Latent Riemannian Flow Matching for Geometry-Grounded 3D Foundation Models

Lisa Weijler TU Wien
Irene Ballester TU Wien
Guofeng Mei Fondazione Bruno Kessler
Tolga Birdal Imperial College London
Pedro Hermosilla TU Wien

Abstract

Geometric foundation models such as the Visual Geometry Grounded Transformer (VGGT) provide strong 3D priors from unposed images, but operate deterministically and cannot generate plausible geometry beyond what the observations directly support. We bridge geometric foundation models and generative 3D modeling by performing flow matching directly in their latent space, leveraging learned 3D priors without committing to explicit representations such as Gaussians, meshes, or video-VAE latents. This requires respecting the latent geometry: LayerNorm places the decoder inputs on a product of zero-mean hyperspheres, and treating this space as unconstrained Euclidean substantially degrades flow matching. We therefore formulate Riemannian flow matching on this product manifold, conditioned on unposed context views and a target camera pose, keeping generated tokens within the geometry expected by the frozen decoder. Our main experiments use VGGT, while additional experiments with Depth Anything 3 show that the same formulation transfers to a different independently pretrained 3D backbone without modification. On RealEstate10K, ScanNet++, and ETH3D, our method achieves strong results against recent scene-generation baselines in both novel-view appearance and aggregated 3D geometry, establishing latent-space flow matching on geometric foundation models as a viable paradigm for 3D generation. Project page: `https://lisaweijler.github.io/geometry-grounded-rfm/`.

1 Introduction

Inferring the complete 3D scene geometry from partial RGB observations is crucial for robotics, AR/VR, and autonomous driving. Agents must not only understand the observed 3D scene, but also reason about what lies outside their field of view before they can safely plan through it. This task, therefore, combines two different problems. Where the input views provide evidence, the scene is determined; where they do not, many plausible geometries and appearances remain consistent with the observations. The unobserved part of the task is therefore ill-posed in the strict sense: it admits a set of possible solutions rather than a single one.

From modern Novel View Synthesis (NVS) methods, such as NeRF [46] and 3D Gaussian Splatting [30], to recent feed-forward geometric models, such as DUSt3R [66] and VGGT [65], 3D reconstruction has made remarkable progress, producing high-quality renderings and accurate scene reconstructions from sets of input images. However, 3D reconstruction methods address only part of the problem: they faithfully recover only the part of the scene supported by the input observations.

To obtain a complete scene from partial observations, recent methods therefore incorporate a generative mechanism rather than relying solely on reconstruction. Due to the scarcity of 3D data, many of these methods rely on multi-view [50] or view-conditioned [42, 13, 26] generative models, or

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

leverage video generative models [41, 69, 81]. However, such approaches often lack explicit 3D priors: they treat 3D reconstruction as a post-processing step and require geometric knowledge to be learned implicitly from large datasets via costly training, often resulting in multi-view inconsistencies. More recent works have explored explicitly incorporating 3D structure, for instance, through depth-conditioning [59, 53, 9], adding depth maps as a training objective [44], or coupling generative models with computationally expensive reconstruction pipelines [73]. Yet, these approaches still require costly training or reconstruction strategies.

Feed-forward geometric foundation models, such as VGGT [65], provide strong geometric priors that can be exploited within generative frameworks. A group of concurrent works has recognized this opportunity, pairing video generative models with the latent space of geometric foundation models [72, 23], or designing generative approaches directly within that latent space [27]. Operating directly in the native representation space of geometric foundation models is attractive because it allows generative models to directly leverage the strong 3D priors without first mapping them into a separate generative latent space. However, existing approaches treat this space as Euclidean and do not account for its underlying geometric structure, which can degrade performance.

In this work, inspired by generative approaches operating on structured latent spaces [85, 35], we propose a generative geometry-grounded foundation model that operates directly in the latent space of feed-forward reconstruction models. However, naively applying standard generative techniques, such as flow matching [40], to this space without accounting for its geometry can reduce generation quality. To address this, we identify the latent representation of VGGT [65] and related models as a product manifold of four hyperspheres. Building on [8], we then introduce a *conditional Riemannian Flow Matching* (RFM) framework tailored to this latent geometry. Our main experiments use VGGT, while additional experiments with Depth Anything 3 (DA3) demonstrate that the same formulation transfers to a different independently pretrained 3D backbone without modifying the manifold construction or generative model. Our model is conditioned on a set of partial, unposed observations and a target camera pose. By leveraging frozen decoding heads, we reconstruct depth and point maps directly from the generated latent codes. Additionally, we train an RGB decoder for VGGT, enabling full RGB reconstruction. We evaluate our framework on sparse-view reconstruction, demonstrating that it generates plausible 3D scenes while benefiting from the strong geometric priors encoded in the underlying reconstruction model.

In summary, **our contributions are**:

- We address 3D reconstruction and NVS from a generative perspective with a flow matching model operating *directly* in the latent space of frozen geometric foundation models, thereby leveraging the strong 3D priors acquired during large-scale pre-training without first mapping them into a separate generative latent space.
- We realize this approach through a context- and pose-conditioned RFM model tailored to the latent structure of geometric foundation models. We instantiate it primarily with VGGT [65] and further demonstrate transfer to DA3, showing that the formulation is not tied to one backbone.
- Our extensive evaluation on RE10K, ScanNet++, and ETH3D demonstrates improved geometric reconstruction and generation while preserving high-quality RGB inference.

Our formulation is fully general with respect to the conditioning configuration: it is order-invariant over context views, handles as few as a single context view, and generates one target view at a time. This avoids the trajectory and adjacency assumptions of video-diffusion models.

2 Related Work

Multi-view 3D reconstruction. Traditional multi-view reconstruction pipelines decompose the problem into feature matching, structure-from-motion, and dense reconstruction [56, 57], while neural approaches such as NeRF [46, 1, 48] and Gaussian Splatting [30, 22] bypass these steps via differentiable rendering at the cost of expensive per-scene optimization. Feed-forward variants generalize across scenes through explicit 3D representations [80, 7, 6, 10, 75] or geometry-free transformer architectures [28, 31], but still require camera poses. Geometric foundation models [66, 36, 65, 39] eliminate this requirement by jointly inferring geometry and camera parameters from uncalibrated images; VGGT [65] unifies camera estimation, dense geometry, and point correspondence in a single feed-forward transformer. Our work exploits VGGT's structured latent space for generative modeling while preserving its geometric priors.

Reconstruction from limited (sparse) views. Sparse-view reconstruction addresses scenarios where limited viewpoint coverage leaves geometry ambiguous and depth under-constrained. Regularization-based approaches [51, 25, 12, 32] adapt per-scene optimization to sparse settings through geometric regularization, semantic consistency, or ray entropy minimization. Feed-forward methods circumvent optimization through learned priors: some leverage multi-view feature matching to predict Gaussian splats [6, 10, 75, 82], while others learn scene-level priors to directly regress radiance fields [80, 7]. These deterministic approaches excel at reconstruction but cannot model distributions over plausible geometries. Our generative framework instead models 3D scenes probabilistically under sparse observations.

Scene generation. For probabilistic scene modeling, methods leverage 2D generative priors or operate in 3D representation spaces. Pixel-space methods synthesize novel views without explicit 3D structure, relying on implicit geometric learning through multi-view generation [42, 50], view-conditioned synthesis [13, 26], or camera-controlled video diffusion [41, 81, 69, 83]. To inject geometric priors while remaining in pixel space, methods align or augment video generators with geometric foundation models [72, 64, 11, 76], condition generation on depth [59, 53, 9], or tightly couple reconstruction and generation pipelines [73, 17, 34, 43]. Other approaches explicitly generate geometry: Matrix3D [44] denoises RGB, depth, and camera rays jointly in a single multi-modal diffusion transformer, and GenRec [5] denoises RGB and scene coordinates while gating supervision by an observation mask, so that observed regions are regressed and unobserved ones generated. Other work applies generative models to refine existing 3D reconstructions, using diffusion [70, 79, 74] or flow matching [16] on rendered views to remove artifacts and complete occluded regions, or directly generates 3D primitives such as Gaussians [86, 71]. Closest to us, Gen3R [23] also builds on geometric foundation model tokens, but treats VGGT tokens as Euclidean vectors and maps them into the latent space of a pretrained video generative model, where generation takes place. In contrast, we generate in VGGT's own token space and employ Riemannian flow matching to respect its hyperspherical geometry.

Latent spaces of deep networks. Stable Diffusion [54] moves generation from pixels to latent space through compressed VAE spaces. Representation Autoencoders [85] instead pair frozen pretrained encoders such as DINOv2 with learned decoders, preserving semantically rich, high-dimensional representations. Independently of our work, Kumar and Patel [35] identify hyperspherical structure in DINOv2 embeddings and apply Riemannian flow matching to avoid the interference caused by Euclidean paths through the sphere's interior.

We observe that VGGT's latent space similarly forms a product manifold of hyperspheres and employ Riemannian flow matching for geometrically consistent 3D generation. Recent work further supports operating in geometric-foundation-model latents: VGGT-World [61] forecasts frozen VGGT tokens for world modeling and reports that velocity-prediction flow matching collapses in this space, Geometry Latent Diffusion (GLD) [27] repurposes DA3 and VGGT features as a multi-view diffusion latent space, and OneWorld [18] generates in the representation space of a fine-tuned π^3 [67], adopting clean-sample prediction following Li and He [38]. All three, however, treat these features as Euclidean; we instead identify and exploit their hyperspherical product structure.

Riemannian flow matching (RFM). Flow matching [40] learns continuous normalizing flows by regressing velocity fields. RFM extends this to Riemannian manifolds, with applications including molecule generation [24, 2], human pose estimation [49, 20], material generation [47], and, as discussed above, representation-encoder latents for image generation [35]. We extend RFM to 3D scene generation, operating directly on VGGT's geometric latent manifold to enable probabilistic modeling of 3D geometry while preserving strong reconstruction priors.

3 Method

Given a set of unposed RGB images $I = \{I^{(1)}, \ldots, I^{(N)}\}$ of a scene and a target camera pose $\pi \in SE(3)$, our goal is to generate latent features that encode the geometry and appearance of the scene from the target viewpoint, remaining multi-view consistent with the observed views. We cast this as a conditional generative modeling problem on the VGGT [65] encoder latent space: given context tokens $c = \{c^{(1)}, \ldots, c^{(N)}\}$ extracted from context views I and a target pose π, we model the conditional distribution $p(x_1 \mid c, \pi)$ on a Riemannian manifold $\mathcal{M}$, where x_1 is the latent code corresponding to the target view. The generated x_1 is then decoded by VGGT's DPT heads into

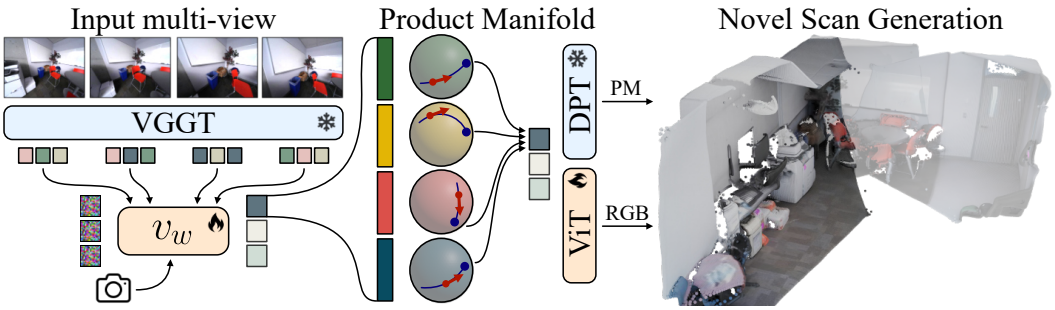


Figure 1: **Method overview.** Given a sparse set of unposed RGB observations, a frozen VGGT encoder extracts per-view latent tokens. A learned Riemannian flow matching model v_w generates latent codes conditioned on the tokens of the input views and a target camera pose, operating directly on the product manifold of four zero-mean hyperspheres. Frozen DPT decoding heads reconstruct point maps (PM) from the generated latents, while a trained RGB decoder produces full photometric reconstructions, enabling novel scan generation of complete, geometrically consistent 3D scenes.

geometry (point clouds, depth) and, via an auxiliary head, into RGB pixels. See Figure 1 for an overview.

Next, we review RFM (Section 3.1), characterize the VGGT latent space as a product of hyperspheres (Section 3.2), specialize conditional RFM to this setting (Section 3.3), and describe our architecture (Section 3.4).

3.1 Riemannian Flow Matching

We train an RFM model [8] on a corpus of latent tokens $\{x^{(i)}\}$ living on a Riemannian manifold $\mathcal{M}$. We briefly review the ingredients we use.

Flow and probability path. A time-dependent flow on $\mathcal{M}$ is a family of diffeomorphisms $\{\psi_t \colon \mathcal{M} \to \mathcal{M}\}_{t\in[0,1]}$ obtained by integrating a time-dependent vector field $u_t \in \Gamma(\mathcal{T}\mathcal{M})$:

$$\tfrac{d}{dt}\,\psi_t(x) = u_t(\psi_t(x)), \qquad \psi_0(x) = x. \tag{1}$$

The flow induces a probability path $p_t = [\psi_t]_{\#}\, p_0$ between a reference distribution p_0 and the data distribution p_1, satisfying the continuity equation $\partial_t p_t + \mathrm{div}_g(p_t u_t) = 0$.

Riemannian Conditional Flow Matching. Since u_t is intractable, RFM regresses a neural velocity field $v_w(x_t, t) \in \mathcal{T}_{x_t}\mathcal{M}$ against a tractable *conditional* field $u_t(x \mid x_1)$ that generates a per-sample path $p_t(x \mid x_1)$ with $p_0(x \mid x_1) = p_0(x)$ and $p_1(x \mid x_1) \approx \delta_{x_1}(x)$. For the geodesic distance d_g and linear schedule $\kappa(t) = 1 - t$, the minimal-norm conditional field admits the closed form

$$u_t(x \mid x_1) = \tfrac{1}{1-t}\,\mathrm{Log}_x(x_1), \tag{2}$$

yielding the RCFM training objective [8]:

$$\mathcal{L}_{\mathrm{RCFM}} = \mathbb{E}_{t,\,p(x_0),\,p(x_1)} \big\| v_w(x_t, t) - u_t(x_t \mid x_1) \big\|^2_{g_{x_t}}, \tag{3}$$

with $t \sim \mathcal{U}(0, 1)$ and $x_t = \mathrm{Exp}_{x_0}(t\mathrm{Log}_{x_0}(x_1))$ the geodesic interpolant. We give more background and detailed definitions for RCFM in Section A.1.

3.2 The VGGT latent space as a product of hyperspheres

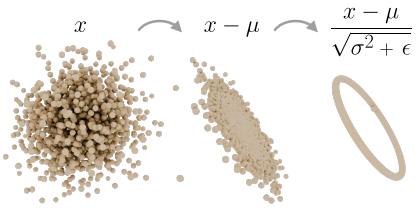
Operating on the latent space of VGGT [65] provides a strong 3D prior that our model can rely on to generate consistent 3D geometry. We define this latent space as the input features of the final DPT heads in the VGGT architecture, *i.e.*, the output of the VGGT encoder, avoiding costly self- and cross-attention operations during decoding. Features in this space take the form $x = \bigoplus_{i\in\{4,11,17,23\}} h_i$, where $\oplus$ is concatenation and h_i is the output of VGGT's ith encoder block. Each token has dimension $4C$ with $C = 2048$, yielding $x \in \mathbb{R}^{8192}$. Operating directly on encoder features avoids the reconstruction loss of an additional auto-encoder. Yet encoder features are typically high-dimensional latent representations that are difficult to model with Euclidean velocity-based diffusion or flow matching [77, 85, 38]. This difficulty is typically attributed to the

manifold hypothesis: data lies on a low-dimensional manifold within ambient space, and Euclidean probability paths traverse large off-manifold regions.

Explicit manifold structure. In our case, the manifold structure is not hypothesized but explicit. Before x is processed by the RGB and DPT heads, each h_i passes through a separate LayerNorm. This first removes the per-token mean (orthogonal projection onto the zero-mean hyperplane $H = \{z \in \mathbb{R}^C : \langle \mathbf{1}, z \rangle = 0\}$), then rescales the result to a fixed norm $\sqrt{C}$, projecting it onto the hypersphere

$$\mathcal{S}^{C-2} = \{x \in \mathbb{R}^C : \|x\|_2 = \sqrt{C},\ \langle \mathbf{1}, x \rangle = 0\} \tag{4}$$

embedded in H. The accompanying figure on the right illustrates this construction for three dimensions. Consequently, operating directly on $\mathcal{S}^{C-2}$ incurs no loss of information relative to the post-LayerNorm representation, allowing us to fully exploit the learned affine parameters of LayerNorm and any downstream weights that act on these normalized features.



The full latent space is therefore the product manifold $\mathcal{M} = \left(\mathcal{S}^{C-2}\right)^K, K = 4$ of four zero-mean hyperspheres of radius $\sqrt{C}$. Euclidean flow matching [40] ignores this structure: the prior, intermediate states, and velocity field are unconstrained in $\mathbb{R}^{4C}$, so trajectories can leave $\mathcal{M}$ and reach the decoder off-manifold. Similar effects have been observed in other feature spaces [35]. We instead perform Riemannian flow matching directly on $\mathcal{M}$, keeping the trajectory within the latent geometry expected by VGGT's frozen decoding heads by construction. We quantify the resulting gains in Section 4.3. For training, we rescale each block to unit radius and undo the scaling before decoding.

Geometry of one block. The tangent space at $x \in \mathcal{S}^{C-2}$ is $\mathcal{T}_x\mathcal{S}^{C-2} = \{v \in \mathbb{R}^C : \langle v, x \rangle = 0,\ \langle \mathbf{1}, v \rangle = 0\}$, i.e., vectors orthogonal to x that also satisfy the zero-mean constraint. With the induced metric, the exponential and logarithm maps (on the unit-radius sphere used for training) are

$$\mathrm{Exp}_x(v) = \cos(\|v\|)\, x + \tfrac{\sin(\|v\|)}{\|v\|}\, v, \qquad \mathrm{Log}_x(y) = \tfrac{\theta}{\sin\theta}\big(y - \langle x, y \rangle x\big), \tag{5}$$

with $\theta = \arccos(\langle x, y \rangle)$, and the tangent projection is $\Pi_x(p) = p - \langle \mathbf{1}, p \rangle\, \mathbf{1}/C - \langle x, p \rangle x$ (enforcing the zero-mean and orthogonality constraints, respectively).

