Surflo: Consistent 3D Surface Flow Model with Global State

Antoine Guédon[1,*] **Shu Nakamura**[2,*] **Nicolas Dufour**[3,*] **Jiahui Lei**[4]
Ko Nishino[2] **Angjoo Kanazawa**[4]

[*]Equal contribution.

[1]LIX, École polytechnique [2]Kyoto University [3]Kytai [4]UC Berkeley

antoine.guedon@enpc.fr

https://anttwo.github.io/surflo



Figure 1: **Surflo turns a handful of unposed RGB views into a detailed 3D surface.** From a variable number of input views *(left)*, Surflo encodes the scene into a fixed-size latent and decodes it via flow matching into an arbitrary number of oriented surface points, yielding a clean mesh *(middle)*. Per-view feed-forward models [68, 46] instead emit overlapping pointmaps *(right)* that complicate mesh extraction. Surflo exceeds them in quality, while running an order of magnitude faster than state-of-the-art optimization-based methods like Gaussian Wrapping [21], which require hundreds of views. All results are from the same 16 unposed input images, a sparse set of screenshots from an in-the-wild video of the *Gallos* sculpture at Tintagel Castle, UK.

Abstract

Geometry is invariant to viewpoint, which makes any collection of images a redundant encoding of a single 3D state. Existing feed-forward reconstruction models fail to exploit this: per-view methods emit overlapping, unaligned pointmaps that grow linearly with input count, while global-latent methods commit to a fixed, low-resolution output. We introduce Surflo, which compresses a variable number of unposed RGB views into K latent tokens—one global state—and decodes oriented 3D surface points by independently transporting them from noise onto the surface via flow matching. This frees the output from any fixed grid or token budget: the same latent yields from a few thousand to a million points in a single forward pass. To suppress the local inconsistencies inherent to independent per-point decoding, an inference-time guidance term correlates nearby points by injecting a photometric gradient during ODE integration. Surflo matches or surpasses feed-forward baselines on surface metrics, runs an order of magnitude faster than optimization-based methods that require hundreds of views, and is the only feed-forward approach to combine a global latent with arbitrary-resolution decoding.

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

1 Introduction

In the spirit of Klein's Erlangen program [42], 3D geometry is precisely what remains invariant under the viewing transformation. It follows that a collection of images of a scene is a redundant encoding of its geometry: the same underlying 3D state is re-projected into every view, modulo the camera. As more views are gathered, the raw pixel data grows linearly while the geometric content does not—additional views serve only to resolve ambiguities and refine a shared, global state. For downstream applications such as real2sim for robotics [4] and agent navigation [80], it is this global geometry that is the target; the images are merely the instruments used to recover it.

This observation has a direct architectural consequence: if many views encode one geometry, the right intermediate representation is a single global state, not a per-view or per-pixel one. We present **Surflo**, a feed-forward model built around this principle. Given a variable number of unposed RGB views, Surflo compresses them into a single compact and consistent global representation of the 3D scene. Rather than being bound to a fixed spatial grid or a rigid token budget, the underlying geometry can be queried from this global state by sampling an arbitrary number of oriented surface points—from a few thousand to a million per scene. The resulting output is global, consistent across views, and readily converted into a mesh.

Existing feed-forward methods fall short of this principle along two distinct axes. *Per-view* models such as VGGT [68, 73, 46] predict a separate pointmap per input image, preserving the very view-wise redundancy a global representation should compress away: their token count grows linearly with views, and the pointmaps overlap heavily without exact alignment, resisting fusion into a clean mesh (Figure 1). The few methods that summarize the scene in a single latent pay a different price: Chen et al. [11] compresses few-view inputs into a fixed-size code but reads it out as a small, fixed set of $10K$ points, and Wang et al. [70] maintains a persistent state decoded as per-view RGB rather than a global geometric object. D4RT [93] also adopts a scene representation queried for independent points, but operates on dense videos and regresses positions of *pixel-anchored* queries rather than generatively sampling free oriented surface points from sparse views as Surflo does (Appendix C).

Building a representation that is both globally compressed and geometrically queryable requires three coupled ingredients: (1) compression of an unbounded number of multi-view tokens into a fixed-size representation without loss of scene-specific detail, (2) decoding of that global representation into an arbitrary number of surface points, and (3) consistency across the decoded geometry. First, the encoder of Surflo builds on a frozen VGGT [68] backbone for its strong geometric features: patch tokens from several intermediate layers are enriched with a 3D positional encoding of their unprojected position and compressed into K latent tokens. The decoder then generates oriented surface points via flow matching [48], but unlike standard generation on fixed grids it denoises each point independently: conditioned on the global latent, it predicts a velocity that transports an individual query point from a source distribution onto the surface. The same independence, however, has a cost—nothing forces independently sampled points to land on the same surface, so ambiguity in occluded regions can produce local inconsistencies. We resolve this with a guidance mechanism, reminiscent of loss-conditioned diffusion guidance [5], that correlates nearby points at inference time: injecting the gradient of a photometric and depth loss into the predicted velocity aligns the output with the visible images and suppresses noisy outliers.

Across 8 3D reconstruction benchmarks, Surflo matches or surpasses recent feed-forward baselines on surface metrics while being the only feed-forward method that decodes geometry at arbitrary resolution from a single shared latent. The contributions of this work can be summarized as follows:

- A feed-forward encoder, built on a frozen VGGT backbone [68], that compresses a variable number of unposed RGB views into a single global state independent of view count.
- A flow-matching decoder that reads out this latent as an oriented surface by transporting oriented points in $\mathbb{R}^3 \times \mathbb{S}^2$ independently, enabling arbitrary output resolution from the same shared latent.
- A rendering-based guidance scheme during ODE integration that correlates nearby points and aligns the decoded surface with the input images, mitigating the outliers typical of independent per-point decoding.
- As an auxiliary contribution, a meshed version of the DL3DV [47] dataset: each of the ~10.5K scenes is enriched with a watertight surface mesh and ~10^7 oriented points obtained via Gaussian Wrapping [21]. To our knowledge, this is the first real scene-level watertight mesh dataset at this scale, and we publicly release the data as well as the code.

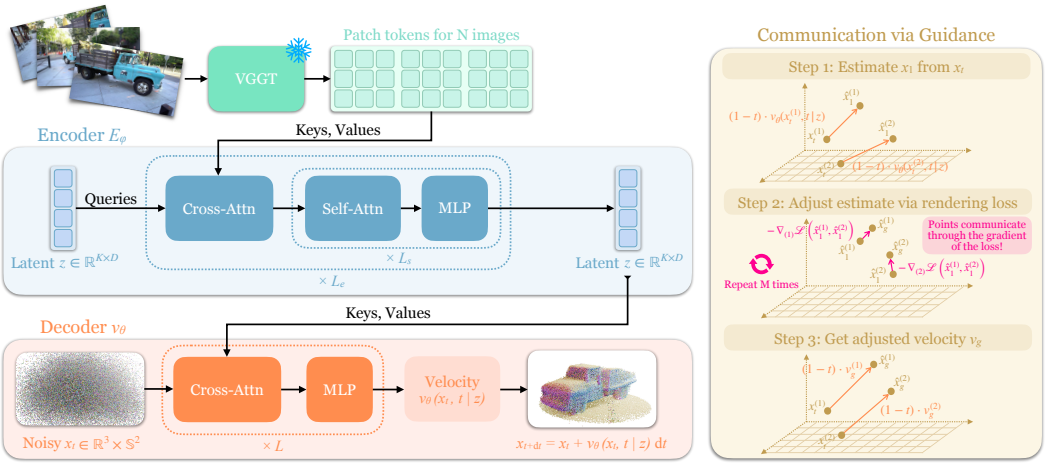

Figure 2: **Three key ingredients define Surflo. Encoder:** a frozen VGGT [68] maps N unposed RGB views to patch tokens, which a Perceiver-style compressor with K learnable queries distills into a fixed-size latent $\mathbf{z} \in \mathbb{R}^{K\times D}$. **Decoder:** each query $\mathbf{x}_t \in \mathbb{R}^3 \times \mathbb{S}^2$ is diffused *independently* via cross-attention with $\mathbf{z}$ to a velocity v_θ; any number of points can be integrated in parallel. **Guidance** (Sec. 2.3): for $t \geq 0.95$, at each step of the ODE integration, per-point velocities are coupled [5] via M gradient steps on a global rendering loss $\mathcal{L}$.

2 Method

Our goal is to recover scene geometry from a variable number of input images at a variable output resolution, with both chosen freely at inference. The design of Surflo follows directly from this goal. To support an arbitrary number of input views N, the conditioning representation must have a size independent of N. Recent multi-view foundation models [68, 74, 46] integrate observations across views and produce geometry-aware features, but their token count grows linearly with N. We therefore compress these features into a *fixed-size* latent $\mathbf{z}$. To support an arbitrary number of output points P, we decode each query point *independently*: a per-point estimator maps each query $\mathbf{x} \in \mathbb{R}^3 \times \mathbb{S}^2$ to a surface sample (3D coordinates and normal). Decoding P points thus reduces to P independent, batched forward passes. To ensure that independently sampled queries fall onto a coherent surface, we introduce an inference-time guidance term that correlates nearby points. Putting these together, Surflo operates as follows (Figure 2):

1. An *encoder* E_ϕ that maps $\{\mathbf{I}_n\}_{n=1}^N$ to a fixed-size latent $\mathbf{z} \in \mathbb{R}^{K\times D}$, independent of N.
2. A *flow-matching decoder* v_θ that, given $\mathbf{z}$, a time $t \in [0, 1]$, and a query $\mathbf{x}_t \in \mathbb{R}^3 \times \mathbb{S}^2$, predicts a velocity transporting $\mathbf{x}_t$ toward the scene surface $\mathcal{S} \subset \mathbb{R}^3$ and its normals.
3. A *guided ODE solver* that, at inference, integrates v_θ together with a rendering-based coupling term to produce a coherent surface.

2.1 Encoder: From Images to a Fixed-size Global State

Frozen VGGT backbone. We rely on a frozen foundation model, namely VGGT [68], as the backbone for its strong geometry-aware features. For each input view $\mathbf{I}_n$, VGGT produces a stack of patch tokens together with a single camera token. Following the procedure of Wang et al. [68], we extract and concatenate patch tokens from four layers $\ell \in \{4, 11, 17, 23\}$, then project them to a working dimension D. This yields a per-view set of tokens $\mathbf{T}_n \in \mathbb{R}^{4N_p\times D}$, where N_p is the number of patches per image. We apply a similar projection to the per-view camera tokens $\mathbf{C}_n \in \mathbb{R}^{4\times D}$.

3D positional encoding. We augment each patch token with a 3D positional encoding. We read off the 3D coordinates of the patch centers in the corresponding VGGT pointmaps, yielding $\mathbf{p}_n \in \mathbb{R}^{N_p\times 3}$; we then map them to Fourier features $\gamma(\mathbf{p}_n) \in \mathbb{R}^{N_p\times D}$ [65], and add the result to the patch tokens. A similar encoding γ is reused by the decoder for query points. This shared encoding is the only mechanism that gives the decoder a notion of *where* in 3D space a query lies relative to the encoded scene. Without it, the latent $\mathbf{z}$ stores observations from VGGT's view-aggregated coordinate frame, while decoder queries arrive as raw coordinates with no a priori relationship to that frame; the

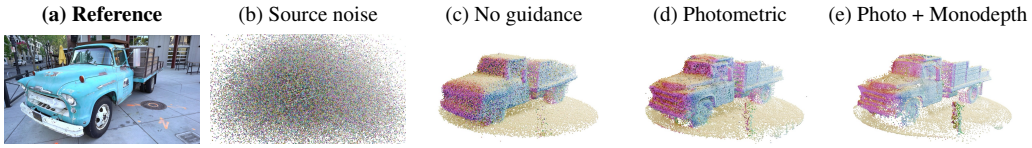

Figure 3: **Surflo denoises 3D points into an oriented surface.** From a variable number of input views (a), Surflo denoises an arbitrary number of noisy 3D points and normals (b) into an oriented surface (c). Integrating the flow-matching velocity without guidance already achieves high accuracy but can exhibit outliers and miss fine details. We therefore introduce two guidance terms. A photometric guidance term $\nabla_{\mathbf{x}}\mathcal{L}_{\text{photo}}$ computed by instantiating Gaussian Splatting [41] on the points, pulls them onto surfaces consistent with the input views and recovers finer details (d). An optional monocular depth [46] guidance further sharpens geometry (e).

alignment would have to be learned implicitly, across whatever coordinate system VGGT produces for each scene. With shared γ, a query point near a VGGT patch shares similar Fourier features with it, so cross-attention localizes relevant scene information by spatial proximity rather than learned association—making coordinate-frame robustness structural rather than something the decoder must absorb during training.

