website
Stage 1 — pre-training (256×256)				
SceneNet-RGBD (McCormac et al., 2017)	indoor	**5,359,500**	1.00	link
Stage 2 — fine-tuning (512×512) mixture				
Hypersim (Roberts et al., 2021)	indoor	70,647	0.12	link
Virtual KITTI 2 (Cabon et al., 2020)	driving	42,520	0.14	link
UrbanSyn (Gómez et al., 2025)	urban driving	7,539	0.05	link
Synscapes (Wrenninge & Unger, 2018)	urban driving	25,000	0.09	link
TartanAir (Wang et al., 2020)	diverse	306,637	0.10	link
OmniWorld-Game (Zhou et al., 2025)	diverse (game)	1,024,252	0.19	link
EDEN (Lê et al., 2021)	garden/outdoor	368,663	0.05	link
IRS (Wang et al., 2019)	indoor	39,342	0.02	link
Dynamic Replica (Karaev et al., 2023)	indoor	150,900	0.03	link
MVS-Synth (Huang et al., 2018)	urban	12,000	0.06	link
TartanGround (Wang et al., 2025d)	diverse	4,170,178	0.15	link
Total (Stage 2)		**6,217,678**	**1.00**	

Table 5. **Zero-shot evaluation datasets.** All are real captured data and disjoint from training. "Boundary" marks the datasets on which the scale-invariant boundary F1 is additionally reported.

Dataset	Domain	#Samples	Boundary
DIODE (Vasiljevic et al., 2019)	indoor + outdoor	771	
KITTI (Geiger et al., 2012)	outdoor (driving)	652	
NYUv2 (Silberman et al., 2012)	indoor	654	
ETH3D (Schöps et al., 2017)	indoor + outdoor	454	
HAMMER (Jung et al., 2022)	indoor (multi-modal)	775	✓
iBims-1 (Koch et al., 2018)	indoor	100	✓
Booster (Zama Ramirez et al., 2022)	indoor (transparent/specular)	38	✓
Total		**3,444**	

Stage-1 checkpoint (the fixed positional embeddings are bicubically interpolated from the 16×16 to the 32×32 grid) and uses blr$=1 \times 10^{-4}$, a constant learning rate (no warmup). We report the per-variant epoch counts, GPU counts, and wall-clock times for both stages in Table 6.

Table 6. **Training cost.** Number of epochs, H100 GPUs, and wall-clock time for the pre-training (256×256) and fine-tuning (512×512) stages of each variant.

Model	Param (M)	Pre-train (256×256)			Fine-tune (512×512)		
		Epoch	GPUs	Time	Epoch	GPUs	Time
PointDiT-B	223	30	16	12h	8	64	2.5h
PointDiT-L	771	30	16	21h	5	64	7h
PointDiT-H	1,807	30	64	22h	3	128	5.5h

B. More Evaluation Results

B.1. Per-Dataset Metrics

Table 1 in the main paper reports metrics averaged over all seven evaluation datasets. For completeness, we provide the full per-dataset breakdown here: Table 7 for point map accuracy (Rel$^{\mathrm{p}}$, δ_1^{p}), Table 8 for depth accuracy (Rel$^{\mathrm{d}}$, δ_1^{d}), and Table 9 for boundary sharpness (BF1). We split the point map and depth metrics into separate tables for readability, and report BF1 only on the three datasets whose ground truth supports boundary evaluation (HAMMER, iBims-1, Booster); per-dataset sample counts are given in Table 5. PointDiT variants use 4 sampling steps, and the Avg column is the sample-weighted mean over all evaluation samples. Overall, PointDiT-H attains the best average depth accuracy, PointDiT achieves the sharpest

Table 7. **Per-dataset point map results.** $\mathrm{Rel}^{\mathrm{p}}\downarrow$ and $\delta_1^{\mathrm{p}}\uparrow$ for each of the seven evaluation datasets at 512×512 resolution. PointDiT variants use 4 sampling steps. The Avg column is the sample-weighted mean over all 3,444 samples.

