FoundSurface: Feedforward Ray-Based 3D Scene Surface Reconstruction from Unposed Images

Shenxing Wei[1] **Jinxi Li**[1] **Zihui Zhang**[1] **Ajay Kumar**[2] **Wei Wang** [3] * **Bo Yang**[1] *

[1]vLAR Group, [2]DSAI, The Hong Kong Polytechnic University; [3]Regressor, Inc.

{shenxing.wei,jinxi.li,zihui.zhang}@connect.polyu.hk, bo.yang@polyu.edu.hk

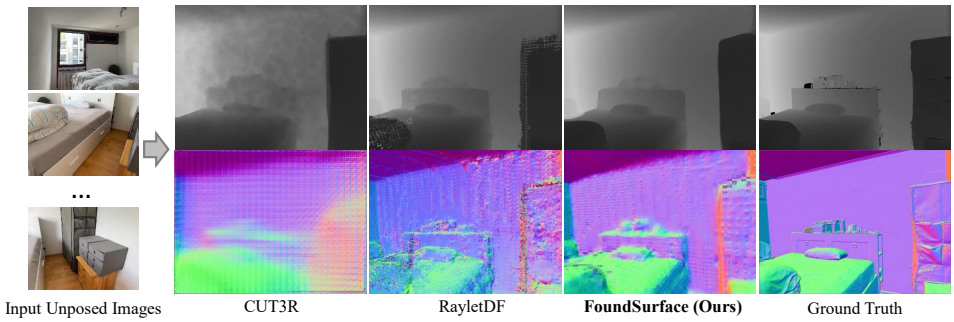

Figure 1: We introduce a large feed-forward model FoundSurface to recover continuous 3D scene surfaces from an arbitrary number of images without needing poses.

Abstract

Accurate 3D surface reconstruction from unposed images remains a fundamental challenge in computer vision. Recent feed-forward pointmap methods eliminate the need for camera poses but inherently produce discrete, sparse representations. In this paper, we present a generalizable pipeline that directly recovers continuous, high-fidelity 3D surfaces from unposed images by seamlessly integrating pointmap- and ray-based representations. Our approach first extracts explicit geometric priors using a pre-trained pointmap model. A query-view-centric selection module then identifies geometrically relevant reference views via a robust surfel voting mechanism. Next, a cross-attention regressor bridges target query rays with reference features to estimate a continuous coarse surface. Lastly, a raylet-based module aggregates local 3D features to carve out high-frequency details and analytical normals. Trained as a single foundation model on 9 diverse datasets, our method demonstrates exceptional zero-shot generalization on 6 unseen datasets, clearly surpassing state-of-the-art baselines in geometric accuracy and detail preservation. Code and data are available at: https://github.com/vLAR-group/FoundSurface

1 Introduction

Accurate 3D surface reconstruction from multiple images plays a key role in emerging applications such as embodied AI and augmented reality. In recent years, significant progress has been made in the field of 3D reconstruction through the development of various 3D representation techniques [82]. These include coordinate-based methods such as radiance fields (NeRF) [56], and point-based methods like 3D Gaussian Splatting (3DGS) [43], and their variants [29, 12]. While these approaches achieve impressive results in novel view synthesis, they usually rely on highly accurate camera poses, which are often unavailable in practice, and typically struggle to recover precise 3D surfaces.

*Corresponding authors.

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

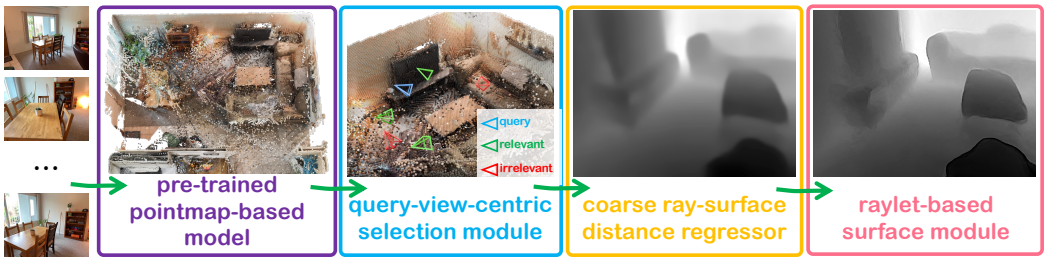

Figure 2: The general workflow and components of our framework.

Furthermore, they generally require time-consuming per-scene optimization rather than functioning as generalizable, feed-forward models [99].

Recently, the pioneering feedforward method DUSt3R [81] has enabled precise 3D geometry reconstruction by directly predicting pixel-aligned pointmaps from unposed image pairs. Subsequent methods [101] have extended this pointmap representation to handle multiple views [78, 92], to improve computational efficiency [65, 52], *etc*.. Leveraging powerful Transformer architectures and large-scale training datasets, these approaches achieve excellent accuracy in recovering surface points from image collections. However, the reconstructed pointmaps are inherently discrete and sparse, lacking the continuous and fine-grained geometric details required for many downstream applications, such as robot manipulation and scene editing.

Another recent line of work focuses on ray-based representations such as RayDF [53] and RayletDF [84]. Their key innovation is learning a ray-surface distance field for any query light ray conditioned on visual inputs, thus enabling the recovery of continuous and high-fidelity surfaces. However, they still rely on accurate camera poses or sparse point clouds as inputs to reconstruct detailed surfaces.

In this paper, we present a generalizable, feed-forward 3D surface reconstruction pipeline that takes only a set of unposed images as input and directly recovers continuous, high-fidelity 3D surfaces. We achieve this by seamlessly integrating pointmap- and ray-based representations. In particular, as illustrated in Figure 2, our pipeline comprises four key components: 1) a pre-trained pointmap-based model such as VGGT [78], which extracts visual tokens and generates pixel-aligned pointmaps along with estimated camera poses; 2) a query-view-centric selection module designed to identify geometrically relevant reference views; 3) a coarse ray-surface distance regressor utilizing cross-attention layers; and 4) a raylet-based surface module for refining surface details and normals.

Among the four modules of our pipeline, the core is the second and third modules, which seamlessly integrate pointmap- and ray-based representations, effectively estimating continuous surfaces for any query views conditioned on the recovered pointmaps from unposed input images. In contrast, directly applying RayletDF to VGGT pointmaps can lead to inaccurate raylet initialization, resulting in multi-surface artifacts and missing regions.

Our pipeline is based on large Transformer and standard neural layers with sufficient capacity, without relying on domain specific assumptions. Therefore, it is rather suitable to consume various indoor and outdoor datasets. We train a single **found**ation 3D **surface** model on a mixture of 9 public datasets with depth annotations, including BlendMVS [94], WildRGB [86], ScanNet [21], HyperSim [63], MVS-Synth [37], ARKitScenes [4], S3DIS [1], SceneNN [33], and GTASfM [79]. Our model is termed **FoundSurface** and Figure 1 shows qualitative results. Our contributions are:

- We introduce a large feed-forward model to recover continuous 3D scene surfaces from an arbitrary number of images without needing poses.
- We design a query-view-centric selection module together with a cross attention based fusion module to seamlessly integrate pointmap- and ray-based 3D representations.
- We train a single foundation model and demonstrate superior 3D surface reconstruction accuracy across multiple datasets, clearly surpassing existing baselines.

2 Related Work

To recover 3D surfaces from images, classical SfM [59] and SLAM [9] systems extract sparse point clouds that can be smoothed into continuous surfaces, though they often fail to capture fine details.

Early deep learning methods utilized explicit 3D representations such as voxel grids [17, 90, 91], point clouds [24], octrees [72], and meshes [41]. Nevertheless, the quality of these discrete structures is inherently bottlenecked by memory footprints and resolution constraints. Comprehensive surveys of these methods are available in [5, 32].

Implicit Representations for Surface Reconstruction: To bypass the resolution limits of explicit 3D formats, implicit representations use simple MLPs to parameterize continuous 3D shapes [**?**]. Techniques such as occupancy fields (OF) [15, 55], un/signed distance fields (U/SDF) [16, 60, 75, 22], vector fields (VF) [93, 61], and NeRF [56, 73] operate by mapping 3D coordinates to local geometric or visual properties. While these approaches and their variants [82] yield highly accurate surface reconstructions and 2D renderings, they rely on dense 3D spatial sampling and exhaustive network evaluations per view and often require images with accurate poses as input. In this paper, our FoundSurface learns a feedforward model from unposed images, directly regressing surface points for any query viewing rays in single pass.

Point-based Representations for Surface Reconstruction: A recent breakthrough is 3D Gaussian Splatting (3DGS) [43], which models environments as a set of 3D Gaussians with optimizable positions, covariances, and colors. Supported by a custom tile-based rasterizer, 3DGS delivers real-time 2D view synthesis. Nevertheless, because it inherently lacks tight geometric constraints, 3DGS falls short in extracting fine surface details and generating highly accurate depth maps. While a multitude of recent works [13, 25, 85, 31, 34, 18, 89, 48, 35, 98, 87, 58, 97, 67, 11, 102, 30] have attempted to enforce surface alignment and improve depth quality, they generally require time-consuming per-scene optimization and images with accurate poses as input.

Another breakthrough is DUSt3R [81] which demonstrates 3D geometry can be represented by pixel-aligned pointmaps and recovered from unposed image pairs in a single forward pass. A series of succeeding methods [101, 47, 8, 23, 14, 78, 80, 92, 38, 100, 50, 88, 39, 104, 46, 26, 49] extend it from different perspectives, including handling more views , modeling videos, processing faster, *etc*.. Although these methods achieve impressive results, they are limited to recovering discrete and sparse point clouds, falling short of producing continuous, high-fidelity surfaces.

Ray-based Representations for Surface Reconstruction: Another line of work models 3D shapes through ray-based neural functions. By taking individual rays as input, these models directly regress either RGB values [51, 27, 57, 69, 10] or explicit surface points [28, 2, 53, 84, 45, 70, 40, 71, 96]. Despite their impressive rendering efficiency and surface accuracy, most of these methods require posed images as input and struggle to generalize across diverse 3D scenes. In this paper, we leverage the robust surface representation capabilities of RayDF [53] and RayletDF [84] and seamlessly integrate them with pointmap-based representations. This synergistic approach enables feedforward 3D surface reconstruction directly from sets of images.

3 FoundSurface

FoundSurface recovers continuous high-fidelity 3D scene surfaces through a coarse-to-fine, discrete-to-continuous progression. Given a set of unposed and uncalibrated multi-view images $\mathcal{I} = \{\mathbf{I}_i \in \mathbb{R}^{H\times W\times 3} \mid i = 1, ..., K\}$, we first extract an explicit geometric prior using a pre-trained pointmap-based model elaborated in Section 3.1. Next, we filter out redundant information by selecting geometrically relevant views for any target query in Section 3.2. We then transition from 3D points to rays by regressing an initial, continuous coarse surface, which resolves global geometry and occlusions detailed in Section 3.3. Finally, we refine this coarse surface using a raylet-based module that aggregates high-resolution local 3D features to recover fine-grained details, yielding the final high-fidelity surfaces, discussed in Section 3.4.

3.1 Pre-trained Pointmap-based Model

This model aims to establish a reliable spatial anchor for building continuous surfaces without camera poses. Given a set of input images $\mathcal{I} = \{\mathbf{I}_i \in \mathbb{R}^{H\times W\times 3}\}$, we employ the pre-trained VGGT model [78] to predict explicit geometric priors. Specifically, the network's prediction heads output a pixel-aligned pointmap $\mathbf{P}_i \in \mathbb{R}^{H\times W\times 3}$, a camera pose matrix $\mathbf{T}_i \in \mathbb{R}^{4\times 4}$, and an intrinsic matrix $\mathbf{C}_i \in \mathbb{R}^{3\times 3}$ for each input image $\mathbf{I}_i$.

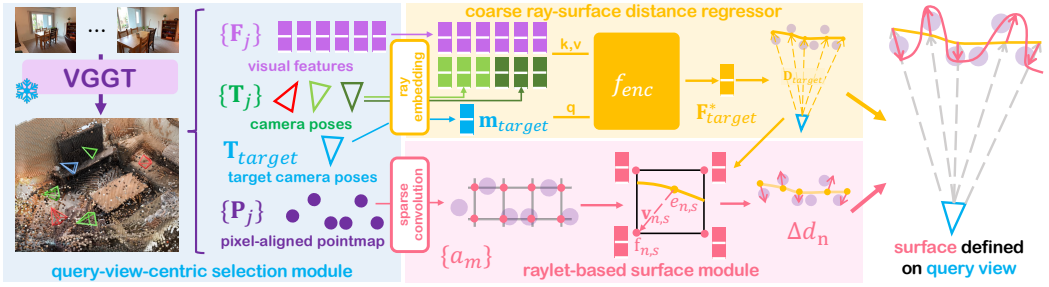

Figure 3: Left: The query-view-centric selection module. Middle: The coarse ray-surface distance regressor, which seamlessly integrates pointmap- and ray-based representations. Right: The raylet-based surface module, which further refines the initially estimated surfaces.

During this forward pass, we concurrently retain intermediate patch features to capture rich contextual information. Let $\mathbf{F}_{i,l} \in \mathbb{R}^{h \times w \times d_l}$ denote the patch feature extracted from the l-th layer of VGGT, where $h = H/s$ and $w = W/s$ are the patch-level spatial dimensions downsampled from H and W by ratio s, and d_l is the channel depth. We specifically collect features from four layers, defined as the set $L = \{4, 11, 17, 23\}$. To prepare compact and geometry-aware visual tokens for subsequent modules, we fuse these multi-scale patch features using a two-layer MLP, denoted as f_{fuse}. The entire extraction and fusion process can be succinctly formulated as:

$$\{\mathbf{P}_i, \mathbf{T}_i, \mathbf{C}_i, \{\mathbf{F}_{i,l}\}_{l \in L}\} \leftarrow \text{VGGT}(\mathbf{I}_i), \quad \mathbf{F}_i = f_{\text{fuse}}([\mathbf{F}_{i,4}, \mathbf{F}_{i,11}, \mathbf{F}_{i,17}, \mathbf{F}_{i,23}]) \tag{1}$$

where the operator $[\cdot]$ denotes concatenation along the channel dimension, yielding the final fused visual feature $\mathbf{F}_i \in \mathbb{R}^{h \times w \times d_{out}}$. The discrete geometric predictions $\mathbf{P}_i/\mathbf{T}_i/\mathbf{C}_i$ and the fused visual tokens $\mathbf{F}_i$ constitute the foundation for our continuous ray-based formulation.