Compression. We compress the union of $N \times 4N_p$ position-encoded patch tokens $\{\mathbf{T}_n + \gamma(\mathbf{p}_n)\}_{n=1}^{N}$ with a Perceiver-style cross-attention module [34, 35]. A fixed set of $K (\ll N \times 4N_p)$ learned latent queries $\mathbf{z}_p^{(0)} \in \mathbb{R}^{K \times D}$ cross-attends to the patch tokens, followed by L_{s} self-attention blocks:

$$\mathbf{z}_p^{(l+1)} \;=\; \text{SelfAttn}^{L_{\text{s}}}\big(\text{CrossAttn}\big(\mathbf{z}_p^{(l)},\, \{\mathbf{T}_n + \gamma(\mathbf{p}_n)\}_n\big)\big) \;\in\; \mathbb{R}^{K \times D}\,. \tag{1}$$

This process is iterated L_{e} times to obtain the final latent $\mathbf{z}_p := \mathbf{z}_p^{(L_{\text{e}})}$. Camera tokens $\{\mathbf{C}_n\}_n$ are processed by a similar but lighter Perceiver into a single latent $\mathbf{z}_c \in \mathbb{R}^{1 \times D}$ encoding additional information about the current world space of the scene, and concatenated into $\mathbf{z} := [\mathbf{z}_p, \mathbf{z}_c]$.

2.2 Flow-Matching Decoder with Independent Query Points

Per-point flow. We treat surface decoding as transporting individual query points in $\mathbb{R}^3 \times \mathbb{S}^2$ from a source distribution p_0 to the surface distribution p_1, conditioned on the latent representation $\mathbf{z}$. We adopt the flow-matching formulation of Lipman et al. [48]: at training time, we sample a target $\mathbf{x}_1 \sim p_1$, a source $\mathbf{x}_0 \sim p_0$, a time $t \sim \text{LogitNormal}(1, 1.6)$, and form the linear interpolant $\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\,\mathbf{x}_1$ with conditional velocity $u(\mathbf{x}_t, t) = \mathbf{x}_1 - \mathbf{x}_0$.

Decoder architecture. The decoder v_θ is an L-layered transformer that ingests a Fourier-encoded query point $\gamma(\mathbf{x}_t)$ and predicts a velocity through cross-attention to the latent $\mathbf{z}_p$ and an MLP. The prediction of the velocity is also conditioned on the time and camera token $[\tau(t), \mathbf{z}_c]$ through Ada-LN [57] blocks before each cross-attention and MLP, where τ is a sinusoidal time embedding, allowing the decoder to adjust its prediction to the current time and world space of the scene.

Training objective. The training objective is the standard flow-matching loss

$$\mathcal{L}_{\text{FM}}(\theta, \phi) \;=\; \mathbb{E}_{t, \mathbf{x}_0, \mathbf{x}_1, \{\mathbf{I}_n\}_n} \left\| v_\theta\big(\mathbf{x}_t,\, t \mid \mathbf{z}\big) \;-\; (\mathbf{x}_1 - \mathbf{x}_0) \right\|_2^2 . \tag{2}$$

The loss decomposes over independent query points, allowing us to train on thousands of points per scene while keeping a single shared latent in memory. Parameters θ and ϕ are learnt jointly, as the decoder and encoder are trained together in an end-to-end fashion.

Source distribution. Each query's normal is initialized uniformly on $\mathbb{S}^2$. Initializing the 3D coordinate from $\mathcal{N}(0, \mathbf{I}_3)$, however, wastes model capacity on empty space and slows convergence. We instead sample from a mixture of Gaussians centered on perturbed VGGT pointmap samples, with noise σ_s large enough to cover surfaces occluded in the input views (Figure 3). This concentrates the flow near the geometry while leaving room to fill in occluded regions.

Inference. At inference, we sample P query points $\{\mathbf{x}_0^{(m)}\}_{m=1}^{P}$ from p_0 and integrate $\frac{d\mathbf{x}_t}{dt} = v_\theta(\mathbf{x}, t, \mathbf{z})$ from t=0 to t=1 with a simple Euler solver. Because all P queries share the same latent $\mathbf{z}$, the per-point cost is dominated by the relatively small decoder, and we can decode $P = 10^3$ to $P = 10^6$ points in a batched pass on a single GPU. As a result, depending on the requested number of samples, the same scene latent supports both a fast coarse preview and a dense surface (Figure 5).

2.3 Communication via Guidance

Per-point independence gives Surflo its flexibility, but at a cost: Equation (2) models each query in isolation, never the joint distribution over surface points. As a result, independently sampled queries are not constrained to land on the same physical surface, producing scattered points between surfaces and drift from the input views. We therefore augment the deterministic flow with a *communication-via-guidance* term, in the spirit of universal guidance [5, 27], injecting the gradient of a differentiable-rendering [41, 90, 25] loss into each ODE step so independently sampled queries can coordinate through a shared global signal.

Specifically, near the end of the ODE integration, for $t \geq 0.95$, we predict at each ODE step the corresponding target points $\hat{\mathbf{x}}_1 = \mathbf{x}_t + (1-t)\, v_\theta$ and run M gradient-descent steps on a global communication loss $\mathcal{L}_{\text{render}}$ that depends on the entire batch. This process couples all points and adjusts the estimated target points, yielding a guided target $\hat{\mathbf{x}}_1^{\text{g}}$ and the corresponding guided velocity $v_{\text{g}} = (\hat{\mathbf{x}}_1^{\text{g}} - \mathbf{x}_t)/(1-t)$ used for the actual Euler step $\mathbf{x}_{t+dt} = \mathbf{x}_t + dt\, v_{\text{g}}$.

In practice, we use a rendering-based loss: we treat the set of predicted targets $\hat{\mathbf{x}}_1$ as a cloud of small oriented Gaussians and *render* them through the cameras recovered by the pretrained VGGT backbone to obtain images $\hat{\mathbf{I}}_n(t)$ at the input view poses. Following Gaussian Wrapping [21], we then compute a rendering loss against the input views:

$$\mathcal{L}_{\text{render}}\big(\{\hat{\mathbf{x}}_1\}\big) \;=\; \frac{1}{N}\sum_{n=1}^{N} \lambda \left\| \hat{\mathbf{I}}_n(t) \;-\; \mathbf{I}_n \right\|_1 + (1-\lambda)\,\text{DSSIM}(\hat{\mathbf{I}}_n(t)\,, \mathbf{I}_n)\,, \tag{3}$$

including additional regularization terms on rendered depth maps [21, 25]. In practice, each of the M gradient steps renders a single input view drawn at random, so the cost of guidance is independent of N (details in Appendix E).

The updated velocities *couple* the query points: nearby queries that disagree about a surface receive a shared corrective signal, which acts as a soft consistency prior. The rendering objective also encourages the points to match the input images and improves the quality of the details considerably. Optionally, we strengthen guidance with a monocular-depth expert [46] (Figure 3). Monocular depth predictions are not multi-view consistent, but their relative ordering provides a strong guidance signal via depth-order regularization [25, 24].

The resulting oriented Gaussians are converted into a triangle mesh via Delaunay triangulation, following [21].

2.4 Training data: Explicit Surfaces for DL3DV

Scene-level 3D surface datasets of real environments do not exist at the scale we need: Tanks & Temples [43] or BlendedMVS [85] contain only a few dozen scenes, and DL3DV [47] provides multi-view images and COLMAP poses but no surface ground truth. As an auxiliary contribution, we close this gap by enriching each of the $\sim$10.5K DL3DV scenes with a watertight mesh and an oriented point cloud. For each scene, we run Gaussian Wrapping [21] on the full set of views to extract a watertight surface, and sample 10^7 points and normals from the mesh, discarding outliers via a visibility check against the input views. We publicly release this dataset and expect it to be useful for scene-level surface learning beyond Surflo.

We use this dataset as our ground-truth surface distributions p_1. At each training step, we first sample 12 scenes; then, we draw $N \in [2, 16]$ input views for each scene as well as a batch of $\sim$ 8K points uniformly on each ground-truth surface; finally, we run our model, align the ground-truth points with our predicted points, and minimize equation (2). More details can be found in Appendix D. Rendering guidance is only applied at inference; the encoder backbone (VGGT) is kept frozen throughout.

3 Experiments

We evaluate Surflo on several few-view 3D reconstruction benchmarks, against representative per-view, latent and optimization-based baselines, and ablate its main design choices. As shown in Figure 4, Surflo remains robust even in challenging capture conditions, including scenes with strong exposure variation across views and transparent objects, such as the semi-transparent figure in the last row.

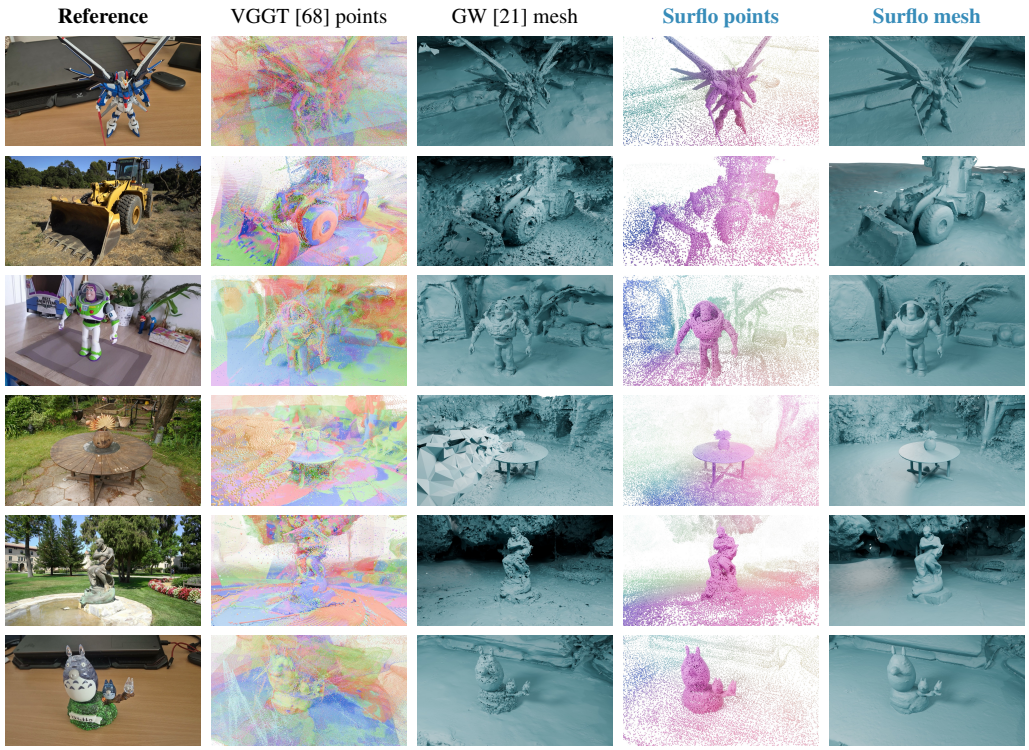

Figure 4: **Qualitative comparisons from** 16 **unposed input views.** From left to right: reference view, VGGT [68] pointmaps (one color per view), Gaussian Wrapping [21] mesh from VGGT initialization, and Surflo's points and mesh. VGGT pointmaps are noisy and duplicated across views, and per-scene GW struggles from few inputs; Surflo yields well-distributed surface points and a clean mesh from the same shared latent.

3.1 Setup

Datasets. We train on the ~10.5K scenes of DL3DV [47] with ground-truth surfaces obtained from Gaussian Wrapping [21], and split out a held-out test set. We evaluate on two groups of out-of-distribution benchmarks. The first group ships with native surface ground truth: ML-Hypersim [59], BlendedMVS [85], DTU [1], and SCRREAM [37]. For the second group, no whole-scene surface ground truth is available, and we generate pseudo-ground-truth surfaces from the dense view sets using the same Gaussian Wrapping pipeline we apply at training time. This group includes our DL3DV test split, Mip-NeRF 360 [6] and DeepBlending [28]. We also include Tanks & Temples [43]: its native ground truth covers only the foreground object of each scene, and our pseudo-GT additionally captures the background geometry. This second evaluation lets us quantify the geometry gap between feed-forward models that observe only a few input views and the best surfaces that can be extracted from a dense RGB capture. For evaluation, each test scene is reconstructed from a fixed set of 16 unposed input views unless stated otherwise.