Product structure. $\mathcal{M}$ is endowed with the product metric $d_{\mathcal{M}}(x, x')^2 = \sum_{b=1}^{K} d_g\big(x^{(b)}, x'^{(b)}\big)^2$, where $x^{(b)}$ denotes the bth block of x, *i.e.*the normalized feature h_i of the bth selected encoder block, and all maps act blockwise: $\mathrm{Exp}_x = (\mathrm{Exp}_{x^{(1)}}, \ldots, \mathrm{Exp}_{x^{(K)}})$, similarly for Log_x and Π_x. Eq. (3) and (6) therefore apply independently to each of the four blocks. Hence, we describe the remaining construction on a single sphere without loss of generality.

3.3 Conditional RFM on VGGT latents

For guided novel view generation, we condition on context tokens c from observed views and a target camera pose $\pi \in SE(3)$. In this conditional regime, our objective shifts from modeling the marginal target distribution $p(x_1)$ to modeling the conditional target distribution $p(x_1 \mid c, \pi)$. Consequently, the true time-dependent vector field driving the flow, $u_t(x \mid c, \pi)$, must be explicitly informed by both c and π to properly guide the probability paths across the product manifold $\mathcal{M}$ toward valid, view-consistent 3D latents.

The conditional RCFM loss expands the expectation over the joint distribution $p(x_1, c, \pi)$:

$$\mathcal{L}_{\mathrm{RCFM}}^{\mathrm{cond}} = \mathbb{E}_{t,\, p(x_0),\, p(x_1, c, \pi)} \big\| v_w(x_t, t, c, \pi) - u_t(x_t \mid x_1) \big\|_{g_{x_t}}^2 . \tag{6}$$

Note that the target field $u_t(x \mid x_1)$ depends only on the endpoints x_0 and x_1 alone: conditioning on (c, π) shapes the joint distribution of (x_1, c, π) from which training pairs are sampled, but not the geodesic from x_0 to x_1 nor the velocity along it.

At inference time, we sample an initial noise state $x_0 \sim \mathrm{Unif}(\mathcal{M})$ by sampling Gaussian noise and projecting it to $\mathcal{M}$. We integrate the learned vector field over the time domain $t \in [0, 1]$, where $t = 0$ corresponds to the noise prior and $t = 1$ to the data distribution. To ensure the integration

path remains strictly on the manifold, we employ a Riemannian ODE solver (*e.g.*, the Riemannian Euler method), iteratively updating the latent state via the exponential map:

$$x_{t+\Delta t} = \mathrm{Exp}_{x_t}\left(\Delta t \cdot v_w(x_t, t, c, \pi)\right). \tag{7}$$

3.4 Network architecture

We parameterize $v_w(x_t, t, c, \pi)$ as a transformer that maps manifold-valued inputs to tangent vectors. The four spherical components of x_t are concatenated and jointly projected to a working representation $z \in \mathbb{R}^D$ with $D = 768$, allowing information across components to mix from the input. Context tokens c are embedded separately into the same working dimension. The backbone follows the pattern SA-(CA-SA)^8, comprising 9 self-attention (SA) blocks, alternated with 8 cross-attention (CA) blocks. All attention blocks use 12 heads of dimension 64, resulting in a model with approximately 120M parameters.

SA blocks use Rotary Position Embeddings (RoPE) [60] to encode spatial patch structure and inject the flow time t through AdaLN modulation. CA blocks attend from the latent representation z to context tokens c, which serve as keys and values, while the target pose π is injected directly into cross-attention through Projective Positional Encoding (ProPE) [37].

A final AdaLN-modulated LayerNorm is followed by a dense linear map back to the ambient space $\mathbb{R}^{4C}$. Its output is partitioned into the four latent blocks and projected onto the corresponding tangent spaces via Π_{x_t}, yielding the valid velocity $v_w(x_t, t, c, \pi) \in \mathcal{T}_{x_t}\mathcal{M}$. For RGB prediction, we additionally train a ViT head on ground-truth VGGT latent tokens to decode the generated representation into RGB colors, enabling novel view generation. Further implementation and training details of our flow model are given in Section B.2 and details on the RGB head in Section B.1. Figure 1 provides an overview of our method.

4 Experiments

The generative nature of our approach enables plausible geometry and appearance to be generated for unseen regions, while faithfully preserving those of observed regions. We evaluate novel-view RGB and depth generation (Section 4.1), aggregated 3D scene reconstruction (Section 4.2), and the effect of Euclidean versus Riemannian latent geometry (Section 4.3). Additional experiments in the supplement evaluate transfer to Depth Anything 3 (Section D.2). We compare against deterministic reconstruction, diffusion-based generation, and recent generative methods built on geometric foundation-model features.

Datasets. The model is trained on the training splits of *RealEstate10K (RE10K)* [87] and *ScanNet++* [78] (59,103 scenes, 150 equidistant frames each). The two are complementary: ScanNet++ captures dense indoor scenes with rich camera motion, while RE10K adds outdoor and large-scale indoor scenes with smoother motion and predominantly forward-facing viewpoints. We evaluate on the RE10K test and ScanNet++ validation splits and, for 3D scan generation, on the *ETH3D* [58] training split as out-of-distribution data ($\sim$7$\times$ physically larger than ScanNet++); the evaluation scenes are listed in Section C.1.

Baselines. We compare against four recent methods covering complementary approaches to novel view generation. DepthSplat [75] is a deterministic reconstruction method based on an explicit 3D representation. Matrix3D [44] is a joint multimodal diffusion model that generates RGB in an image-VAE latent space while denoising depth and ray maps directly in 2D. Gen3R [23] builds on VGGT features but maps them into the latent space of a pretrained video model for generation. GLD [27] operates directly in VGGT feature space using Euclidean flow matching, making it the closest existing baseline to our latent-space formulation. GLD nevertheless differs from our method in several architectural choices: it generates features from two VGGT encoder levels in a cascade, propagates the remaining features, and uses a substantially larger generative model. We therefore isolate the effect of latent geometry separately in Section 4.3, where Euclidean and Riemannian flow matching use exactly the same architecture and training procedure. Finally, all compared baselines use ground-truth context camera poses, whereas our method operates on unposed context views. Additional baseline details are provided in Section B.3.

Evaluation protocols. We evaluate one-sided and two-sided context setups following Gen3R [23] with $N \in \{1, 2, 3, 4\}$ context views: one-sided uses the first N images of a sequence as context and

Table 1: **Geometry evaluation.** (a) Novel-view depth on ScanNet++ (metric ground truth) and RE10K (VGGT pseudo-depth; only VGGT-based baselines). (b) 3D scan generation: accuracy, completeness and Chamfer distance of fused context and target point maps, in cm. Best in **bold**.

(a) Novel-view depth generation

			ScanNet++							
			High overlap				Low overlap			
	N	Method	RMSE↓	ARel↓	δ↑	FID↓	RMSE↓	ARel↓	δ↑	FID↓
one-sided	1	Gen3R	0.386	0.223	0.723	55.86	0.543	0.301	0.571	64.75
		Matrix3D	0.473	0.277	0.707	54.30	0.608	0.319	0.537	69.71
		GLD	0.431	0.253	0.671	60.84	0.604	0.310	0.534	72.31
		Ours	**0.223**	**0.118**	**0.891**	**45.99**	**0.299**	**0.138**	**0.852**	**53.55**
	2	DepthSplat	0.488	0.295	0.582	48.47	0.540	0.289	0.590	52.85
		Matrix3D	0.390	0.221	0.751	46.87	0.516	0.255	0.658	59.89
		GLD	0.280	0.144	0.844	48.44	0.442	0.210	0.707	62.39
		Ours	**0.207**	**0.102**	**0.914**	**39.66**	**0.283**	**0.121**	**0.882**	**49.80**
	3	DepthSplat	0.461	0.274	0.606	40.98	0.541	0.294	0.581	50.45
		Matrix3D	0.351	0.189	0.787	38.80	0.468	0.235	0.679	54.82
		GLD	0.253	0.129	0.865	38.40	0.407	0.188	0.745	58.45
		Ours	**0.211**	**0.100**	**0.915**	**32.89**	**0.282**	**0.120**	**0.876**	**46.58**
	4	DepthSplat	0.441	0.249	0.634	32.99	0.536	0.289	0.581	48.83
		Matrix3D	0.341	0.175	0.802	29.98	0.453	0.228	0.689	52.23
		GLD	0.226	0.117	0.893	27.77	0.390	0.179	0.770	53.15
		Ours	**0.217**	**0.102**	**0.907**	**24.73**	**0.280**	**0.118**	**0.876**	**44.04**
two-sided	2	DepthSplat	0.457	0.258	0.630	30.72	0.526	0.298	0.557	53.30
		Gen3R	0.344	0.185	0.786	28.50	0.512	0.279	0.585	51.82
		Matrix3D	0.354	0.183	0.810	29.30	0.442	0.227	0.702	49.91
		GLD	0.257	0.120	0.873	27.94	0.340	0.168	0.778	49.24
		Ours	**0.220**	**0.094**	**0.908**	**23.64**	**0.276**	**0.128**	**0.856**	**41.41**
	4	DepthSplat	0.378	0.201	0.713	20.37	0.453	0.263	0.607	35.73
		Matrix3D	0.306	0.145	0.852	18.28	0.372	0.190	0.760	33.45
		GLD	0.216	0.087	0.905	15.93	0.247	0.115	0.877	30.18
		Ours	**0.211**	**0.085**	**0.921**	**13.23**	**0.228**	**0.100**	**0.907**	**26.14**

			RE10K							
			High overlap				Low overlap			
	N	Method	RMSE↓	ARel↓	δ↑	FID↓	RMSE↓	ARel↓	δ↑	FID↓
one-sided	1	Gen3R	0.248	0.217	0.700	46.94	0.271	0.240	0.664	45.58
		GLD	0.275	0.257	0.650	42.22	0.303	0.289	0.607	43.34
		Ours	**0.176**	**0.161**	**0.842**	**32.46**	**0.203**	**0.176**	**0.813**	**34.18**
	2	GLD	0.240	0.215	0.729	39.16	0.272	0.249	0.677	39.97
		Ours	**0.171**	**0.157**	**0.851**	**32.04**	**0.198**	**0.172**	**0.820**	**34.38**
	3	GLD	0.215	0.184	0.773	37.13	0.246	0.211	0.736	38.73
		Ours	**0.168**	**0.153**	**0.859**	**31.81**	**0.195**	**0.170**	**0.827**	**34.32**
	4	GLD	0.197	0.170	0.812	35.72	0.228	0.194	0.772	37.90
		Ours	**0.166**	**0.151**	**0.860**	**31.69**	**0.192**	**0.169**	**0.831**	**33.89**
two-sided	2	Gen3R	0.131	0.119	0.886	50.90	0.145	0.150	0.828	63.08
		GLD	**0.064**	**0.044**	**0.965**	**30.04**	**0.096**	**0.091**	**0.925**	**43.75**
		Ours	0.094	0.075	0.940	30.28	0.118	0.107	0.901	44.17
	4	GLD	**0.043**	**0.026**	**0.986**	**24.19**	**0.055**	**0.040**	**0.974**	**33.02**
		Ours	0.082	0.061	0.951	28.57	0.094	0.083	0.931	39.13

(b) 3D scan generation

	ScanNet++				ETH3D			
Method	Acc↓	Comp↓	CH↓	Rel%↓	Acc↓	Comp↓	CH↓	Rel%↓
Gen3R	25.65	26.24	25.95	3.66	48.64	40.27	44.45	1.65
Matrix3D	22.59	16.87	19.73	2.75	41.00	22.20	31.60	1.26
GLD	13.90	12.58	13.24	1.85	**19.53**	19.28	**19.40**	**0.69**
Ours	**12.45**	**10.42**	**11.43**	**1.55**	25.97	**14.76**	20.36	0.74

the last as target, two-sided uses the middle image as target with $N/2$ context views on each side (Fig. C.2). DepthSplat requires at least two context views and Gen3R one or two, depending on the setting. We report two overlap regimes between target and context, *high* ($\sim$60% for ScanNet++, 95% for RE10K) and *low* ($\sim$35% and 88%). For 3D scan generation we use the two-sided low-overlap setting on ScanNet++ and context and target frames of ETH3D with $\sim$40% overlap. The point maps of the context and the two enclosed target views are fused into one scene cloud, aligned to ground truth with the Umeyama algorithm [62], and both are downsampled to 20 k points by farthest point sampling. All results use the Riemannian Euler ODE solver with 20 steps ($\sim$0.7 s per sample on an RTX A6000; step-count ablation in Section D.3). Resolution unification, overlap categories, depth scale and point alignment are detailed in Section C.

Metrics. Following prior work [23, 75, 71, 65], we report PSNR, SSIM [68], LPIPS [84], and FID [21] for RGB, RMSE, absolute relative error (ARel), δ<1.25, and FID for depth, and Accuracy, Completeness, and Chamfer distance for point maps. On ScanNet++, where ground-truth (GT) depth is available, a per-target visibility mask separates the regions observed in the context views from unobserved ones: FID is computed on the prediction inpainted into the unobserved region of the GT frame, all other metrics on observed regions only, so that reconstruction and the generation of plausible unobserved content are evaluated separately (Section C.4).

4.1 Novel view generation

Depth prediction. Table 1a reports depth prediction on ScanNet++ and RE10K. ScanNet++ provides metric depth ground truth, while RE10K is evaluated against VGGT pseudo-depth and is therefore compared only with Gen3R and GLD, which likewise use VGGT as their geometric prior. On ScanNet++, our method achieves the best performance across all metrics, context counts, overlap regimes, and both one-sided and two-sided settings. The gains are largest in the more challenging one-sided, low-overlap regime, where larger camera offsets require greater extrapolation beyond the observed scene. In this setting, our method reduces RMSE by 28-45% and ARel by 34-54% relative to the strongest baseline for each context count, while improving δ by 0.11-0.28 in absolute terms. On RE10K, where depth is evaluated against pseudo ground truth, we outperform both generative baselines in the one-sided setting, while GLD is better in the two-sided one. Across both datasets, our largest gains occur in the more extrapolative settings, where the target is less constrained by

Table 2: **Novel-view RGB generation on ScanNet++ (left) and RE10K (right).** Best in **bold**.

	N	Method	ScanNet++ High overlap PSNR↑	SSIM↑	LPIPS↓	FID↓	ScanNet++ Low overlap PSNR↑	SSIM↑	LPIPS↓	FID↓	RE10K High overlap PSNR↑	SSIM↑	LPIPS↓	FID↓	RE10K Low overlap PSNR↑	SSIM↑	LPIPS↓	FID↓
one-sided	1	Gen3R	15.39	0.543	0.143	65.78	13.86	0.475	0.153	77.66	14.66	0.475	0.367	**40.94**	14.24	0.463	0.397	**43.27**
		Matrix3D	14.78	0.535	0.159	63.09	13.21	0.477	0.159	76.18	14.03	0.467	0.410	47.07	13.36	0.449	0.452	49.81
		GLD	13.97	0.509	0.189	82.72	13.25	0.463	0.170	98.40	14.41	0.455	0.458	66.96	14.06	0.445	0.484	69.35
		Ours	**18.99**	**0.677**	**0.085**	**49.73**	**17.61**	**0.620**	**0.086**	**63.86**	**15.80**	**0.511**	**0.327**	47.20	**15.19**	**0.496**	**0.356**	49.98
	2	DepthSplat	13.51	0.514	0.239	94.03	11.20	0.365	0.294	137.2	10.39	0.321	0.537	148.1	9.63	0.277	0.591	169.0
		Matrix3D	15.95	0.572	0.151	51.57	14.33	0.515	0.154	70.52	14.17	0.470	0.401	47.65	13.64	0.458	0.436	49.69
		GLD	16.10	0.601	0.149	60.22	14.50	0.527	0.156	84.30	15.09	0.488	0.408	57.29	14.52	0.471	0.444	60.30
		Ours	**19.13**	**0.681**	**0.096**	**42.16**	**17.76**	**0.630**	**0.095**	**58.86**	**15.87**	**0.513**	**0.324**	**46.64**	**15.23**	**0.498**	**0.353**	**49.32**
	3	DepthSplat	15.15	0.607	0.201	67.18	11.94	0.417	0.280	117.7	11.38	0.371	0.511	145.6	10.49	0.322	0.563	166.32
		Matrix3D	16.82	0.599	0.145	41.55	14.56	0.528	0.158	67.16	14.28	0.475	0.392	48.59	13.74	0.462	0.423	49.58
		GLD	16.93	0.639	0.147	45.90	15.04	0.553	0.154	74.46	15.55	0.512	0.378	53.06	14.92	0.489	0.417	56.99
		Ours	**19.35**	**0.685**	**0.105**	**35.30**	**17.78**	**0.632**	**0.102**	**55.10**	**15.92**	**0.514**	**0.321**	**45.92**	**15.29**	**0.500**	**0.350**	**49.02**
	4	DepthSplat	16.63	0.661	0.181	44.24	12.47	0.441	0.280	108.6	11.62	0.385	0.507	146.6	10.79	0.339	0.556	162.6
		Matrix3D	17.93	0.629	0.142	30.43	14.56	0.532	0.167	62.34	14.53	0.484	0.378	47.94	13.93	0.469	0.413	50.64
		GLD	17.86	0.681	0.142	31.99	15.37	0.573	0.154	67.42	15.86	**0.531**	0.360	49.60	15.24	**0.508**	0.395	53.44
		Ours	**19.59**	**0.691**	**0.113**	**25.67**	**17.73**	**0.632**	**0.110**	**51.86**	**15.98**	0.516	**0.317**	**45.11**	**15.36**	0.502	**0.347**	**48.91**
two-sided	2	DepthSplat	17.50	0.647	0.166	41.02	12.56	0.469	0.270	92.42	**24.61**	**0.833**	**0.092**	28.99	**20.54**	**0.724**	**0.192**	61.53
		Gen3R	17.27	0.596	0.159	34.77	14.00	0.498	0.214	67.73	21.27	0.718	0.113	**25.84**	18.02	0.598	0.214	**47.81**
		Matrix3D	17.40	0.605	0.153	31.70	14.94	0.545	0.178	59.42	19.69	0.659	0.155	43.36	17.28	0.578	0.236	63.06
		GLD	17.17	0.640	0.163	35.11	14.98	0.555	0.178	68.91	19.68	0.684	0.174	39.58	17.93	0.604	0.251	60.56
		Ours	**19.58**	**0.687**	**0.112**	**26.10**	**17.43**	**0.631**	**0.121**	**50.67**	18.44	0.592	0.195	40.83	16.83	0.539	0.265	57.03
	4	DepthSplat	**20.31**	**0.730**	0.143	20.45	16.34	0.599	0.224	44.62	**24.20**	**0.845**	**0.094**	33.40	**21.98**	**0.778**	**0.143**	52.32
		Matrix3D	19.23	0.661	0.133	17.55	15.89	0.572	0.203	35.26	21.82	0.722	0.111	36.55	19.76	0.655	0.157	49.60
		GLD	18.95	0.719	0.140	17.25	16.85	0.632	0.171	35.74	21.36	0.752	0.132	**31.77**	20.42	0.705	0.161	**42.51**
		Ours	20.18	0.703	**0.116**	**12.89**	**18.39**	**0.656**	**0.135**	**29.69**	19.08	0.610	0.170	38.11	17.77	0.567	0.217	49.41

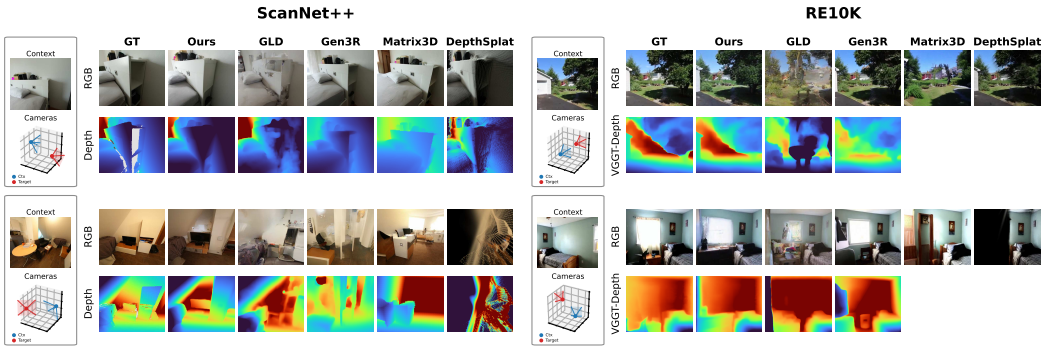


Figure 2: **Qualitative novel view generation on ScanNet++ (left) and RE10K (right).** For each scene, the columns show the target RGB and depth for the GT and all methods. On RE10K, depth is evaluated against VGGT pseudo-depth and is therefore shown only for the methods that share this backbone. Reconstruction-based DepthSplat leaves holes and streak artifacts where the target leaves its reconstructed support, GLD renders blurred or ghosted appearance, and Matrix3D and Gen3R produce plausible images with coarse or inconsistent depth, whereas our model recovers both appearance and geometry close to the ground truth.

the observations. Notably, our method achieves these results without requiring poses for the context views, unlike all competing baselines, and with a substantially smaller flow model than GLD ($\sim$120M vs. $\sim$1.3B parameters).