Method	DIODE		KITTI		NYUv2		ETH3D		HAMMER		iBims-1		Booster		Avg	
	$\mathrm{Rel}^{\mathrm{p}}\downarrow$	$\delta_1^{\mathrm{p}}\uparrow$	$\mathrm{Rel}^{\mathrm{p}}\downarrow$	$\delta_1^{\mathrm{p}}\uparrow$	$\mathrm{Rel}^{\mathrm{p}}\downarrow$	$\delta_1^{\mathrm{p}}\uparrow$	$\mathrm{Rel}^{\mathrm{p}}\downarrow$	$\delta_1^{\mathrm{p}}\uparrow$	$\mathrm{Rel}^{\mathrm{p}}\downarrow$	$\delta_1^{\mathrm{p}}\uparrow$	$\mathrm{Rel}^{\mathrm{p}}\downarrow$	$\delta_1^{\mathrm{p}}\uparrow$	$\mathrm{Rel}^{\mathrm{p}}\downarrow$	$\delta_1^{\mathrm{p}}\uparrow$	$\mathrm{Rel}^{\mathrm{p}}\downarrow$	$\delta_1^{\mathrm{p}}\uparrow$
GeometryCrafter	5.88	95.06	4.84	97.97	4.10	98.64	5.46	97.03	6.75	95.46	5.50	97.23	3.73	99.31	5.45	96.75
PPD	6.35	93.68	7.04	96.16	4.10	98.27	5.08	97.36	5.19	97.69	4.55	97.97	3.32	98.73	5.54	96.59
Depth Pro	6.36	94.11	7.33	95.97	4.08	98.48	5.90	96.85	5.27	98.03	4.54	98.11	2.87	99.81	5.71	96.71
UniDepthV2	6.02	94.35	**4.03**	**98.26**	3.29	98.52	4.02	98.79	4.57	97.53	4.07	98.34	3.78	98.90	4.45	97.35
DA3	5.47	93.68	6.37	96.72	3.67	98.20	3.44	98.20	4.60	96.95	4.08	97.81	2.92	99.03	4.77	96.63
MoGe	**4.25**	**96.37**	4.10	98.10	3.48	98.63	**3.35**	**98.99**	5.51	95.95	3.95	97.82	2.74	99.06	**4.21**	97.45
MoGe-2	4.57	95.28	5.87	97.86	**3.13**	**98.83**	3.73	98.71	5.19	97.19	4.00	98.27	**2.59**	**99.90**	4.53	97.46
PointDiT-B	6.62	93.83	5.65	97.08	5.05	97.54	6.67	96.97	5.67	98.52	4.87	97.69	3.81	99.58	5.85	96.80
PointDiT-L	5.69	95.05	5.18	97.55	4.35	98.08	4.95	97.96	4.26	99.18	4.25	98.14	3.15	99.88	4.85	97.55
PointDiT-H	5.12	95.97	5.09	97.86	3.95	98.40	4.56	98.30	**3.52**	**99.54**	**3.91**	**98.57**	3.01	99.87	4.40	**98.02**

Table 8. **Per-dataset depth map results.** $\mathrm{Rel}^{\mathrm{d}}\downarrow$ and $\delta_1^{\mathrm{d}}\uparrow$ for each of the seven evaluation datasets at 512×512 resolution. PointDiT variants use 4 sampling steps. The Avg column is the sample-weighted mean over all 3,444 samples.