Metrics. We report Chamfer Distance (CD↓) between predicted and ground-truth point clouds (normalized by the spatial extent of the scene) and F1-score (F1↑) at a fixed scene-relative threshold of 1% of the scene diagonal, following standard practice.

Baselines. We compare Surflo against three families of baselines for scene-level surface reconstruction, all of which receive the same input views (from 2 to 16 views) unless stated otherwise. (i) *Per-view feed-forward*: the raw pointmaps of VGGT [68] and DepthAnything-3 [46], together with their TSDF-fused [16] variants for turning per-view predictions into a single mesh. Because pointmap outputs do not represent a single coherent surface, they trade physical consistency for purely local accuracy; we therefore report them *for reference only* and exclude them from the ranking. Our aim is indeed not higher raw point accuracy, but a globally consistent, view-count-independent and mesh-ready surface; Surflo nonetheless also achieves a lower Chamfer distance than VGGT pointmaps on 5 of the 8 benchmarks. (ii) *Latent feed-forward*: NOVA3R [11], which compresses

Table 1: **Quantitative Evaluation** on four datasets where reference surfaces are generated from Gaussian Wrapping [21] on dense views, covering both foreground and background. DL3DV [47] corresponds to our held-out test split (in-distribution), Tanks & Temples [43], Mip-NeRF 360 [6] and DeepBlending [28] are out-of-distribution benchmarks. All methods are evaluated from the same set of 16 unposed input views per scene. Per-view feed-forward baselines come in two flavours: (i) the raw *pointmap* variants (greyed), which simply concatenate per-view depth predictions into a multi-layered point set without enforcing inter-view consistency, and (ii) their TSDF [16]-fused counterparts, which collapse the per-view depths into a single global mesh.

	DL3DV		Tanks&Temples		Mip-NeRF 360		DeepBlending	
Method	CD↓	F1↑	CD↓	F1↑	CD↓	F1↑	CD↓	F1↑
Per-view feed-forward (no global surface)								
VGGT pointmap [68]	0.0100	74.84	0.0076	81.04	0.0132	69.18	0.0110	76.89
DA3 pointmap [46]	0.0097	77.42	0.0156	71.96	0.0144	67.59	0.0132	67.84
Per-view feed-forward (fused global surface)								
VGGT + TSDF [68]	0.0126	69.23	0.0113	77.46	0.0178	60.64	0.0193	62.30
DA3 + TSDF [46]	0.0120	72.30	0.0177	70.80	0.0182	59.91	0.0210	54.03
Latent feed-forward (fixed-size output)								
NOVA3R [11]	0.0459	30.51	0.0432	32.99	0.0429	25.60	0.0550	27.61
Per-scene optimization (sparse-view)								
2DGS [32]	0.0163	60.10	0.0161	62.95	0.0222	51.08	0.0204	59.54
RaDe-GS [90]	0.0166	59.48	0.0170	61.67	0.0224	50.83	0.0202	60.04
Gaussian Wrapping [21]	0.0168	60.67	0.0157	64.94	0.0201	57.86	0.0164	64.54
Surflo (ours) - No guidance	0.0072	81.92	0.0053	88.57	0.0068	82.00	0.0116	70.96
Surflo (ours) - With guidance	0.0083	78.55	0.0056	86.40	0.0103	76.57	0.0109	75.09

the input views into a latent but reads it out as a fixed-size point cloud. NOVA3R is evaluated with randomly sampled 2 views every time, as it is trained with 2 input views only and feeding more than 2 views systematically results in blurry reconstructions. We restrict our comparisons to methods targeting scene-level reconstruction and therefore exclude object-centric feed-forward models such as ReconViaGen [9]: their outputs cover only foreground objects and would yield uninformative scene-level metrics, as most of the surrounding geometry would be missing. (iii) *Per-scene optimization*: 2DGS [32], RaDe-GS [90], and Gaussian Wrapping [21] run on the same sparse views with camera parameters and initial points estimated by VGGT. These baselines highlight how strongly per-scene optimization underperforms in the few-view regime, and motivate the use of a learned prior. Gaussian Wrapping also generates our training supervision, but from the full DL3DV view sets (200 to 300 views) with COLMAP cameras: the gap between the two settings reflects the reliance of optimization on dense views and accurate poses, not the quality of our supervision.

Implementation details. The encoder uses a frozen VGGT-1B backbone, a Perceiver compressor with $K = 128$ latent tokens of dimension $D = 512$, $L_s = 4$ self-attention blocks per cross attention, and Gaussian Fourier features [65] with $F = 512$ frequencies. The decoder is a 12-layer transformer that cross-attends to the latent for the first 6 layers. We train with AdamW, batch size 12 scenes, 8K query points per scene per step, on $4\times$H100 GPUs for 400K iterations. At inference, we use an Euler ODE solver with 150 steps, and decode 100K points by default, though this number can be adjusted at will; Figure 5 illustrates reconstructions with different numbers of points. Optionally, we apply photometric guidance starting from $t = 0.95$, and perform $M = 32$ loss updates at each ODE step.

3.2 Results

Main Results. Table 2 reports surface metrics on the four out-of-distribution benchmarks with native surface ground truth, and Table 1 reports the same metrics on DL3DV, T&T, Mip-NeRF 360 and DeepBlending, where reference surfaces are generated by Gaussian Wrapping on the dense view sets. All methods receive the same 16 input views. The two tables tell a consistent story: per-view baselines, while accurate per-frame, accumulate misalignments when their pointmaps are merged, and their fused meshes show duplicated surfaces and eroded or floating geometry, particularly at view boundaries. NOVA3R [11] produces coarse and eroded outputs as its decoder is forced to a fixed point budget. Surflo outperforms the best optimization-based references in absolute Chamfer at a fraction of the cost (see Efficiency in Section 3.2), and is the only feed-forward model that also supports decoding the same latent at any chosen resolution. Figure 4 shows qualitative comparisons.

Table 2: **Quantitative Evaluation** on four out-of-distribution benchmarks with native surface ground truth: ML-Hypersim [59], BlendedMVS [85], DTU [1], and SCRREAM [37]. We report Chamfer Distance (CD, ↓) and F1-score (F1, ↑). All methods are evaluated from the same set of 16 unposed input views per scene. Per-view feed-forward baselines come in two flavours: (i) the raw *pointmap* variants (greyed), which simply concatenate per-view depth predictions into a multi-layered point set without enforcing inter-view consistency, and (ii) their TSDF [16]-fused counterparts, which collapse the per-view depths into a single global mesh.

	ML-Hypersim		BlendedMVS		DTU		SCRREAM	
Method	CD↓	F1↑	CD↓	F1↑	CD↓	F1↑	CD↓	F1↑
Per-view feed-forward (no global surface)								
VGGT pointmap [68]	0.0070	91.91	0.0105	79.98	0.0249	47.80	0.0063	83.36
DA3 pointmap [46]	0.0071	87.16	0.0739	66.74	0.0714	26.35	0.0063	83.26
Per-view feed-forward (fused global surface)								
VGGT + TSDF [68]	0.0138	74.08	0.0270	59.64	0.0380	28.93	0.0226	59.89
DA3 + TSDF [46]	0.0151	69.07	0.0875	52.61	0.0801	17.93	0.0220	60.95
Latent feed-forward (fixed-size output)								
NOVA3R [11]	0.0635	27.65	0.0413	32.13	0.0307	31.41	0.0771	27.41
Per-scene optimization (sparse-view)								
2DGS [32]	0.0176	62.03	0.0295	48.19	0.0394	28.78	0.0234	53.57
RaDe-GS [90]	0.0174	62.68	0.0303	48.85	0.0393	28.30	0.0242	53.52
Gaussian Wrapping [21]	0.0145	66.86	0.0259	55.64	0.0460	30.17	0.0123	62.96
Surflo (ours) - No guidance	0.0097	77.98	0.0103	76.50	0.0242	39.23	0.0114	61.20
Surflo (ours) - With guidance	0.0079	87.97	0.0114	77.28	0.0240	42.05	0.0070	81.11

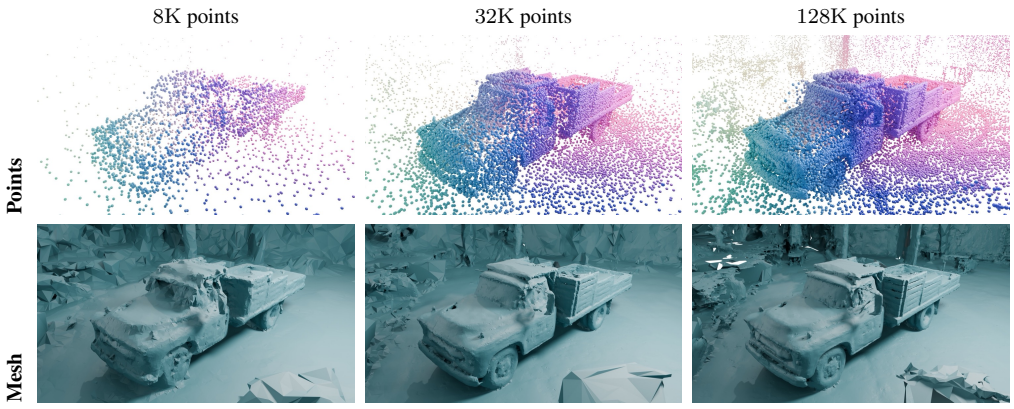

Figure 5: **Variable-resolution decoding from a single shared latent.** Left to right: the flow-matching decoder is queried more densely; *the latent is unchanged across columns*. Bottom: mesh extracted from each point set.

Normal accuracy. As CD and F1 ignore the oriented half of our output, we also evaluate normal angular errors (Appendix H): with 16 views, Surflo has the most accurate normals, reducing the median error by 24% on T&T and 15% on Mip-NeRF 360 relative to the strongest baseline, and its meshes are watertight by construction.

Varying number of input views. Beyond the 16-view protocol used on all datasets above, we further study how Surflo behaves as the number of input views changes. Because the encoder compresses any number of views into a fixed-size latent, the same model can be evaluated across capture densities without retraining or any architectural change. Table 3 reports this experiment on Tanks & Temples as the number of unposed input views per scene varies from 2 to 32, and Table 6 in Appendix G on Mip-NeRF 360. Surflo is consistently best across all view counts on both datasets, including the very sparse 2-view regime. While Surflo benefits from additional views, the size of the latent does not grow with the number of views. Figure 6 shows that even from very few views the model already produces a coherent surface; additional views progressively fill previously missing geometry and sharpen fine detail, while the decoding cost remains unchanged.

Multi-resolution decoding. A key feature of Surflo is that the latent itself does not change with the requested output resolution. Figure 5 shows the same scene latent decoded at three resolutions, from 8K to 128K points, on the same GPU. The coarse output is suitable for fast collision queries or scene previews, while the dense output offers sharper details and background geometry. Quantitatively,

Table 3: **Varying number of input views** on Tanks&Temples. We report Chamfer Distance (CD, ↓) and F1-score (F1, ↑) as the number of unposed input views per scene varies from 2 to 32. Best, second-best and third-best are highlighted; greyed per-view pointmap variants are reported for reference and excluded from the ranking.