3.2 Query-View-Centric Selection Module

As shown in the leftmost block of Figure 3, when reconstructing the geometry of a novel query view, which is defined explicitly by a given target camera pose $\mathbf{T}_{target}$ and $\mathbf{C}_{target}$, intuitively, not all input views are helpful. Views that do not observe the target surfaces will only introduce noise and computational overhead. Therefore, instead of globally querying features of all input views, we design a selection module to identify only the overlapping, geometrically relevant reference views by a *surfel-based* voting mechanism, ensuring the selected views provide reliable geometric coverage.

Specifically, we first establish a canonical geometric reference for this novel query view. We project all available input pointmaps onto the camera plane defined by the target query pose, and apply a standard depth buffer (z-buffer) to render a dense target pseudo-depth map, $\mathbf{D}^{target}$. We then uniformly divide $\mathbf{D}^{target}$ into non-overlapping patches, conceptualized as target surfels.

To evaluate a candidate view j out of all input views, we similarly divide its pointmap $\mathbf{P}_j$ into candidate surfels and project them onto the target view. For each pixel (x, y) within a projected candidate surfel, we compare its depth value $\mathbf{D}^j_{x,y}$ against the canonical target depth $\mathbf{D}^{target}_{x,y}$. The pixel casts a valid match vote according to an indicator function:

$$v_{x,y} = \mathbb{I}\left(\left|\mathbf{D}^{target}_{x,y} - \mathbf{D}^j_{x,y}\right| < \delta_{depth}\right) \tag{2}$$

A candidate surfel is considered successfully "matched" if the proportion of positive votes $v_{x,y}$ within it exceeds a threshold τ_{vote}. By aggregating these surfel-level results, we compute the percentage of matched surfels for the entire candidate view j, yielding a robust, coverage-aware overlap score. Finally, we rank all input views based on this surfel overlap score and retain only the *top-k* candidates. This geometric filtering eliminates irrelevant regions and provides our subsequent attention modules with a highly curated and effective query candidate set: $\{\mathbf{T}_j\}_{j=1}^K, \{\mathbf{C}_j\}_{j=1}^K, \{\mathbf{P}_j\}_{j=1}^K, \{\mathbf{F}_j\}_{j=1}^K$. More implementation details are provided in Appendix A.1.

3.3 Coarse Ray-Surface Distance Regressor

With the relevant views selected, the next challenge is locating the actual surface intersection for the target camera rays. This module aims to seamlessly integrate pointmap- and ray-based representations to directly estimate an initial, continuous coarse surface. To robustly regress this initial surface, we

establish a local coordinate system centered at the target query view. As shown in the yellow block of Figure 3, given the target camera pose $\mathbf{T}_{target}$ and intrinsics $\mathbf{C}_{target}$, we align the candidate views to this target viewpoint and tokenize their pixel-aligned Plücker rays via a convolution layer (Conv) to yield ray embeddings $\mathbf{m}_j$ for each relevant view j as:

$$\mathbf{m}_j = \text{Conv}\big(\mathcal{R}(\mathbf{T}_j^{rel}, \mathbf{C}_j)\big) \in \mathbb{R}^{hw \times d}, \quad \text{where } \mathbf{T}_j^{rel} = \mathbf{T}_{target}^{-1}\mathbf{T}_j \tag{3}$$

Here, $\mathbf{T}_j^{rel}$ defines the relative transformation to the target viewpoint, and $\mathcal{R}(\cdot)$ denotes the operation that casts camera parameters into dense Plücker ray maps. To inject spatial awareness into the visual features $\mathbf{F}_j$ (obtained from the backbone in Section 3.1), we condition them with the corresponding Plücker ray tokens using an MLP-based network f_{cond} and get the reference features $\hat{\mathbf{F}}_j$ as:

$$\hat{\mathbf{F}}_j = f_{\text{cond}}([\mathbf{m}_j, \mathbf{F}_j]), \tag{4}$$

where $[\cdot]$ denotes concatenation along the channel dimension.

Similarly, for the target query view, we cast Plücker rays using its parameters $(\mathbf{I}_{4\times4}, \mathbf{C}_{target})$ and tokenize them into target ray embeddings $\mathbf{m}_{target}$. To aggregate spatially aligned features from the candidates, we employ a Transformer network f_{enc} consisting of alternating self- and cross-attention layers. Specifically, in the cross-attention mechanism, the target query ray embeddings act as the queries ($\mathbf{Q}$), while the conditioned candidate features $\{\hat{\mathbf{F}}_j\}_{j=1}^K$ act as the keys ($\mathbf{K}$) and values ($\mathbf{V}$). This allows the target query rays to adaptively fetch relevant geometric information:

$$\mathbf{F}_{target}^* = f_{\text{enc}}\left(\mathbf{Q} = \mathbf{m}_{target};\ \mathbf{K}, \mathbf{V} = \{\hat{\mathbf{F}}_j\}_{j=1}^K\right) \tag{5}$$

Decoding the aggregated feature map $\mathbf{F}_{target}^*$ yields a ray distance map $\mathbf{D}^{target}$. This distance map effectively establishes our continuous, albeit coarse, ray-surface representation. Specifically, to locate the 3D surface point $\mathbf{p}_{x,y}$ for a given pixel (x, y) in the target query view, we cast a ray using the known target query camera parameters:

$$\mathbf{p}_{x,y} = \mathbf{p}_{target} + \mathbf{D}_{x,y}^{target} \cdot \mathbf{r}_{x,y} \tag{6}$$

where $\mathbf{p}_{target} \in \mathbb{R}^3$ is the optical center and $\mathbf{r}_{x,y} \in \mathbb{R}^3$ is the normalized ray direction. Both are uniquely determined by the target query pose $\mathbf{T}_{target}$ and intrinsics $\mathbf{C}_{target}$. More implementation details can be found in Appendix A.2.

While this coarse surface successfully captures the overall scene geometry and resolves occlusions, it inherently lacks fine-grained, high-frequency local details due to the patch-level resolution of the visual tokens. Therefore, we treat these coarse surface points and their corresponding ray directions as a set of sampled raylets conceptualized by RayletDF [84], denoted as $\{\mathbf{l}_n = (\mathbf{p}_n, \mathbf{r}_n)\}_{n=1}^N$, which serve as the geometric foundation for our subsequent raylet-based surface module to refine the details.

3.4 Raylet-based Surface Module

To elevate our coarse surface to a high-fidelity continuous representation, we must compensate for the missing local geometric details. We achieve this by refining the coarse raylets $\{\mathbf{l}_n\}_{n=1}^N$ through the aggregation of high-resolution local 3D features.

We first voxelize the initial pointmaps $\{\mathbf{P}_j\}_{j=1}^K$ and index the corresponding visual features $\{\mathbf{F}_j\}_{j=1}^K$ obtained in Section 3.3. We then utilize a 3D sparse convolution network (SPConv) [19] to process these indexed features, constructing a robust 3D local feature volume. As shown in the rightmost block of Figure 3, the sparse convolution network produces a set of spatial anchors and their associated local features, denoted as $\mathcal{A} = \{\mathbf{a}_m\}_{m=1}^M$. Here, the anchor points are generated through sparse convolution over the voxelized 3D representation, rather than being directly sampled from the original VGGT pointmaps. Each anchor is associated with a learned local geometric feature, which is subsequently used to refine nearby raylets.

For each coarse raylet $\mathbf{l}_n = (\mathbf{p}_n, \mathbf{r}_n)$ generated from the previous module, we search for its S-nearest anchor points in 3D space, yielding a local neighborhood $\mathcal{A}_n = \{(\mathbf{a}_{n,s}, \mathbf{f}_{n,s})\}_{s=1}^S$. To make the network highly sensitive to local geometry, we explicitly calculate the spatial relationships between the coarse raylet and its local anchors. Specifically, we extract the Euclidean distances $e_{n,s} = \|\mathbf{p}_n - \mathbf{a}_{n,s}\|_2$ and the relative direction vectors $\mathbf{v}_{n,s} = (\mathbf{p}_n - \mathbf{a}_{n,s})/e_{n,s}$.

Table 1: Quantitative results of estimated ray-surface distances.

	test on → ScanNet++					*test on* → 7Scenes					*test on* → MultiScan				
	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta\uparrow$	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta\uparrow$	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta\uparrow$
Projection [78]	0.580	0.959	0.360	0.554	0.636	0.188	0.391	0.091	0.078	0.887	0.566	0.767	0.462	0.521	0.510
PriorDA [83]	0.198	0.290	0.123	0.101	0.868	0.154	0.266	0.074	0.034	0.913	0.384	0.471	0.307	0.310	0.646
CUT3R [80]	0.335	0.446	0.222	0.173	0.703	0.197	0.305	0.105	0.054	0.878	0.452	0.530	0.454	0.410	0.463
FLARE [100]	0.312	0.495	0.157	0.156	0.771	0.228	0.363	0.112	0.066	0.841	0.396	0.589	0.305	0.343	0.677
RayletDF [84]	0.179	0.334	0.119	0.205	0.896	0.158	0.295	0.084	0.049	0.907	0.331	0.447	0.306	0.425	0.719
Ours *w/o refine*	0.162	0.270	0.097	0.063	0.894	0.130	**0.230**	0.068	**0.029**	**0.945**	0.233	0.309	0.222	0.147	0.737
Ours	**0.132**	**0.259**	**0.079**	**0.067**	**0.910**	**0.123**	0.233	**0.064**	0.031	0.943	**0.205**	**0.287**	**0.190**	**0.137**	**0.774**
	test on → Replica					*test on* → MuSHRoom					*test on* → Tanks&Temples				
Projection [78]	0.495	1.093	0.201	0.453	0.802	1.192	1.596	0.759	1.539	0.333	3.135	6.044	0.289	5.825	0.771
PriorDA [83]	0.116	0.189	0.045	0.018	0.974	0.341	0.506	0.309	0.476	0.760	1.572	**2.586**	**0.136**	**0.970**	0.850
CUT3R [80]	0.365	0.473	0.149	0.106	0.772	0.689	0.868	0.610	1.028	0.485	4.982	6.482	0.579	8.779	0.444
FLARE [100]	0.307	0.453	0.115	0.063	0.843	0.553	0.882	0.323	0.528	0.649	2.014	3.390	0.143	1.218	0.803
RayletDF [84]	0.071	0.179	0.032	0.023	0.975	0.374	0.640	0.390	0.996	0.795	3.970	7.940	0.442	10.93	0.748
Ours *w/o refine*	0.137	0.215	0.053	0.021	0.976	0.350	0.511	0.299	0.430	0.734	1.622	3.144	0.183	1.929	0.865
Ours	**0.064**	**0.165**	**0.027**	**0.015**	**0.981**	**0.254**	**0.458**	**0.232**	**0.429**	**0.822**	**1.233**	2.977	0.144	2.012	**0.908**

Finally, a refinement network processes the local anchor features alongside these explicit geometric relationships to regress a precise distance residual Δd_n. This aggregation is formulated as:

$$\Delta d_n = f_{\text{refine}}\left(\left[\underset{s=1}{\overset{S}{\text{MEAN}}}(\mathbf{f}_{n,s}),\ e_{n,1}, \mathbf{v}_{n,1}, \ldots, e_{n,S}, \mathbf{v}_{n,S}\right]\right) \tag{7}$$

where $[\cdot]$ denotes feature concatenation, and MEAN represents the mean-pooling function applied exclusively to the visual features of the K local neighbors. The final refined surface point is obtained by updating the coarse raylet:

$$\mathbf{p}_n^* = \mathbf{p}_n + \Delta d_n \cdot \mathbf{r}_n \tag{8}$$

This refinement dynamically carves out the high-frequency details, yielding a continuous, high-fidelity 3D surface. As a byproduct of this continuous ray-based formulation, our method natively supports the analytical derivation of surface normals. Given the input raylet and the final refined distance, we can easily compute a closed-form normal vector, effectively circumventing the noise typical of numerical approximations. We defer the full derivation and other details to the Appendix A.3.

3.5 Training

We employ a decoupled two-stage training strategy to train our core modules.

Stage 1: Coarse Ray-Surface Distance Regression. In this stage, our goal is to establish an accurate global geometric framework. We train the coarse ray-surface distance regressor to predict the continuous coarse depth map $\mathbf{D}^{target}$. Supervision is provided by the aligned ground truth depth $\mathbf{D}_{gt}$, using a loss function that combines a pixel-wise ℓ_1 distance with a spatial gradient loss to enforce surface consistency:

$$\ell_{coarse} = |\mathbf{D}_{gt} - \mathbf{D}^{target}| + \|\nabla\mathbf{D}_{gt} - \nabla\mathbf{D}^{target}\| \tag{9}$$

Stage 2: Raylet-based Surface Refinement. Once the coarse geometry is reasonably established, we freeze or isolate the first-stage predictions and focus entirely on training the raylet-based surface module. To capture high-frequency geometric details without destabilizing the learned global structure, we predict a residual distance map $\Delta\mathbf{D}$. During this stage, we apply a stop-gradient operation to the coarse predictions, formulating the fine-stage loss as:

$$\ell_{fine} = \left|\mathbf{D}_{gt} - \left(\text{sg}(\mathbf{D}^{target}) + \Delta\mathbf{D}\right)\right| \tag{10}$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operator. Here, the global residual map $\Delta\mathbf{D}$ is formed by aggregating the local raylet residuals corresponding to each pixel, i.e., $\Delta\mathbf{D} = \{\Delta d_n\}_{n=1}^{N}$, where Δd_n is computed following the local feature aggregation mechanism detailed in Section 3.4. This two-stage training paradigm effectively decouples global topology localization from local surface refinement, thereby enabling high-fidelity 3D reconstructions. More details are in Appendix A.4.