Method	DIODE		KITTI		NYUv2		ETH3D		HAMMER		iBims-1		Booster		Avg	
	$\mathrm{Rel}^{\mathrm{d}}\downarrow$	$\delta_1^{\mathrm{d}}\uparrow$	$\mathrm{Rel}^{\mathrm{d}}\downarrow$	$\delta_1^{\mathrm{d}}\uparrow$	$\mathrm{Rel}^{\mathrm{d}}\downarrow$	$\delta_1^{\mathrm{d}}\uparrow$	$\mathrm{Rel}^{\mathrm{d}}\downarrow$	$\delta_1^{\mathrm{d}}\uparrow$	$\mathrm{Rel}^{\mathrm{d}}\downarrow$	$\delta_1^{\mathrm{d}}\uparrow$	$\mathrm{Rel}^{\mathrm{d}}\downarrow$	$\delta_1^{\mathrm{d}}\uparrow$	$\mathrm{Rel}^{\mathrm{d}}\downarrow$	$\delta_1^{\mathrm{d}}\uparrow$	$\mathrm{Rel}^{\mathrm{d}}\downarrow$	$\delta_1^{\mathrm{d}}\uparrow$
GeometryCrafter	3.74	96.89	3.73	98.20	2.99	98.82	3.26	98.38	3.86	97.24	2.95	98.06	2.25	99.14	3.52	97.84
PPD	4.69	95.86	4.81	97.47	3.57	98.38	4.15	97.69	2.57	99.33	3.11	98.44	2.56	99.21	3.88	97.78
Depth Pro	4.57	95.85	4.36	97.18	3.37	98.65	4.34	97.45	3.00	98.81	2.87	98.60	1.88	**99.92**	3.84	97.63
UniDepthV2	3.62	97.11	3.25	**98.44**	2.60	98.84	2.65	99.00	2.23	99.42	2.46	98.49	2.27	98.98	2.86	98.52
DA3	3.73	96.26	3.96	97.40	2.96	98.46	2.74	98.45	2.74	98.63	2.54	98.35	2.22	99.42	3.22	97.81
MoGe	3.17	97.32	3.26	98.33	2.64	98.90	2.53	**99.17**	3.75	96.92	2.57	98.14	2.21	99.16	3.10	98.01
MoGe-2	2.99	97.61	**3.16**	98.36	**2.58**	**98.95**	2.71	99.03	3.13	98.52	**2.26**	**98.83**	**1.77**	99.75	2.90	98.45
PointDiT-B	3.98	96.71	4.33	97.38	3.50	98.02	3.22	98.00	3.22	98.99	2.91	98.65	2.55	99.70	3.64	97.86
PointDiT-L	3.37	97.26	4.00	97.67	3.02	98.47	2.77	98.40	2.42	99.31	2.46	98.80	1.98	99.87	3.09	98.25
PointDiT-H	**2.90**	**97.79**	3.88	97.94	2.75	98.61	**2.46**	98.69	**1.91**	**99.53**	2.33	98.73	1.79	99.87	**2.75**	**98.54**

boundaries on average, and PointDiT leads on the challenging HAMMER set across point map, depth, and boundary metrics. A notable exception is the datasets containing outdoor scenes: across KITTI, DIODE, and ETH3D, PointDiT is inferior to the strongest regression baselines such as MoGe and UniDepthV2. We attribute this gap mainly to the limited coverage of outdoor scenes in our synthetic training mixture, and we expect that introducing more outdoor datasets for training would further narrow it.

B.2. More Visualizations

We show additional depth comparisons in Figure 7, and our PointDiT is significantly better in terms of reconstructing thin structures, transparent objects, and maintaining a more accurate relative scale across the global scene.

We provide more results in our project page: `https://haofeixu.github.io/pointdit`

Table 9. **Per-dataset boundary sharpness (BF1 ↑).** BF1 is reported only on the three datasets whose ground truth supports boundary evaluation (HAMMER, iBims-1, Booster). PointDiT variants use 4 sampling steps. The Avg column is the sample-weighted mean over these 913 boundary-annotated samples.

Method	HAMMER	iBims-1	Booster	Avg
GeometryCrafter	3.34	11.25	13.78	4.64
PPD	7.60	**16.95**	22.78	9.26
Depth Pro	7.87	15.01	25.91	9.41
UniDepthV2	6.22	9.54	14.91	6.94
DA3	5.33	11.64	12.77	6.33
MoGe	4.28	11.87	16.39	5.61
MoGe-2	5.97	14.36	18.28	7.40
PointDiT-B	7.82	14.29	22.95	9.16
PointDiT-L	**9.13**	14.94	26.58	**10.50**
PointDiT-H	9.03	14.85	**28.66**	10.49