	2 views		4 views		8 views		32 views	
Method	CD↓	F1↑	CD↓	F1↑	CD↓	F1↑	CD↓	F1↑
Per-view feed-forward (no global surface)								
VGGT pointmap [68]	0.1461	5.27	0.0224	65.67	0.0082	80.61	0.0064	87.44
DA3 pointmap [46]	0.1595	6.79	0.0247	54.88	0.0177	67.38	0.0128	74.63
Per-view feed-forward (fused global surface)								
VGGT + TSDF [68]	0.1444	6.83	0.0285	53.62	0.0138	70.85	0.0094	84.83
DA3 + TSDF [46]	0.1428	6.97	0.0293	47.90	0.0210	61.64	0.0140	74.59
Latent feed-forward (fixed-size output)								
NOVA3R [11]	0.2620	5.78	0.0502	30.29	0.0557	31.78	0.0423	35.54
Per-scene optimization (sparse-view)								
2DGS [32]	0.1453	6.09	0.0316	43.05	0.0187	55.95	0.0152	68.43
RaDe-GS [90]	0.1454	6.30	0.0314	42.60	0.0191	55.67	0.0156	66.24
Gaussian Wrapping [21]	0.1476	5.24	0.0313	45.28	0.0176	60.04	0.0133	72.10
Surflo (ours) - No guidance	0.1304	10.11	0.0098	80.91	0.0055	87.94	0.0048	90.54
Surflo (ours) - With guidance	0.1303	9.37	0.0095	81.28	0.0051	88.96	0.0044	91.38

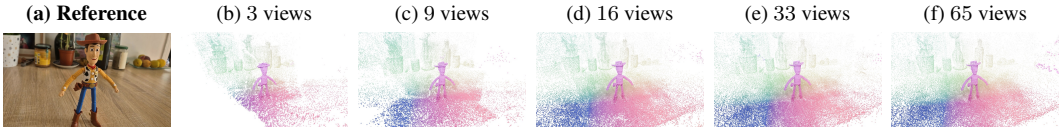

Figure 6: **Surflo reconstructions with a varying number of input views.** More views progressively complete the scene and sharpen fine detail; the decoder cost is unchanged, since each reconstruction is decoded from a latent of the same fixed size. Surflo handles a wide range of capture densities without retraining.

accuracy improves steadily with density, and Surflo outperforms the TSDF-fused baselines at every density from 10^3 to 10^5 points (Appendix I).

Ablations. Table 4 ablates the three main design choices of Surflo. **Latent size.** Increasing K from 32 to 128 tokens consistently improves both metrics. We use $K = 128$ throughout. **Source distribution.** Replacing our noisy-VGGT-pointmap source with a pure Gaussian source degrades reconstruction, especially for large scenes. **3D positional encoding.** Augmenting VGGT tokens with our 3D positional encoding improves both metrics.

Decoder vs. guidance. The unguided rows measure the flow-matching decoder alone, which already outperforms all baselines at every view count (Table 3 and Appendix Table 6). Conversely, the optimization baselines are initialized from VGGT points and cameras, and thus amount to VGGT points refined by rendering, with similar losses and a $\sim 8\times$ larger budget than our guidance; guided Surflo still roughly halves their Chamfer distance on average. The learned prior therefore drives performance, while guidance improves visual quality by removing outliers and sharpening details, which Chamfer distance, dominated by a few noisy points, only partially reflects.

Efficiency. We additionally report wall-clock numbers on a single H100 GPU: encoding a 16-view scene takes a single forward pass through VGGT and the Perceiver compressor; the latent is then re-used for all subsequent queries. Decoding 10^5 points from the cached latent takes about 8 seconds, dominated by the ODE solve, which is two orders of magnitude faster than per-scene optimization baselines such as 2DGS or Gaussian Wrapping. Guidance adds a modest overhead (Appendix I): a complete guided reconstruction takes under 50 seconds with our default setting, and under 16 seconds with a lighter one using fewer ODE and guidance steps, as rendering Gaussians is inexpensive and independent of the number of views N (Section 2.3).

4 Related Work

We provide a focused summary here; a comprehensive discussion is available in Appendix A.

Per-view feed-forward 3D. One class of models predicts geometry—typically depth or pointmaps—separately for each input image. DUSt3R [73] and VGGT [68] produce highly ac-

Table 4: Ablation study on the DL3DV test split (in-distribution) and Tanks & Temples (out-of-distribution), with 16 input views. We vary one design choice at a time and report Chamfer Distance (CD ↓) and F1 (F1 ↑).

		DL3DV		Tanks & Temples	
Component	Variant	CD↓	F1↑	CD↓	F1↑
Latent size	$K = 32$ tokens	0.0089	72.76	0.0068	79.51
	$K = 128$ tokens	0.0073	81.21	0.0055	88.02
3D PE on VGGT tokens	None (raw VGGT tokens)	0.0083	76.05	0.0068	80.78
	Gaussian Fourier 3D PE	0.0073	81.21	0.0055	88.02
Source distribution	Pure Gaussian $\mathcal{N}(0, \sigma^2 I)$	0.0088	73.57	0.0074	77.09
	Mixture of Gaussians	0.0073	81.21	0.0055	88.02

curate local geometry, but their output is *view-bound*: the number of predicted tokens grows linearly with the number of inputs. These per-view outputs are often redundant and misaligned, making them difficult to fuse into a consistent global mesh [4]. Surflo differs by producing a single, fixed-size latent that represents the entire scene regardless of the number of input views.

Global latent models. Other works seek a view-agnostic representation. CUT3R [70] maintains a persistent state but still decodes it into per-view pointmaps. NOVA3R [11] is the closest to our work, using a flow-matching head to decode points from a latent. However, NOVA3R is restricted to a fixed output of 10K points and is trained on only two views, limiting its ability to reconstruct full scenes. In contrast, Surflo supports decoding an arbitrary number of oriented surface points from a global state.

Flow matching and guidance. Our decoder builds on flow matching [48, 51], which has shown success in generating point clouds and meshes for single objects [56, 81, 92]. Unlike these methods, which often operate on fixed grids or structured latents, Surflo predicts a velocity field that transports query points independently, reminiscent of MAR [45]. This independence is regularized by a guidance mechanism reminiscent of loss-conditioned diffusion [5, 62]. While previous guidance methods typically use external classifiers, our guidance signal is derived directly from a differentiable photometric loss to ensure consistency with the input images.

Per-scene optimization. High-fidelity reconstruction at the scene scale is traditionally dominated by optimization-based methods like NeuS [69] or 3D Gaussian Splatting [41]. These methods produce excellent surfaces but require hundreds of *posed* images and significant per-scene compute time. Surflo provides a feed-forward alternative: it produces a reusable scene latent in a single pass, decoupling the cost of the final reconstruction from the number of input images.

5 Conclusion

We presented Surflo, a feed-forward 3D reconstruction model that compresses an arbitrary number of unposed RGB views into a single fixed-size latent, and decodes an arbitrary number of oriented surface points from that latent through flow matching. Per-point independence in the decoder makes the model both flexible and scalable: the same shared latent supports a coarse global preview as well as million-point dense surfaces. To compensate for the lack of explicit surface coupling between independent points, we proposed a photometric guidance scheme that injects the gradient of a differentiable rendering loss into the ODE, correlating nearby points and aligning the output with the input images. We built and released a meshed version of DL3DV [47], including $\sim$10.5K scenes augmented with watertight surfaces. This dataset made it possible to train Surflo at scale, and we hope it will support future scene-level surface learning. On few-view 3D reconstruction benchmarks, Surflo outperforms strong baselines on standard surface metrics while being the only feed-forward model with this combination of fixed-size latent and arbitrary-resolution decoding.

Limitations. Surflo builds on a frozen VGGT backbone, a practical choice imposed by the scarcity of scene-level surface data, and inherits its failure modes: a grossly wrong camera layout cannot be recovered, and training from scratch may perform better given enough data. As points are generated in a space normalized by the scene extent, much larger captures remain globally correct but become locally coarser. Our Gaussian Wrapping [21] supervision also carries biases into Surflo, e.g., sky reconstructed as a curved shell, and occluded regions are noisier since guidance only constrains observed geometry. Finally, very low densities ($<$1K points per scene) may cause tearing when meshing, and Surflo represents geometry but not appearance. Appendix J discusses these points and future directions.

Acknowledgments and Disclosure of Funding

We are grateful to Maks Ovsjanikov, Patrick Pérez, and Vincent Lepetit for insightful discussions. Parts of this work were supported by the ERC Advanced Grant "explorer" (No. 101097259). This work was in part supported by JSPS KAKENHI 21H04893 and 26H02534, and JST JPMJAP2305 to KN. This work was granted access to the HPC resources of IDRIS under the allocation 2025-AD011013387R3 made by GENCI. Part of the computation was carried out using Google Cloud Platform (GCP) credits. Angjoo Kanazawa holds concurrent appointments as an Associate Professor at UC Berkeley and as an Amazon Scholar. This paper describes work performed at UC Berkeley and is not associated with Amazon.

References

[1] Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision (IJCV)*, 120:153–168, 2016.

[2] Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *International Conference on Learning Representations (ICLR)*, 2023.

[3] Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research (JMLR)*, 2025. arXiv:2303.08797.

[4] Arthur Allshire, Hongsuk Choi, Junyi Zhang, David McAllister, Anthony Zhang, Chung Min Kim, Trevor Darrell, Pieter Abbeel, Jitendra Malik, and Angjoo Kanazawa. Visual imitation enables contextual humanoid control. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2025.

[5] Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 843–852, 2023.

[6] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[7] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-NeRF: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[8] Eric R. Chan, Koki Nagano, Matthew A. Chan, Alexander W. Bergman, Jeong Joon Park, Axel Levy, Miika Aittala, Shalini De Mello, Tero Karras, and Gordon Wetzstein. Generative novel view synthesis with 3D-aware diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4217–4229, 2023.

[9] Jiahao Chang, Chongjie Ye, Yushuang Wu, Yuantao Chen, Yidan Zhang, Zhongjin Luo, Chenghong Li, Yihao Zhi, and Xiaoguang Han. ReconViaGen: Towards accurate multi-view 3D object reconstruction via generation. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.23306.

[10] 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. Best Paper Runner-Up.

[11] Weirong Chen, Chuanxia Zheng, Ganlin Zhang, Andrea Vedaldi, and Daniel Cremers. NOVA3R: Non-pixel-aligned visual transformer for amodal 3D reconstruction. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2603.04179.

[12] Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, et al. SAM 3D: 3Dfy anything in images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[13] 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 *European Conference on Computer Vision (ECCV)*, 2024.

[14] Ziwen Chen, Hao Tan, Kai Zhang, Sai Bi, Fujun Luan, Yicong Hong, Li Fuxin, and Zexiang Xu. Long-LRM: Long-sequence large reconstruction model for wide-coverage Gaussian splats. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4349–4359, 2025.

[15] Hyungjin Chung, Jeongsol Kim, Michael T. McCann, Marc L. Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *International Conference on Learning Representations (ICLR)*, 2023.

[16] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *ACM SIGGRAPH Conference Proceedings*, 1996.

[17] Prafulla Dhariwal and Alex Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[18] Nicolas Dufour, Victor Besnier, Vicky Kalogeiton, and David Picard. Don't drop your samples! coherence-aware training benefits conditional diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6264–6273, 2024.

[19] Nicolas Dufour, Lucas Degeorge, Arijit Ghosh, Vicky Kalogeiton, and David Picard. MIRO: Multi-reward conditioned pretraining improves T2I quality and efficiency. In *International Conference on Machine Learning (ICML)*, 2026.

[20] 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[21] Diego Gomez, Antoine Guédon, Nissim Maruani, Bingchen Gong, and Maks Ovsjanikov. From blobs to spokes: High-fidelity surface reconstruction via oriented Gaussians. In *European Conference on Computer Vision (ECCV)*, 2026.

[22] Shrisudhan Govindarajan, Daniel Rebain, Kwang Moo Yi, and Andrea Tagliasacchi. Radiant foam: Real-time differentiable ray tracing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4135–4145, 2025.

[23] Antoine Guédon and Vincent Lepetit. SuGaR: Surface-aligned Gaussian splatting for efficient 3D mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5354–5363, 2024.

[24] Antoine Guédon, Diego Gomez, Nissim Maruani, Bingchen Gong, George Drettakis, and Maks Ovsjanikov. MILo: Mesh-in-the-loop Gaussian splatting for detailed and efficient surface reconstruction. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 44(6), 2025.

[25] Antoine Guédon, Tomoki Ichikawa, Kohei Yamashita, and Ko Nishino. MAtCha Gaussians: Atlas of Charts for High-Quality Geometry and Photorealism From Sparse Views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[26] Moayed Haji-Ali, Willi Menapace, Ivan Skorokhodov, Dogyun Park, Anil Kag, Michael Vasilkovsky, Sergey Tulyakov, Vicente Ordonez, and Aliaksandr Siarohin. One model, many budgets: Elastic latent interfaces for diffusion transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[27] Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, Wei-Hsiang Liao, Yuki Mitsufuji, J. Zico Kolter, Ruslan Salakhutdinov, and Stefano Ermon. Manifold Preserving Guided Diffusion. In *International Conference on Learning Representations (ICLR)*, 2024.

[28] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel J. Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 37(6), 2018.

[29] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS Workshop on Deep Generative Models and Downstream Applications*, 2021.

[30] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[31] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. LRM: Large reconstruction model for single image to 3D. In *International Conference on Learning Representations (ICLR)*, 2024.

[32] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH Conference Proceedings*, 2024.