RGB prediction. Table 2 reports RGB novel view generation results on ScanNet++ and RE10K. On ScanNet++, our method achieves the strongest results throughout the one-sided setting and under low overlap, with the advantage again increasing for larger camera offsets similarly to depth. With a single context view in the low-overlap regime, we improve PSNR from 13.86 to 17.61 and FID from 76.18 to 63.86 over the strongest competing values. The same trend holds with additional context and in the two-sided setting. Only in the easiest two-sided, high-overlap configuration with $N{=}4$ DepthSplat obtains higher PSNR and SSIM (GLD also higher SSIM), while our method retains better LPIPS and FID. This is consistent with reconstruction-based methods being particularly effective when the target is densely bracketed by observed views, whereas our generative formulation is most

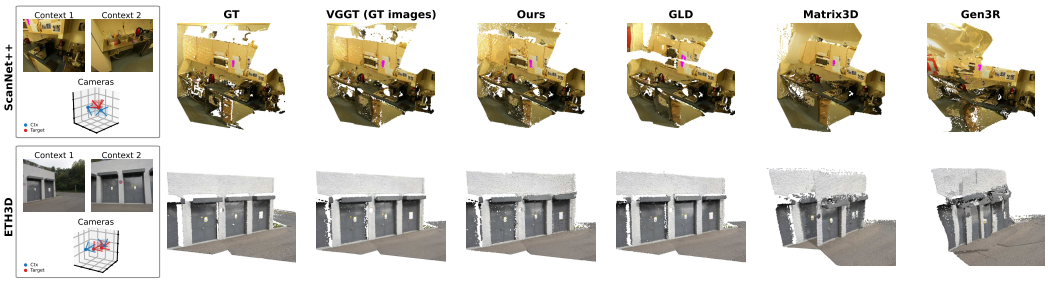


Figure 3: **Qualitative 3D scan generation on ScanNet++ (top) and ETH3D (bottom).** The columns show the point maps of the context and generated target views fused into a single scene, for the GT, for VGGT applied to the GT target images as an upper bound, and for all methods. Our model and GLD stay close to the reference geometry, whereas Matrix3D and Gen3R produce misaligned or duplicated structure.

beneficial when substantial content must be extrapolated beyond the observations. On RE10K, where the smoother camera motion makes the task more interpolation-dominated, the differences are smaller. In the one-sided setting, our method achieves the strongest PSNR and LPIPS across all context counts and is best or competitive in SSIM and FID. In the two-sided setting, DepthSplat and the other posed baselines benefit strongly from the dense interpolation setting and achieve higher reconstruction metrics. Overall, the RGB results complement the geometric evaluation: the largest gains occur in the more extrapolative ScanNet++ regimes, while differences narrow when the target is well supported by nearby context views.

Figure 2 provides qualitative examples for depth and RGB across these settings, highlighting the improved geometric consistency of our predictions, particularly under large camera offsets and limited context.

4.2 3D scan generation

To measure not only 3D reconstruction and generation quality but also the consistency between generated and observed frames, we evaluate the point maps of the context frames jointly with those of their enclosed target views. Table 1b reports the resulting fused-point-cloud metrics on ScanNet++ and ETH3D. On ScanNet++, our method achieves the best performance across all metrics, reducing the Chamfer distance from $13.24\,\text{cm}$ to $11.43\,\text{cm}$ relative to the strongest baseline, GLD, by improving both accuracy and completeness.

On the out-of-distribution ETH3D benchmark, our method remains competitive despite the substantially larger physical scene scale. GLD obtains the best accuracy and slightly lower Chamfer distance (19.40 vs. $20.36\,\text{cm}$), whereas our method achieves the best completeness (14.76 vs. $19.28\,\text{cm}$). The scale-normalized relative error remains similarly low for both methods (0.69% vs. 0.74%), indicating that our generated geometry transfers well to out-of-distribution scenes.

Figure 3 qualitatively illustrates these results. Fusing the generated target point maps with those of the context frames yields coherent, well-aligned 3D scans, whereas competing methods exhibit larger geometric distortions or incomplete structure. Additional qualitative results for novel view generation and 3D scan generation are shown in Figs. D.2 to D.4.

4.3 Euclidean vs. Riemannian flow matching

We directly ablate the effect of latent geometry by comparing Euclidean and Riemannian velocity flow matching with matched architecture, data, and training schedule. For the Euclidean baseline, latent channels are standardized using training-set statistics and flow matching is performed in the resulting unconstrained Euclidean space. Before decoding, we invert the standardization transform. Our model instead defines the flow on $\mathcal{M}$ and keeps sampling on it.

As shown in Table 3 and Fig. 4 (more examples in Fig. D.5), respecting the latent geometry substantially improves both RGB and depth generation. On the two-sided, low-overlap ScanNet++ setting with $N{=}4$ context views, RGB PSNR increases by $1.41\,\text{dB}$ and SSIM by 0.103, while LPIPS and FID decrease by 41.8% and 23.9%, respectively. For depth, RMSE and ARel decrease by 21.9%

Table 3: **Ablation on flow geometry.** Euclidean vs. Riemannian flow matching, matched architecture and training (two-sided ScanNet++, $N{=}4$, low overlap). Best in **bold**.

	RGB				Depth			
Method	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	RMSE ↓	ARel ↓	δ ↑	FID ↓
Eucl.	16.98	0.553	0.232	39.00	0.292	0.178	0.775	28.68
Riem.	**18.39**	**0.656**	**0.135**	**29.69**	**0.228**	**0.100**	**0.907**	**26.14**

and 43.8%, δ increases from 0.775 to 0.907, and FID decreases by 8.9%. The same trend holds across all context and overlap configurations (Section D.5).

Why the latent geometry matters. Euclidean flow is harder for two related reasons. Its straight-line transport contains radial motion, and the optimal velocity $u^\star(x,t) = \big(\mathbb{E}[x_1 \mid x_t = x] - x\big)/(1-t)$ requires the model to reproduce the current state through the $-x/(1-t)$ term, which near $t = 0$ is dominated by high-dimensional isotropic noise that cannot in general be preserved through the model's lower-dimensional representation. Riemannian flow restricts the velocity to the tangent space, removing both the radial component and this state-reconstruction burden. The learned trajectories show the difference: Euclidean samples terminate at $\|x_1\|/R = 1.33$, where R is the empirical radius of real feature blocks, while Riemannian samples stay at unit radius. In addition to this radial deviation, the Euclidean endpoint also deviates in the directions it occupies. To quantify it, we measure the off-subspace energy E_{off} (eq. (D.1)), the fraction of centered token energy that lies outside the per-block principal subspace capturing 99% of the real-token variance. Intuitively, it is the share of generated-token variation in directions that real decoder inputs never occupy. Real tokens have $E_{\text{off}} = 1\%$, compared with 33% for Euclidean and 2% for Riemannian generated tokens, and Riemannian endpoints are better aligned with the ground-truth tokens (cosine similarity 0.85 versus 0.66). Euclidean flow thus leaves the decoder-relevant manifold both in radius and in direction. Trajectory, capacity, and prediction-target analyses are provided in Section D.6.

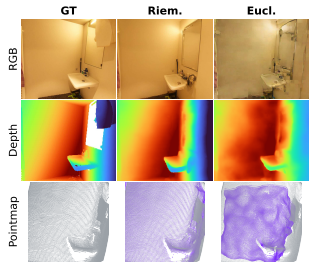


Figure 4: Eucl. vs. Riem. flow on a ScanNet++ scene.

5 Conclusion

We introduce a generative framework for 3D scene modeling that performs flow matching directly in the latent space of a frozen geometric foundation model, VGGT, rather than committing to an explicit downstream representation such as Gaussians, meshes, or video-VAE latents. The key observation is that VGGT's post-LayerNorm tokens do not occupy a Euclidean space but a product manifold of four zero-mean hyperspheres, which standard flow matching ignores. Our conditional Riemannian flow matching formulation respects this geometry, keeping generated tokens on the valid data manifold required by the frozen decoding heads. The resulting model is order-invariant in the context views, operates from as few as a single unposed view, and requires only the relative target pose, removing the trajectory and adjacency assumptions of video-diffusion and explicit-reconstruction baselines. On ScanNet++ and RE10K it improves over recent scene-generation methods in per-view geometry and remains competitive in appearance, and on the 3D scan generation task it produces coherent, well-aligned point clouds that generalize to the out-of-distribution ETH3D setting. Together these results suggest latent-space Riemannian flow matching on geometric foundation models is a viable paradigm for generative 3D reconstruction.

Limitations and future work. Our current framework generates one target view at a time. The 3D scan experiment shows that jointly decoded frames already remain mutually consistent, yet extending it to joint multi-view generation could improve global consistency across synthesized views. More broadly, casting reconstruction as a flow on VGGT's latent manifold opens directions unavailable to a purely feed-forward model: sampling the flow yields a distribution over plausible scenes, offering a natural route to uncertainty estimation in unobserved regions, while the structure of the learned trajectories invites exploration for tasks such as multi-view alignment and scene editing.

Acknowledgements and disclosures

Funding transparency statement. Funding in direct support of this work: Austrian Research Promotion Agency (FFG) project FO999922723; a TU Wien KUWI grant (kurzfristige wissenschaftliche Arbeiten und fachspezifische Kurse im Ausland) supporting LW's research visit at Imperial College London; UKRI Engineering and Physical Sciences Research Council (EPSRC) Future Leaders Fellowship MR/Y018818/1 (TB); compute from the UK AI Research Resource (AIRR), grant 0251-4584-0945-1, from the Austrian Scientific Computing (ASC) infrastructure, in particular the MUSICA system, project 201194, and from the EuroHPC Joint Undertaking, project EHPC-DEV-2025D11-012, on the EuroHPC supercomputer LEONARDO hosted by CINECA.

Competing interests. The authors declare no competing financial interests related to this work.

References

[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5855–5864, 2021.

[2] Joey Bose, Tara Akhound-Sadegh, Guillaume Huguet, Kilian Fatras, Jarrid Rector-Brooks, Cheng-Hao Liu, Andrei Cristian Nica, Maksym Korablyov, Michael M Bronstein, and Alexander Tong. SE(3)-stochastic flow matching for protein backbone generation. In *International Conference on Learning Representations (ICLR)*, 2024.

[3] Nicolas Boumal. *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023.

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9630–9640, 2021.

[5] Ata Çelen, Jaewoo Jung, Federico Tombari, Marc Pollefeys, Sunghwan Hong, Michael Niemeyer, and Daniel Barath. GenRec: Knowing where to reconstruct and where to generate. *arXiv preprint arXiv:2608.17832*, 2026.

[6] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelSplat: 3D Gaussian splats from image pairs for scalable generalizable 3D reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19457–19467, 2024.

[7] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14124–14133, 2021.

[8] Ricky T. Q. Chen and Yaron Lipman. Flow matching on general geometries. In *International Conference on Learning Representations (ICLR)*, 2024.

[9] Weiliang Chen, Jiayi Bi, Yuanhui Huang, Wenzhao Zheng, and Yueqi Duan. SceneCompleter: Dense 3D scene completion for generative novel view synthesis. *arXiv preprint arXiv:2506.10981*, 2025.

[10] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. MVSplat: Efficient 3D Gaussian splatting from sparse multi-view images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 370–386, 2024.

[11] Yixiang Dai, Fan Jiang, Chiyu Wang, Mu Xu, and Yonggang Qi. FantasyWorld: Geometry-consistent world modeling via unified video and 3D prediction. In *International Conference on Learning Representations (ICLR)*, 2026.

[12] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12882–12891, 2022.

[13] Noam Elata, Bahjat Kawar, Yaron Ostrovsky-Berman, Miriam Farber, and Ron Sokolovsky. Novel view synthesis with pixel-space diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26756–26766, 2025.

[14] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12873–12883, 2021.

[15] Kenneth Falconer. *Fractal Geometry: Mathematical Foundations and Applications*. Wiley, 2nd edition, 2003.

[16] Tobias Fischer, Samuel Rota Bulò, Yung-Hsu Yang, Nikhil Keetha, Lorenzo Porzi, Norman Müller, Katja Schwarz, Jonathon Luiten, Marc Pollefeys, and Peter Kontschieder. FlowR: Flowing from sparse to dense 3D reconstructions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 27702–27712, 2025.

[17] Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul P. Srinivasan, Jonathan T. Barron, and Ben Poole. CAT3D: Create anything in 3D with multi-view diffusion models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[18] Sensen Gao, Zhaoqing Wang, Qihang Cao, Dongdong Yu, Changhu Wang, Tongliang Liu, Mingming Gong, and Jiawang Bian. OneWorld: Taming scene generation with 3D unified representation autoencoder. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2026.

[19] Boris Hanin and Mark Sellke. Approximating continuous functions by ReLU nets of minimal width. *arXiv preprint arXiv:1710.11278*, 2017.

[20] Yannan He, Garvita Tiwari, Tolga Birdal, Jan Eric Lenssen, and Gerard Pons-Moll. NRDF: Neural Riemannian distance fields for learning articulated pose priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1661–1671, 2024.

[21] 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 (NeurIPS)*, 30, 2017.

[22] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024.

[23] Jiaxin Huang, Yuanbo Yang, Bangbang Yang, Lin Ma, Yuewen Ma, and Yiyi Liao. Gen3R: 3D scene generation meets feed-forward reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 25358–25369, 2026.

[24] Guillaume Huguet, James Vuckovic, Kilian Fatras, Eric Thibodeau-Laufer, Pablo Lemos, Riashat Islam, Chenghao Liu, Jarrid Rector-Brooks, Tara Akhound-Sadegh, Michael Bronstein, Alexander Tong, and Avishek Joey Bose. Sequence-augmented SE(3)-flow matching for conditional protein generation. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:33007–33036, 2024.

[25] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting NeRF on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5885–5894, 2021.

[26] Wonbong Jang, Jonathan Tremblay, and Lourdes Agapito. DT-NVS: Diffusion transformers for novel view synthesis. *arXiv preprint arXiv:2511.08823*, 2025.

[27] Wooseok Jang, Seonghu Jeon, Jisang Han, Jinhyeok Choi, Minkyung Kwon, Seungryong Kim, Saining Xie, and Sainan Liu. Repurposing geometric foundation models for multi-view diffusion. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 609–627, 2026.

[28] Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. LVSM: A large view synthesis model with minimal 3D inductive bias. In *International Conference on Learning Representations (ICLR)*, 2025.

[29] Jesse Johnson. Deep, skinny neural networks are not universal approximators. In *International Conference on Learning Representations (ICLR)*, 2019.

[30] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):139:1–139:14, 2023.

[31] Evan Kim, Hyunwoo Ryu, Thomas W. Mitchel, and Vincent Sitzmann. Scaling view synthesis transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[32] Mijeong Kim, Seonguk Seo, and Bohyung Han. InfoNeRF: Ray entropy minimization for few-shot neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12912–12921, 2022.

[33] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[34] Hanyang Kong, Xingyi Yang, and Xinchao Wang. Generative sparse-view Gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26745–26755, 2025.

[35] Amandeep Kumar and Vishal M Patel. Learning on the manifold: Unlocking standard diffusion transformers with representation encoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2026.

[36] Vincent Leroy, Yohann Cabon, and Jerome Revaud. Grounding image matching in 3D with MASt3R. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

[37] Ruilong Li, Brent Yi, Junchen Liu, Hang Gao, Yi Ma, and Angjoo Kanazawa. Cameras as relative positional encoding. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[38] Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 36115–36125, 2026.

[39] Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth Anything 3: Recovering the visual space from any views. In *International Conference on Learning Representations (ICLR)*, 2026.

[40] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

[41] Fangfu Liu, Wenqiang Sun, Hanyang Wang, Yikai Wang, Haowen Sun, Junliang Ye, Jun Zhang, and Yueqi Duan. ReconX: Reconstruct any scene from sparse views with video diffusion model. *arXiv preprint arXiv:2408.16767*, 2024.

[42] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3D object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9298–9309, 2023.

[43] Xiaohan Lu, Jiaye Fu, Jiaqi Zhang, Zetian Song, Chuanmin Jia, and Siwei Ma. ProSplat: Improved feed-forward 3D Gaussian splatting for wide-baseline sparse views. *arXiv preprint arXiv:2506.07670*, 2025.

[44] Yuanxun Lu, Jingyang Zhang, Tian Fang, Jean-Daniel Nahmias, Yanghai Tsin, Long Quan, Xun Cao, Yao Yao, and Shiwei Li. Matrix3D: Large photogrammetry model all-in-one. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11250–11263, 2025.

[45] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[46] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[47] Benjamin Kurt Miller, Ricky TQ Chen, Anuroop Sriram, and Brandon M Wood. FlowMM: Generating materials with Riemannian flow matching. In *International Conference on Machine Learning (ICML)*, pages 35664–35686, 2024.

[48] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):1–15, 2022.