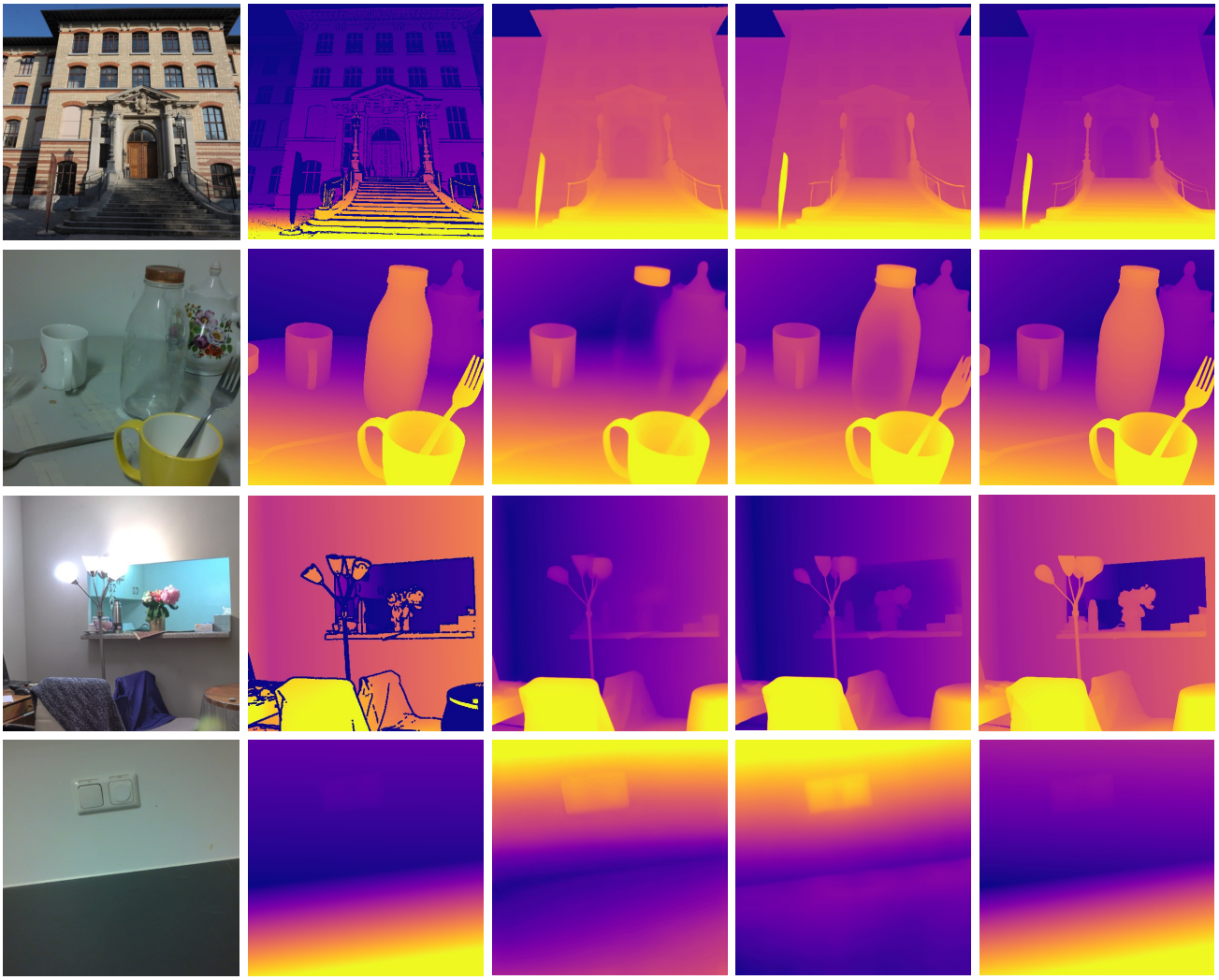


Figure 7. **Depth comparisons.** Our PointDiT is significantly better in terms of reconstructing thin structures (1st row), transparent objects (2nd rows), and maintaining a more accurate relative scale across the global scene (3rd and 4th rows). The corresponding point map comparisons are provided in the main paper (Figure 4).