[33] Allan Jabri, David Fleet, and Ting Chen. Scalable adaptive computation for iterative generation. In *International Conference on Machine Learning (ICML)*, 2023.

[34] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning (ICML)*, 2021.

[35] Andrew Jaegle, Sébastien Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver IO: A general architecture for structured inputs and outputs. In *International Conference on Learning Representations (ICLR)*, 2022.

[36] Wonbong Jang, Philippe Weinzaepfel, Vincent Leroy, Lourdes Agapito, and Jérôme Revaud. Pow3R: Empowering unconstrained 3D reconstruction with camera and scene priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1071–1081, 2025.

[37] Hyun Jun Jung, Weihang Li, Shun-Cheng Wu, William Bittner, Nikolas Brasch, Jifei Song, Eduardo Pérez-Pellitero, Zhensong Zhang, Arthur Moreau, Nassir Navab, and Benjamin Busam. SCRREAM: SCan, register, REnder and map: A framework for annotating accurate and dense 3D indoor scenes with a benchmark. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2024.

[38] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[39] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Symposium on Geometry Processing (SGP)*, 2006.

[40] Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, Jonathon Luiten, Manuel Lopez-Antequera, Samuel Rota Bulò, Christian Richardt, Deva Ramanan, Sebastian Scherer, and Peter Kontschieder. MapAnything: Universal feed-forward metric 3D reconstruction. In *International Conference on 3D Vision (3DV)*, 2026.

[41] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 42(4), 2023.

[42] Felix Klein. Vergleichende Betrachtungen über neuere geometrische Forschungen. *Mathematische Annalen*, 43:63–100, 1893.

[43] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and Temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 36(4), 2017.

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

[45] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[46] Haotong Lin, Sili Chen, Jun Hao 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.

[47] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, Xuanmao Li, Xingpeng Sun, Rohan Ashok, Aniruddha Mukherjee, Hao Kang, Xiangrui Kong, Gang Hua, Tianyi Zhang, Bedrich Benes, and Aniket Bera. DL3DV-10K: A large-scale scene dataset for deep learning-based 3D vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22160–22169, 2024.

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

[49] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, T. Mukund Varma, Zexiang Xu, and Hao Su. One-2-3-45: Any single image to 3D mesh in 45 seconds without per-shape optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[50] 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)*, 2023.

[51] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023.

[52] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, and Wenping Wang. Wonder3D: Single image to 3D using cross-domain diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[53] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *ACM SIGGRAPH Computer Graphics*, volume 21, pages 163–169, 1987.

[54] 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. In *European Conference on Computer Vision (ECCV)*, 2020.

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

[56] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-E: A system for generating 3D point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022.

[57] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4195–4205, 2023.

[58] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. In *International Conference on Learning Representations (ICLR)*, 2023.

[59] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[60] 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.

[61] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021.

[62] Jiaming Song, Qinsheng Zhang, Hongxu Yin, Morteza Mardani, Ming-Yu Liu, Jan Kautz, Yongxin Chen, and Arash Vahdat. Loss-guided diffusion models for plug-and-play controllable generation. In *International Conference on Machine Learning (ICML)*, 2023.

[63] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

[64] Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3D reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10208–10217, 2024.

[65] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[66] 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.

[67] Hengyi Wang and Lourdes Agapito. 3D reconstruction with spatial memory. In *International Conference on 3D Vision (3DV)*, 2025. arXiv:2408.16061, 2024.

[68] 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)*, pages 5294–5306, 2025. Best Paper Award.

[69] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[70] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A. Efros, and Angjoo Kanazawa. Continuous 3D perception model with persistent state. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10510–10522, 2025.

[71] Ruicheng Wang, Sicheng Xu, Cassie Dai, Jianfeng Xiang, Yu Deng, Xin Tong, and Jiaolong Yang. MoGe: Unlocking accurate monocular geometry estimation for open-domain images with optimal training supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5261–5271, 2025.

[72] Ruicheng Wang, Sicheng Xu, Yue Dong, Yu Deng, Jianfeng Xiang, Zelong Lv, Guangzhong Sun, Xin Tong, and Jiaolong Yang. MoGe-2: Accurate monocular geometry with metric scale and sharp details. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

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

[74] 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. arXiv:2507.13347.

[75] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. NeuS2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[76] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. ProlificDreamer: High-fidelity and diverse text-to-3D generation with variational score distillation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[77] Frederik Warburg, Ethan Weber, Matthew Tancik, Aleksander Holynski, and Angjoo Kanazawa. NeRFbusters: Removing ghostly artifacts from casually captured NeRFs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[78] Dongxu Wei, Qi Xu, Zhiqi Li, Hangning Zhou, Cong Qiu, Hailong Qin, Mu Yang, Zhaopeng Cui, and Peidong Liu. Any 3D scene is worth 1K tokens: 3D-grounded representation for scene generation at scale. *arXiv preprint arXiv:2604.11331*, 2026. Project: `https://wswdx.github.io/3DRAE/`.

[79] Rundi Wu, Ben Mildenhall, Philipp Henzler, Keunhong Park, Ruiqi Gao, Daniel Watson, Pratul P. Srinivasan, Dor Verbin, Jonathan T. Barron, Ben Poole, and Aleksander Holynski. ReconFusion: 3D reconstruction with diffusion priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21551–21561, 2024.

[80] Ziyuan Xia, Jingyi Xu, Chong Cui, Yuanhong Yu, Jiazhao Zhang, Qingsong Yan, Tao Ni, Junbo Chen, Xiaowei Zhou, Hujun Bao, Ruizhen Hu, and Sida Peng. Habitat-GS: A high-fidelity navigation simulator with dynamic Gaussian splatting. In *European Conference on Computer Vision (ECCV)*, 2026.

[81] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3D latents for scalable and versatile 3D generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. Spotlight; arXiv:2412.01506, 2024.

[82] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. InstantMesh: Efficient 3D mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024.

[83] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. PointFlow: 3D point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[84] Jianing Yang, Alexander Sax, Kevin J. Liang, Mikael Henaff, Hao Tang, Ang Cao, Joyce Chai, Franziska Meier, and Matt Feiszli. Fast3R: Towards 3D reconstruction of 1000+ images in one forward pass. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21924–21935, 2025.

[85] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. BlendedMVS: A large-scale dataset for generalized multi-view stereo networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[86] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[87] Botao Ye, Sifei Liu, Haofei Xu, Xueting Li, Marc Pollefeys, Ming-Hsuan Yang, and Songyou Peng. No pose, no problem: Surprisingly simple 3D Gaussian splats from sparse unposed images. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.24207.

[88] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 2024.

[89] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. LION: Latent point diffusion models for 3D shape generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[90] Baowen Zhang, Chuan Fang, Rakesh Shrestha, Yixun Liang, Xiaoxiao Long, and Ping Tan. RaDe-GS: Rasterizing depth in Gaussian splatting. *ACM Transactions on Graphics*, 2026. arXiv:2406.01467.

[91] Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3DShape2VecSet: A 3D shape representation for neural fields and generative diffusion models. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 42(4), 2023.

[92] Biao Zhang, Jing Ren, and Peter Wonka. Geometry distributions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[93] Chuhan Zhang, Guillaume Le Moing, Skanda Koppula, Ignacio Rocco, Liliane Momeni, Junyu Xie, Shuyang Sun, Rahul Sukthankar, Joëlle K. Barral, Raia Hadsell, Zoubin Ghahramani, Andrew Zisserman, Junlin Zhang, and Mehdi S. M. Sajjadi. Efficiently Reconstructing Dynamic Scenes One D4RT at a Time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[94] 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 *European Conference on Computer Vision (ECCV)*, 2024.

[95] Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. CLAY: A controllable large-scale generative model for creating high-quality 3D assets. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 43(4), 2024.

[96] Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, Bin Fu, Tao Chen, Gang Yu, and Shenghua Gao. Michelangelo: Conditional 3D shape generation based on shape-image-text aligned latent representation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[97] Zibo Zhao, Zeqiang Lai, Qingxiang Lin, Yunfei Zhao, Haolin Liu, Shuhui Yang, Yifei Feng, Mingxin Yang, Sheng Zhang, Xianghui Yang, et al. Hunyuan3D 2.0: Scaling diffusion models for high resolution textured 3D assets generation. *arXiv preprint arXiv:2501.12202*, 2025.

A Related work

Per-view feed-forward 3D geometry. The first family of feed-forward 3D models predicts a pointmap or depth map per input image. DUSt3R [73] pioneered this paradigm by regressing paired pointmaps in a common frame from two views, and Leroy et al. [44] extended it to more accurate matching. More recent work scales the idea to many views with a single transformer: VGGT [68] processes hundreds of images in one forward pass, Lin et al. [46] pushes the formulation further with a depth–ray prediction target, Yang et al. [84] parallelises DUSt3R-style decoding across $1000+$ views, and Wang et al. [74] removes the dependency on a fixed reference frame. Streaming variants such as Spann3R [67] and ingredients such as metric monocular pointmaps [71, 72], prior-conditioned regression [36], and unified factored representations [40] further enrich this space. These models are remarkably accurate per view, but the output remains *view-bound*: the number of predicted tokens grows linearly with the number of inputs, the per-view pointmaps overlap heavily, and they are rarely consistent enough to be fused into a clean mesh [4].

Latent and global feed-forward 3D. A second line of work attempts to keep a single, view-agnostic representation. CUT3R [70] maintains a persistent state through cross-attention but still reads it out with per-view metric pointmaps. Concurrent global tokenization approaches such as 3DRAE [78] target this goal but, in practice, still query the latent through per-view decodings. Recent work NOVA3R [11] is the closest to our setting: it compresses unposed multi-view inputs into a fixed-size latent and decodes points with a flow-matching head, but emits a fixed number of 3D points with no ability to sample more or less. Furthermore, it is trained with at most 2 input views, limiting its ability to fully reconstruct the entire scene. Closely related in spirit, D4RT [93] also encodes the input into a single global scene representation that is decoded by independent point queries. It is, however, designed for *dense video*: its encoder operates on a temporally-patchified frame sequence to target 4D reconstruction, and it decodes by regressing the 3D position of *pixel-anchored* spatio-temporal queries rather than generatively transporting free oriented points onto a surface. Surflo instead targets sparse images and decodes arbitrary 3D points through flow matching. Generalisable Gaussian-splatting models follow a related philosophy: pixelSplat, MVSplat, NoPoSplat [10, 13, 87] and the LRM family [31, 94, 14, 64] predict per-pixel or per-view 3D Gaussians from a few sparse images, sharing the redundancy issue of pointmap methods. On the architectural side, our encoder is a Perceiver [34, 35], an architecture that has been repurposed as a generative backbone in recurrent interface networks (RIN) [33, 18, 19, 26] for image generation; we adapt it here to feed-forward 3D by compressing multi-view tokens into a fixed-size latent that is then exposed through a decoder that can sample arbitrary number of oriented points directly in the 3D space. Surflo differs from this entire family in that it produces a single, fixed-size, view-independent latent *and* exposes the geometry through an arbitrary-resolution decoder.

Per-scene optimization for surfaces. At the scene scale, fine-grained 3D reconstruction is still dominated by per-scene optimization. One branch targets photo-realistic novel-view synthesis without explicit surfaces: NeRF [54] and its accelerated or anti-aliased successors [55, 6, 7], 3D Gaussian Splatting [41], and recent variants such as Radiant Foam [22], all of which fit a radiance field that produces faithful renderings but no clean geometry. The other branch is explicitly geometry-focused: NeuS [69] and VolSDF [86] render an implicit signed-distance field, while SuGaR [23], 2D Gaussian Splatting [32], Gaussian Opacity Fields [88], RaDe-GS [90], MILo [24] and Gaussian Wrapping [21] build on radiance-field representations to extract a manifold-quality mesh, often combined with classical surface-extraction machinery [53, 16, 39]. Both branches reach high fidelity but share the same practical limits: they rely on dense captures of hundreds of images, require tens of minutes of per-scene optimization, and yield no reusable latent. Sparse-view variants such as Guédon et al. [25] relax the input requirement, yet their time and memory cost scales linearly with the number of views, limiting their scalability. Surflo instead leverages Gaussian Wrapping [21] only at training time, as a source of dense surface supervision, and is itself feed-forward and scene-level at inference: it decodes surfaces at arbitrary resolution from a single shared latent from unposed input images, decoupling the cost of reconstruction from the number of input views.