[49] Jebastin Nadar, Simone Foti, and Tolga Birdal. PoseD-Flow: Versatile and guided flow matching model of human pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21165–21175, 2026.

[50] Nithin Gopalakrishnan Nair, Srinivas Kaza, Xuan Luo, Vishal M. Patel, Stephen Lombardi, and Jungyeon Park. Scaling transformer-based novel view synthesis models with token disentanglement and synthetic data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 28567–28576, 2025.

[51] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. RegNeRF: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5480–5490, 2022.

[52] Sejun Park, Chulhee Yun, Jaeho Lee, and Jinwoo Shin. Minimum width for universal approximation. In *International Conference on Learning Representations (ICLR)*, 2021.

[53] Xuanchi Ren, Tianchang Shen, Jiahui Huang, Huan Ling, Yifan Lu, Merlin Nimier-David, Thomas Müller, Alexander Keller, Sanja Fidler, and Jun Gao. GEN3C: 3D-informed world-consistent video generation with precise camera control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6121–6132, 2025.

[54] 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 (CVPR)*, pages 10684–10695, 2022.

[55] Axel Sauer, Tero Karras, Samuli Laine, Andreas Geiger, and Timo Aila. StyleGAN-T: Unlocking the power of GANs for fast large-scale text-to-image synthesis. In *International Conference on Machine Learning (ICML)*, pages 30105–30118, 2023.

[56] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[57] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[58] Thomas Schöps, Johannes L Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3260–3269, 2017.

[59] Junyoung Seo, Kazumi Fukuda, Takashi Shibuya, Takuya Narihira, Naoki Murata, Shoukang Hu, Chieh-Hsin Lai, Seungryong Kim, and Yuki Mitsufuji. GenWarp: Single image to novel views with semantic-preserving generative warping. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

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

[61] Xiangyu Sun, Shijie Wang, Fengyi Zhang, Lin Liu, Caiyan Jia, Ziying Song, Zi Huang, and Yadan Luo. VGGT-World: Transforming VGGT into an autoregressive geometry world model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2026.

[62] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.

[63] Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge University Press, 2018.

[64] Yuhao Wan, Lijuan Liu, Jingzhi Zhou, Zihan Zhou, Xuying Zhang, Dongbo Zhang, Shaohui Jiao, Qibin Hou, and Ming-Ming Cheng. GeoWorld: Providing full-frame geometry features to facilitate 3D scene generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 38–57, 2026.

[65] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[66] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[67] Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. In *International Conference on Learning Representations (ICLR)*, 2026.

[68] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[69] Zhouxia Wang, Ziyang Yuan, Xintao Wang, Yaowei Li, Tianshui Chen, Menghan Xia, Ping Luo, and Ying Shan. MotionCtrl: A unified and flexible motion controller for video generation. In *ACM SIGGRAPH 2024 Conference Papers*, 2024.

[70] Jiaxin Wei, Stefan Leutenegger, and Simon Schaefer. GSFix3D: Diffusion-guided repair of novel views in Gaussian splatting. In *International Conference on 3D Vision (3DV)*, 2026.

[71] Christopher Wewer, Kevin Raj, Eddy Ilg, Bernt Schiele, and Jan Eric Lenssen. latentSplat: Autoencoding variational Gaussians for fast generalizable 3D reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 456–473, 2024.

[72] Haoyu Wu, Diankun Wu, Tianyu He, Junliang Guo, Yang Ye, Yueqi Duan, and Jiang Bian. Geometry forcing: Marrying video diffusion and 3D representation for consistent world modeling. In *International Conference on Learning Representations (ICLR)*, 2026.

[73] Rundi Wu, Ben Mildenhall, Philipp Henzler, Keunhong Park, Ruiqi Gao, Daniel Watson, Pratul P. Srinivasan, Dor Verbin, Jonathan T. Barron, Ben Poole, and Aleksander Hołyński. ReconFusion: 3D reconstruction with diffusion priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[74] Sibo Wu, Congrong Xu, Binbin Huang, Andreas Geiger, and Anpei Chen. GenFusion: Closing the loop between reconstruction and generation via videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6078–6088, 2025.

[75] Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. DepthSplat: Connecting Gaussian splatting and depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16453–16463, 2025.

[76] Haibo Yang, Yang Chen, Yingwei Pan, Zhineng Chen, Ting Yao, and Tao Mei. DreamWorld: Geometry-grounded video diffusion for 3D-consistent world modeling. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2026.

[77] Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[78] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++: A high-fidelity dataset of 3D indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12–22, 2023.

[79] Xingyilang Yin, Qi Zhang, Jiahao Chang, Ying Feng, Qingnan Fan, Xi Yang, Chi-Man Pun, Huaqi Zhang, and Xiaodong Cun. GSFixer: Improving 3D Gaussian splatting with reference-guided video diffusion priors. In *International Conference on Machine Learning (ICML)*, 2026.

[80] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4578–4587, 2021.

[81] Wangbo Yu, Jinbo Xing, Li Yuan, Wenbo Hu, Xiaoyu Li, Zhipeng Huang, Xiangjun Gao, Tien-Tsin Wong, Ying Shan, and Yonghong Tian. ViewCrafter: Taming video diffusion models for high-fidelity novel view synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[82] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. GS-LRM: Large reconstruction model for 3D Gaussian splatting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 1–19, 2024.

[83] Qihang Zhang, Shuangfei Zhai, Miguel Ángel Bautista Martin, Kevin Miao, Alexander Toshev, Joshua Susskind, and Jiatao Gu. World-consistent video diffusion with explicit 3D modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21685–21695, 2025.

[84] 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/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018.

[85] Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *International Conference on Learning Representations (ICLR)*, 2026.

[86] Junsheng Zhou, Weiqi Zhang, and Yu-Shen Liu. DifFGS: Functional Gaussian splatting diffusion. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[87] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Transactions on Graphics*, 37 (4):65:1–65:12, 2018.

Appendix

A Theoretical background

A.1 Riemannian Flow Matching

Definition 1 (Riemannian Flow [8]). *A time-dependent **flow** is a one-parameter family of diffeomorphisms $\{\psi_t : \mathcal{M} \to \mathcal{M}\}_{t=0}^{1}$ defined by integrating instantaneous deformations represented by a time-dependent vector field $u_t \in \Gamma(\mathcal{T}\mathcal{M})$ on the tangent space (Riemannian flow-matching field). ψ_t is defined by solving the following Riemannian ordinary differential equation (ODE) on $\mathcal{M}$ over $t \in [0, 1]$:*

$$\frac{d}{dt}\psi_t(x) = u_t\left(\psi_t(x)\right), \quad \psi_0(x) = x. \tag{A.1}$$

Definition 2 (Probability path). *Let $\mathcal{P}(\mathcal{M})$ denote the space of probability distributions on $\mathcal{M}$. A probability path $p_t : [0, 1] \to \mathcal{P}(\mathcal{M})$ interpolates between two distributions $p_0, p_1 \in \mathcal{P}(\mathcal{M})$ indexed by $t \in [0, 1]$. p_t is said to be **generated** by ψ_t if it* pushes forward *$p_0 := p(x_0)$ to $p_1 := p(x_1)$ following u_t,* i.e.*$p_t = [\psi_t]_{\#}(p_0)$. We define a smooth probability path between data p_1 and a reference p_0 as*

$$p_t(x) = \int_{\mathcal{M}} p_t(x \mid x_1)\, p_1(x_1)\ \mathrm{d}V(x_1), \tag{A.2}$$

where $p_t(x \mid x_1)$ is the conditional probability path obtained by pushing the source forward along the conditional flow $\psi_t(\cdot \mid x_1)$ generated by the conditional vector field $u_t(x \mid x_1)$ defined below, i.e.*$p_t(\cdot \mid x_1) = [\psi_t(\cdot \mid x_1)]_{\#}\, p_0$, so that $p_0(\cdot \mid x_1) = p_0$ and $p_1(\cdot \mid x_1)$ concentrates at x_1.*

Remark 1 (Velocity field). *p_t satisfies a continuity (Liouville) equation on the manifold: $\partial_t p_t + \mathrm{div}_g\left(p_t u_t\right) = 0$, where u_t is the **velocity field** transporting probability mass along the manifold, where div_g is the divergence on $\mathcal{M}$.*

We are now ready to define Riemannian-FM (RFM).

Definition 3 (RFM). *Given a probability path p_t, subject to the boundary conditions $p_0 = p_{\mathrm{source}}$ and $p_1 = p_{\mathrm{target}}$, as well as an associated flow ψ_t, Riemannian flow matching learns a* continuous normalizing flow *by directly regressing u_t through a neural network $v_w(x, t)$ parametrized by w.*

Definition 4 (Riemannian Conditional FM). *The vanilla RFM objective is intractable as we do not have access to the closed-form u_t generating p_t. Instead, we regress v_w against a tractable* conditional vector field *$u_t(x \mid x_1)$, generating a* conditional probability path *$p_t(x \mid x_1)$ which can recover the target unconditional path by marginalization:*

$$u_t(x) = \int_{\mathcal{M}} u_t(x \mid x_1) \frac{p_t(x \mid x_1) p(x_1)}{p_t(x)}\, \mathrm{dV}_{x_1}. \tag{A.3}$$

Definition 5 (Generating conditional vector field). *RFM defines a vector field $u_t\left(x \mid x_1\right)$ that generates $p_t(x \mid x_1)$ through a distance d by enforcing $d\left(\psi_t\left(x \mid x_1\right), x_1\right) = \kappa(t) d\left(x, x_1\right)$. The minimal-norm conditional field is [8]:*

$$u_t\left(x \mid x_1\right) = \frac{d}{dt} \log \kappa(t) \frac{d\left(x, x_1\right)}{\left\|\nabla d\left(x, x_1\right)\right\|_g^2} \nabla d\left(x, x_1\right). \tag{A.4}$$

For the geodesic distance $d := d_g$ and $\kappa(t) := 1 - t$, $\left\|\nabla d_g\right\|_g = 1$ and $d_g \nabla d_g = \nabla \frac{1}{2} d_g^2 = -\mathrm{Log}_x\left(x_1\right)$, giving

$$u_t\left(x \mid x_1\right) = \frac{1}{1-t} \mathrm{Log}_x\left(x_1\right) \tag{A.5}$$

With this choice of time scheduling Chen & Lipman [8] then define an explicit Riemannian conditional FM (RCFM) objective for learning as:

$$\mathcal{L}_{\mathrm{RCFM}} = \mathbb{E}_{t, p(x_1), p(x_0)} \left\| v_w(x_t, t) + d(x_0, x_1) \frac{\mathrm{grad}\ d(x_t, x_1)}{\|\mathrm{grad}\ d(x_t, x_1)\|_g^2} \right\|_g^2, \tag{A.6}$$

whose gradient is the same as that of RFM. We use eq. (A.6) to train our model, *i.e.*, to obtain the parameters w of v_w. Here, $t \in \mathcal{U}(0,1)$ and $d(\cdot,\cdot)$ is the geodesic distance.

Parallel-Transport Form of the RCFM Target. In our implementation, we compute the conditional target velocity via parallel transport rather than the gradient form of eq. (A.6). The two are equivalent on constant-speed geodesics, as we now show.

Recall that the conditional vector field at x_t is

$$u_t(x_t \mid x_1) = \frac{1}{1-t}\mathrm{Log}_{x_t}(x_1). \tag{A.7}$$

Let $P_{x_0 \to x_t} : \mathcal{T}_{x_0}\mathcal{M} \to \mathcal{T}_{x_t}\mathcal{M}$ denote parallel transport[1] along the geodesic from x_0 to x_t defined by $x_t = \mathrm{Exp}_{x_0}(t \cdot \mathrm{Log}_{x_0}(x_1))$. Then

$$u_t(x_t \mid x_1) \;=\; P_{x_0 \to x_t}\big(\mathrm{Log}_{x_0}(x_1)\big). \tag{A.8}$$

To see this, note that the conditional path is the geodesic $\gamma(t) = \mathrm{Exp}_{x_0}(t\,\mathrm{Log}_{x_0}(x_1))$ and the target velocity is $\dot\gamma(t)$. A geodesic is self-parallel, $\nabla_{\dot\gamma}\dot\gamma = 0$, so its velocity at time t is the parallel transport of its initial velocity $\mathrm{Log}_{x_0}(x_1)$; since $d_g(x_t, x_1) = (1-t)\, d_g(x_0, x_1)$, this is the same vector as $\frac{1}{1-t}\mathrm{Log}_{x_t}(x_1)$.

We use eq. (A.8) in our implementation because it is well-conditioned for all t: the logarithm is evaluated once, between x_0 and x_1, which are far apart, and transported by a rotation in the plane they span. The gradient form instead requires $\mathrm{Log}_{x_t}(x_1)$ between points that coincide as $t \to 1$, where both the $\arccos$ and the projection lose precision, and then divides the result by $1-t$.

B Implementation details

B.1 RGB head: architecture and training

Since VGGT [65] produces depth, point, and camera predictions but no RGB output, we train a dedicated RGB head on frozen VGGT tokens. Like VGGT's native prediction heads, our RGB head consumes the concatenated patch tokens from the same encoder blocks $4, 11, 17, 23$, $x = \bigoplus_{i\in\{4,11,17,23\}} h_i$, with $x \in \mathbb{R}^{4\cdot 2\cdot 1024}$. The RGB head reconstructs a 3-channel RGB image at the input resolution of the images used to obtain the latent tokens. Its design follows a masked-autoencoder (MAE) ViT decoder trained with ℓ_1 + LPIPS + a patch-GAN, following the general design of GLD [27].

Architecture. The decoder is a non-hierarchical ViT that operates on each view independently, meaning that there is no cross-view attention. Given patch tokens $\mathbf{X} \in \mathbb{R}^{N\times 8192}$:

1. a shared `LayerNorm`(2048) is applied independently to each of the four feature slices, matching the normalization used by VGGT's DPT heads,
2. the concatenated 8192-dimensional token is projected to decoder width D,
3. a learned CLS token is prepended and a fixed 2D sine-cosine positional embedding,
4. L pre-norm transformer blocks with multi-head self-attention and GELU MLPs are applied,
5. a final `LayerNorm` and linear projection $D \to 14\cdot 14\cdot 3$ predict RGB pixels for each patch, which are subsequently unpatchified.

The decoder predicts ImageNet-normalized RGB, which is de-normalized to $[0,1]$ for evaluation. Architectural details are summarized in Table B.1 (right).

Training data. The RGB head is trained on the ScanNet++ and RealEstate10K training splits, sampled with equal probability. Sequence lengths are drawn from $2, 3, 4, 5$. Since decoding is performed independently per view, a batch of B sequences containing S views contributes $B \cdot S$ independent image reconstructions. We use $B = 8$ sequences per GPU, corresponding to 16-40 images per GPU and step (28 on average).

Objective. Let x_{rgb} denote a ground-truth frame and $\hat{x}_{\mathrm{rgb}}$ its reconstruction obtained by encoding the frame with frozen VGGT and decoding the resulting tokens. The generator objective is

[1] Often used to compare tangent vectors at distinct points [3].

Table B.1: **RGB decoder.** Left: reconstruction from GT VGGT tokens on ScanNet++ validation and RE10K test views. Right: decoder architecture. The parameter count includes the $8192 \to D$ input projection and excludes the fixed positional embedding.

Dataset	PSNR ↑	SSIM ↑	LPIPS ↓
ScanNet++	27.63	0.897	0.090
RealEstate10K	25.68	0.838	0.114

Component	Value
Width D	768
Layers L	12
Heads	12
MLP	3072
Params	91.8M

$$\mathcal{L}_G \;=\; \underbrace{\|\hat{x}_{\mathrm{rgb}} - x_{\mathrm{rgb}}\|_1 \;+\; \mathrm{LPIPS}(\hat{x}_{\mathrm{rgb}}, x_{\mathrm{rgb}})}_{\mathcal{L}_{\mathrm{rec}}} \;+\; 0.75\,\lambda_{\mathrm{ada}}\; \underbrace{\big(-\mathbb{E}\, D(\mathcal{A}(\mathcal{C}(\hat{x}_{\mathrm{rgb}})))\big)}_{\mathcal{L}_{\mathrm{gan}}}, \tag{B.1}$$

where LPIPS uses the VGG-16 variant of [84], evaluated on inputs mapped to $[-1, 1]$, $\mathcal{C}$ denotes a random 224×224 crop, $\mathcal{A}$ differentiable augmentation, and D the discriminator described below. The ℓ_1 term is the mean absolute error over pixels and color channels. Both reconstruction terms have unit weight and are active from the start of training.

Following VQGAN [14], the adversarial term is rescaled by the adaptive weight

$$\lambda_{\mathrm{ada}} \;=\; \mathrm{clip}\!\left(\frac{\|\nabla_W \mathcal{L}_{\mathrm{rec}}\|_2}{\|\nabla_W \mathcal{L}_{\mathrm{gan}}\|_2 + 10^{-6}},\; 0,\; 10^4\right), \tag{B.2}$$

computed with respect to the weights W of the final prediction layer.

The discriminator is trained with the hinge loss

$$\mathcal{L}_D \;=\; \tfrac{1}{2}\Big(\mathbb{E}\,\max(0, 1 - D(\mathcal{A}(\mathcal{C}(x_{\mathrm{rgb}})))) + \mathbb{E}\,\max(0, 1 + D(\mathcal{A}(\mathcal{C}(\hat{x}_{\mathrm{rgb}}))))\Big), \tag{B.3}$$

and returns one logit per patch position rather than a single score per image. $\mathbb{E}$ denotes a uniform mean over all discriminator outputs and all images in the batch. Within each discriminator step, the same crop window is applied to x_{rgb} and $\hat{x}_{\mathrm{rgb}}$.

Differentiable augmentation follows StyleGAN-T [55] and consists of random translation by up to $\pm 12.5\%$ together with brightness, saturation, and contrast jitter, applied with probability 1. Augmentation is used in both generator and discriminator updates. The discriminator is trained from step 8k onward, and the generator receives the adversarial term from step 12k onward; before that, training is purely reconstructive.

Discriminator. We use a frozen DINO ViT-S/8 [4] patch discriminator with features extracted after blocks $\{2, 5, 8, 11\}$. Following GLD [27], five lightweight heads operate on these features, with two heads applied to the final block. Each head is a lightweight convolutional network with spectral normalization and LeakyReLU activations, producing per-token discriminator logits used in eq. (B.1) and (B.3). Only the discriminator heads are trained (7.4M parameters).

Optimization. We use Adam [33] with $\beta = (0.9, 0.95)$ and no gradient clipping. The decoder uses a learning rate of 2×10^{-4} with 5k warm-up steps, followed by cosine decay to 2×10^{-5} at step 100k; the discriminator uses the same learning-rate range with 2k warm-up steps and decay completed at step 60k. We maintain an EMA of the decoder weights with decay 0.9978 and use the EMA model for evaluation. Training runs for 200k steps on four H100 GPUs, corresponding to approximately 22.4M decoded images and 53 hours.