4 Experiments

Training Datasets: We train our model on a mixture of 9 datasets, including the training splits of ScanNet [21], GTASfM [79] and ARKitScenes [4], together with the full datasets of Blend-

MVS [94], WildRGB [86], HyperSim [63], MVS-Synth [37], S3DIS [1] and SceneNN [33] . For evaluation, we use ScanNet++ [95], 7Scenes [66], MultiScan [54], Mushroom [62], Replica [68], and Tanks&Temples [44]. Detailed dataset splits are provided in Appendix A.17.

Baselines: We compare our approach against five representative baselines: **1) VGGT Projection** [78], evaluated via the direct projection of its pointmaps; **2) PriorDA** [83], a state-of-the-art depth completion model applied to VGGT pointmaps; **3) CUT3R** [80], a feed-forward model utilizing state tokens for novel query views; **4) FLARE** [100], the latest feed-forward Gaussian splatting model; and **5) RayletDF** [84], a state-of-the-art generalizable ray-based method.

Metrics: We evaluate surface reconstruction by comparing the predicted ray–surface distance against the ground-truth distance. We report the absolute distance error (**ADE**, in meters), together with four commonly used metrics [74]: root mean squared error (**RMSE**), absolute relative distance (**Abs Rel**), squared relative distance (**Sq Rel**), and the Threshold accuracy ($\delta \leq 1.25$) which measures the percentage of predictions that fall within the specified threshold of the ground truth.

For normal evaluation, we follow the metrics used in normal estimation [3]. We report both the mean and median angular errors on normal. Additionally, we evaluate performance by measuring the percentage of pixels with an angular error below thresholds of 11.25°, 22.5°, and 30.0°. Detailed computation of all metrics is in Appendix A.7&A.8.

4.1 Evaluation of Out-of-domain Reconstruction

In this setting, our model takes 50 unposed input views and predicts ray–surface distances for query rays. For each scene, we uniformly sample 50 images from the test RGB sequence as input views. We further ablate the number of input views in Tables 3& 4.

We evaluate on ScanNet++, 7Scenes, MultiScan, MuSHRoom, and Tanks&Temples. These datasets are excluded from both the VGGT and our training datasets, thereby forming a strictly out-of-domain evaluation setting. For **VGGT Projection** [78], we directly project the pointmaps predicted by VGGT into the query view using Open3D [103]. We further combine VGGT projection with **PriorDA** [83], a state-of-the-art depth completion model, to fill missing regions after projection. For the feed-forward reconstruction model **CUT3R** [80], we first forward all input views and then query the scene state token to predict query-view point coordinates. For the feed-forward Gaussian splatting model **FLARE** [100], we render query-view depth maps from the predicted 3D Gaussians. For the ray-based method **RayletDF** [84], which is originally optimized for Gaussians and point clouds, we retrain it on VGGT-predicted pointmaps from our training datasets for a fair comparison. Standard deviation results are provided in Appendix A.6.

Results & Analysis: Table 1 and Figure 4 report quantitative and qualitative comparisons on ray-surface distance estimation for all query rays of test images. To assess the contribution of our Raylet-based refinement module, we also report the performance of our coarse ray-surface distance regressor in Table 1, denoted as *w/o refine*.

Our method achieves the best ADE across all out-of-domain datasets, outperforming all baselines by a large margin. For example, on ScanNet++, our ADE is 0.132m compared with 0.198m for PriorDA. These results demonstrate the strong generalization ability of FoundSurface on unseen datasets. Compared with the state-token-based model CUT3R, our method preserves finer geometric details, as illustrated in Figure 4. Removing the proposed refinement module substantially degrades reconstruction accuracy, confirming its effectiveness. More results are provided in Appendix A.13.

4.2 Evaluation of Surface Normal Accuracy

We further evaluate the normal estimation performance of FoundSurface and all baselines. Since some baselines, such as PriorDA, cannot directly produce analytical surface normals, we additionally compute normals from the predicted depth maps using numerical differentiation for reference.

Results & Analysis. As shown in Table 2 and Figure 5, our method achieves the best performance using only depth supervision, demonstrating the effectiveness of FoundSurface as a robust surface representation. Additional qualitative comparisons are provided in Appendix A.10.

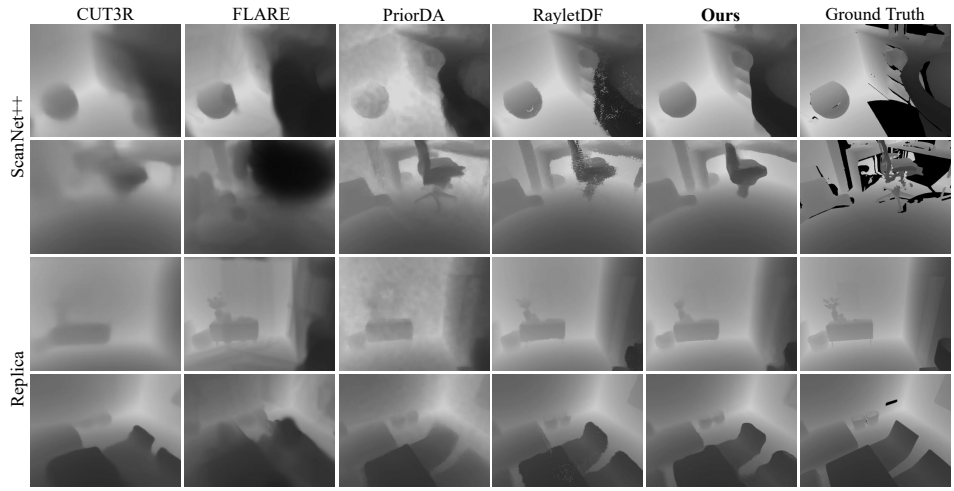

Figure 4: Qualitative results of our method and baselines for ray distance on novel views.

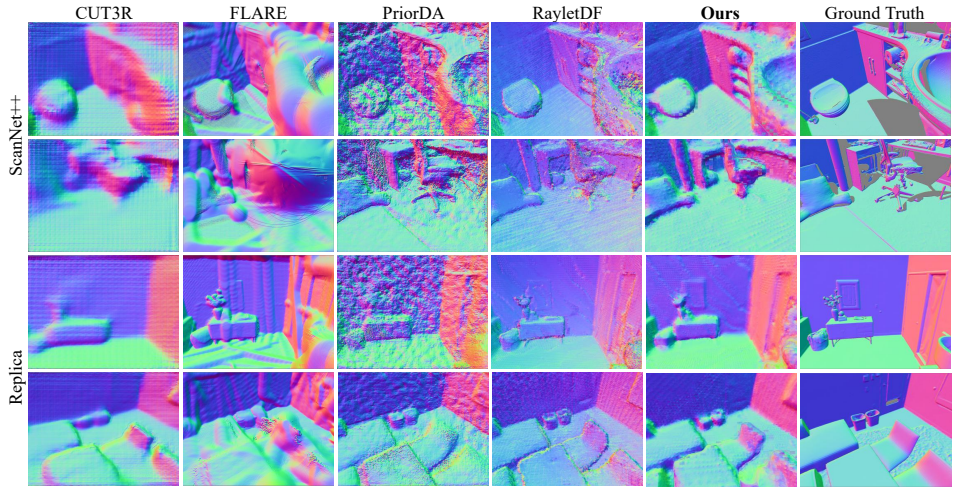

Figure 5: Qualitative results of our method and baselines for normal estimation on novel views.

4.3 Influence of the Number of Input Views

We evaluate the effect of input-view density by varying the number of input views from 5 to 100. As shown in Tables 3& 4, FoundSurface consistently achieves superior reconstruction accuracy across all settings. Its advantage is particularly pronounced given sparse views, demonstrating stronger robustness under limited observations. Additional qualitative results are provided in Appendix A.11.

4.4 Evaluation of Single View Reconstruction

In this section, we evaluate single-view reconstruction. For each test sequence, we sample one image as the input view and use its two adjacent frames as query views for evaluation. Quantitative results are reported in Table 5. Our method still achieves the best performance across all datasets. CUT3R obtains the second-best results on most datasets, likely benefiting from its training design for unseen-view inpainting. Additional qualitative results are provided in Appendix A.12.

4.5 Ablation Study

We investigate the effectiveness of surfel matching, hierarchical features, local coordinate, and reference view selection. Due to computational constraints, as full training requires approximately 288 A800 GPU hours, we train the ablation variants on ScanNet and evaluate them on the test split of MuSHRoom dataset. Additional results are provided in Appendix A.15.

Table 2: Quantitative results of estimated normals.

	test on → ScanNet++					*test on* → Replica				
	mean↓	med↓	11.25°↑	22.5°↑	30°↑	mean↓	med↓	11.25°↑	22.5°↑	30°↑
Projection [78]	75.06	80.31	0.066	0.124	0.156	67.23	69.79	0.072	0.149	0.197
PriorDA [83]	48.00	36.29	0.097	0.261	0.363	36.26	30.30	0.181	0.405	0.519
CUT3R [80]	39.93	32.83	0.168	0.371	0.474	34.64	26.20	0.219	0.447	0.548
FLARE [100]	52.29	46.31	0.111	0.259	0.346	33.25	24.63	0.255	0.471	0.579
RayletDF [84]	39.87	34.70	0.084	0.288	0.439	28.96	24.57	0.150	0.451	0.630
Ours *w/o refine*	36.49	28.29	0.193	0.380	0.488	40.47	38.56	0.061	0.220	0.353
Ours	**29.83**	**21.21**	**0.292**	**0.557**	**0.654**	**20.72**	**12.67**	**0.476**	**0.711**	**0.781**

Table 3: Quantitative results of estimated ray-surface distances under sparse views on ScanNet++.

	5 Views					10 Views					25 Views				
	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta\uparrow$	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta\uparrow$	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta\uparrow$
Projection [78]	1.604	1.812	0.884	1.618	0.117	1.364	1.650	0.773	1.386	0.234	0.927	1.302	0.551	0.923	0.450
PriorDA [83]	0.635	0.776	0.397	0.583	0.565	0.395	0.525	0.252	0.338	0.729	0.256	0.360	0.164	0.168	0.822
CUT3R [80]	0.484	0.616	0.306	0.352	0.589	0.406	0.524	0.262	0.251	0.642	0.303	0.413	0.199	0.146	0.736
FLARE [100]	1.164	1.456	0.615	1.114	0.353	0.816	1.126	0.438	0.731	0.512	0.448	0.702	0.234	0.326	0.708
RayletDF [84]	0.679	0.920	0.440	1.001	0.633	0.417	0.622	0.270	0.613	0.763	0.231	0.396	0.149	0.285	0.872
Ours	**0.370**	**0.514**	**0.253**	**0.260**	**0.690**	**0.252**	**0.394**	**0.164**	**0.168**	**0.800**	**0.166**	**0.300**	**0.104**	**0.093**	**0.878**

Table 4: Quantitative results of estimated ray-surface distances under dense views on ScanNet++.

	50 Views					70 Views					100 Views				
	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta\uparrow$	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta\uparrow$	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta\uparrow$
Projection [78]	0.580	0.959	0.360	0.554	0.636	0.469	0.831	0.294	0.434	0.696	0.363	0.691	0.226	0.317	0.762
PriorDA [83]	0.198	0.290	0.123	0.101	0.868	0.190	0.283	0.116	0.086	0.874	0.180	0.278	0.105	0.077	0.887
CUT3R [80]	0.335	0.446	0.222	0.173	0.703	0.359	0.470	0.239	0.181	0.667	0.395	0.507	0.266	0.208	0.623
FLARE [100]	0.312	0.495	0.157	0.156	0.771	0.288	0.458	0.143	0.123	0.784	0.281	0.464	0.139	0.128	0.796
RayletDF [84]	0.179	0.334	0.119	0.205	0.896	0.170	0.322	0.111	0.159	0.893	0.162	0.617	0.107	0.138	0.896
Ours	**0.132**	**0.259**	**0.079**	**0.067**	**0.910**	**0.126**	**0.248**	**0.076**	**0.062**	**0.916**	**0.121**	**0.240**	**0.072**	**0.059**	**0.922**

Table 5: Quantitative results of single-view reconstruction.

	test on → ScanNet++					*test on* → 7Scenes					*test on* → MultiScan				
	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta\uparrow$	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta\uparrow$	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta\uparrow$
Projection [78]	1.454	1.759	0.766	1.444	0.233	0.910	1.323	0.467	0.848	0.533	0.517	0.807	0.398	0.479	0.602
PriorDA [83]	0.174	0.288	0.096	0.092	0.895	0.155	0.247	0.076	**0.031**	0.924	0.172	0.275	0.138	0.082	0.820
CUT3R [80]	0.139	0.249	0.079	0.047	0.922	0.153	0.249	0.077	0.037	0.925	0.151	0.225	0.130	0.083	0.852
FLARE [100]	0.301	0.559	0.154	0.203	0.835	0.257	0.497	0.127	0.145	0.866	0.265	0.434	0.254	0.221	0.781
RayletDF [84]	0.150	0.280	0.129	0.117	0.869	0.162	0.277	0.085	0.046	0.908	0.212	0.341	0.193	0.155	0.775
Ours	**0.100**	**0.221**	**0.061**	**0.044**	**0.930**	**0.134**	**0.230**	**0.068**	**0.031**	**0.941**	**0.122**	**0.209**	**0.108**	**0.056**	**0.882**
	test on → Replica					*test on* → MuSHRoom					*test on* → Tanks&Temples				
Projection [78]	2.466	2.768	0.863	2.459	0.137	1.690	1.957	0.905	1.685	0.099	9.517	11.87	0.895	13.40	0.241
PriorDA [83]	0.219	0.365	0.087	0.648	0.885	0.359	0.550	0.213	0.209	0.725	1.619	**2.828**	0.198	1.163	0.802
CUT3R [80]	0.230	0.363	0.089	0.059	0.920	0.245	0.401	0.170	0.149	0.825	1.919	3.075	**0.146**	**1.137**	0.833
FLARE [100]	0.901	1.536	0.313	0.795	0.689	0.758	1.220	0.405	0.676	0.559	3.830	5.995	0.254	3.102	0.709
RayletDF [84]	0.236	0.496	0.105	0.162	0.882	0.357	0.599	0.298	0.486	0.772	3.034	6.299	0.548	14.79	0.774
Ours	**0.121**	**0.252**	**0.050**	**0.034**	**0.952**	**0.185**	**0.347**	**0.141**	**0.131**	**0.872**	**1.480**	3.386	0.309	6.846	**0.899**

Effects of Surfel Voting. For reference-view selection, we compare two strategies: (1) estimating view overlap via per-pixel comparison, and (2) our surfel voting strategy. As shown in Table 6, surfel voting consistently improves performance, indicating that surfel-level aggregation provides a more reliable measure of cross-view relevance. Additional comparisons are provided in Appendix A.14.