Object-level learned 3D and shape latents. Feed-forward learned mesh predictors offer an alternative to optimization, but remain largely *object*-level: LRM and its Gaussian-splatting variant GS-LRM [31, 94] regress NeRFs or 3DGS from one or a few posed views, InstantMesh [82] adds an iso-surface extraction head, and TRELLIS [81] learns a structured latent for object generation with a

rectified-flow transformer. Sam3D [12] predicts 3D geometry and texture from a single image. Closer to our own design, 3DShape2VecSet [91], Michelangelo [96] and CLAY [95] represent shapes as a small set of latent vectors decoded by cross-attention, and a wide ecosystem of 3D-native generators built on this idea reconstructs single objects from images or text [50, 49, 52, 97]. None of these methods targets full scenes from a few unposed views with a shared, queryable latent, which is precisely Surflo's regime.

Diffusion / flow priors and guidance. A growing line of work uses 2D diffusion as a prior for novel-view synthesis or 3D reconstruction: GeNVS [8], CAT3D [20] and ReconFusion [79] hallucinate plausible views that are then turned into geometry through an external pipeline, NeRFbusters [77] works directly in 3D as a voxel-based regulariser, and ReconViaGen [9] distils generative priors into object reconstruction. Score-distillation methods [58, 76] push the same idea further at the price of long per-scene optimization. Their target is pixels or single objects rather than a global, queryable scene representation. The closest mathematical tool to ours is flow matching and its relatives [48, 51, 2, 3], which build on diffusion foundations [30, 61, 63, 38, 60, 57] and were already shown to be effective for point clouds and meshes in PointFlow, LION, Point-E, Geometry Distributions, and TRELLIS [83, 89, 56, 92, 81]. Instead of denoising images, our decoder learns a velocity field that transports query points in $\mathbb{R}^3$ *independently* from a noisy source to the surface, conditioned on the scene latent. Conceptually, this design echoes MAR [45] in image generation: MAR pairs a global autoregressive prior over image tokens with a per-token diffusion head, sampling each patch conditioned on a global state; Surflo analogously pairs a global feed-forward prior, built on VGGT [68], with a per-point flow-matching head, replacing patch-wise transport in pixel space with point-wise transport in $\mathbb{R}^3$. The guidance mechanism we use during ODE integration is reminiscent of classifier and classifier-free guidance [17, 29] and of training-free inverse-problem solvers [15, 5, 62], but here the gradient signal comes from a differentiable photometric rendering of the partially solved point cloud rather than from an external classifier or likelihood.

B Architecture details

We give in this section the exact hyperparameters of the encoder and decoder used in all main experiments.

Encoder. The frozen VGGT-1B [68] backbone (24 transformer blocks, patch size 14, input resolution 518×280) produces $37 \times 20 = 740$ patch tokens per view per layer and 1 camera token per view, all of dimension $d_v = 2048$. We extract tokens from layers $\ell \in \{4, 11, 17, 23\}$. We then project to a working dimension $D = 512$ with a single linear layer; a separate linear layer projects the camera tokens. The 3D Gaussian Fourier features [65] use $F = 512$ frequencies sampled from $\mathcal{N}(0, \sigma^2 \mathbf{I}_3)$ distributed over 16 log-spaced bands with $\sigma \in \{100, 64.94, \ldots, 0.237, 0.154\}$ (geometric ratio ≈ 0.65), 32 frequencies per band, expressed in scene-normalized coordinates (coordinates are normalized by the median distance to the median VGGT point). The same encoder configuration is shared with the query-point spatial encoder of the decoder, and the resulting 3D PE is broadcast across the four intermediate VGGT layers. The Perceiver encoder uses $K = 128$ learnable latent tokens and $L_\mathrm{s} = 4$ self-attention blocks with 16 heads after each of the $L_\mathrm{e} = 4$ cross-attention with patch tokens, with $\mathrm{mlp_ratio} = 4.0$, qk-norm, LayerScale initialized at 0.01. The Perceiver encoder for camera tokens applies cross-attention of 1 latent token with camera tokens for 4 times, with 16 heads and no self-attention. The patch and camera latents are kept on separate paths and consumed by the decoder through different mechanisms (cross-attention for the patch latents, AdaLN conditioning for the camera latent).

Decoder. The flow-matching decoder is a $L = 12$-layer transformer with hidden dimension $D = 512$ and 16 heads, $\mathrm{mlp_ratio} = 4.0$, and qk-norm. There is no self-attention over query points: each block processes points independently, and all spatial information sharing is delegated to the encoder latents. Each of the first 6 blocks alternates cross-attention to the latent $\mathbf{z}$ and an MLP, each of which is prepended with Ada-LN with conditioning tokens (time and camera token). The latter 6 blocks apply only Ada-LN and MLP. Each AdaLN conditioning network produces shift, scale and gate parameters in the DiT [57] style ($\mathrm{LN}(x) \cdot (1 + \mathrm{scale}) + \mathrm{shift}$, residual gated by gate), with the projection layer zero-initialized so that every block starts as the identity. All AdaLN modules are fed with $\tau(t) + \mathrm{MLP}_{\text{cam}}(\mathbf{z}_{\text{cam}})$, where $\mathbf{z}_{\text{cam}}$ is the encoded camera latent and $\mathrm{MLP}_{\text{cam}}$ is a 2-layer

2048-dimensional MLP. The time embedding $\tau(t)$ uses the standard sinusoidal schedule with 512 frequencies log-spaced in $[0.1, 1000]$ (output dimension 1024, projected to $D = 512$). Query-point coordinates are normalized per-scene with the same statistics as the patch 3D PE and encoded with the 16-band Gaussian RFF encoder described above before entering the first block. A small two-layer MLP head on top of the last block predicts the per-point normal in $\mathbb{S}^2$ alongside the velocity, yielding a 6-dimensional output; the final LayerNorm and head are executed in FP32 for precision. Following VGGT [68], we use GeLU as the activation function, except for Ada-LN and MLP for the camera token or Fourier-encoded token projections, where we use SiLU.

Query Formulation. For numerical stability, normals are represented as *a residual* within a query point $\mathbf{x}$. That is, given the sampled 3D point $\mathbf{m} \in \mathbb{R}^3$ and a normal $\mathbf{n} \in \mathbb{S}^2$, the query point is initialized as $\mathbf{x}_0 = (\mathbf{m}, \mathbf{m} + \epsilon\mathbf{n}) \in \mathbb{R}^6$, where $\epsilon = 10^{-3}$ multiplied by the scene scale. Velocities are predicted in the same space. For decoding the normal from this representation at inference time, we simply take the difference between the second and the first half and normalize the resulting vector.

C Role of the flow-matching decoder

Why a generative decoder? Arbitrary output cardinality does not by itself require a generative model: a deterministic, coordinate-based decoder can also be queried at any number of locations, as in D4RT [93]. D4RT queries, however, are pixel coordinates in a source frame, and dense reconstruction is obtained by querying observed pixels. Two consequences follow, neither of which is a criticism of D4RT, which targets dense videos rather than sparse-view surface reconstruction. First, no query addresses a region that appears in no input frame, so occluded geometry cannot be predicted; with 2 to 32 unposed images, much of a scene can be occluded in every view. Second, sampling in pixel space inherits the biases of depth back-projection: local density decreases with the distance to the camera, varies with the incidence angle, and accumulates unevenly depending on the camera layout. The queries of Surflo are instead free 3D points and normals, which the ODE transports from a source distribution onto the surface distribution. Point density is therefore a property of the surface rather than of the camera configuration, which yields a more homogeneous sampling and facilitates meshing, and the predicted distribution also covers occluded regions. This is where modeling a distribution rather than a function helps: sparse-view reconstruction is one-to-many, and while a regression decoder would return the conditional mean of the plausible surfaces, placing points between them, a generative decoder commits to one of them. Finally, flow matching refines its prediction over several forward passes, which lets rendering feedback enter the ODE itself (Section 2.3), whereas a regression decoder would require a two-stage pipeline separating network inference from a decoupled test-time optimization. Other regression alternatives we considered have further drawbacks: regressing a fixed-size point set caps the output resolution and requires self-attention across all points to make them mutually consistent, while regressing a geometric field on a grid trades scene extent against level of detail, as dense voxels are costly for full scenes with background, and sparse voxels require additional machinery and are mostly used for single objects.

What does the decoder learn beyond VGGT? Although Surflo relies on VGGT for its image features, its source distribution, its 3D positional encoding and the cameras used for guidance, the decoder does not merely reformat VGGT's geometry. Unguided Surflo is more accurate than the raw pointmaps of its own frozen backbone at every view count (Tables 3 and 6), and on 5 of the 8 benchmarks with 16 views (Tables 2 and 1). Moreover, replacing the VGGT-seeded source distribution with pure Gaussian noise still reconstructs the scene (Table 4): the decoder genuinely transports points from noise onto the surface, rather than denoising VGGT points.

D Training

Meshed DL3DV dataset. We build a scene-level surface dataset by enriching every scene of DL3DV [47] with a watertight mesh and an associated oriented point cloud. For each scene we run Gaussian Wrapping [21] to extract a watertight surface, and finally sample $\sim 10^7$ points uniformly on this surface together with their normals. We release the dataset as an auxiliary contribution at `https://huggingface.co/datasets/AntoineGuedon/DL3DV-10K-Meshed`, and the code and model weights at `https://github.com/Anttwo/Surflo`. We cache the VGGT tokens once per

(a) Reference



(b) Watertight mesh



Figure 7: **Meshed DL3DV dataset.** We enrich every scene of DL3DV [47] (a) with a watertight mesh (b) and an associated oriented point cloud. For each scene we run Gaussian Wrapping [21] to extract a watertight surface, and finally sample 10^7 points uniformly on this surface together with their normals.

scene and per view set to avoid recomputing them at every epoch. Figure 7 shows examples from the dataset.

Training. The decoder is trained as a velocity-prediction flow-matching model with the conditional optimal-transport scheduler of [48]. The starting state $\mathbf{x}_0$ for each query point is sampled as the VGGT world-point prediction (estimated depth values back-projected using estimated camera parameters) plus isotropic Gaussian noise of standard deviation 0.1 (in scene-normalized coordinates). Time samples are drawn from a logit-normal distribution $t = \sigma(z)$ with $z \sim \mathcal{N}(1.0, 1.6^2)$, biasing supervision towards $t \to 1$. With probability 0.1 the VGGT features are masked to zero during training to enable classifier-free guidance at inference.

Aligning GT to Predicted Points. The flow-matching objective supervises each query point against the ground-truth surface, but the two live in different coordinate frames: the ground-truth points are given in the original COLMAP frame of the dataset, whereas the predicted points follow the VGGT convention. We must therefore align them before computing the loss. This alignment has to be accurate enough to preserve fine geometric detail in the supervision, yet fast enough to run at every training step. We estimate an affine transform $(\mathbf{L}, \mathbf{t})$ that maps COLMAP world coordinates into the VGGT frame, i.e. $\mathbf{p}_{\text{vggt}} \approx \mathbf{p}_{\text{colmap}}\mathbf{L} + \mathbf{t}$. The transform is fitted from depth maps of the ground-truth mesh rendered from the COLMAP cameras, which we precompute for each scene and store as part of the dataset.

We first discard unreliable points by keeping only the top 25% of pixels ranked by VGGT depth confidence, then proceed in two steps. The first step produces a coarse but robust similarity alignment from the *reference camera alone*: we take only its ground-truth depth points, express them in the reference camera's view space through the COLMAP extrinsics, isotropically rescale them by the ratio of the VGGT and COLMAP camera-baseline extents (the mean distance of the camera centers to their centroid), and move them into the VGGT world frame using the corresponding VGGT camera pose. Because the rotation, translation, and scale are derived from camera geometry rather than from matching depth values, this step is insensitive to per-pixel depth noise and yields a reliable initialization. The second step refines this initialization into a full affine transform. For each camera, we measure the residual between the coarsely aligned ground-truth depth points and the VGGT depth points, normalize it by a characteristic spatial scale, and turn it into robust per-point weights via a soft-min; correspondences whose weight falls below the per-camera median are treated as outliers and dropped. A single weighted least-squares fit over the surviving depth points from all cameras then yields the final transform.

Table 5: **Robustness of the guidance to camera errors** on Tanks & Temples with 16 input views. Perturbed poses have an average translation error of 1% of the scene extent and an average rotation error of 1°. The last column reports the average rotation correction applied by the in-loop pose refinement, which is enabled by default.