Reconstruction quality. Table B.1 reports reconstruction from ground-truth VGGT tokens, i.e. real images encoded by the frozen backbone and decoded by the RGB head. These results characterize the fidelity of the decoder independently of the generative model and provide a reference for the RGB quality attainable from real VGGT tokens. They should not be interpreted as novel view generation results.

B.2 Training details of our Riemannian flow matching model

We train our generative model for 330k steps using Adam [33] without weight decay. The learning rate is linearly warmed up for the first 10k steps to 10^{-3} and then cosine-decayed to 10^{-6}. We maintain an exponential moving average (EMA) of the model parameters with decay 0.9999 and use the EMA weights at inference. Gradients are clipped to a maximum norm of 1.0. Training uses bfloat16 mixed precision and a global batch size of 256 frame sequences distributed across 16 H100 GPUs. Training takes approximately six days, dominated by the VGGT forward passes, which are computed online for every batch rather than precomputed.

Frame sequences are sampled with equal probability from RealEstate10K and ScanNet++. Within each scene, we randomly sample the target view, context views, and the number of context views $N \in \{1, 2, 3, 4\}$, exposing the model during training to varying conditioning configurations.

Rather than sampling flow times uniformly, $t \sim \mathcal{U}(0, 1)$, we use a shifted uniform distribution that places more probability mass near the noisy end of the trajectory. We first sample $\tau \sim \mathcal{U}(0, 1)$ and transform it as $t = \frac{\tau}{\tau + s(1-\tau)}$. Under our convention, $t = 0$ corresponds to noise and $t = 1$ to clean tokens, therefore $s > 1$ biases training toward noisier states. We use $s = 6$ in all experiments.

B.3 Baselines

The compared methods differ in their conditioning interfaces and assumptions. Our model accepts a variable number of unposed context images together with an arbitrary target camera pose and does not require a prescribed camera trajectory. In contrast, Gen3R operates along a camera trajectory, DepthSplat requires at least two posed context views, and Matrix3D and GLD also use known context poses. We therefore evaluate each baseline using its native interface while matching the context and target views as closely as possible. Below, we detail the checkpoints, inputs, and inference settings used for each method.

Gen3R. Gen3R [23] is a video-diffusion novel-view synthesizer built on the VGGT geometry backbone. Unlike our method, it expects a full camera trajectory and iteratively extends a partially observed scene, rather than being conditioned on a single arbitrary target pose. To ensure a fair comparison, for each context-target pairing we condition on the context view(s) and synthesize the target by generating a short video along a camera trajectory that passes through the target pose. The novel view is then read off at the trajectory index assigned to the target camera. We follow Gen3R's official preprocessing throughout.

Trajectory. We use a 49-frame anchor-based trajectory, with one generation pass per target. The anchors are the context and target cameras: each anchor's pose is kept exactly at its assigned index, and intermediate frames between two consecutive anchors are filled by interpolation. Rotations are interpolated using spherical linear interpolation (SLERP), and translations linearly.

One-sided vs. two-sided. In the two-sided setting the context views are spread evenly around the target, so the target is an interpolation between them (context $\rightarrow$ target $\rightarrow$ context). In the one-sided setting the context views are the ones farthest from the target, all lying on one side, so the target is reached by extrapolating the camera path outward.

Sampling. We use classifier-free guidance with scale 5.0, the text prompt `"a realistic scene"` and the negative prompt `"bad detailed"` during sampling.

DepthSplat. DepthSplat [75] is a deterministic feed-forward method that combines multi-view stereo with 3D Gaussian splatting, predicting depth and Gaussians in a single forward pass. We use the public `gs-large` checkpoint (`depthsplat-gs-large-re10k-256x256-view2`), trained on RealEstate10K at 256×256. The encoder is a large ViT with a cost-volume module whose depth candidates span $[\text{near}, \text{far}]$, sampled uniformly in inverse depth. We set this range per dataset to match each domain's metric extent: 0.5–100 for RealEstate10K (the training values) and 0.1–10 for ScanNet++ (estimated from ScanNet++ training scenes). Target depth is obtained as the z-buffer of the splatted Gaussians at the target pose, so RGB and depth share the same 256×256 resolution.

Matrix3D. Matrix3D [44] is a latent multi-modal diffusion transformer that jointly models RGB, camera (Plücker ray maps), and depth under masked any-to-any conditioning, where each (view, modality) cell is marked condition, generate, or unused. Because target poses can be supplied as conditions, it addresses our task directly. We use the public `matrix3d_512.pt` checkpoint in all

experiments. Following the authors' instructions, target RGB and depthmaps are generated in two passes: target RGB at guidance 1.5 (50 DDPM steps), then depth at guidance 1.0, conditioned on the generated RGB. Pointmaps are obtained by back-projecting the predicted depth given no native pointmap head.

Matrix3D predicts depth in normalized scene units, and fixes that unit from the input poses alone if no ground truth depth is provided. It sets the largest camera-to-first-camera distance in the sequence to 1.0.

If necessary, when this derived unit falls outside its training distribution, we set the scale ourselves. To do so we rescale each sequence poses before inference so the expected scene depth is ≈ 1, by $k \times$ spread, where spread is the maximum pairwise camera distance and k is a single global constant per dataset directly calculated by the evaluation scenes (ScanNet++ 2.296, ETH3D 1.431), and undo the factor on the outputs.

GLD. GLD [27] performs multi-view generation in the feature space of a frozen geometric foundation model. We use the VGGT variant with the public checkpoints for their best configuration, the *cascade* mode (`vggt_level1.pt` and `vggt_cascade.pt`). The latent space consists of VGGT patch tokens from encoder blocks $[4, 11, 17, 23]$ and the cascade first diffuses the level-1 (block-11) tokens and then maps them to level 0, with levels 2/3 obtained by propagation. Sampling uses a linear rectified-flow ODE (Euler, 100 steps), classifier-free guidance with scale 1.5. RGB is decoded by its own RGB head, depth maps by the frozen VGGT depth head, and point maps from the frozen VGGT pointmap head, all from the same synthesized tokens.

C Additional details of the evaluation protocol

C.1 Evaluation scenes

All novel-view results are computed on 50 scenes of the ScanNet++ validation split and on 50 scenes of the RE10K test split, the latter chosen as scenes in which context–target windows satisfying the overlap criteria of Section C.5 exist. For 3D scan generation, ScanNet++ uses the same 50 scenes and ETH3D averages are over all 13 scenes of its training split.

C.2 Resolution unification for cross-method evaluation

Each method we compare against preprocesses its inputs differently and therefore predicts a different sub-region of the scene at a different resolution. Per-pixel metrics such as PSNR, SSIM, RMSE, ARel and δ are only comparable if every method is scored on the same scene region sampled at the same pixel grid. Naively scoring each method on its own preprocessed output would conflate synthesis quality with differences in field of view (FOV) and sampling density.

We resolve this with a single common grid per dataset, constructed so that all methods are evaluated on exactly the scene region that all of them predict. Both the predictions and the ground truth are mapped onto this grid before any metric is computed, including the distributional FID.

Common-grid construction. Let the original frame have resolution (H, W). For each method m, we determine the rectangle of the original frame it actually predicts, expressed in original pixel coordinates, which we call its FOV $\mathcal{F}_m = (H_m^{\text{fov}}, W_m^{\text{fov}})$. The FOVs are obtained by inverting each method's preprocessing:

- **DepthSplat** fits the original into its 256×256 crop with the cover scale $\sigma = \max(256/H,\, 256/W)$, retaining $\mathcal{F}_{\text{D}} = \big(\min(\lfloor 256/\sigma \rceil, H),\, \min(\lfloor 256/\sigma \rceil, W)\big)$.
- **Gen3R** resizes the shorter side to 560 and center-crops a square, so it predicts a centered square of the original: $\mathcal{F}_{\text{G}} = (s, s)$ with $s = \min(H, W)$.
- **Matrix3D** centre-crops the original to the largest centred square and resizes it to 512×512, so $\mathcal{F}_{\text{M}} = (s, s)$ with $s = \min(H, W)$ (its depth is predicted at a native 128×128; see below).
- **GLD** resizes the shorter side to 504 and center-crops a square, giving the same centered square $\mathcal{F}_{\text{GLD}} = (s, s)$, $s = \min(H, W)$.

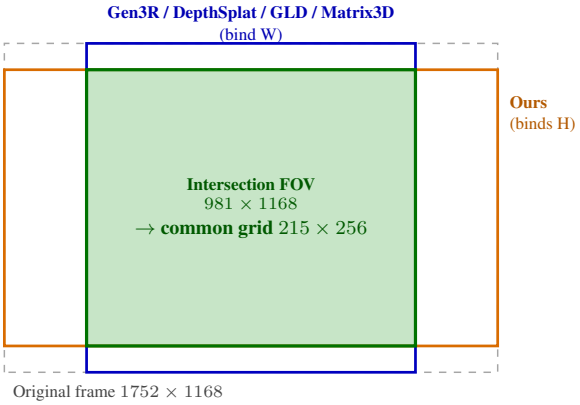


Figure C.1: **Resolution unification on ScanNet++.** Gen3R, DepthSplat, GLD and Matrix3D bind the width; our method binds the height. Their green intersection defines the 215×256 common grid. The four square-crop methods coincide on these datasets; in general they need not.

Table C.1: **Common evaluation grids.** Grids resulting from the unification, fixed per dataset. ETH3D DSLR images vary slightly across scenes, hence the $\approx$.

Dataset	Orig. ($W \times H$)	Grid ($H \times W$)
RE10K	640×360	256×256
ScanNet++	1752×1168	215×256
ETH3D	$\approx 6205 \times 4135$	215×256

- **Ours** resizes width to 518, sets height to the nearest multiple of 14, and center-crops/pads to the canonical 294×518. This keeps the full original width and crops height, so $\mathcal{F}_{\mathrm{O}}$ retains the full width and a centred vertical band of the original.

The only region predicted by *all* methods is the elementwise intersection of these rectangles:

$$\mathcal{F}_{\cap} = \Big(\min_m H_m^{\mathrm{fov}},\ \min_m W_m^{\mathrm{fov}}\Big). \tag{C.1}$$

To avoid upsampling any method, we sample $\mathcal{F}_{\cap}$ at the coarsest native resolution among the methods (the one exception is Matrix3D's depth, whose native 128×128 lies below this density and is nearest-neighbour upsampled to the grid; its per-pixel depth metrics are unchanged when re-scored at its native rate). For method m with output size S_m on a given axis covering an FOV extent F_m of the original, the native sampling density is $\rho_m = S_m / F_m$ (output pixels per original pixel). We use a single isotropic scale

$$r = \min_m \rho_m, \tag{C.2}$$

taken over both axes of all methods, which preserves the aspect ratio of $\mathcal{F}_{\cap}$ and guarantees $r \le \rho_m$ for every method, i.e. no method is upsampled. The common grid is

$$(H_g, W_g) = \big(\lfloor r\, H_{\cap}^{\mathrm{fov}} \rceil,\ \lfloor r\, W_{\cap}^{\mathrm{fov}} \rceil\big). \tag{C.3}$$

Applying this construction yields a fixed grid per dataset, summarised in Table C.1 and an illustration for the ScanNet++ dataset is given in Fig. C.1 .

Mapping predictions and ground truth onto the grid. Each method's prediction is center-cropped to the intersection sub-rectangle within its own output frame and then resampled to (H_g, W_g). The ground truth is center-cropped from the original frame to $\mathcal{F}_{\cap}$ and resampled identically. For RGB, we resample with Lanczos interpolation. For depth, we resample with nearest-neighbor. Invalid pixels are preserved exactly rather than averaged into their neighbors. All metrics are then computed on the common grid.

C.3 3D metric scales alignment

VGGT's predicted depth and point maps are up to an unknown scale and shift. So before calculating the metrics, we align predictions to the ground truth's metric scale. The same procedure is applied to every method.

Depth maps. For depth maps, we do so by a single least-squares scale-and-shift fit over the valid pixels of the first context frame:

$$(a^\star, b^\star) = \arg\min_{a,b} \sum_{i \in \text{valid}} \left(a\,\hat{d}_i + b - d_i\right)^2, \tag{C.4}$$

where d_i and $\hat{d}_i$ are the ground truth and predicted depth for a valid pixel i. The valid set of pixels consists of those for which the ground truth depth is positive and the prediction is finite and positive. Pixels without a sensor return (or without a laser hit) are excluded from the fit. The resulting affine is then applied to the predicted target frame,

$$\tilde{d}_i = a^\star\,\hat{d}_i + b^\star. \tag{C.5}$$

If fewer than $1{,}000$ valid pixels support the fit, the sample falls back to fitting on the target frame itself. This is applied identically to all methods. Depth metrics are calculated over all valid pixels after alignment.

Pseudo-depth evaluation on RE10K. RE10K provides no ground-truth depth, so we score depth predictions against a pseudo ground truth from VGGT. For every context target pairing, target view and number of context views N, we run VGGT once on the N context images together with the ground-truth target image and take its predicted depth of the target view as the reference. All methods are scored against the same VGGT geometry. Predictions and pseudo ground truth are mapped to the common 256×256 grid (see Section C.2) and compared on valid pixels without any scale or shift alignment: since all compared methods produce depth in VGGT's normalised scale (the per-sample median scale ratio to the reference is 0.99–1.03 for every method).

Point maps. For point maps we align via a similarity transform $(R, s, t) \in SO(3) \times \mathbb{R}_{>0} \times \mathbb{R}^3$ using Umeyama's closed-form solution [62],

$$(R^\star, s^\star, t^\star) = \arg\min_{R,s,t} \sum_{i \in V_{\text{ctx}}} \left\| s\,R\,\hat{p}_i + t - p_i \right\|^2, \tag{C.6}$$

where $\hat{p}_i$ and p_i are corresponding predicted and ground-truth world-space points. We write V_F for the joint validity mask over a set of frames F: those pixels at which the predicted point is finite and the ground-truth depth is positive. The fit uses V_{ctx}, the context frames only, and the single resulting transform is applied uniformly to every frame's point cloud.

To prevent a small number of noisy predicted points from biasing the fit we apply Umeyama in two passes. A tentative fit brings the predicted cloud into the ground-truth frame. We then run a statistical outlier filter on the aligned prediction, dropping any point whose mean distance to its 25 nearest neighbors exceeds $\mu + 1.25\,\sigma$ (with μ and σ the mean and standard deviation of these per-point mean distances across the cloud). V_{ctx} is updated to exclude the removed points, and Umeyama is re-fit on the cleaned correspondences.

Each method's point maps are obtained as follows. Our method generates each target in its own pass from the real context tokens and decodes contexts and both targets through VGGT's point head once, giving one cloud and one fit. GLD, Gen3R and Matrix3D predict both targets jointly in one pass, so their clouds are likewise aligned once on the context frames. Matrix3D has no point-map head, so its point maps are obtained by back-projecting its depths of all four views with the same given cameras that conditioned the generation. Ground-truth point maps are back-projected from the original-resolution depth, predicted and ground-truth maps are mapped onto the common grid of Section C.2 before correspondences are formed.

Metrics are computed over the context and generated target frames jointly ($V_{\text{ctx}\cup\text{tgt}}$). The predicted and ground-truth clouds are restricted to this joint mask, concatenated across frames, and then uniformly voxel-downsampled (voxel size 5 mm) followed by farthest-point subsampling to at most $K = 20$k points per side (skipped when a cloud already has fewer points), so that scenes are compared on comparable point budgets rather than raw per-frame pixel counts. No outlier filtering is applied to the merged predicted cloud before scoring. We report accuracy (mean nearest-neighbour distance from prediction to ground truth), completeness (ground truth to prediction), their mean (Chamfer), and Chamfer relative to the ground-truth bounding-box diagonal.

Table C.2: **Overlap-tier thresholds.** Thresholds used to classify context–target view pairings. Rotation bounds are relative to the context camera's horizontal field of view ($\mathrm{FoV}_x \approx 109^\circ$ on ScanNet++, $\approx 91^\circ$ on RE10K); translations are absolute (m).

	ScanNet++			RE10K		
Category	θ_{rot}	t_{lat}	t_{long}	θ_{rot}	t_{lat}	t_{long}
High overlap	$\leq 0.10\,\mathrm{FoV}_x$	≤ 0.20	≤ 0.50	$\leq 0.15\,\mathrm{FoV}_x$	≤ 0.25	≤ 0.70
Low overlap	$\leq 0.60\,\mathrm{FoV}_x$	≤ 1.50	≤ 3.00	$\leq 0.70\,\mathrm{FoV}_x$	≤ 1.50	≤ 3.20

C.4 Reconstructed and generated region split

To separate *reconstruction* of observed content from *hallucination* of occluded or unseen content, we additionally split each unified target view into a visible and a generated region. The split is a per-pixel mask over the same unified pixels (see Section C.2). Crucially, the mask depends only on ground-truth geometry and is thus identical across methods, making the per-region numbers directly comparable.

A target pixel is labeled *visible* if back-projecting its ground-truth depth into world space and reprojecting it into at least one context view lands on a consistent depth, $|z_{\text{ctx}} - D_{\text{ctx}}(\pi)| < \tau\, D_{\text{ctx}}$ with $\tau = 0.05$ (relative). It is labeled *generated* if the target pixel has valid depth but is seen by no context view. The same mask is shared across RGB and depth, since both live on the same grid.

On the visible region, we report reconstruction metrics (PSNR/SSIM/LPIPS for RGB, RMSE/ARel/δ for depth). On the generated region we report the generation metric FID. For depth, we apply a single global scale-and-shift alignment over all valid pixels as explained in Section C.3 and only then split.

Region-wise LPIPS and FID. LPIPS was computed on the full unified 215×256 image with non-visible pixels zeroed in both prediction and ground truth before passing to AlexNet-LPIPS, the identical masking on both sides drives the non-visible contribution toward zero, so the reported value is dominated by the visible pixels. FID was accumulated over the generated region only via a composite-onto-GT variant, $\mathrm{fake} = \mathrm{gt} \cdot (1 - m_{\text{gen}}) + \mathrm{pred} \cdot m_{\text{gen}}$ against $\mathrm{real} = \mathrm{gt}$, so the distribution shift measured by FID isolates the hallucinated pixels.

C.5 High and low overlapping categories description

In order to evaluate a smooth transition from reconstruction to generation, the model's ability to generate consistent and plausible geometry for seen as well as hallucinated areas, we categorize context and target pairings in *high overlapping* or *low overlapping* categories, based on the relative camera poses.

Pairings are characterized by three measures computed from the world-to-camera extrinsics:

- **Rotation** θ_{rot} (degrees): the geodesic distance (angle) in SO(3) between the two rotation matrices.
- **Lateral translation** t_{lat} (meters): the magnitude of the relative camera position projected onto the context camera's image plane (i.e. the x-y components in context-camera coordinates).
- **Longitudinal translation** t_{long} (meters): the absolute value of the context-camera z-component of the relative position (motion along the optical axis).