Effects of Hierarchical Features. To evaluate the impact of hierarchical features, we ablate our model to use only the final-layer VGGT tokens. As Table 6 shows, this removal increases all error metrics, highlighting that multi-scale representations are vital for capturing the global structure and local details required for accurate ray-surface distance regression.

Effects of Local Coordinates. We evaluate the choice of coordinate space by comparing global VGGT coordinates against a query-view-centric local coordinates. Table 6 shows that adopting local coordinates provides distinct performance gains, highlighting its stronger generalization.

Effects of Reference-view Selection. As shown in the last row of Table 6, removing the *top-k* reference-view selection severely degrades performance. This confirms that querying all views without filtering introduces detrimental noise and irrelevant observations.

4.6 Sensitivity to Geometric Priors

Since FoundSurface builds upon geometric priors predicted by VGGT, we further investigate its sensitivity to errors in the estimated pointmaps and camera poses. Specifically, we group test samples

Table 6: Ablation results on the test split of MuSHRoom.

	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta \uparrow$
w/o surfel vote	0.341	0.550	0.278	0.401	0.754
w/o hierarchy feature	0.369	0.575	0.307	0.444	0.732
w/o local coordinate	0.457	0.658	0.367	0.628	0.685
w/o *top-k* reference view	0.372	0.577	0.285	0.375	0.722
Full FoundSurface	**0.331**	**0.538**	**0.266**	**0.372**	**0.761**

according to their VGGT pointmap errors and camera pose errors, measured by Absolute Distance Error (ADE) and Absolute Trajectory Error (ATE), respectively. As shown in the following Table 7 and 8, our method maintains a consistent error level across different VGGT results, demonstrating its robustness.

Table 7: Results across different VGGT pointmap error levels on ScanNet++.

VGGT ADE	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta \uparrow$
0–0.05m	0.041	0.107	0.032	0.012	0.973
0.05–0.10m	0.113	0.250	0.070	0.051	0.928
0.10–0.20m	0.167	0.324	0.107	0.080	0.875
>0.20m	0.534	0.765	0.265	0.350	0.617

Table 8: Results across different VGGT camera error levels on ScanNet++.

VGGT ATE	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta \uparrow$
0–0.05m	0.069	0.172	0.044	0.027	0.958
0.05–0.10m	0.123	0.260	0.079	0.058	0.916
0.10–0.20m	0.153	0.287	0.099	0.071	0.895
>0.20m	0.444	0.645	0.227	0.678	0.799

4.7 Evaluation of Surface Reconstruction with 3D Metrics

We further evaluate the reconstructed 3D geometry on ScanNet++ using surface completeness, Chamfer distance (CD), accuracy, and F-scores. For a controlled comparison, we apply TSDF, Poisson, and NKSR to identical VGGT pointmaps. We also compare with generalizable feed-forward methods and PGSR, a per-scene optimization baseline. For all remaining methods, we reconstruct meshes by TSDF fusion of their predicted novel-view and VGGT points.

Table 9: Scores of 3D metrics for surface reconstruction metrics on ScanNet++.

Category	Method	Comp.↓	CD↓	Acc.↓	F1@5cm↑	F1@10cm↑
Surface Fitting	VGGT + TSDF [20]	0.148	0.095	**0.042**	0.724	0.847
	VGGT + Poisson [42]	0.102	0.074	0.047	0.787	0.897
	VGGT + NKSR [36]	0.166	0.124	0.082	0.647	0.758
Generalizable Feed-forward Reconstruction	PriorDA [83]	0.108	0.080	0.052	0.721	0.873
	CUT3R [80]	0.205	0.191	0.177	0.212	0.423
	FLARE [100]	0.122	0.131	0.140	0.440	0.704
Per-scene Optimization	PGSR [11]	0.081	0.090	0.100	0.616	0.779
Feed-forward Ray-based Model	RayletDF [84]	0.076	0.088	0.099	0.702	0.838
	Ours w/o refine	0.078	0.064	0.049	0.729	0.879
	Ours (voxel 0.025)	0.091	0.066	**0.042**	0.761	0.883
	Ours (voxel 0.01)	**0.066**	**0.056**	0.046	**0.792**	**0.910**

Results & Analysis. As shown in Table 9, our method achieves the best completeness, CD, and F-scores, while maintaining competitive accuracy. Notably, it outperforms classical surface fitting methods using identical VGGT pointmaps, demonstrating the advantage of our learned ray-based surface representation. The results also confirm the effectiveness of raylet-based refinement in improving 3D surface reconstruction quality.

5 Conclusion

In this paper, we introduced a generalizable, feed-forward model that reconstructs continuous, high-fidelity 3D surfaces from unposed image collections. By seamlessly integrating pointmap- and ray-based representations through a query-view-centric selection module and a raylet-based refinement network, our method bridges the gap between discrete geometric priors and continuous surface generation. Extensive cross-dataset evaluations demonstrate that our approach achieves superior zero-shot generalization and geometric accuracy compared to existing baselines.

Limitations: Our model has a large size due to the reliance on a pre-trained pointmap backbone. Future work will focus on optimizing the network architecture for deployment on mobile devices.

Acknowledgments: This work was supported in part by Research Grants Council of Hong Kong under Grants 15225522 &15228626 & 15219125, in part by National Natural Science Foundation of China under Grant 62271431, in part by Otto Poon Charitable Foundation Smart Cities Research Institute (8-CDCQ), in part by Research Center for Unmanned Autonomous Systems (1-CE3D), and in part by PolyU Kunpeng & Ascend Technology Innovation Incubation Center, The Hong Kong Polytechnic University.

References

[1] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *arXiv:1702.01105*, 2017.

[2] T. Aumentado-Armstrong, S. Tsogkas, S. Dickinson, and A. Jepson. Representing 3D Shapes with Probabilistic Directed Distance Fields. *CVPR*, 2022.

[3] G. Bae and A. J. Davison. Rethinking inductive biases for surface normal estimation. In *CVPR*, 2024.

[4] G. Baruch, Z. Chen, A. Dehghan, T. Dimry, Y. Feigin, P. Fu, T. Gebauer, B. Joffe, D. Kurz, A. Schwartz, and E. Shulman. ARKitScenes: A Diverse Real-World Dataset For 3D Indoor Scene Understanding Using Mobile RGB-D Data. *NeurIPS*, 2021.

[5] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, J. A. Levine, A. Sharf, and C. T. Silva. A Survey of Surface Reconstruction from Point Clouds. *Computer Graphics Forum*, 2017.

[6] E. Brachmann, M. Humenberger, C. Rother, and T. Sattler. On the Limits of Pseudo Ground Truth in Visual Camera Re-Localization. *ICCV*, 2021.

[7] R. Brégier. Deep regression on manifolds: a 3D rotation case study. *3DV*, 2021.

[8] Y. Cabon, L. Stoffl, L. Antsfeld, G. Csurka, B. Chidlovskii, J. Revaud, and V. Leroy. MUSt3R: Multi-view Network for Stereo 3D Reconstruction. *CVPR*, 2025.

[9] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. D. Reid, and J. J. Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Towards the Robust-Perception Age. *TRO*, 2016.

[10] J. Cao, H. Wang, P. Chemerys, V. Shakhrai, J. Hu, Y. Fu, D. Makoviichuk, S. Tulyakov, and J. Ren. Real-Time Neural Light Field on Mobile Devices. *CVPR*, 2023.

[11] D. Chen, H. Li, W. Ye, Y. Wang, W. Xie, S. Zhai, N. Wang, H. Liu, H. Bao, and G. Zhang. PGSR: Planar-based Gaussian Splatting for Efficient and High-Fidelity Surface Reconstruction. *TVCG*, 2024.

[12] G. Chen and W. Wang. A Survey on 3D Gaussian Splatting. *ACM Computing Surveys*, 2024.

[13] H. Chen, F. Wei, C. Li, T. Huang, Y. Wang, and G. H. Lee. VCR-GauS: View Consistent Depth-Normal Regularizer for Gaussian Surface Reconstruction. *NeurIPS*, 2024.

[14] Z. Chen, M. Qin, T. Yuan, Z. Liu, and H. Zhao. LONG3R: Long Sequence Streaming 3D Reconstruction. *ICCV*, 2025.

[15] Z. Chen and H. Zhang. Learning Implicit Fields for Generative Shape Modeling. *CVPR*, 2019.

[16] J. Chibane, A. Mir, and G. Pons-Moll. Neural Unsigned Distance Fields for Implicit Function Learning. *NeurIPS*, 2020.

[17] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. *ECCV*, 2016.

[18] J. Chung, J. Oh, and K. M. Lee. Depth-Regularized Optimization for 3D Gaussian Splatting in Few-Shot Images. *CVPRW*, 2024.

[19] S. Contributors. Spconv: Spatially sparse convolution library. `https://github.com/traveller59/spconv`, 2022.

[20] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996.

[21] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. *CVPR*, 2017.

[22] X. Deng, B. Yang, Y. Shi, and B. Wang. PrimitiveUDF: Primitive-Based Unsigned Distance Fields for Surface Reconstruction from Point Clouds. *ECCV*, 2026.

[23] S. Elflein and L. Leal-taix. Light3R-SfM: Towards Feed-forward Structure-from-Motion. *CVPR*, 2025.

[24] H. Fan, H. Su, and L. Guibas. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. *CVPR*, 2017.

[25] L. Fan, Y. Yang, M. Li, H. Li, and Z. Zhang. Trim 3D Gaussian Splatting for Accurate Geometry Representation. *arXiv:2406.07499*, 2024.

[26] K. Fang, C. Zhou, Y. Fu, H. H. Li, and Y. Chen. IncVGGT: Incremental VGGT for Memory-Bounded Long-Range 3D Reconstruction. *ICLR*, 2026.

[27] B. Y. Feng and A. Varshney. SIGNET: Efficient Neural Representation for Light Fields. *ICCV*, 2021.

[28] B. Y. Feng, Y. Zhang, D. Tang, and R. Du. PRIF: Primary Ray-based Implicit Function. *ECCV*, 2022.

[29] K. Gao, Y. Gao, H. He, D. Lu, L. Xu, and J. Li. NeRF: Neural Radiance Field in 3D Vision, A Comprehensive Review. *arXiv:2210.00379*, 2022.

[30] Z. Gao, J.-W. Bian, G. Lin, H. Chen, and C. Shen. SurfaceSplat: Connecting Surface Reconstruction and Gaussian Splatting. *ICCV*, 2025.

[31] A. Guédon and V. Lepetit. SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering. *CVPR*, 2024.

[32] X.-f. Han, H. Laga, and M. Bennamoun. Image-based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era. *TPAMI*, 2019.

[33] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung. SceneNN: A Scene Meshes Dataset with Annotations. *3DV*, 2016.

[34] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. *SIGGRAPH*, 2024.

[35] H. Huang, Y. Wu, C. Deng, G. Gao, M. Gu, and Y.-S. Liu. FatesGS: Fast and Accurate Sparse-View Surface Reconstruction using Gaussian Splatting with Depth-Feature Consistency. *AAAI*, 2025.

[36] J. Huang, Z. Gojcic, M. Atzmon, O. Litany, S. Fidler, and F. Williams. Neural kernel surface reconstruction. In *CVPR*, 2023.

[37] P.-H. Huang, K. Matzen, J. Kopf, N. Ahuja, and J.-B. Huang. DeepMVS: Learning Multi-view Stereopsis. *CVPR*, 2018.

[38] W. Jang, P. Weinzaepfel, V. Leroy, L. Agapito, and J. Revaud. Pow3R: Empowering Unconstrained 3D Reconstruction with Camera and Scene Priors. *CVPR*, 2025.

[39] H. Jin, R. Wu, T. Zhang, R. Gao, J. T. Barron, N. Snavely, and A. Holynski. ZipMap: Linear-Time Stateful 3D Reconstruction via Test-Time Training. *CVPR*, 2026.

[40] L. Jin, N. Kulkarni, and D. Fouhey. 3DFIRES: Few Image 3D REconstruction for Scenes with Hidden Surface. *CVPR*, 2024.

[41] H. Kato, Y. Ushiku, and T. Harada. Neural 3D Mesh Renderer. *CVPR*, 2018.

[42] M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM TOG*, 2013.

[43] B. Kerbl, U. Côte, G. Kopanas, U. Côte, T. Leimkühler, M.-p.-i. Informatik, and G. R. Aug. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *SIGGRAPH*, 2023.

[44] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM TOG*, 2017.