Camera poses	Pose refinement	CD↓	F1↑	Rotation correction
VGGT	Off	0.0049	89.33	–
VGGT	On	0.0048	89.50	0.239°
Perturbed VGGT	Off	0.0052	88.72	–
Perturbed VGGT	On	0.0050	89.23	0.446°

E Guidance

Time grid. We use a bi-phase time grid of $50 + 100$ steps with a phase switch at $t = 0.95$, sampling $P = 10^5$ query points by default. At each step $t_k \geq 0.95$ the network predicts the velocity, from which we recover an estimate $\hat{\mathbf{x}}_1$ of the clean sample (point positions and, in the $\mathbb{R}^6$ point–normal parameterization, per-point normals).

Rendering guidance. The key idea of *rendering guidance* is to nudge this trajectory, in its final phase, so that the implied surface is photometrically and geometrically consistent with the input images. Guidance is activated only once $t \geq \tau_{\mathrm{g}} = 0.95$. At activation, the current estimate $\hat{\mathbf{x}}_1$ is used to instantiate a set of 3D Gaussians, one per generated point, whose learnable attributes are anisotropic log-scales, rotation quaternions, opacity, and view-dependent color (spherical harmonics up to degree 3).

Gaussians as a scaffold. These Gaussians are a scaffold, not an output: they only make the point set differentiably renderable, and are discarded once integration ends. Surflo thus predicts no appearance: each Gaussian's color is initialized to a neutral grey and is a free variable fitted in the inner loop, exactly like its opacity, scale and rotation. Color is hence only a proxy that converts multi-view photometric disagreement into a geometric gradient, and no color is read from the input images except through the rendering loss. While all Gaussian attributes receive gradients, only the resulting position offsets and normals are read back into the flow, where they define the guided target. Visibility and occlusions need no special treatment, as they follow from the front-to-back alpha compositing of the rasterizer: guidance only affects the parts of the scene visible in the input views, while occluded parts are entirely handled by the flow-matching decoder.

Inner loop. At every subsequent ODE step we run an inner loop of $M = 32$ gradient descent iterations that differentiably renders these Gaussians (via the RaDe-GS rasterizer [90]) into a single input view per iteration, drawn at random and cycling through all views, and minimizes a composite rendering loss [25, 21]. After the inner loop, the Gaussian displacements define a refined target $\hat{\mathbf{x}}_g$, which we convert back into a velocity; the ODE step is then taken with this *guided* velocity, so that the rendering gradients steer the generative trajectory. We additionally backpropagate the rendering gradients to the camera poses to account for potential, small pose errors in the VGGT predictions.

Robustness to camera errors. VGGT poses provide a good initialization but are not always pixel-accurate, while photometric supervision benefits from accurate alignment. Since guidance is already iterative, the in-loop pose refinement described above comes at no extra cost. Table 5 evaluates it on Tanks & Temples with 16 views, using either the VGGT poses or poses perturbed with an average translation error of 1% of the scene extent and an average rotation error of 1°. Perturbing the poses only increases the Chamfer distance by 4 to 6%, and the refinement responds by roughly doubling its applied rotation correction. Refinement, however, only corrects small pose errors: if the VGGT camera layout is grossly wrong, guidance cannot recover it, and the unguided output, which performs no rendering, is the more robust choice.

Monocular geometric experts. An additional, optional guidance term injects priors from an off-the-shelf monocular expert [46]. Following [25, 24], we use a scale-invariant depth-order loss between the rendered depth and the monocular depth prior. Only relative ordering is enforced, preventing the multi-view inconsistency of monocular depth predictions from affecting the flow trajectory.

Filtering outliers. The rendering guidance mechanism provides a natural criterion to detect outliers in the flow: noisy points and floaters that do not contribute to any input view converge to a low opacity. We exploit this signal to prune low-opacity points on the fly during the ODE integration, following the standard pruning threshold of 3D Gaussian Splatting [41].

Meshing. We do not triangulate the bare points. Following Gaussian Wrapping [21], we place Delaunay pivots along the anisotropic extent of each Gaussian fitted during guidance, tetrahedralize them, and extract the isosurface of the resulting occupancy field with marching tetrahedra, which yields watertight meshes by construction. This meshing is density-adaptive: decoding fewer points yields larger Gaussians and a wider pivot spacing, hence a coarser but still closed surface (Figure 5), so that the mesh loses detail rather than gaining holes. Only at extremely low densities (below $\sim$1K points for a full scene) may structures thinner than the local point spacing cause topology breaks or tearing.

F Evaluation protocol.

To ensure a fair comparison, all baselines receive the same set of unposed views for each scene, and latent methods are evaluated from their official checkpoints. Per-view, pointmap-based methods [68, 46] do not produce a single surface, so we fuse their predictions with TSDF fusion [16]. To recover both foreground and background geometry while preserving fine detail in the foreground, we use a multi-resolution voxel grid: three nested grids of 100^3 voxels each, with side lengths of 1, 3 and 10 times the scene diagonal, and a TSDF truncation margin of 1.25% of the scene diagonal.

Point cloud alignment. To compare a predicted point cloud against the ground-truth surface, we first bring the prediction into the ground-truth coordinate frame and then measure geometric agreement. All distance thresholds are expressed as fractions of the ground-truth bounding-box diagonal $d = \|\mathbf{p}_{\max} - \mathbf{p}_{\min}\|$, so that the protocol is invariant to the absolute scale of each scene. Alignment proceeds in two stages. First, we estimate a closed-form 7-DoF similarity transform (scale, rotation, and translation) with the Umeyama algorithm [66], using the predicted and ground-truth camera centers as correspondences; this fixes the global scale. Second, we refine the rigid component with a scale-locked, robust point-to-point ICP, in which the Umeyama scale is held fixed and only rotation and translation are updated. ICP runs for up to 30 iterations on voxel-downsampled clouds (voxel size $10^{-3}d$). In each iteration, correspondences beyond a truncation distance are discarded, where the truncation radius follows a geometric schedule from $5 \times 10^{-2}d$ down to $10^{-2}d$ so that early iterations are permissive and later ones strict. Of the surviving correspondences we retain only the closest 70% (least-trimmed squares), weight their residuals with a Huber loss (with δ set to half the current truncation radius), and solve a weighted Procrustes problem for the rigid update.

Metrics. After alignment, we clip prediction points lying outside the axis-aligned ground-truth bounding box to avoid asymmetrically inflating the prediction-to-ground-truth distance term. Finally, both clouds are voxel-downsampled to a common density (voxel size $10^{-3}d$), and we report the symmetric Chamfer distance together with the F-score at a distance threshold $\tau = 10^{-2}d$. Chamfer distances are reported normalized by d to allow for aggregation across scenes of differing absolute scale.

G Varying number of input views on Mip-NeRF 360

Table 6 complements the view-count experiment of Table 3 on Mip-NeRF 360, with the same protocol. As on Tanks & Temples, Surflo is the most accurate method at every view count, from 2 to 32 unposed input views, while the size of its latent remains fixed.

H Normal accuracy

Chamfer distance and F1 only evaluate point positions, and ignore the oriented half of the output of Surflo. We therefore also evaluate normals, against the TSDF-fused per-view baselines and against Gaussian Wrapping [21], which we use as the optimization-based reference as it outperforms NeuS2 [75] on multi-view mesh reconstruction. Results are reported in Table 7.

Table 6: **Varying number of input views** on Mip-NeRF 360. We report Chamfer Distance (CD, ↓) and F1-score (F1, ↑) as the number of unposed input views per scene varies from 2 to 32. Best, second-best and third-best are highlighted; greyed per-view pointmap variants are reported for reference and excluded from the ranking.

	2 views		4 views		8 views		32 views	
Method	CD↓	F1↑	CD↓	F1↑	CD↓	F1↑	CD↓	F1↑
Per-view feed-forward (no global surface)								
VGGT pointmap [68]	0.0765	8.33	0.0310	47.62	0.0166	67.93	0.0133	74.90
DA3 pointmap [46]	0.0750	8.35	0.0300	45.45	0.0168	66.73	0.0139	73.72
Per-view feed-forward (fused global surface)								
VGGT + TSDF [68]	0.0755	9.63	0.0326	40.79	0.0244	55.29	0.0161	65.56
DA3 + TSDF [46]	0.0746	9.24	0.0322	40.31	0.0220	55.17	0.0162	64.75
Latent feed-forward (fixed-size output)								
NOVA3R [11]	0.0795	8.46	0.0642	21.41	0.0492	28.01	0.0562	18.95
Per-scene optimization (sparse-view)								
2DGS [32]	0.0754	8.73	0.0344	37.15	0.0283	47.45	0.0217	52.13
RaDe-GS [90]	0.0756	8.76	0.0352	37.11	0.0293	47.89	0.0215	53.08
Gaussian Wrapping [21]	0.0766	7.28	0.0339	41.61	0.0251	53.97	0.0162	62.41
Surflo (ours) - No guidance	0.0714	13.07	0.0123	68.23	0.0121	75.52	0.0071	81.02
Surflo (ours) - With guidance	0.0736	10.40	0.0134	67.25	0.0122	75.57	0.0137	80.37

Table 7: **Normal accuracy** with 16 input views: angular error (in degrees, ↓) between the normals of each reconstructed mesh and the ground-truth normals. *Full scene*: all reconstructed geometry, including background and occluded regions. *Visible foreground*: background culled with the ground-truth bounding box, and evaluation restricted to mesh faces visible from the input views. Best and second-best are highlighted. Protocols are detailed in Appendix H.

	Full scene				Visible foreground	
	Tanks & Temples		Mip-NeRF 360		T&T	DTU
Method	Mean	Median	Mean	Median	Mean	Mean
VGGT + TSDF [68]	52.05	39.28	68.45	62.36	19.73	17.29
DA3 + TSDF [46]	53.40	42.21	67.45	60.37	25.42	37.83
Gaussian Wrapping [21]	75.33	74.55	70.59	65.04	45.99	36.46
Surflo (ours)	42.45	29.84	55.06	51.17	15.12	15.37

Full-scene protocol. We extract a mesh for each method and align it to the ground-truth frame as in Appendix F. We then sample points and normals on the mesh, associate each sampled point with its nearest ground-truth point and normal, and report the mean and median angular errors. All meshes are consistently outward-oriented by construction, since both TSDF fusion and marching tetrahedra operate on space partitions where each cell carries a signed volume; we also verified this quantitatively and visually. Surflo has the most accurate normals of all methods, reducing the median angular error by 24% on Tanks & Temples and 15% on Mip-NeRF 360 relative to the strongest baseline.

The absolute errors are high for all methods, as this protocol includes regions that are hard to reconstruct from sparse views. First, a substantial part of the scene can be occluded in all input views: contrary to Surflo, per-view and optimization methods only reconstruct visible surfaces, and the meshing step fills unobserved regions with potentially broken geometry. Second, many background elements appear in at most one image, and resolving this ambiguity from RGB alone requires a strong learned prior. Optimization baselines have none, and their large number of primitives produces chaotic surfaces when underconstrained, even with VGGT initialization (Figures 1 and 4). The full-scene protocol thus also measures the ability to predict accurate normals for background and occluded geometry.

Visible-foreground protocol. To evaluate normals in a more controlled setting, we additionally cull the background with the ground-truth bounding box, rasterize each mesh from all input viewpoints to identify its visible faces, and sample points and normals on these visible faces only. We use Tanks & Temples and the object-centric DTU [1], which both provide ground-truth bounding boxes. For DTU, we use the dense ground-truth foreground point clouds; since they come without normals,

Table 8: **Varying number of output points P**, with 8 unposed input views. We report Chamfer Distance (CD, ↓) and F1-score (F1, ↑). Best and second-best are highlighted; greyed per-view pointmap variants are reported for reference and excluded from the ranking.

	Tanks & Temples						Mip-NeRF 360					
	$P = 10^3$		$P = 10^4$		$P = 10^5$		$P = 10^3$		$P = 10^4$		$P = 10^5$	
Method	CD↓	F1↑	CD↓	F1↑	CD↓	F1↑	CD↓	F1↑	CD↓	F1↑	CD↓	F1↑
Per-view feed-forward (no global surface)												
VGGT pointmap [68]	0.0139	49.52	0.0097	73.20	0.0083	80.53	0.0226	37.74	0.0179	61.34	0.0165	67.98
DA3 pointmap [46]	0.0231	44.76	0.0188	62.62	0.0177	67.38	0.0233	38.89	0.0187	60.76	0.0167	66.71
Per-view feed-forward (fused global surface)												
VGGT + TSDF [68]	0.0191	39.60	0.0150	65.33	0.0138	70.88	0.0313	25.24	0.0265	49.16	0.0243	55.39
DA3 + TSDF [46]	0.0258	37.50	0.0218	57.89	0.0210	61.55	0.0291	24.62	0.0243	48.06	0.0220	55.27
Surflo (ours) - No guidance	0.0121	52.36	0.0073	81.06	0.0055	87.93	0.0201	36.34	0.0147	63.78	0.0121	75.52
Surflo (ours) - With guidance	0.0120	53.02	0.0071	81.25	0.0051	88.99	0.0207	37.83	0.0145	65.49	0.0122	75.59

Table 9: **Runtime and peak GPU memory** on a single H100 GPU as the number of output points P varies, with 8 input views, averaged over Tanks & Temples and Mip-NeRF 360. Guided inference uses our default guidance setting (Table 10).