The rotation threshold is *scaled by the context camera's horizontal field of view* FoV_x, since a fixed angular displacement removes a larger fraction of pixels under a narrow lens. Translations are absolute. A context - target pairing is labeled *high overlapping* if there is one context frame for the target frame at hand, where all three measures lie within the high-overlap bounds $(\theta^{\text{h}}, \ell^{\text{h}}, z^{\text{h}})$, and *low overlapping* if all three measures lie within the looser low-overlap bounds $(\theta^{\text{l}}, \ell^{\text{l}}, z^{\text{l}})$ but the pairing is not already classified as high overlapping. Table C.2 lists the FoV-relative bounds. We compute FoV_x from the camera intrinsics of the ground truth, which gives $\mathrm{FoV}_x \approx 109^\circ$ on ScanNet++ and $\mathrm{FoV}_x \approx 91^\circ$ on RE10K; the absolute rotation bounds are then $\theta^{\text{h}}_{\text{ScanNet++}} \approx 11^\circ$, $\theta^{\text{l}}_{\text{ScanNet++}} \approx 66^\circ$, $\theta^{\text{h}}_{\text{RE10K}} \approx 14^\circ$, $\theta^{\text{l}}_{\text{RE10K}} \approx 64^\circ$.

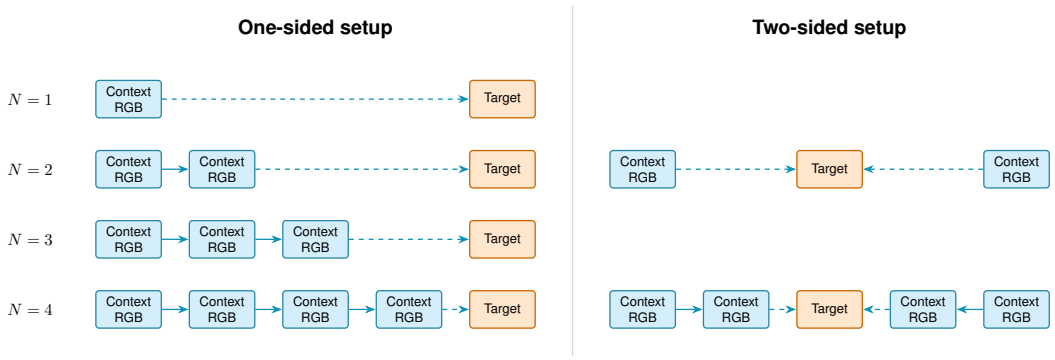


Figure C.2: **Evaluation setup.** We use two setups, *one-sided* and *two-sided* setups. In *one-sided*, we provide the model a set of consecutive frames within a trajectory without pose and task the model to predict a target view defined by the relative pose to the first frame. We provide experiments for a variable context length, $N \in \{1, 2, 3, 4\}$. In *two-sided*, we provide unposed frames from the beginning and end of a trajectory and request the model to predict an intermediate frame of such a trajectory. We evaluate model with two context lengths, $N \in \{2, 4\}$.

RE10K's bounds are widened relative to ScanNet++ to account for the smoother, slower camera motion typical of real-estate captures. To prevent high-overlap 2-sided sequences from collapsing onto temporally adjacent, meaning almost identical, frames on RE10K, every edge of the four-context setup ($S_0 \to S_1 \to T \to S_2 \to S_3$) and every context-target pair is required to span at least 12 frames. ScanNet++ uses a frame gap of 1, since the aequidistant subsampling to 150 frames per scene already provides a wider gap. An illustration of the one-sided vs. two-sided evaluation setup is given in Fig. C.2.

Percentage overlap. We measure the actual per-pixel overlap between each target view and its context views on the unified evaluation grid (Table C.3). For ScanNet++ we compute overlap exactly from ground-truth depth; for RE10K, which lacks GT depth, we use a constant-depth ($Z = 4\,\mathrm{m}$) proxy, so the two datasets are not directly comparable in absolute terms. We report the mean fraction of target pixels covered under four context-aggregation rules, the union of all context frames, and the maximum, median, and minimum coverage from any single context frame, which together bracket the best- and worst-case visibility a model can exploit.

On ScanNet++, coverage drops sharply from the high- to the low-overlap tier under every aggregation rule (e.g. union coverage falls from 0.82 to 0.47 one-sided and from 0.92 to 0.79 two-sided), so the low-overlap tier genuinely demands more generation of unobserved content. The two-sided setup consistently yields higher coverage than the one-sided one, as bracketing the target between context views leaves fewer unseen regions. On RE10K the overlaps are uniformly higher even in the low-overlap tier (≥ 0.89 union), reflecting the smooth, forward-facing motion of real-estate captures.

For the ETH3D scenes used in the 3D scan generation experiment, we report the analogous overlap statistics in Table C.4. Here, the overlap is again computed exactly from ground-truth depth on the unified 215×256 crop, over all $n = 26$ target samples (13 scenes, one two-context/two-target window each), and we report the median with [p5, p95] range rather than the mean, given the small sample count. Even under the union of both context frames the median target-pixel coverage is only 0.61, and drops to 0.29 for the worst single context frame. This makes ETH3D a substantially harder and sparser setting than ScanNet++.

D Additional experiments and ablations

Unless stated otherwise, all ScanNet++ results in this section follow the evaluation protocol of the main paper: pixel-aligned metrics (PSNR, SSIM, LPIPS, RMSE, ARel, δ) are computed on the regions observed in the context views and FID on the unobserved regions (Section C.4).

Table C.3: **Context–target overlap on ScanNet++ and RE10K.** Mean fraction of target pixels covered by the context views on the unified evaluation grid, under four aggregation rules (union of all context views; max / median / min over single context views). ScanNet++: exact overlap from GT depth; RE10K: constant-depth ($Z = 4$ m) proxy.

		ScanNet++		RE10K	
		high	low	high	low
one-sided	n_{targets}	500	500	500	500
	union-of-4	0.82	0.47	0.93	0.89
	max-ctx	0.78	0.44	0.93	0.89
	med-ctx	0.59	0.31	0.93	0.88
	min-ctx	0.42	0.21	0.92	0.87
two-sided	n_{targets}	500	494	364	228
	union-of-4	0.92	0.79	1.00	1.00
	max-ctx	0.85	0.60	1.00	1.00
	med-ctx	0.65	0.38	0.95	0.90
	min-ctx	0.44	0.17	0.82	0.59

Table C.4: **Context–target overlap on ETH3D.** Same quantity as Table C.3, computed exactly from GT depth over valid target pixels; median across the 26 samples with $[\text{p5}, \text{p95}]$ in brackets. Union-of-2 counts pixels visible from either context view; max / med / min-ctx report the best / typical / worst single-context-view coverage.

ETH3D ($n_{\text{targets}} = 26$)	
	Median [p5, p95]
Union-of-2	0.61 [0.47, 0.89]
Max-ctx	0.45 [0.29, 0.83]
Med-ctx	0.38 [0.27, 0.64]
Min-ctx	0.29 [0.01, 0.59]

Table D.1: **Robustness to context view ordering.** One-sided generation from $N=4$ context views on ScanNet++. Random #1–#3 are three random permutations of the context views.

		RGB				Depth			
	Context order	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	RMSE ↓	ARel ↓	δ ↑	FID ↓
High overlap	Furthest first	19.62	0.689	0.111	26.19	0.218	0.103	0.905	24.80
	Closest first	20.55	0.722	0.091	24.69	0.202	0.091	0.916	24.35
	Random #1	20.02	0.703	0.103	25.10	0.214	0.099	0.907	24.53
	Random #2	20.02	0.703	0.103	25.37	0.207	0.095	0.913	24.38
	Random #3	20.00	0.702	0.103	25.40	0.212	0.098	0.908	24.36
Low overlap	Furthest first	17.65	0.628	0.110	51.82	0.287	0.120	0.871	43.95
	Closest first	17.85	0.635	0.107	52.03	0.284	0.119	0.874	43.66
	Random #1	17.74	0.631	0.109	52.08	0.285	0.119	0.874	43.96
	Random #2	17.75	0.632	0.108	51.92	0.285	0.119	0.872	43.82
	Random #3	17.78	0.633	0.107	51.99	0.285	0.119	0.869	43.93

D.1 Context order shuffling

Our formulation does not assume a fixed ordering or camera trajectory for the context views. To verify this, we evaluate the same $N=4$ context set under different permutations in Table D.1. Across three random orderings, performance is highly stable: in the high-overlap setting, the standard deviation is only 0.01 dB in PSNR and 0.003 in depth RMSE, with similarly small variation across the remaining metrics. The low-overlap regime is equally stable. Thus, unlike trajectory-based generation methods, our model does not rely on a particular temporal or spatial ordering of the context views.

A small systematic effect appears when the context view closest to the target is placed first. This ordering yields slightly better performance, most noticeably in the high-overlap setting (e.g. PSNR 20.55 vs. 20.01 for random orderings). We attribute this to the underlying VGGT representation, for which the first input frame plays a distinguished role. Importantly, the small variance across random permutations shows that this positional bias does not translate into a requirement for any fixed context trajectory or ordering.

D.2 Generality of the latent space

Our Riemannian flow-matching formulation is not specific to VGGT. It only requires a frozen backbone and decoder for which the features consumed by the decoding heads are LayerNormed. Given any set of such intermediate features, we apply LayerNorm and project the resulting features onto the unit sphere, and perform flow matching directly on the resulting manifold. If a single intermediate feature is used, the latent space is a single sphere, if multiple intermediate features are used,

Table D.2: **Transfer to a different 3D backbone.** Novel view generation on ScanNet++ using our method with VGGT and DA3 latents. The same generative formulation and training recipe are used for both backbones. Results are reported for high- and low-overlap settings with one-sided and two-sided context.

			High overlap				Low overlap			
	N	Method	RMSE↓	ARel↓	δ↑	FID↓	RMSE↓	ARel↓	δ↑	FID↓
one-sided	1	Ours-DA3	0.255	0.134	0.861	46.10	0.317	0.150	0.822	54.75
		Ours-VGGT	0.232	0.124	0.874	45.98	0.319	0.149	0.846	53.78
	2	Ours-DA3	0.236	0.116	0.895	39.70	0.319	0.150	0.828	51.49
		Ours-VGGT	0.209	0.102	0.909	39.63	0.293	0.124	0.876	49.09
	3	Ours-DA3	0.237	0.117	0.897	33.43	0.309	0.142	0.842	48.15
		Ours-VGGT	0.213	0.101	0.913	32.70	0.290	0.120	0.875	46.48
	4	Ours-DA3	0.237	0.116	0.897	25.28	0.307	0.138	0.852	45.13
		Ours-VGGT	0.218	0.103	0.905	24.80	0.287	0.120	0.871	43.95
two-sided	2	Ours-DA3	0.236	0.105	0.898	24.32	0.307	0.147	0.828	42.47
		Ours-VGGT	0.220	0.094	0.907	23.64	0.276	0.128	0.859	41.05
	4	Ours-DA3	0.218	0.092	0.919	13.13	0.247	0.114	0.881	27.09
		Ours-VGGT	0.212	0.086	0.920	13.03	0.222	0.099	0.888	26.45

it is a product of spheres. The construction therefore does not depend on a particular backbone architecture, feature dimensionality, or number of intermediate features.

The geometric interpretation is also natural in high dimensions: high-dimensional representations already exhibit an approximate spherical structure, as their norms tend to concentrate near an outer shell. LayerNorm makes this structure explicit by normalizing each feature to a fixed scale, after which projection onto the unit sphere gives an explicit spherical manifold on which the flow is defined. The frozen decoder already applies the same normalization to the features it consumes, so generated representations can be passed to the decoder without architectural modification.

Application to Depth Anything 3. To test this generality, we apply the same formulation to DA3 [39], another 3D backbone with a DINOv2 ViT architecture and alternating cross-view attention, QK normalization, and RoPE in its later blocks, trained for 3D reconstruction. For training in the DA3 latent space, we use the four intermediate features consumed by its DualDPT decoder (blocks $11, 15, 19, 23$), $x = \bigoplus_{i\in 11,15,19,23} h_i$, with $x \in \mathbb{R}^{4\cdot 2\cdot 1024}$. Thus, the resulting latent dimensionality and product-manifold structure match those used for VGGT, allowing us to reuse the generative model architecture without modification.

Applying the same training recipe to DA3 produces a working model across all context settings and overlap regimes (Table D.2). The DA3 model exhibits trends closely aligned with VGGT as the amount of context increases, with competitive depth accuracy and similar FID in both high- and low-overlap settings. While VGGT achieves somewhat stronger depth metrics overall, the comparable performance across the two backbones demonstrates that the proposed Riemannian flow-matching formulation transfers to a substantially different, independently pretrained 3D backbone without requiring changes to the generative formulation, latent geometry, or model architecture.

D.3 ODE solver step count

We ablate the number of ODE integration steps at inference in Table D.3 on the one-sided $N{=}2$ ScanNet++ protocol for both high- and low-overlap settings. Very low step counts reveal a clear difference between pixel-aligned metrics (observed regions) and distributional metrics (FID, unobserved regions). With only one to three steps, PSNR, SSIM, and depth errors are often strongest, but FID degrades substantially, particularly in the low-overlap regime. For example, RGB FID drops from 93.29 at one step to 60.24 at five steps and 58.35 at ten steps, while high-overlap FID similarly improves from 57.13 to 42.89 and 42.16, respectively.

Beyond this low-step regime, performance is comparatively stable. From 5 to 200 steps, RGB PSNR varies by only 0.67dB and 0.65dB in the high-and low-overlap settings, respectively, while depth RMSE changes by 0.018 and 0.042. FID largely saturates after approximately 5–10 steps, with little benefit from substantially increasing the integration budget. Pixel-aligned metrics gradually favor fewer steps, whereas qualitative inspection shows that additional integration produces finer structures and more detailed objects in regions not constrained by the context (Fig. D.1). The effect

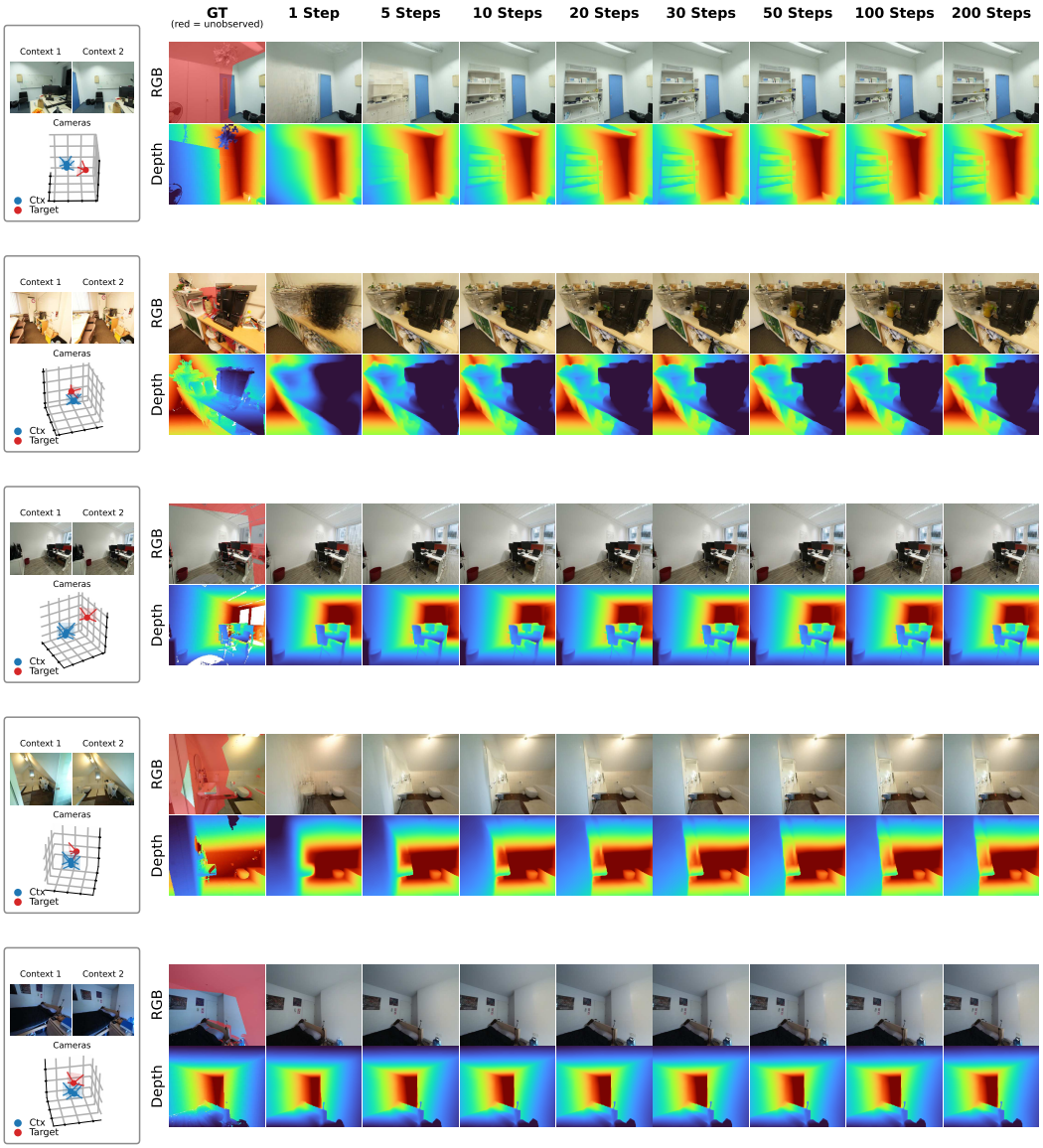


Figure D.1: **Effect of ODE solver steps.** Novel view generation on ScanNet++ using the one-sided protocol with two context views. Each example shows predicted RGB (top) and depth (bottom); red in the target marks regions unobserved by all context views. Across each row, the initial noise, context views, and target pose are fixed, and only the number of Euler integration steps changes. Very low step counts favor pixel-aligned reconstruction but produce visibly smoother generated regions, while additional steps recover finer appearance and object detail; beyond roughly 10-20 steps, changes become increasingly subtle. Quantitative results are reported in Table D.3.

is most apparent in generated regions, while visible-region reconstruction changes comparatively little.

We use 20 steps for all main experiments as a practical compromise between generation detail and computational efficiency. A 20-step ODE solve takes approximately 0.67s per sample on an RTX A6000 ($\approx$ 33ms per ODE step), plus 0.02s and 0.04s for depth and RGB decoding, respectively. The strong performance obtained with fewer steps also allows inference to be further accelerated when generation speed is prioritized.

Table D.3: **Effect of ODE solver steps.** Region-split RGB and depth metrics on the one-sided ScanNet++ protocol with $N=2$ context views, for high and low overlap.