[45] N. Kulkarni, J. Johnson, and D. F. Fouhey. Directed Ray Distance Functions for 3D Scene Reconstruction. *ECCV*, 2022.

[46] Y. Lan, Y. Luo, F. Hong, S. Zhou, H. Chen, Z. Lyu, S. Yang, B. Dai, C. C. Loy, and X. Pan. STream3R: Scalable Sequential 3D Reconstruction with Causal Transformer. *ICLR*, 2026.

[47] V. Leroy, Y. Cabon, and J. Revaud. Grounding Image Matching in 3D with MASt3R. *ECCV*, 2024.

[48] J. Li, J. Zhang, X. Bai, J. Zheng, X. Ning, J. Zhou, and L. Gu. DNGaussian: Optimizing Sparse-View 3D Gaussian Radiance Fields with Global-Local Depth Normalization. *CVPR*, 2024.

[49] J. Li, T. Zhang, Y. Yang, Z. Zhang, P. Huang, K. W. M. Lin, and B. Yang. NeoMap: Training-free Novel-View Synthesis from Single Images and Videos. *ECCV*, 2026.

[50] Z. Li, J. Zhou, Y. Wang, H. Guo, W. Chang, Y. Zhou, H. Zhu, J. Chen, C. Shen, and T. He. WinT3R: Window-Based Streaming Reconstruction with Camera Token Pool. *ICLR*, 2026.

[51] C. Liu, Z. Li, J. Yuan, and Y. Xu. NeuLF: Efficient Novel View Synthesis with Neural 4D Light Field. *Eurographics Symposium on Rendering*, 2022.

[52] Y. Liu, S. Dong, S. Wang, Y. Yin, Y. Yang, Q. Fan, and B. Chen. SLAM3R: Real-Time Dense Scene Reconstruction from Monocular RGB Videos. *CVPR*, 2025.

[53] Z. Liu, B. Yang, Y. Luximon, A. Kumar, and J. Li. RayDF: Neural Ray-surface Distance Fields with Multi-view Consistency. *NeurIPS*, 2023.

[54] Y. Mao, Y. Zhang, H. Jiang, A. X. Chang, and M. Savva. MultiScan: Scalable RGBD scanning for 3D environments with articulated objects. *NeurIPS*, 2022.

[55] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. *CVPR*, 2019.

[56] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *ECCV*, 2020.

[57] J. Ost, I. Laradji, A. Newell, Y. Bahat, and F. Heide. Neural Point Light Fields. *CVPR*, 2022.

[58] W. Ouyang, G. Zhang, Y. Qiao, and T. He. GigaGS: Scaling up Planar-Based 3D Gaussians for Large Scene Surface Reconstruction. *arXiv:2409.06685*, 2024.

[59] O. Ozyesil, V. Voroninski, R. Basri, and A. Singer. A Survey of Structure from Motion. *Acta Numerica*, 2017.

[60] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. *CVPR*, 2019.

[61] E. M. Rella, A. Chhatkuli, E. Konukoglu, and L. Van Gool. Neural Vector Fields for Surface Representation and Inference. *IJCV*, 2024.

[62] X. Ren, W. Wang, D. Cai, T. Tuominen, J. Kannala, and E. Rahtu. Mushroom: Multi-sensor hybrid room dataset for joint 3d reconstruction and novel view synthesis. *WACV*, 2024.

[63] M. Roberts and N. Paczan. Hypersim: A Photorealistic Synthetic Dataset for Holistic Indoor Scene Understanding. *ICCV*, 2021.

[64] P. Schröppel, J. Bechtold, A. Amiranashvili, and T. Brox. A benchmark and a baseline for robust multi-view depth estimation. *3DV*, 2022.

[65] Y. Shen, Z. Zhang, Y. Qu, and L. Cao. FastVGGT: Training-Free Acceleration of Visual Geometry Transformer. *ICLR*, 2026.

[66] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. *CVPR*, 2013.

[67] G. Song, C. Cheng, and H. Wang. GVKF: Gaussian Voxel Kernel Functions for Highly Efficient Surface Reconstruction in Open Scenes. *NeurIPS*, 2024.

[68] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv:1906.05797*, 2019.

[69] M. Suhail, C. Esteves, L. Sigal, and A. Makadia. Light Field Neural Rendering. *CVPR*, 2022.

[70] P. B. Sundt and T. Theoharis. MARF: The Medial Atom Ray Field object representation. *Computers & Graphics*, 2023.

[71] P. B. Sundt and T. Theoharis. Towards Multi-view Consistency in Neural Ray Fields using Parametric Medial Surfaces. *Computers & Graphics*, 2024.

[72] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs. *ICCV*, 2017.

[73] A. Trevithick and B. Yang. GRF: Learning a General Radiance Field for 3D Representation and Rendering. *ICCV*, 2021.

[74] M. Turkulainen, X. Ren, I. Melekhov, O. Seiskari, E. Rahtu, and J. Kannala. DN-Splatter: Depth and Normal Priors for Gaussian Splatting and Meshing. *WACV*, 2025.

[75] B. Wang, Z. Yu, B. Yang, J. Qin, T. Breckon, L. Shao, N. Trigoni, and A. Markham. RangeUDF: Semantic Surface Reconstruction from 3D Point Clouds. *arXiv:2204.09138*, 2022.

[76] H. Wang and L. Agapito. 3d reconstruction with spatial memory. *arXiv:2408.16061*, 2024.

[77] H. Wang and L. Agapito. Amb3r: Accurate feed-forward metric-scale 3d reconstruction with backend. *arXiv:2511.20343*, 2025.

[78] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny. VGGT: Visual Geometry Grounded Transformer. *CVPR*, 2025.

[79] K. Wang and S. Shen. Flow-Motion and Depth Network for Monocular Stereo and Beyond. *RAL*, 2020.

[80] Q. Wang, Y. Zhang, A. Holynski, A. A. Efros, and A. Kanazawa. Continuous 3D Perception Model with Persistent State. *CVPR*, 2025.

[81] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud. DUSt3R: Geometric 3D Vision Made Easy. *CVPR*, 2024.

[82] Z. Wang. 3D Representation Methods: A Survey. *arXiv:2410.06475*, 2024.

[83] Z. Wang, S. Chen, L. Yang, J. Wang, Z. Zhang, H. Zhao, and Z. Zhao. Depth anything with any prior. *arXiv:2505.10565*, 2025.

[84] S. Wei, J. Li, Y. Yang, S. Zhou, and B. Yang. RayletDF: Raylet Distance Fields for Generalizable 3D Surface Reconstruction from Point Clouds or Gaussians. *ICCV*, 2025.

[85] Q. Wu, J. Zheng, and J. Cai. Surface Reconstruction from 3D Gaussian Splatting via Local Structural Hints. *ECCV*, 2024.

[86] H. Xia, Y. Fu, S. Liu, and X. Wang. RGBD Objects in the Wild: Scaling Real-World 3D Object Learning from RGB-D Videos. *CVPR*, 2024.

[87] H. Xiang, X. Li, X. Lai, W. Zhang, Z. Liao, K. Cheng, and X. Liu. GaussianRoom: Improving 3D Gaussian Splatting with SDF Guidance and Monocular Cues for Indoor Scene Reconstruction. *arXiv:2405.19671*, 2024.

[88] T. Xie, P. Yang, Y. Jin, Y. Cai, W. Yin, W. Ren, Q. Zhang, W. Hua, S. Peng, X. Guo, and X. Zhou. Scal3R: Scalable Test-Time Training for Large-Scale 3D Reconstruction. *CVPR*, 2026.

[89] H. Xu, S. Peng, F. Wang, H. Blum, D. Barath, A. Geiger, and M. Pollefeys. DepthSplat: Connecting Gaussian Splatting and Depth. *CVPR*, 2025.

[90] B. Yang, S. Rosa, A. Markham, N. Trigoni, and H. Wen. Dense 3D Object Reconstruction from a Single Depth View. *TPAMI*, 2019.

[91] B. Yang, S. Wang, A. Markham, and N. Trigoni. Robust Attentional Aggregation of Deep Feature Sets for Multi-view 3D Reconstruction. *IJCV*, 2020.

[92] J. Yang, A. Sax, K. J. Liang, M. Henaff, H. Tang, A. Cao, J. Chai, F. Meier, and M. Feiszli. Fast3R: Towards 3D Reconstruction of 1000+ Images in One Forward Pass. *CVPR*, 2025.

[93] X. Yang, G. Lin, Z. Chen, and L. Zhou. Neural Vector Fields: Implicit Representation by Explicit Learning. *CVPR*, 2023.

[94] Y. Yao, Z. Luo, S. Li, J. Zhang, Y. Ren, L. Zhou, T. Fang, and L. Quan. BlendedMVS: A Large-scale Dataset for Generalized Multi-view Stereo Networks. *CVPR*, 2020.

[95] C. Yeshwanth, Y.-C. Liu, M. Nießner, and A. Dai. ScanNet++: A High-Fidelity Dataset of 3D Indoor Scenes. *ICCV*, 2023.

[96] R. Yin, Y. Chen, S. Karaoglu, and T. Gevers. Ray-Distance Volume Rendering for Neural Scene Reconstruction. *ECCV*, 2024.

[97] M. Yu, T. Lu, L. Xu, L. Jiang, Y. Xiangli, and B. Dai. GSDF: 3DGS Meets SDF for Improved Rendering and Reconstruction. *NeurIPS*, 2024.

[98] Z. Yu, T. Sattler, and A. Geiger. Gaussian Opacity Fields: Efficient and Compact Surface Reconstruction in Unbounded Scenes. *SIGGRAPH Asia*, 2024.

[99] J. Zhang, Y. Li, A. Chen, M. Xu, K. Liu, J. Wang, X.-X. Long, H. Liang, Z. Xu, H. Su, C. Theobalt, C. Rupprecht, A. Vedaldi, K. Zhou, P. P. Liang, S. Lu, and F. Zhan. Advances in Feed-Forward 3D Reconstruction and View Synthesis: A Survey. *arXiv:2507.14501*, 2025.

[100] S. Zhang, J. Wang, Y. Xu, N. Xue, C. Rupprecht, X. Zhou, Y. Shen, and G. Wetzstein. FLARE: Feed-Forward Geometry, Appearance and Camera Estimation from Uncalibrated Sparse Views. *CVPR*, 2025.

[101] W. Zhang, Y. Wu, S. Li, W. Ma, X. Ma, Q. Li, and Q. Wang. Review of Feed-forward 3D Reconstruction: From DUSt3R to VGGT. *arXiv:2507.08448*, 2025.

[102] Z. Zhang, B. Huang, H. Jiang, L. Zhou, X. Xiang, and S. Shen. Quadratic Gaussian Splatting for Efficient and Detailed Surface Reconstruction. *arXiv:2411.16392*, 2024.

[103] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[104] D. Zhuo, W. Zheng, J. Guo, Y. Wu, J. Zhou, and J. Lu. Streaming Visual Geometry Transformer. *ICLR*, 2026.

A Appendix

The appendix includes:

- Details for query-view-centric selection.
- Details of coarse ray-surface distance regressor.
- Details of raylet-based surface module.
- Details of training.
- Computation cost analysis.
- Results of standard deviation.
- The definitions of all the evaluation metrics.
- More quantitative and qualitative results.
- Details of datasets.
- Implementation details for our model.
- Robustness evaluation of the S-nearest neighbor.
- Qualitative results of 3D outputs.
- Failure cases.
- Effectiveness of raylet-Based refinement.
- Efficiency of ray-Based surface queries.

A.1 Details of Query-View-Centric Selection

For reference view selection, we first project the N point coordinate frames to the query camera plane using the extrinsics and intrinsics predicted by VGGT, resulting in $N \times H \times W$ projected images. A pseudo-depth map for the query view is then generated by taking the minimum positive value across these projections. Lastly, both the projected images and the pseudo-depth map are divided into 8×8 non-overlapping patches. The candidate view evaluation uses thresholds of δ_{depth}=0.05 and τ_{vote}=0.3. For all experimental evaluations, the number of selected reference views is fixed at 2.

A.2 Details of Coarse Ray-Surface Distance Regressor

A.2.1 Details of Ray Embedding Generation

Given the camera parameters $\mathbf{T}_{target}$ and $\mathbf{C}_{target}$, ray parameter of pixel (u, v) is computed by

$$o = \mathbf{T}_{target}[:3, 3] \tag{11}$$

$$d = \text{Normalize}(\mathbf{T}_{target} \cdot \mathbf{C}_{target}^{-1} \cdot [u, v, 1]) \tag{12}$$

Then the Plücker parameter of (o, d) is calculated by: $(d,\ o \times d)$. Subsequently, the dense ray map is tokenized via a convolutional layer with a kernel and stride of 14.

A.2.2 Details of Spatial Feature Aggregation Transformer

The aggregation module, denoted as f_{enc}, is implemented as a Transformer-based architecture comprising four alternating blocks of self-attention and cross-attention layers. To balance computational efficiency and modeling capacity, each attention layer is configured with four heads. This structure facilitates the effective fusion of global contextual information with localized ray-specific features.

A.3 Details of Raylet-based Surface Module

A.3.1 Details of Raylet Network

The Raylet refinement network f_{refine} consists of three modules: (1) a dimensionality reduction network that contains two linear layers and a ReLU layer. (2) a Raylet prediction network which is an 8-layer MLP network. (3) a sparse convolution network to aggregate local features in 3D space. Considering that the pointmaps estimated by VGGT are defined within a normalized coordinate space, we set the voxelization resolution to 0.025. This configuration inherently facilitates an adaptive voxelization strategy, where the effective resolution scales dynamically according to the scene's spatial extent. Structure of Raylet prediction network and sparse convolution network is available in Section A.18.