Guidance	P	Encoding	ODE	Guidance	Total	Peak VRAM
Off	10^3	0.32 s	1.25 s	–	1.57 s	6.16 GiB
Off	10^4	0.32 s	1.46 s	–	1.78 s	6.15 GiB
Off	10^5	0.32 s	8.07 s	–	8.39 s	8.49 GiB
On	10^3	0.32 s	0.59 s	35.55 s	36.47 s	13.91 GiB
On	10^4	0.32 s	1.22 s	34.51 s	36.04 s	13.92 GiB
On	10^5	0.32 s	10.39 s	37.83 s	48.53 s	13.94 GiB

we estimate normals by local PCA and orient them by propagation. Errors decrease for every method, which supports the explanation above, and Surflo remains the most accurate on both datasets, even though it targets full scenes rather than single foreground objects.

Limitations of the metric. Since points are sampled uniformly on the surface, the average angular error is dominated by large, smooth regions and reflects fine details poorly. The visual gap between Surflo and VGGT+TSDF, which over-smooths and removes much of the detail, is therefore wider than these numbers suggest (Figures 1 and 4).

I Efficiency and output resolution

Output resolution. Table 8 complements the multi-resolution experiment of Section 3.2: with 8 input views, we vary the number of output points P from 10^3 to 10^5 on Tanks & Temples and Mip-NeRF 360. Accuracy improves steadily with P, and Surflo outperforms the TSDF-fused baselines at every density, with or without guidance.

Runtime and memory. Table 9 reports runtime and peak GPU memory on a single H100 GPU. With 8 input views, peak memory stays below 9 GiB without guidance and 14 GiB with it, from 10^3 to 10^5 decoded points. Up to 10^4 points, runtime is dominated by fixed costs rather than by the output resolution. The ODE cost then grows with the number of points, whereas the guidance cost stays essentially constant: it is mostly set by the number of rendering iterations rather than by the number of Gaussians, as more points means smaller Gaussians, each cheaper to rasterize. The guidance cost is also independent of the number of input views N, since each iteration renders a single, randomly selected input view.

Guidance settings. The cost of guidance is controlled by the number of guided ODE steps after $t = 0.95$, the number M of inner iterations per step, and the number of final *polish* iterations, during which the Gaussian attributes are refined with fixed point positions after integration. As in 3D Gaussian Splatting [41], guidance also applies adaptive density control (densification and pruning) to the Gaussians, so that the number of output points may differ from the requested P. Table 10 compares our default setting, used in all other experiments, with two lighter ones. The fastest reconstructs a complete scene of 10^5 points in under 16 seconds, about twice the cost of unguided

Table 10: **Guidance settings** at $P = 10^5$, averaged over Tanks & Temples and Mip-NeRF 360. *Steps* $\times$ M: number of guided ODE steps (after $t = 0.95$) times inner-loop iterations per step; *Polish*: final iterations refining the Gaussians with fixed point positions. The default setting is used in all other experiments.

Setting	Steps $\times$ M	Polish	ODE	Guidance	Total	Peak VRAM	CD↓	F1↑
Minimal	50×16	100	5.21 s	10.04 s	15.71 s	13.98 GiB	0.0103	76.10
Short	50×32	100	5.20 s	17.28 s	22.79 s	13.99 GiB	0.0096	77.39
Default	100×32	500	10.41 s	36.20 s	46.92 s	13.98 GiB	0.0094	78.15

inference, while already producing very clean point clouds and competitive metrics, though slightly noisier meshes. Guidance is in any case optional: unguided Surflo already outperforms all baselines at every view count (Tables 3 and 6).

J Extended limitations and future work

Frozen backbone. Using a frozen VGGT backbone was a practical choice driven by data scarcity rather than a requirement of the formulation: the encoder only needs multi-view features, and any multi-view transformer, such as π^3 [74] or Depth Anything 3 [46], could be substituted. The binding constraint is data: existing scene-level surface supervision is either tiny, object-centric, or synthetic and indoor-only, and even our $\sim$10.5K meshed scenes are too few to learn an image encoder from scratch. With enough data, training the whole pipeline from scratch might outperform the frozen backbone. Surflo also inherits the failure modes of VGGT: with very few views or extreme baselines, its pointmaps can be unreliable, and a grossly wrong camera layout cannot be recovered, as guidance only corrects small pose errors (Appendix E).

Scene extent and local precision. Surflo generates points in a space normalized by the scene extent, so that its accuracy is roughly a fixed fraction of this extent. Growing coverage therefore trades local precision for reach: a much larger capture is still reconstructed correctly at the global scale, but a small region of it is coarser than when reconstructed from its own views. Adding locality to the latent, e.g., with a chunk-wise encoding whose decoded distribution is conditioned on a bounded region rather than on the whole scene, should decouple local precision from global extent.

Supervision biases. Our supervision runs Gaussian Wrapping on the full DL3DV view sets (200 to 300 views) with COLMAP cameras, which yields detailed surfaces, and Surflo surpasses Gaussian Wrapping applied in our sparse-view setting at every view count. As it is trained on $\sim$10.5K diverse scenes with largely uncorrelated per-scene noise, Surflo is fairly robust to the noise of its supervision, but it inherits its systematic biases. The sky is reconstructed as a curved shell; we keep these artifacts in all reported results for a fair comparison, although they can be removed at inference by segmenting the sky and culling the points that rasterize into it. Good angular coverage of the scene is preferred, although captures need not be object-centric, and transparent or textureless structures remain imperfect. Finally, occluded geometry is noisier, as guidance only constrains observed regions.

Scaling supervision. Our data pipeline requires no human annotation: COLMAP and Gaussian Wrapping produce meshes and oriented points automatically from casual captures, so the cost of scaling is compute rather than labeling. Synthetic data with perfect ground truth could help where our supervision is weakest, i.e., thin structures, sky and occluded interiors, and a mixture with real captures would preserve the photometric statistics the encoder needs. Another direction is self-distillation from a multi-view teacher: running VGGT on many views yields pseudo-ground-truth surfaces that could supervise our objective from far fewer views. This asymmetric setup already drives our training, as our supervision is computed from 200 to 300 views while Surflo sees at most 16. Since Surflo remains stable far beyond this training regime (Figure 6), it might even be distilled into itself, enabling training on captures without any 3D ground truth.

Appearance. Surflo represents geometry but not appearance. Guided inference does produce a renderable Gaussian model as a byproduct, since the auxiliary Gaussians carry colors fitted to the input views, but this is not a feed-forward prediction of appearance. As the latent already supports arbitrary 3D queries, predicting appearance from the same latent with an additional head is a natural extension.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction (Section 1) state four contributions—(i) a fixed-size latent encoder, (ii) a flow-matching decoder with arbitrary-resolution output, (iii) rendering-based guidance, and (iv) a meshed version of DL3DV. Each is described in the corresponding part of Section 2 and evaluated quantitatively in Section 3.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: A dedicated *Limitations* paragraph in Section 5, extended in Appendix J, discusses the reliance on a frozen VGGT backbone and its failure modes, the trade-off between scene extent and local precision, the biases inherited from the Gaussian Wrapping supervision, noisier occluded regions, tearing at very low point densities, and the fact that Surflo models geometry but not appearance. The inference-time cost of rendering-based guidance is quantified in Section 3.2 and Appendix I.

   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 introduce theorems or formal theoretical claims. The equations in Section 2 are method definitions (flow-matching objective, guidance update), not statements requiring proof.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Section 2 fully describes the encoder (frozen VGGT backbone, 3D positional encoding, Perceiver compressor) and decoder (per-point flow-matching transformer with AdaLN conditioning) architectures. Section 3.1 reports training (AdamW, batch size, $4\times$H100, 400K iterations), inference (Euler ODE with 150 steps), and evaluation protocols. Appendix B gives exhaustive hyperparameters (head counts, MLP ratios, qk-norm, LayerScale, sigma bands, time schedule, classifier-free dropout, etc.), Appendix D details the dataset construction and training, Appendix E the guidance, and Appendix F the evaluation protocol used for every baseline.

   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: Training and inference code, model checkpoints, and the meshed-DL3DV dataset (Section 2.4, Appendix D) are publicly released, with usage instructions, at `https://github.com/Anttwo/Surflo`, which is also linked from the project page given on the first page. Architecture and training details are further disclosed in Section 2, Section 3.1, and Appendix 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: Section 3.1 reports the training/test split for DL3DV, the eight evaluation benchmarks (with native vs. pseudo-ground-truth surfaces), the optimizer (AdamW), batch size, query budget per scene, and the inference settings (Euler solver with 150 steps, 100K query points by default, guidance starting at t=0.95 with 32 updates per step). Appendix B provides full architecture and flow-matching hyperparameters, and Appendix F details the evaluation protocol (TSDF voxel size, baseline checkpoints, etc.).

   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: We do not report error bars across training seeds, as a single training run takes 400K iterations on $4\times$H100 GPUs and repeating it under several seeds for every ablation is computationally prohibitive. The reported metrics are means over many test scenes per benchmark (between tens and hundreds of scenes depending on the dataset), which makes the comparison between methods stable in practice; the ranking between Surflo and the baselines is consistent across all eight benchmarks (Tables 2 and 1).

   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: Section 3.1 reports that training uses $4\times$H100 GPUs for 400K iterations with batch size 12 scenes and 8K query points per scene per step. Section 3.2 reports inference cost on a single H100: a 16-view encode is one forward pass through VGGT and the Perceiver compressor, decoding 10^5 points from the cached latent takes about 8 seconds, and a complete guided reconstruction takes under 50 seconds; Appendix I details runtime and peak memory. Total project compute (including preliminary runs) is larger than the headline numbers and is acknowledged as such.

   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 have reviewed the NeurIPS Code of Ethics and the research conforms to it. The work uses publicly released multi-view image datasets in line with their academic-research terms, does not involve human subjects or private data, and the manuscript and any submitted artifacts have been anonymized for double-blind review.

   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: [N/A]

    Justification: Our work introduces a fundamental framework for modeling 3D surfaces from images that can support a broad range of downstream applications (robotics, AR/VR, simulation, content creation). We do not see a direct path to specific negative societal impacts beyond those that already apply to multi-view 3D reconstruction in general.

    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: The released artifacts—a 3D-surface reconstruction model and a derived mesh dataset built on top of the publicly released DL3DV scenes—do not pose dual-use risks comparable to those of pre-trained language models, image generators, or face-recognition systems. The meshed-DL3DV dataset is derived from images already released by the DL3DV authors and inherits their distribution policy.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: All third-party models (VGGT, Depth-Anything-3, NOVA3R, 2DGS, RaDe-GS, Gaussian Wrapping) and datasets (DL3DV, BlendedMVS, DTU, Tanks & Temples, Mip-NeRF 360, DeepBlending, SCRREAM, ML-Hypersim) are credited via citation in Sections 2 and 3 and are used in line with their respective public release terms for academic research.

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: The paper introduces two new assets, both publicly released at `https://github.com/Anttwo/Surflo`: (i) a meshed version of DL3DV (~10.5K scenes, each

enriched with a watertight surface mesh and ~10^7 oriented surface points), described in Section 2.4 and Appendix D, and (ii) the trained Surflo encoder/decoder weights together with training and inference code. The construction pipeline and sampling protocol are described in Appendix D, and documentation of the data format accompanies the release.

Guidelines:

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

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

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

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects. All evaluations are automatic geometric metrics (Chamfer distance, F1) computed against existing public benchmarks.

Guidelines:

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

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

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

Answer: [N/A]

Justification: The paper does not involve research with human subjects, so no IRB approval was required.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: LLMs are not part of the proposed method or experimental pipeline. The model uses a frozen vision backbone (VGGT) and a flow-matching transformer decoder; no LLM is used as an important, original, or non-standard component of the core methodology.

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.