		RGB				Depth			
	Steps	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	RMSE ↓	ARel ↓	δ ↑	FID ↓
High overlap	1	20.27	0.707	0.086	57.13	0.188	0.093	0.917	44.84
	2	20.09	0.708	0.083	47.87	0.192	0.093	0.919	42.08
	3	19.88	0.703	0.085	44.02	0.194	0.094	0.919	40.92
	5	19.62	0.696	0.088	42.89	0.198	0.096	0.916	39.98
	10	19.35	0.687	0.091	42.16	0.204	0.099	0.913	39.55
	20	19.13	0.681	0.096	42.16	0.207	0.102	0.914	39.66
	30	19.08	0.678	0.095	42.81	0.212	0.104	0.907	39.46
	50	19.02	0.675	0.096	43.17	0.214	0.105	0.904	39.52
	100	18.97	0.673	0.097	43.44	0.215	0.106	0.903	39.67
	200	18.95	0.672	0.098	43.66	0.216	0.107	0.902	39.75
Low overlap	1	18.91	0.656	0.087	93.29	0.262	0.112	0.883	59.56
	2	18.62	0.654	0.086	73.69	0.261	0.112	0.887	55.31
	3	18.39	0.647	0.088	64.99	0.264	0.113	0.888	53.08
	5	18.09	0.638	0.090	60.24	0.271	0.116	0.889	50.90
	10	17.81	0.630	0.093	58.35	0.282	0.120	0.884	49.66
	20	17.76	0.630	0.095	58.86	0.283	0.121	0.882	49.80
	30	17.58	0.622	0.096	57.66	0.302	0.128	0.873	49.30
	50	17.51	0.619	0.097	58.18	0.306	0.130	0.873	49.45
	100	17.46	0.617	0.098	58.40	0.311	0.133	0.871	49.60
	200	17.44	0.616	0.098	58.75	0.313	0.134	0.870	49.62

D.4 Additional qualitative results

Figures D.2 and D.3 show additional qualitative results for novel-view RGB and depth generation on ScanNet++ and RE10K in the one-sided and two-sided settings, and Fig. D.4 shows further 3D scan generation results on ScanNet++ and ETH3D. The same observations as in Fig. 2 hold across scenes: the reconstruction-based DepthSplat degrades sharply once the target leaves the support of the context views, GLD tends to blur appearance, Matrix3D and Gen3R render plausible images whose depth is coarse or inconsistent with the view, and our model keeps appearance and geometry consistent with each other and with the context, most visibly in the one-sided setting. For 3D scan generation (Fig. D.4), the picture matches Table 1b: the fused scans of our model and GLD stay close to the reference geometry on both datasets, including the much larger ETH3D scenes, whereas Matrix3D and Gen3R produce misaligned or duplicated structure that no longer lines up with the context point maps.

D.5 Full tables of Euclidean vs. Riemannian flow matching results

Table D.4 reports the full comparison between Euclidean and Riemannian flow matching across all context configurations for RGB (left) and depth (right). Respecting the latent geometry consistently improves performance: the Riemannian formulation outperforms the Euclidean formulation on every reported metric across all context counts and overlap regimes. For RGB, this yields substantial gains in both reconstruction quality (PSNR, SSIM, and LPIPS) and generation quality (FID). The same pattern holds for depth, with improvements in RMSE, ARel, δ, and FID throughout.

The consistency of these gains across modalities, context regimes, and difficulty levels indicates that the benefit is not tied to a particular evaluation setting, but instead arises from performing flow matching on the latent manifold induced by the backbone representation rather than treating the latent as an unconstrained Euclidean space.

Figure D.5 shows the two models side by side on six ScanNet++ scenes. The Euclidean samples decode to washed-out appearance and flattened depth, and their fused point maps bend away from the context geometry, whereas the Riemannian samples stay close to the ground truth in all three modalities.

Off-subspace energy. To quantify how far generated tokens leave the directions occupied by real decoder inputs, let P_b project onto the principal subspace holding 99% of the centred real-token variance of VGGT encoder block $b \in \{4, 11, 17, 23\}$, with mean μ_b. The pooled off-subspace

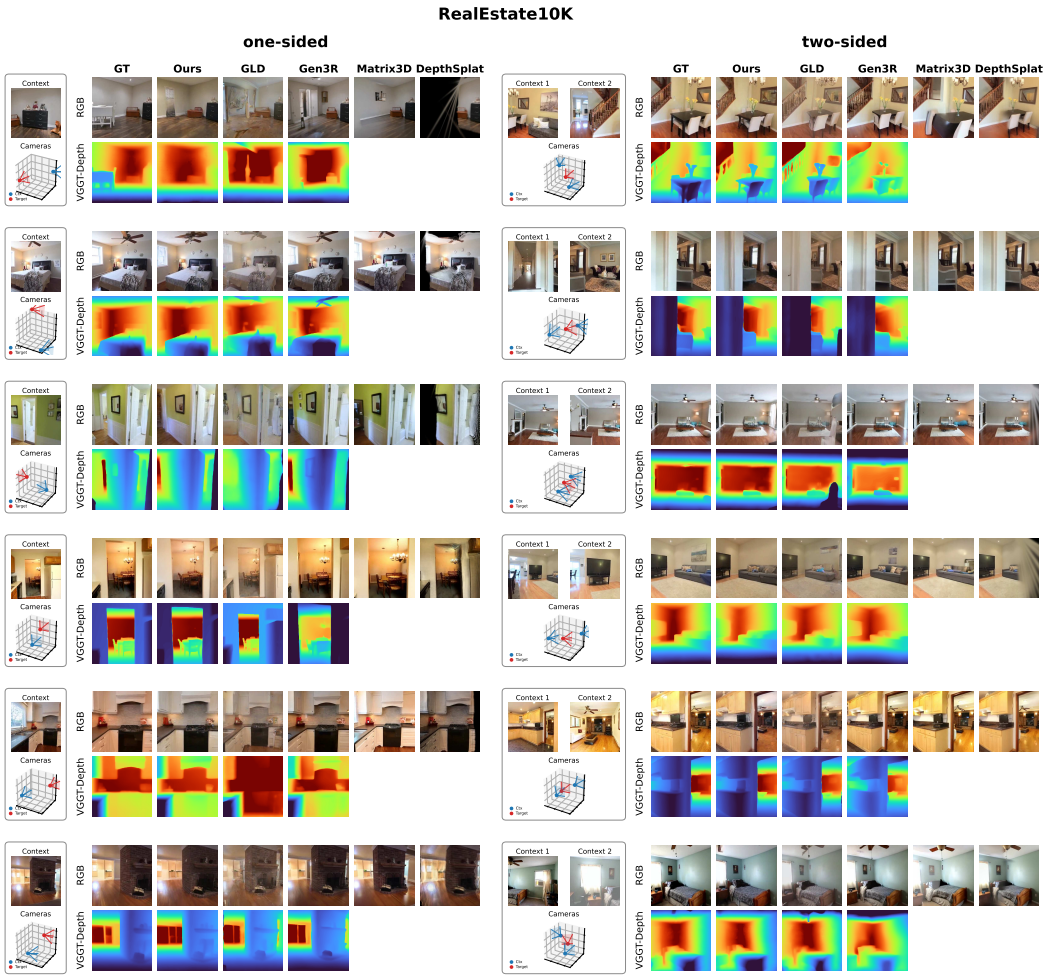


Figure D.2: **Additional qualitative results on RE10K.** One-sided (left) and two-sided (right) novel view generation. The context panel shows one (one-sided) or both (two-sided) context views and the camera layout (blue: context, red: target); columns show target RGB and VGGT pseudo-depth for the GT and all methods, the latter only for the VGGT-based methods.

energy

$$E_{\text{off}} = \frac{\sum_{i,b} \left\| (I - P_b)\left(x_i^{(b)} - \mu_b\right) \right\|_2^2}{\sum_{i,b} \left\| x_i^{(b)} - \mu_b \right\|_2^2}, \tag{D.1}$$

where i runs over the generated tokens, is the fraction of their variation that lies outside the directions real tokens occupy. By construction it is 0.01 for real tokens.

Table D.5 characterizes the generated tokens (two-sided, $N{=}4$, low overlap configuration). The Euclidean endpoint leaves the data shell and ends at $1.33R$. In addition, a third of its energy ($E_{\text{off}}{=}0.33$) lies in directions real tokens do not occupy (0.02 for the Riemannian model), and its cosine to the ground-truth token is 0.66 against 0.85. After normalization, the token therefore sits on the shell, but at an untypical place, and that is what the decoder sees. The Riemannian model never leaves the shell and its direction is close to the real token's. Where this behavior comes from is analyzed in Section D.6.

D.6 Trajectory analysis: geometry and prediction target

The results in Section D.5 show that respecting the latent geometry substantially improves generation. We next disentangle the effects of geometry and prediction parameterization. Under Euclidean velocity prediction, the model must both reproduce the state-dependent term in the optimal velocity

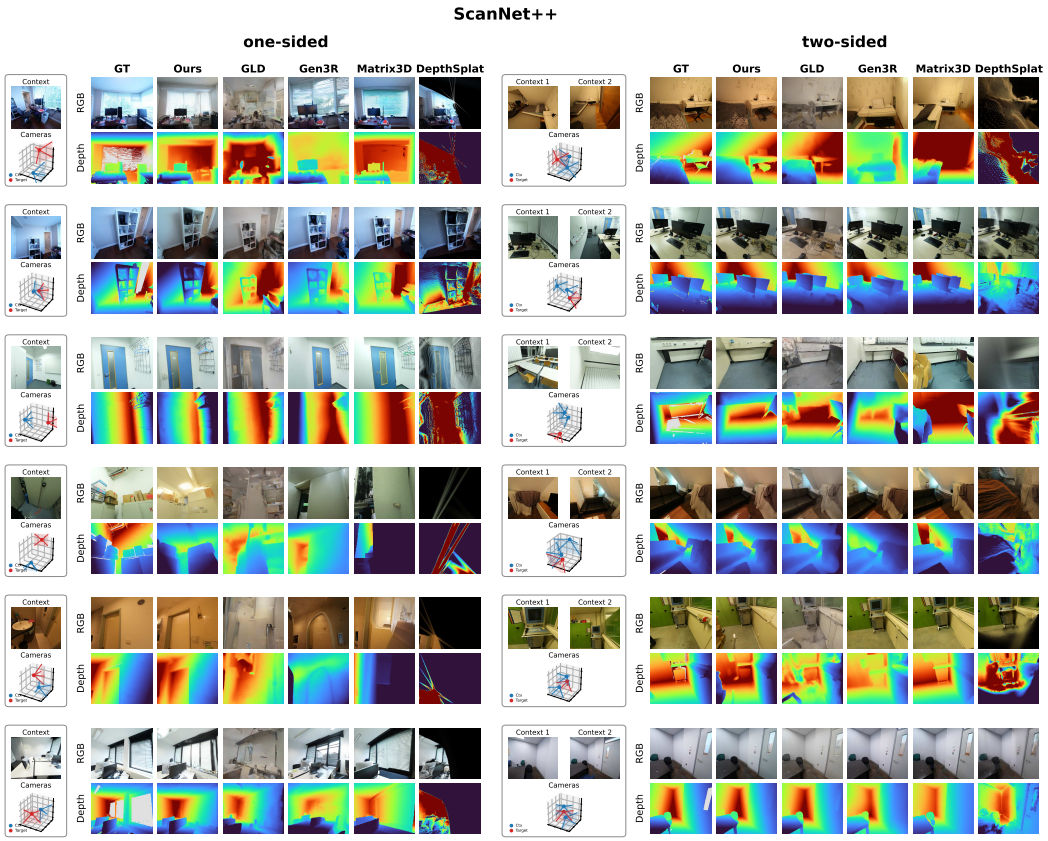


Figure D.3: **Additional qualitative results on ScanNet++.** One-sided (left) and two-sided (right) novel view generation. The context panel shows one (one-sided) or both (two-sided) context views and the camera layout (blue: context, red: target); columns show target RGB and depth for the GT and all methods.

field and model the radial motion induced by straight-line transport. Endpoint prediction reparameterizes the same Euclidean flow so that the state term is constructed analytically by the sampler, removing the associated representational burden. However, the underlying path is unchanged and still contracts through the interior of the latent shell. Riemannian transport removes this radial component by construction. Finally, comparing velocity and endpoint prediction within the same Riemannian geometry isolates the remaining effect of the prediction objective itself.

Euclidean transport contains a radial component. We write a latent token as $x = \bigoplus_{i\in\{4,11,17,23\}} h_i$, where $\oplus$ is concatenation and $h_i \in \mathbb{R}^C$ is the output of VGGT's ith encoder block.

The norm of ground-truth tokens of each encoder block, as well as the norm of the Gaussian source distribution tokens, is strongly concentrated around an empirical radius $\|h_i\|_2 \approx R$. Since the product manifold of spheres is realized by concatenation of each sphere, we analyze one block in the following without loss of generality.

The Euclidean conditional path is the chord

$$x_t = (1-t)x_0 + tx_1, \qquad u_t = \dot{x}_t = x_1 - x_0. \tag{D.2}$$

For $\|x_0\|_2 \approx \|x_1\|_2 \approx R$ and $\langle x_0, x_1\rangle \approx 0$, as expected for high-dimensional source and target samples,

$$\frac{\|x_t\|_2^2}{R^2} \approx (1-t)^2 + t^2, \tag{D.3}$$

$$\frac{\|x_t\|_2}{R} \approx \sqrt{(1-t)^2 + t^2}. \tag{D.4}$$



Figure D.4: **Additional qualitative results for 3D scan generation on ScanNet++ (top) and ETH3D (bottom).** The context panel shows the two context views and the camera layout (blue: context, red: target); the columns show the point maps of the context and generated target views fused into a single scene, for the GT, for VGGT applied to the GT target images as an upper bound, and for all methods.

Hence,

$$\min_t \frac{\|x_t\|_2}{R} = \frac{1}{\sqrt{2}} \qquad \text{at} \qquad t = \frac{1}{2}. \tag{D.5}$$

This shows that although both endpoints lie approximately at the data shell, their Euclidean interpolation contracts to approximately $0.707R$ before expanding again (see the *analytic path* curve in Fig. D.6).

The radial component can be made explicit by decomposing u_t into radial and tangential components,

$$u_t = u_t^{\text{rad}} + u_t^{\text{tan}}, \qquad u_t^{\text{rad}} = \langle u_t, \hat{x}_t \rangle \hat{x}_t, \qquad \hat{x}_t = \frac{x_t}{\|x_t\|_2}. \tag{D.6}$$

We define the radial fraction

$$\rho(t) := \frac{\|u_t^{\text{rad}}\|_2^2}{\|u_t\|_2^2} = \frac{\langle u_t, \hat{x}_t \rangle^2}{\|u_t\|_2^2}. \tag{D.7}$$

Intuitively, $\rho(t)$ measures the fraction of motion at time t devoted to changing the radius, i.e. moving normal to the sphere, while $1 - \rho(t)$ measures the fraction of motion along the sphere.

For the Euclidean conditional velocity $u_t = x_1 - x_0$, the orthogonality approximation gives

$$\rho_{\text{Euc}}(t) \approx \frac{(2t-1)^2}{2\big((1-t)^2 + t^2\big)}, \tag{D.8}$$

and therefore

$$\rho_{\text{Euc}}(0) = \rho_{\text{Euc}}(1) = \frac{1}{2}, \qquad \rho_{\text{Euc}}\left(\frac{1}{2}\right) = 0. \tag{D.9}$$

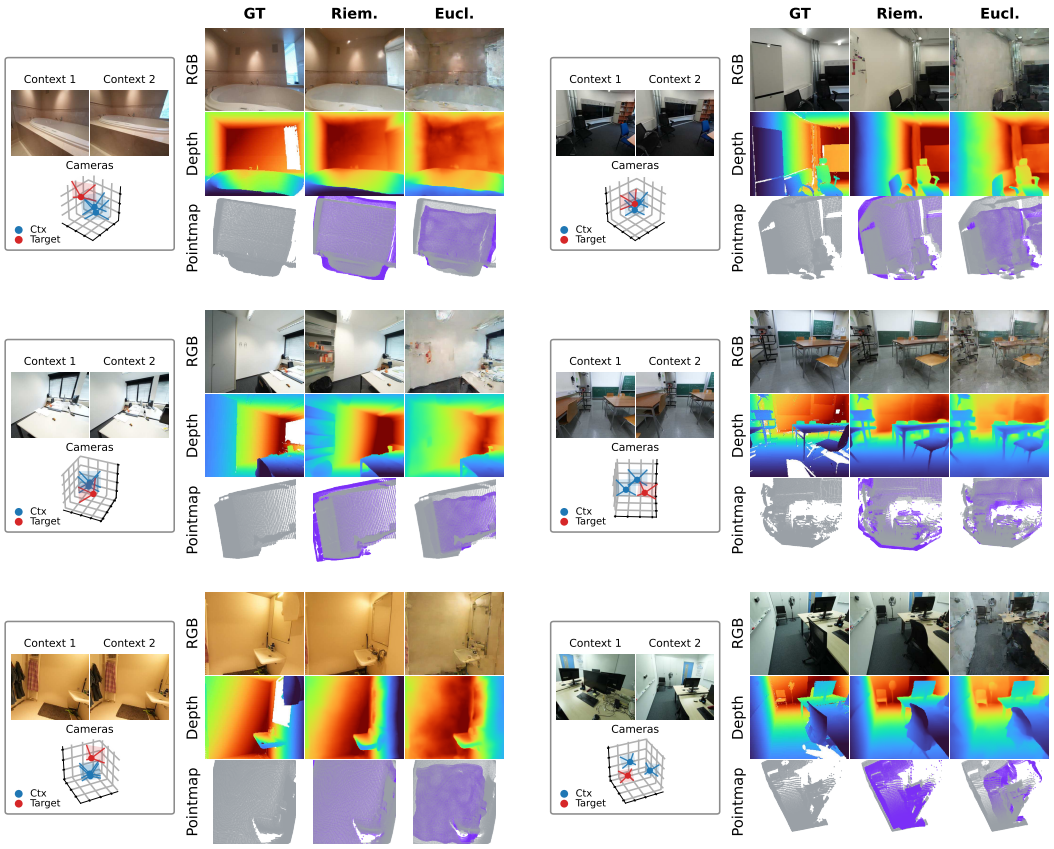


Figure D.5: **Euclidean vs. Riemannian flow matching on ScanNet++** (two-sided, $N{=}4$). For each scene: the two context views and the camera layout (blue: context, red: target), and the target RGB, depth and fused point map (context points in gray, generated target points in color) for the GT, the Riemannian and the Euclidean model.

Thus, at either endpoint, approximately half of the squared Euclidean velocity is radial. The Riemannian path instead follows a geodesic on the sphere. Its velocity satisfies

$$u_t \in \mathcal{T}_{x_t}\mathcal{S}^{C-2}, \qquad \langle u_t, x_t \rangle = 0, \tag{D.10}$$

and consequently

$$\rho_{\mathrm{Riem}}(t) = 0 \qquad \forall t. \tag{D.11}$$

The two geometries therefore define genuinely different transport problems: Euclidean flow must model radial contraction and expansion, whereas Riemannian flow uses all predicted motion in tangent directions.