A.3.2 Surface Normal Derivation

Similar to RayletDF, our method also has a nice property of deriving a closed-form expression for a normal vector, thanks to the nature of our ray-based formulation. Given a ray from camera center, we can parameter its direction in spherical coordinate system as $(\theta, \phi, 1)$ and convert the ray direction back to world coordinate as $\boldsymbol{r} = (x, y, z) = (\sin\theta\cos\phi, \cos\theta, \sin\theta\sin\phi)$. Then we utilize the coarse ray-surface distance regressor to compute the initial intersection ray distance D^{target}. Consequently, we construct a raylet $\boldsymbol{l}$ and calculate the complete ray distance by $D = D^{target} + \Delta D$; the final estimated surface point can be formed as:

$$\Phi(\theta, \phi) = (D\sin\theta\cos\phi, D\cos\theta, D\sin\theta\sin\phi) \tag{13}$$

Since the full network is differentiable, we can calculate the derivative of the surface point with respect to θ, ϕ by:

$$\frac{\partial\Phi}{\partial\phi} = \begin{bmatrix} \left(\frac{\partial D}{\partial\phi}\cos\phi - D\sin\phi\right)\sin\theta \\ \frac{\partial D}{\partial\phi}\cos\theta \\ \left(\frac{\partial D}{\partial\phi}\sin\phi + D\cos\phi\right)\sin\theta \end{bmatrix} \tag{14}$$

$$\frac{\partial\Phi}{\partial\theta} = \begin{bmatrix} \left(\frac{\partial D}{\partial\theta}\sin\theta + D\cos\theta\right)\cos\phi \\ \frac{\partial D}{\partial\theta}\cos\theta - D\sin\theta \\ \left(\frac{\partial D}{\partial\theta}\sin\theta + D\cos\theta\right)\sin\phi \end{bmatrix} \tag{15}$$

Lastly, the formula for a unit normal vector is :

$$\mathbf{n} = \frac{\frac{\partial\Phi}{\partial\phi} \times \frac{\partial\Phi}{\partial\theta}}{\left\|\frac{\partial\Phi}{\partial\phi} \times \frac{\partial\Phi}{\partial\theta}\right\|}. \tag{16}$$

A.4 Details of Training

The VGGT prediction and ground truth are not located in the same coordinate system; thus, we need to align novel view camera pose for supervision. Though the VGGT is expected to predict at a canonical normalized scale during training stage, in inference, it does not perfectly align with it. Directly training the network to predict novel views on data normalized by VGGT preprocessing pipeline would force the network to compensate for the coordinate system mismatch. Thus, we re-calculate the transformation from VGGT prediction to ground truth data during training. Specifically, we first filter out the points whose confidence scores are within the top 85%. Then least-square fit algorithm [7] is utilized to compute the scale, rotation and translation between the VGGT prediction to ground truth data, forming a transformation matrix. Subsequently, we use the transformation matrix to align the ground truth data for supervision.

Our models are trained using the AdamW optimizer and a OneCycle learning rate schedule. Specifically, the coarse regressor is trained for 200 epochs with a learning rate of 1×10^{-4}, while the refinement network is trained for 100 epochs with a learning rate of 1×10^{-4}. In each epoch, we sample 4,200 data points for training. The model is trained on 4 A800 GPUs for about 72 hours.

A.5 Computation Cost Analysis

We report the memory consumption and rendering speed in test for PriorDA, CUT3R, FLARE, RayletDF and ours on ScanNet test split with 50 input views in Table 10. All methods are tested on a single A800 GPU with Intel(R) Xeon(R) Platinum 8358 CPU. Voxel grid KNN `https://github.com/janericlenssen/torch_knnquery/` is employed to accelerate our algorithm. Since our method only needs 1 raylet sampled on each ray and each raylet only has 5 neighboring points sampled, for a fair comparison we also only sample 1 raylet for RayletDF. We can see that: 1) our method has a clear advantage over the depth completion method in rendering speed; 2) though being slower than CUTR3R and FLARE, our method achieves much higher accuracy in surface reconstruction.

Table 10: The time, memory consumption, and the speed of rendering a 640×480 image of ScanNet.

	Memory (GB)	No. of Samples per ray	Render Speed (FPS)	ADE ↓
PiorDA [83]	17.1	-	1.55	0.125
CUT3R [80]	8.9	-	16.7	0.298
FLARE [100]	19.4	-	31.9	0.238
RayletDF (5 samples) [84]	11.2	5×5	6.74	0.136
RayletDF (1 sample) [84]	3.17	1×5	26.2	0.202
Ours	20.89	1×5	8.68	0.089

We further evaluate the computational efficiency of our method under varying image resolutions and numbers of input views. As shown in Tables 11 and 19, our method maintains moderate GPU memory consumption and supports high-resolution rendering and larger input-view sets with manageable computational overhead.

Table 11: The memory consumption and rendering speed for varying image resolutions.

Resolution	Render Speed (FPS)	GPU Memory (GB)
320*240	18.49	20.69
640*480	8.68	20.89
960*720	4.59	21.27
1440*1080	2.13	22.18

A.6 Standard Deviation Results

We report standard deviation results on out-of-domain datasets in Table 12. As shown in the Table, our method achieves the most robust performance.

Table 12: Quantitative results of standard derivation.

	test on → ScanNet++					*test on* → 7Scenes					*test on* → MultiScan				
	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta\uparrow$	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta\uparrow$	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta\uparrow$
Projection [78]	±0.129	±0.170	±0.103	±0.134	±0.127	±0.073	±0.155	±0.032	±0.056	±0.042	±0.287	±0.302	±0.211	±0.327	±0.190
PriorDA [83]	±0.129	±0.170	±0.103	±0.134	±0.127	±0.047	±0.082	±0.183	±0.017	±0.042	±0.268	±0.294	±0.210	±0.319	±0.205
CUT3R [80]	±0.146	±0.190	±0.135	±0.200	±0.200	±0.045	±0.092	±0.025	±0.028	±0.040	±0.209	±0.227	±0.352	±0.493	±0.251
FLARE [100]	±0.334	±0.416	±0.136	±0.293	±0.206	±0.103	±0.174	±0.040	±0.073	±0.090	±0.411	±0.454	±0.273	±0.432	±0.276
RayletDF [84]	±0.145	±0.229	±0.121	±0.385	±0.080	±0.045	±0.092	±0.025	±0.028	±0.040	±0.251	±0.308	±0.276	±0.671	±0.165
Ours *w/o refine*	±0.093	**±0.151**	±0.065	**±0.080**	±0.097	±0.036	**±0.070**	±0.019	**±0.017**	**±0.026**	±0.131	±0.168	±0.179	±0.207	±0.203
Ours	**±0.087**	±0.155	**±0.060**	±0.087	**±0.079**	**±0.033**	±0.073	**±0.018**	±0.018	**±0.026**	**±0.121**	**±0.165**	**±0.158**	**±0.204**	**±0.160**
	test on → Replica					*test on* → MuSHRoom					*test on* → Tanks&Temples				
Projection [78]	±0.146	±0.203	±0.076	±0.154	±0.077	±0.304	±0.422	±0.374	±1.333	±0.165	±1.507	±2.414	±0.219	±5.519	±0.126
PriorDA [83]	±0.046	±0.069	±0.029	±0.021	±0.037	±0.254	±0.338	±0.421	±1.080	±0.204	±1.022	**±1.638**	±0.125	**±1.326**	±0.144
CUT3R [80]	±0.119	±0.140	±0.071	±0.082	±0.129	±0.363	±0.431	±0.641	±1.641	±0.259	±2.990	±3.838	±0.491	±11.83	±0.247
FLARE [100]	±0.124	±0.191	±0.051	±0.063	±0.126	±0.456	±0.576	±0.296	±0.822	±0.254	±2.209	±3.710	±0.144	±2.935	±0.171
RayletDF [84]	±0.033	±0.079	±0.024	±0.026	±0.023	±0.350	±0.504	±0.587	±2.258	±0.177	±1.773	±2.670	±0.343	±10.59	±0.124
Ours *w/o refine*	±0.037	**±0.058**	±0.020	**±0.014**	±0.024	±0.227	**±0.332**	±0.374	**±1.087**	±0.213	±1.132	±2.404	±0.129	±2.658	±0.129
Ours	**±0.028**	±0.062	**±0.016**	**±0.014**	**±0.017**	**±0.213**	±0.354	**±0.359**	±1.228	**±0.166**	**±0.950**	±2.370	**±0.117**	±2.743	**±0.082**

A.7 Ray-surface Distance Metrics

We report ray distance metrics including Absolute Distance Error (ADE), RMSE, Absolute Relative Distance (Abs-Rel), Squared Relative Distance (Sq Rel) and Threshold accuracy ($\delta < t$) to measure the error between the predicted i^{th} ray-surface distance t_i^{pred} and the ground truth t_i^{gt}. These metrics are calculated over N samples and defined as:

Absolute Distance Error (ADE):

$$\frac{1}{N}\sum_{i=1}^{N}|t_i^{pred}-t_i^{gt}|$$

RMSE:

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(t_i^{pred}-t_i^{gt})^2}$$

Absolute Relative Distance (Abs Rel):

$$\frac{1}{N}\sum_{i=1}^{N}\frac{|t_i^{pred}-t_i^{gt}|}{t_i^{gt}}$$

Squared Relative Distance (Sq Rel):

$$\frac{1}{N}\sum_{i=1}^{N}\frac{(t_i^{pred}-t_i^{gt})^2}{t_i^{gt}}$$

Threshold accuracy, δ:

$$\frac{1}{N}\sum_{i=1}^{N}[\max(\frac{t_i^{pred}}{t_i^{gt}},\frac{t_i^{gt}}{t_i^{pred}})<\delta]$$

Absolute Trajectory Error (ATE):

$$\frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{c}_i^{pred}-\mathbf{c}_i^{gt}\right\|_2$$

where $\mathbf{c}_i^{pred},\mathbf{c}_i^{gt}\in\mathbb{R}^3$ denote the predicted and ground-truth 3D camera centers of the i-th frame, respectively.

A.8 Evaluation Metrics on Normals

Given the ground truth normal $\mathbf{n}_{g,i}$ and predicted normal $\mathbf{n}_{p,i}$, per-pixel angle errors are first computed by:

$$\theta_i=\arccos(\mathbf{n}_{p,i}\cdot\mathbf{n}_{g,i})$$

Then metrics reported in our paper are calculated over N samples and defined as following:

Mean Absolute Error (MAE):

$$\text{MAE}=\frac{1}{N}\sum_{i=1}^{N}\theta_i$$

Median Absolute Error (MAE):

$$\text{Median}=\text{median}(\{\theta_i\}_{i=1}^{N})$$

Fraction of pixels where $\theta<t$:

$$\delta<t=\frac{1}{N}\sum_{i=1}^{N}[\theta_i<t]\times 100\%$$

A.9 Evaluation Metrics on Meshes

We uniformly sample points from the predicted and ground-truth meshes, denoted as $\hat{P}$ and P, respectively. The evaluation metrics are defined as follows:

Accuracy:

$$\text{Acc} = \frac{1}{|\hat{P}|} \sum_{\hat{\mathbf{p}} \in \hat{P}} \min_{\mathbf{p} \in P} \|\hat{\mathbf{p}} - \mathbf{p}\|_2.$$

Completeness:

$$\text{Comp} = \frac{1}{|P|} \sum_{\mathbf{p} \in P} \min_{\hat{\mathbf{p}} \in \hat{P}} \|\mathbf{p} - \hat{\mathbf{p}}\|_2.$$

Chamfer-L1:

$$\text{CD}_{L1} = \frac{\text{Acc} + \text{Comp}}{2}.$$

Precision and Recall: For a distance threshold $\tau \in \{0.05, 0.10\}$ m,

$$\text{Precision}_\tau = \frac{1}{|\hat{P}|} \sum_{\hat{\mathbf{p}} \in \hat{P}} \mathbb{I}\left[\min_{\mathbf{p} \in P} \|\hat{\mathbf{p}} - \mathbf{p}\|_2 < \tau\right],$$

$$\text{Recall}_\tau = \frac{1}{|P|} \sum_{\mathbf{p} \in P} \mathbb{I}\left[\min_{\hat{\mathbf{p}} \in \hat{P}} \|\mathbf{p} - \hat{\mathbf{p}}\|_2 < \tau\right].$$

F-score:

$$\text{F1}_\tau = \frac{2\,\text{Precision}_\tau\,\text{Recall}_\tau}{\text{Precision}_\tau + \text{Recall}_\tau}.$$

We report F1@5cm and F1@10cm using $\tau = 0.05$ m and $\tau = 0.10$ m, respectively.

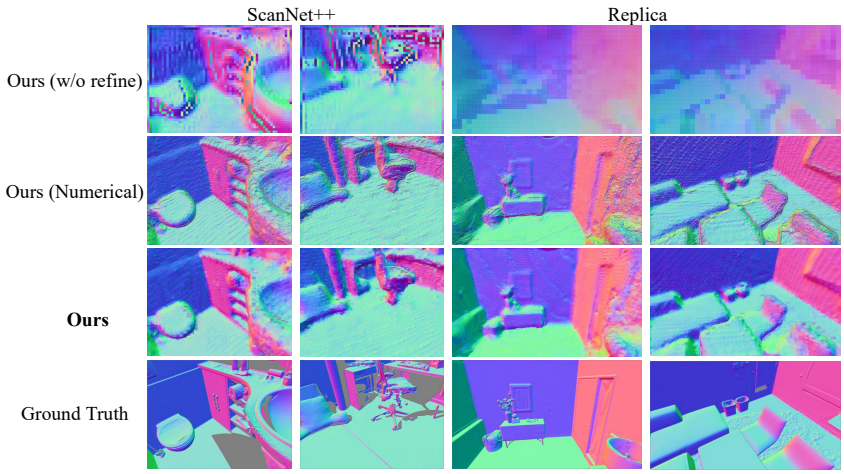

Figure 6: Qualitative results of normal derivation on novel views by different techniques.

A.10 Comparison of Normal Estimation

The qualitative results in Figure 6 show a clear performance ranking. The coarse regressor produces a mosaic-like output lacking geometry details. Numerical differentiation of the refined field yields better but still noisy normals (with characteristic vein-like artifacts) compared to the clean analytical solution, highlighting the limitations of numerical methods.

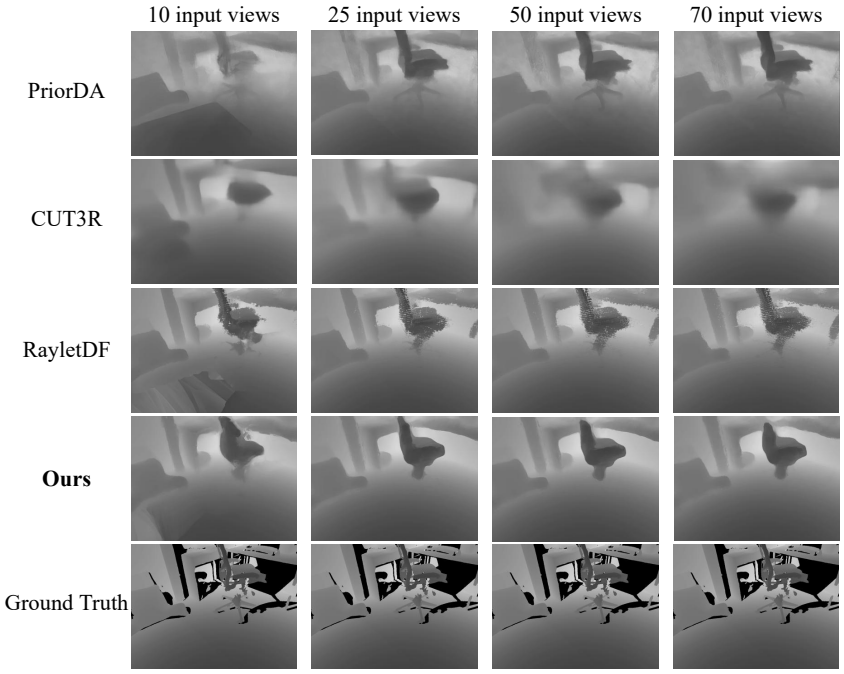

Figure 7: Qualitative results of our method and baselines for 3D surface reconstruction under different input views.

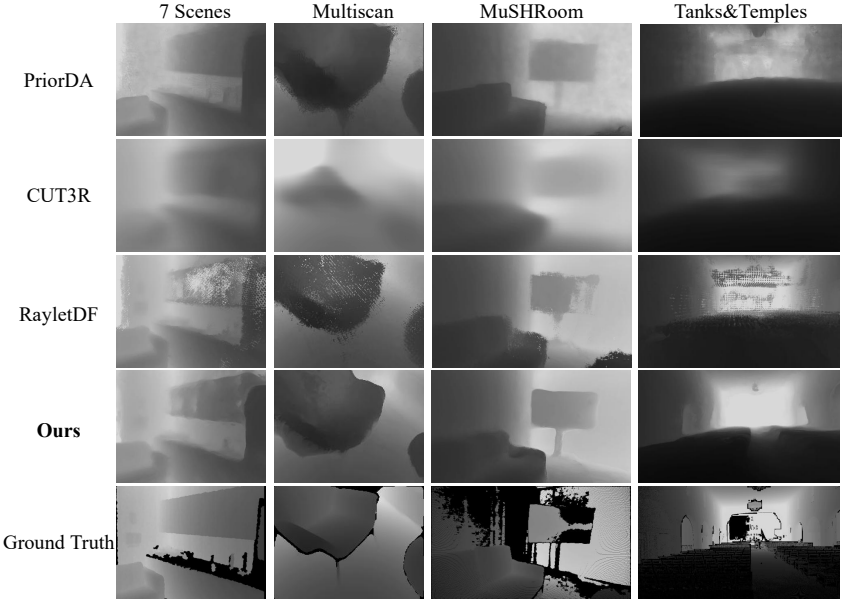

Figure 8: Qualitative results of our method and baselines for 3D surface reconstruction on multiple datasets.

A.11 Qualitative Results of Reconstruction under Different Input Views

Qualitative results under varying numbers of input views are presented in Figure 7. As shown, all methods except CUT3R improve as the number of views increases. Notably, FoundSurface exhibits a clear performance advantage over other baselines in sparse-view settings. In contrast, the predictions of CUT3R deteriorate significantly at 70 views, likely due to information loss caused by its compression mechanism.

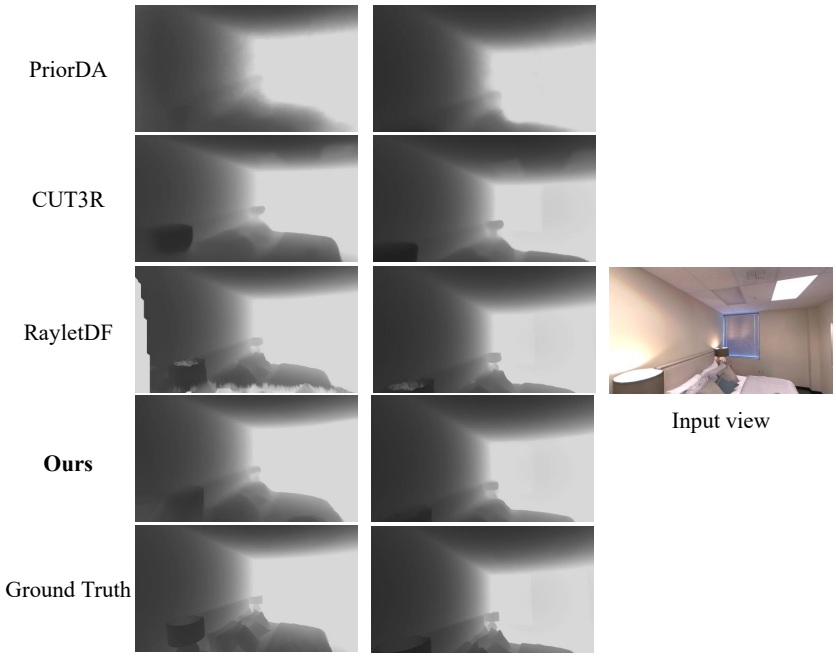

Figure 9: Qualitative results of our method and baselines for single view 3D surface reconstruction on Replica.

A.12 Qualitative Results of Single View Reconstruction

The qualitative results for single-view reconstruction are shown in Figure 9. Notably, FoundSurface again demonstrates superior performance compared to other baselines, underscoring its robustness even under highly sparse input conditions.

A.13 Additional Qualitative Results on Out-of-domain Datasts

We show more qualitative results of out-of-domain datasets in Figure 8.

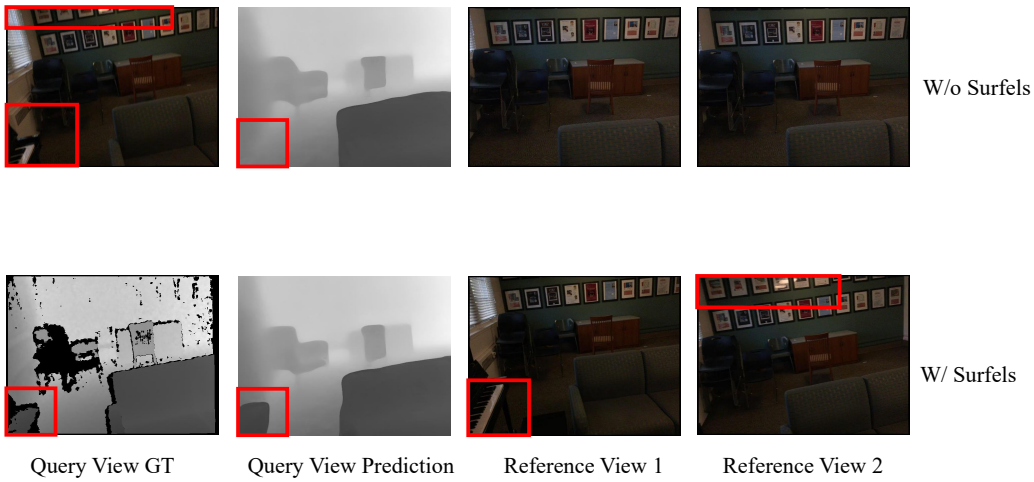

Figure 10: Qualitative comparison of surfel-based reference selection.

A.14 Qualitative Results on Surfel Voting

A case study in Figure 10 illustrates a limitation of per-pixel comparison for view selection (first row): it fails to capture some corner geometry (highlighted in red). In contrast, our surfel-based voting approach (second row) effectively covers these regions, validating its robustness in capturing widespread geometry.

A.15 Additional Quantitative Ablation Experiment Results

We provide ablation results on ScanNet dataset in Table 13 and Table 14. The performance gains from full pipeline ablations on in-domain ScanNet are less pronounced than those on the challenging MuSHRoom dataset. Nevertheless, the results still demonstrate that models incorporating our raylet-based surface module consistently achieve superior performance across most configurations. These findings validate the robust representational capacity of the raylet-based architecture. Notably, the significant sensitivity of the coarse ray-surface regressor's accuracy to these components underscores their critical role in ensuring efficient and precise estimation. As shown in Table 14, on in-domain dataset ScanNet *top-1* or *top-2* is sufficient. We also investigate the *top-k* robustness of our model trained with 2 reference views on MuSHRoom dataset, results are summarized in Table 15. As shown, a large number of reference views is not required.

Table 13: Ablation results for evaluating on the test split of ScanNet.

		ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta \uparrow$
Full	w/o Refine	0.084	0.169	0.044	**0.019**	**0.969**
	w/ Refine	**0.081**	0.171	**0.042**	0.020	**0.969**
w/o surfel vote	w/o Refine	0.086	0.171	0.045	0.020	0.968
	w/ Refine	**0.081**	0.173	0.043	0.020	0.968
w/o local coordinate	w/o Refine	0.120	0.213	0.066	0.031	0.945
	w/ Refine	0.098	0.202	0.053	0.029	0.953
w/o hierarchy feature	w/o Refine	0.090	0.176	0.047	0.021	0.967
	w/ Refine	0.082	0.175	0.043	0.021	0.968
w/o *top-k* reference view	w/o Refine	0.094	0.180	0.049	0.021	0.966
	w/ Refine	**0.081**	**0.170**	**0.042**	**0.019**	**0.969**

Table 14: *Top-k* ablation results for evaluating on the test split of ScanNet.

		ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta \uparrow$
top-1	w/o Refine	0.082	0.169	0.043	**0.019**	**0.969**
	w/ Refine	**0.080**	0.171	0.043	0.020	0.968
top-2	w/o Refine	0.084	0.169	0.044	0.020	**0.969**
	w/ Refine	0.081	0.173	0.043	0.020	0.968
w/o *top-k* reference view	w/o Refine	0.094	0.180	0.049	0.021	0.966
	w/ Refine	0.081	**0.170**	**0.042**	**0.019**	**0.969**

Table 15: Robustness of *top-k* on the test split of MuSHRoom.

	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta \uparrow$
top-1	0.341	0.548	0.272	**0.371**	0.756
top-2	**0.331**	**0.538**	**0.266**	0.372	**0.761**
top-5	0.337	0.540	0.270	0.367	0.753
top-10	0.352	0.555	0.287	0.393	0.738

A.16 Evaluation of In-domain Reconstruction

In this setting, we further evaluate the performance of FoundSurface on in-domain dataset. ScanNet is selected as evaluation dataest because it is within all of CUT3R, VGGT and our training dataset. We also uniformly sample 50 images as input which is the same as the out-of-domain reconstruction

evaluation. The quantitative results are reported in Table 16, our FoundSurface demonstrates clearly better reconstruction accuracy on the in-domain dataset, which is similar to Section 4.1.

Table 16: Qualitative results for in-domain dataset evaluating on the test split of ScanNet.

	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta \uparrow$
Projection [78]	0.247	0.504	0.124	0.139	0.876
PriorDA [83]	0.125	0.240	0.057	0.029	0.937
CUT3R [80]	0.298	0.411	0.167	0.114	0.780
FLARE [100]	0.238	0.435	0.114	0.115	0.855
RayletDF [84]	0.136	0.271	0.071	0.059	0.919
Ours *w/o refine*	0.105	0.195	0.055	**0.025**	0.961
Ours	**0.089**	**0.188**	**0.047**	**0.025**	**0.962**

A.17 Details of Datasets

A.17.1 ScanNet++

We adopt the official test splits for novel view synthesis but exclude scene "1ada7a0617" due to corruptions in its COLMAP data . For each scene, we uniformly sample every 5th frame from the video sequence for training, and reserve every 10th frame in the test set for testing. We use the iPhone sequences with COLMAP registered poses and render mesh scans to obtain high-resolution depth maps. All the images are resized to 960×720.

A.17.2 ScanNet

We use the official training and test splits of ScanNetV2 dataset, with 1201 training scenes and 100 test scenes. Similar to ScanNet++, we also load every 5th frame from the trajectories for training, and then reserve every 10th frame in test set for evaluation.

A.17.3 ARKitScenes

We use the training split of DUSt3R. For training views, we sample every 10th RGB image from the "vga-wide" dataset set and unsample the depth to 640×480 as in DUSt3R.

A.17.4 BlendMVS

All the images of 507 scenes in raw dataset are used in our training dataset. For convenience of sampling during training, we calculate the overlap between each image pair and employ greedy algorithm to rank images in every scenes in video frame sequence order.

A.17.5 HyperSim

We use the "cam_00" trajectory in each scene and all the images are used for training. Similar to BlendMVS images are ranked in video frame sequence order for training convenience.

A.17.6 WildRGBD

Given the extensive scale of the dataset, we select the first ten scenes from each category to construct our training set. Following the preprocessing pipeline in DUSt3R, all images are cropped and resized to a fixed 4:3 aspect ratio to maintain consistency across samples.