Euclidean velocity prediction additionally requires state reconstruction. For the linear path in eq. (D.2), the MSE-optimal marginal velocity field is

$$u^\star(x,t) = \mathbb{E}[x_1 - x_0 \mid x_t = x]. \tag{D.12}$$

Using $x_t = (1-t)x_0 + tx_1$ gives

$$x_0 = \frac{x_t - tx_1}{1-t}, \tag{D.13}$$

and hence

$$u^\star(x,t) = \frac{\mathbb{E}[x_1 \mid x_t = x] - x}{1-t}. \tag{D.14}$$

The target therefore, contains two different terms:

$$u^\star(x,t) = \underbrace{\frac{\mathbb{E}[x_1 \mid x_t = x]}{1-t}}_{\text{data-dependent}} - \underbrace{\frac{x}{1-t}}_{\text{state recon.}}. \tag{D.15}$$

Table D.4: **Ablation on flow geometry on ScanNet++, RGB (left) and depth (right).** Euclidean and Riemannian flow matching under a matched architecture and training setup. Best in **bold**.

			RGB								Depth							
			High overlap				Low overlap				High overlap				Low overlap			
	N	Method	PSNR↑	SSIM↑	LPIPS↓	FID↓	PSNR↑	SSIM↑	LPIPS↓	FID↓	RMSE↓	ARel↓	δ↑	FID↓	RMSE↓	ARel↓	δ↑	FID↓
one-sided	1	Eucl.	17.25	0.566	0.142	78.33	16.30	0.522	0.120	104.17	0.271	0.169	0.820	50.08	0.358	0.187	0.761	61.67
		Riem.	**18.99**	**0.677**	**0.085**	**49.73**	**17.61**	**0.620**	**0.086**	**63.86**	**0.223**	**0.118**	**0.891**	**45.99**	**0.299**	**0.138**	**0.852**	**53.55**
	2	Eucl.	17.27	0.565	0.167	63.53	16.43	0.531	0.136	93.09	0.265	0.161	0.825	42.61	0.345	0.176	0.781	56.45
		Riem.	**19.13**	**0.681**	**0.096**	**42.16**	**17.76**	**0.630**	**0.095**	**58.86**	**0.207**	**0.102**	**0.914**	**39.66**	**0.283**	**0.121**	**0.882**	**49.80**
	3	Eucl.	17.42	0.569	0.190	50.88	16.49	0.534	0.150	86.61	0.275	0.164	0.823	35.61	0.346	0.177	0.776	52.15
		Riem.	**19.35**	**0.685**	**0.105**	**35.30**	**17.78**	**0.632**	**0.102**	**55.10**	**0.211**	**0.100**	**0.915**	**32.89**	**0.282**	**0.120**	**0.876**	**46.58**
	4	Eucl.	17.57	0.572	0.217	36.16	16.48	0.535	0.162	78.37	0.288	0.171	0.813	26.07	0.354	0.182	0.764	48.69
		Riem.	**19.59**	**0.691**	**0.113**	**25.67**	**17.73**	**0.632**	**0.110**	**51.86**	**0.217**	**0.102**	**0.907**	**24.73**	**0.280**	**0.118**	**0.876**	**44.04**
two-sided	2	Eucl.	17.40	0.564	0.217	36.53	16.38	0.537	0.180	71.84	0.276	0.156	0.828	25.62	0.328	0.193	0.751	44.84
		Riem.	**19.58**	**0.687**	**0.112**	**26.10**	**17.43**	**0.631**	**0.121**	**50.67**	**0.220**	**0.094**	**0.908**	**23.64**	**0.276**	**0.128**	**0.856**	**41.41**
	4	Eucl.	17.77	0.578	0.247	17.42	16.98	0.553	0.232	39.00	0.272	0.152	0.832	13.69	0.292	0.178	0.775	28.68
		Riem.	**20.18**	**0.703**	**0.116**	**12.89**	**18.39**	**0.656**	**0.135**	**29.69**	**0.211**	**0.085**	**0.921**	**13.23**	**0.228**	**0.100**	**0.907**	**26.14**

Table D.5: **Generated tokens under Euclidean and Riemannian flow matching.** Two-sided generation from $N{=}4$ context views on the low-overlap split of ScanNet++. Endpoint norm relative to the real-token radius R, cosine to the GT token, and off-subspace energy (eq. (D.1); real tokens 0.01), the latter two in decoder-input coordinates. Best in **bold**.

Method	$\|x_1\|/R$	$\cos(x_1, x_{\mathrm{gt}})$ ↑	E_{off} ↓
Eucl.	1.332	0.657	0.334
Riem.	1.000	**0.846**	**0.023**

The second term requires the network to reproduce the state, which is dominated by the isotropic source distribution sample near $t = 0$. The state term in eq. (D.15) is a d_{amb}-dimensional isotropic draw, where $d_{\mathrm{amb}} = 4C$ denotes the ambient token dimension, and a network whose width drops to $D < d_{\mathrm{amb}}$ anywhere cannot reproduce it: its output is a function of a D-dimensional hidden state, so the set of outputs is a D-dimensional surface in $\mathbb{R}^{d_{\mathrm{amb}}}$ (Lipschitz maps do not increase dimension [15, Cor. 2.4]), whatever the depth, attention or nonlinearities around the waist. With a linear output layer that surface is a subspace, the best the network can do is project onto it, and $1 - D/d_{\mathrm{amb}}$ of the state's energy is lost (91% for $d_{\mathrm{amb}} = 4C = 8192$, $D = 768$).

A nonlinear output layer can do better only by folding that surface, e.g., a curve snaking across a disc, so that it passes within δ of every point the noise can take, a d_{amb}-dimensional volume of radius $\approx \sqrt{d_{\mathrm{amb}}}$. Covering that region takes $\sim (\sqrt{d_{\mathrm{amb}}}/\delta)^{d_{\mathrm{amb}}}$ balls of radius δ, while the image of a bounded D-dimensional hidden state under a map of Lipschitz constant L is covered by $\sim (L/\delta)^D$ of them [63, Ch. 4], so matching the two requires an output sensitivity L growing like $(\sqrt{d_{\mathrm{amb}}}/\delta)^{d_{\mathrm{amb}}/D-1}$, an exponent of $8192/768-1 \approx 9.7$ for our dimensions, far beyond the weight norms of trained networks, and consistent with the general result that networks narrower than their input are not universal approximators [45, 19, 29, 52].

Our measurements agree with the linear bound: with $\|x_0\| \approx R$, if the uncancelled fraction $1 - D/d_{\mathrm{amb}}$ of the noise energy were carried to the end orthogonally to the data, the velocity model's endpoint would be $\sqrt{R^2 + (1 - D/d_{\mathrm{amb}})R^2} = \sqrt{2 - D/d_{\mathrm{amb}}}\, R = 1.38R$ ($D = 768$) and $1.39R$ ($D = 512$). We measure $1.33R$ for both widths: the predicted magnitude, with no gain from the wider model.

This provides a network width-dependent explanation for Euclidean velocity prediction. Its conditional transport requires radial motion, and its velocity parameterization asks the network itself to reconstruct the state needed to produce that motion. The Riemannian formulation instead removes the radial component from the prediction problem.

Endpoint prediction removes the state-reconstruction term, but not the Euclidean path. Increasing the model width to the output dimension, can be circumvented by choosing a different prediction target. With endpoint prediction, the network estimates

$$\hat{x}_1 = f_w(x_t, t, c, \pi), \tag{D.16}$$

Table D.6: **Effect of flow geometry and prediction target across model size.** Two-sided generation from $N{=}4$ context views on the low-overlap split of ScanNet++. D denotes the transformer width and #CA the number of cross-attention layers. Best per model size in **bold**.

			RGB				Depth			
D	#CA	Method	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	RMSE ↓	ARel ↓	δ ↑	FID ↓
512	4	Eucl.-v	16.67	0.544	0.244	40.44	0.284	0.167	0.794	28.80
		Eucl.-x	17.17	0.620	0.173	32.55	0.270	0.124	0.857	28.73
		Riem.-v	**17.80**	**0.639**	**0.156**	**31.49**	**0.242**	**0.108**	**0.891**	**27.74**
768	8	Eucl.-v	16.98	0.553	0.232	39.00	0.292	0.178	0.775	28.68
		Eucl.-x	18.11	0.647	0.143	29.84	0.239	0.107	0.897	26.27
		Riem.-v	**18.39**	**0.656**	**0.135**	**29.69**	**0.228**	**0.100**	**0.907**	**26.14**

and the corresponding Euclidean velocity is assembled analytically as

$$\hat{u}_t = \frac{\hat{x}_1 - x_t}{1-t}. \tag{D.17}$$

The full-dimensional $-x_t/(1-t)$ term is therefore computed directly by the sampler rather than reconstructed through the model. Consequently, the model width no longer limits this component.

This explains why Euclidean endpoint prediction performs substantially better than Euclidean velocity prediction. However, changing the prediction target does not change the geometry. If $\hat{x}_1 = x_1$, integration of eq. (D.17) recovers the same straight conditional path

$$x_t = (1-t)x_0 + tx_1, \tag{D.18}$$

and therefore the same contraction in eq. (D.4). Endpoint prediction removes the need to *learn* the radial state-reconstruction term, but the resulting transport still contains radial motion. This is consistent with the preference for clean-sample prediction reported for high-dimensional pixel space [38], for frozen VGGT tokens [61] and for π^3 features [18]: the parameterization removes the state-reconstruction term of the Euclidean velocity target, which recovers most of the gap, while the remaining difference to the Riemannian velocity flow at matched width (Table D.6) is attributable to the radial transport that the parameterization leaves in place.

Table D.6 and Fig. D.6 further show how this effect interacts with model capacity. For Euclidean velocity prediction, increasing both width and the number of cross-attention layers yields only limited gains: the additional depth cannot fully compensate for the narrow state bottleneck, since the 8192-dimensional input is still compressed to $D = 768$. Euclidean endpoint prediction does not suffer from this state-reconstruction constraint and therefore benefits substantially more from the larger model. As a result, it moves much closer to Riemannian velocity prediction at $D = 768$ than at $D = 512$, although a gap remains. This is consistent with the view that additional capacity can compensate for part of the Euclidean transport burden once the state-copy term is removed, whereas the Riemannian formulation eliminates the radial component by construction.

Why endpoint prediction also improves Riemannian flow. We also train a Riemannian flow matching model with endpoint prediction, using the squared geodesic distance between the predicted and the true endpoint as the loss, with the prediction projected onto the manifold and converted to a velocity at sampling time.

The comparison between Riemannian velocity and endpoint prediction isolates a different effect: the geometry is identical in both cases and all velocities remain tangent, so the representability argument above does not apply. The remaining difference arises from the prediction objective.

For the geodesic conditional path, the velocity target can be written as

$$u_t = \frac{\mathrm{Log}_{x_t}(x_1)}{1-t}. \tag{D.19}$$

Any tangent prediction $\hat{u}_t$ can be written in the same form through the endpoint it implies, $\hat{x}_1 = \mathrm{Exp}_{x_t}\big((1-t)\,\hat{u}_t\big)$, so that

$$\hat{u}_t = \frac{\mathrm{Log}_{x_t}(\hat{x}_1)}{1-t}, \tag{D.20}$$

and the velocity error is exactly

$$\|\hat{u}_t - u_t\|_2^2 = \frac{\big\|\mathrm{Log}_{x_t}(\hat{x}_1) - \mathrm{Log}_{x_t}(x_1)\big\|_2^2}{(1-t)^2}. \tag{D.21}$$

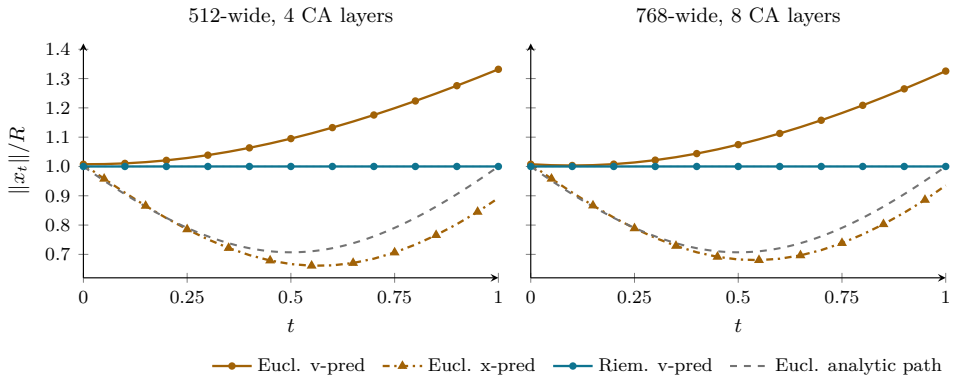


Figure D.6: **Effect of geometry, prediction target, and model capacity on the sampling trajectory.** Normalized latent radius $\|x_t\|/R$ along the learned flow for the $(D, \#\text{CA}) = (512, 4)$ and $(768, 8)$ models. Riemannian flow remains on the manifold ($\|x_t\|/R = 1$) by construction. Euclidean velocity prediction does not reproduce the radial component of the transport and drifts outward instead, with little change at larger capacity. Endpoint prediction removes this state-reconstruction burden and follows the analytic Euclidean chord substantially more closely; the larger model further reduces the discrepancy, but the trajectory still contracts through the interior of the latent shell.

Table D.7: **Prediction target under Riemannian flow matching.** Two-sided generation from N=4 context views on the low-overlap split of ScanNet++. v denotes velocity prediction, x sample prediction. Best in **bold**.

	RGB				Depth			
Method	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	RMSE ↓	ARel ↓	δ ↑	FID ↓
Riem.-v	18.39	0.656	0.135	29.69	0.228	0.100	**0.907**	26.14
Riem.-x	**18.56**	**0.662**	**0.130**	**28.61**	**0.227**	**0.099**	0.905	**25.92**

For a small endpoint perturbation $\delta = \hat{x}_1 - x_1$, first-order expansion of the logarithmic map gives

$$\text{Log}_{x_t}(\hat{x}_1) - \text{Log}_{x_t}(x_1) = J_{\text{Log}_{x_t}}(x_1)\,\delta + O(\|\delta\|_2^2), \tag{D.22}$$

and therefore

$$\|\hat{u}_t - u_t\|_2^2 \approx \frac{\left\| J_{\text{Log}_{x_t}}(x_1)\,\delta \right\|_2^2}{(1-t)^2}. \tag{D.23}$$

Velocity regression is therefore an endpoint regression with an explicit $(1-t)^{-2}$ weight, whereas the geodesic endpoint loss carries weight one. The weight matters because of what the velocity target becomes late on the path. As $t \to 1$ the residual $x_1 - x_t$ vanishes while the model's uncertainty about x_1 does not, so the direction of that residual, which is the velocity target, can no longer be resolved from the state: the relative velocity error $\|v_w - u_t\|^2/\|u_t\|^2$ of the trained model rises from 0.09 at t=0.2 to 0.99 at t=0.99, the value of an uninformative prediction, whereas the geodesic loss of the endpoint model falls monotonically and is flat for $t \geq 0.8$. At the same time a late velocity error barely affects the sample, since it enters only through the remaining path length $\propto (1-t)$. Under the shifted time sampler the velocity objective nevertheless spends 11% of its expected loss on $t > 0.9$ and 37% on $t > 0.5$, against 1% and 11% for the endpoint objective: a substantial part of its gradient goes where the target is unresolvable and the sample insensitive, a regime the endpoint loss de-weights automatically. The effect on the final samples is modest (Table D.7), because the velocity objective still draws most of its signal from the early path.

E Broader impacts

Our method for generative 3D scene reconstruction from sparse views offers potential applications in robotics, AR/VR, and autonomous navigation. As with any generative model, there are important considerations regarding its use.

Generative uncertainty. Our method produces plausible completions of unobserved regions rather than guaranteed accurate reconstructions. In applications requiring high spatial fidelity (such as architectural documentation or safety-critical navigation), users must understand that generated geometry represents one plausible completion among many possibilities. The probabilistic nature of the output should be clearly communicated in any deployment context where geometric accuracy is essential.

Responsible use. As a research contribution, this work is intended to advance understanding of geometry-grounded generative models. Any deployment beyond research settings should include appropriate validation, uncertainty quantification, and human oversight commensurate with the application requirements.

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: Our main claims are summarized in Figure 1, with detailed explanations and evidence provided in Section 3 and Section 4.

   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: We discuss the limitations of our work in Section 5.

   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 new theoretical results requiring formal proofs. We apply existing Riemannian flow matching theory [8] to the VGGT latent space. The mathematical framework is described in Sections 3.1 and 3.2 with detailed definitions provided in Section A.1.

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: We describe our architecture in Section 3.4, the evaluation protocols and datasets (RE10K, ScanNet++ and ETH3D) in Sections C and 4, training details in Sections B.1 and B.2, and baselines in Section B.3. Code and model weights are publicly available at `https://github.com/lisaweijler/latent-rfm`.

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: [Yes]

   Justification: Code and model weights, together with instructions to reproduce our results, are publicly available at `https://github.com/lisaweijler/latent-rfm`. All datasets used are publicly available: RE10K [87], ScanNet++ [78] and ETH3D [58]. The architecture is described in Section 3.4 and implementation details in Section B.

   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: We describe the architecture in Section 3.4, training details in Sections B.1 and B.2, and evaluation protocols with datasets and baselines in Sections B.3, C and 4. Additional implementation details are in Section B.

   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: [No]

   Justification: While we train a single model, as a flow-based generative model, each test prediction involves stochastic sampling, and our metrics are computed over the test sets across three datasets, two protocols, and four context sizes, demonstrating robustness across diverse conditions.

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: Training details including compute resources (GPU type and count, batch size, number of steps) are provided in Section B.2 for the flow model and in Section B.1 for the RGB head; inference time per sample is reported in Section D.3.

   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: We followed 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: We include such discussion in the appendix (Section E).

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: Our method generates 3D scene reconstructions conditioned on input images and does not pose high-risk misuse concerns comparable to unconditioned generative models. The method requires visual observations as input and produces geometric reconstructions for research applications.

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: We use publicly available datasets and credit the authors properly in the paper.

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: [Yes]

    Justification: We release model weights and code at `https://github.com/lisaweijler/latent-rfm`. The method is fully described in Section 3 with architecture details in Section 3.4. Implementation details are provided in Sections B and B.1.

    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: This work does not involve crowdsourcing experiments or research with human subjects. All experiments use existing publicly available datasets of 3D scenes.

    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: Our research does not involve human subjects and therefore does not require IRB approval. All experiments use existing public datasets of indoor and outdoor scenes.

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: This work does not use LLMs as an important or original or non-standard component of the core methods.

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.