A.17.7 MVS-Synth

The official raw dataset, featuring a resolution of 960×540, is employed. All images within the sequences are incorporated into our training framework without further downsampling.

A.17.8 S3DIS

Our training set comprises all areas except for Area 5 and every 10th frame is reserved for validation. Since the raw data is unordered, we employ a custom ranking algorithm to establish a coherent sequence for training.

A.17.9 SceneNN

All scenes in the dataset are utilized for training. To reduce temporal redundancy while maintaining scene coverage, we sample frames from each trajectory at a stride of 10.

A.17.10 GTASfM

We adhere to the official training split of GTASfM. All images within the designated split are used to construct our training dataset.

A.17.11 Multiscan

We use official test split of Multiscan, loading every 50th image from trajectory and then reserve every 10th frame in test set for evaluation.

A.17.12 Replica

We use the pre-processed Replica dataset provided by DN-Splatter [74], which contains 8 scenes with 1200×680 resolution images. Each scene consists of 64 training images and 16 testing images.

A.17.13 7Scenes

Since the original ground truth is imperfect, the same as [77], we adopt COLMAP GT [6] provided that is obtained via global optimization across all sequences.For evaluation, we use the 18-scene split, consistent with the setup in Spann3R [76].

A.17.14 MuSHRoom

We employ the official MuSHRoom split, using the long-sequence RGB-D data captured by Kinect sensors. Every 10th frame is sampled for testing.

A.17.15 Tanks&Temples

Data is sourced from the RMVDB [64] variant, which provides high-quality processed depth maps. For evaluation purposes, we reserve every 10th frame of the sequences as the test set.

A.18 Details of Sparse Convolution and Raylet Distance Network

We simply employ the SparseConv architecture as our Feature extractor. Particularly, we use the existing U-Net implementation of SPconv PyTorch package `https://github.com/traveller59/spconv`. As illustrated in Figure 11, the encoder and decoder have 8 layers respectively, which produce 512-dimensional features. Block means a sequential application of SparseConvolution in encoder or SparseInverseConvolution in decoder, GroupNorm and ReLU activation.

Our Raylet distance network architecture consists of: 1) an initial MLP layer with ReLU that projects input features to 512 dimensions; 2) a block of 8 successive dense layers (512 neurons, ReLU); and 3) a final linear output layer that predicts the raylet distance.

A.19 Robustness Evaluation of the S-nearest Neighbor

To evaluate the robustness of our S-nearest neighbor search against potential outliers, we progressively downsample the pointmaps generated by VGGT to simulate varying levels of data degradation. Specifically, we retain only a certain percentage of the pointmaps (e.g., 80%, 40%) as input to our method. As shown in the following Table 17, our method demonstrates strong robustness against these challenging conditions, essentially because the voxelization process is particularly robust to potential outliers.

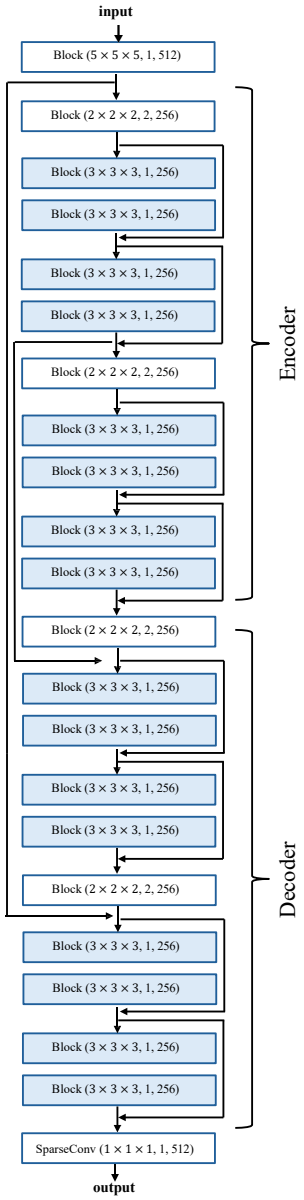

Figure 11: Details of U-Net feature extractor. Block means a sequential application of SparseConvolution in encoder or SparseInverseConvolution in decoder, GroupNorm and ReLU activation.

Table 17: Quantitative results under varying levels of downsampled data prior on ScanNet++.

	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta \uparrow$
100%	0.132	0.259	0.079	0.067	0.910
80%	0.132	0.259	0.079	0.067	0.910
60%	0.132	0.259	0.079	0.067	0.910
40%	0.132	0.259	0.079	0.067	0.910
20%	0.132	0.259	0.079	0.067	0.910
5%	0.133	0.260	0.080	0.068	0.909
1%	0.134	0.261	0.080	0.069	0.907

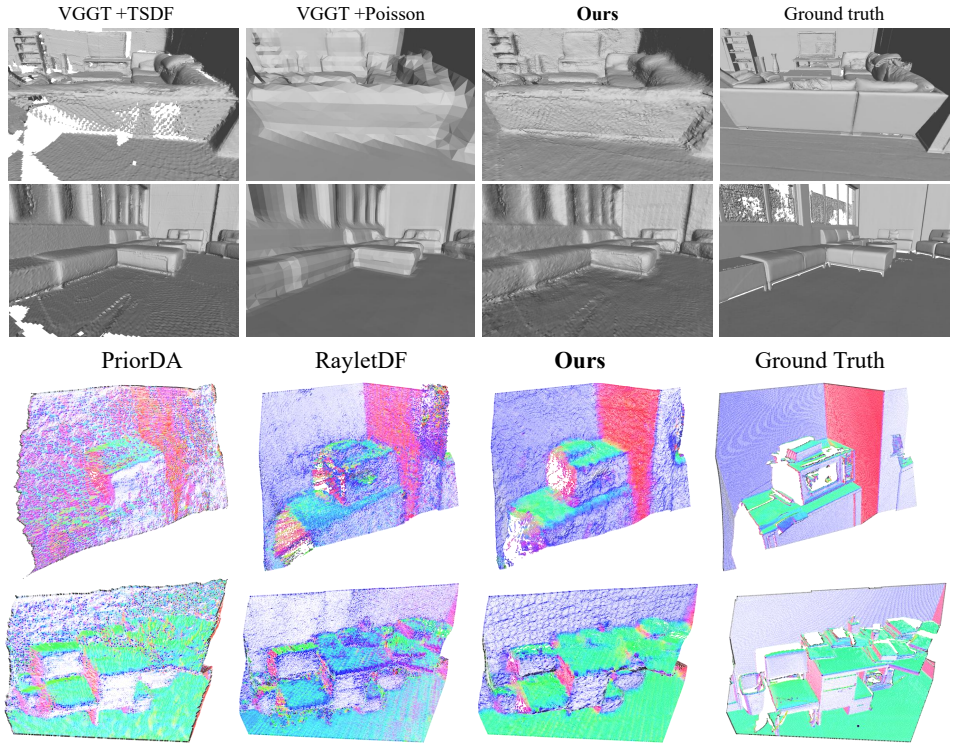

Figure 12: Qualitative 3D results of our method and baselines on ScanNet++. The upper two rows compare 3D meshes, and the bottom two rows compare 3D point clouds.

A.20 Qualitative Results of 3D Outputs

we provide qualitative results of the reconstructed 3D meshes and point clouds in the attached Figure 12.

A.21 Failure Cases

Additionally, we include failure cases in the attached Figure 13. For example, the *mirror* and *transparent glasses* on the wall cause VGGT to generate incorrect pointmaps, ultimately leading to the failure of our method.

A.22 Effectiveness of Raylet-Based Refinement

To quantitatively assess the contribution of our raylet-based refinement module, particularly in high-frequency regions, we applied the Canny edge detector to the RGB images (test views) to categorize all pixels into two groups: 1) edge pixels, representing high-frequency regions, and 2) non-edge pixels, representing low-frequency regions. The following Table 18 reports the performance with and without the raylet-based refinement module across both high- and low-frequency regions on Replica dataset. We can see that our refinement module can consistently improve the surface reconstruction accuracy in both high- and low-frequency regions, demonstrating its general effectiveness.

A.23 Efficiency of Ray-Based Surface Queries

Unlike TSDF, Poisson, and NKSR, which require explicit surface reconstruction before querying target views, our feed-forward ray-based model directly predicts ray-surface distances without explicit mesh construction. As shown in Table 20, our method achieves the lowest target-view query time (0.179 s), demonstrating the efficiency of our direct surface querying approach.

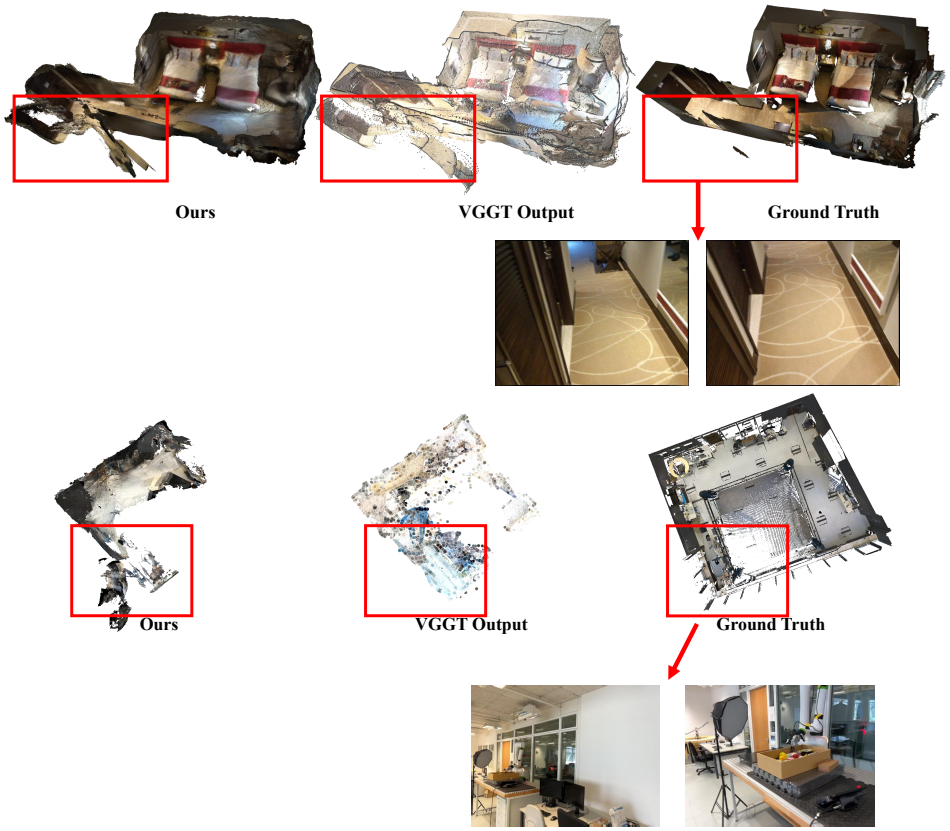

Figure 13: Failure cases of our method. For example, the mirror and transparent glasses on the wall cause inaccurate pointmap estimation by VGGT, which ultimately leads to errors in our approach.

Table 18: Quantitative results of high/low frequency regions on Replica.

		ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta \uparrow$
high freq	w/o refine	0.173	0.283	0.068	0.038	0.949
	w/ refine	0.116	0.258	0.048	0.037	0.951
low freq	w/o refine	0.132	0.202	0.051	0.018	0.980
	w/ refine	0.057	0.145	0.024	0.012	0.985

Table 19: The memory consumption and rendering speed given different numbers of input views.

# of input views	Render Speed (FPS)	GPU Memory (GB)
25	9.29	16.52
50	8.68	20.89
70	8.19	23.60
100	7.68	28.06

A.24 Additional Comparison with Mesh Reconstruction Methods

We further compare our method with conventional mesh reconstruction approaches, including TSDF, Poisson, and NKSR, across six datasets. Specifically, we reconstruct explicit meshes from VGGT-predicted pointmaps and render depth maps at the query views to obtain ray–surface distances. All methods are evaluated using the same metrics as in the main paper. The quantitative results are reported in Table 21.

Table 20: Time required to obtain a target view from the VGGT output.

Method	Time (seconds)↓
VGGT + TSDF [20]	0.216
VGGT + Poisson [42]	55.564
VGGT + NKSR [36]	3.924
Ours	**0.179**

Table 21: Quantitative results of estimated ray-surface distances using meshes reconstructed from VGGT pointmaps.

	test on → ScanNet++					*test on* → 7Scenes					*test on* → MultiScan				
	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta\uparrow$	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta\uparrow$	ADE↓	RMSE↓	Abs-Rel↓	Sq-Rel↓	$\delta\uparrow$
VGGT + TSDF [20]	0.436	0.777	0.235	0.404	0.766	0.228	0.489	0.113	0.135	0.885	0.794	0.986	0.620	0.803	0.381
VGGT + Poisson [42]	0.175	0.327	0.094	0.106	0.881	0.165	0.307	0.081	0.048	0.888	0.384	0.482	0.296	0.276	0.616
VGGT + NKSR [36]	0.446	0.648	0.231	0.298	0.656	0.213	0.371	0.104	0.072	0.832	0.509	0.626	0.379	0.379	0.489
Ours *w/o refine*	0.162	0.270	0.097	0.063	0.894	0.130	**0.230**	0.068	**0.029**	**0.945**	0.233	0.309	0.222	0.147	0.737
Ours	**0.132**	**0.259**	**0.079**	**0.067**	**0.910**	**0.123**	0.233	**0.064**	0.031	0.943	**0.205**	**0.287**	**0.190**	**0.137**	**0.774**
	test on → Replica					*test on* → MuSHRoom					*test on* → Tanks&Temples				
VGGT + TSDF [20]	0.116	0.366	0.050	0.074	0.958	0.606	1.009	0.348	0.652	0.657	1.391	3.577	**0.112**	**1.741**	**0.910**
VGGT + Poisson [42]	**0.063**	0.182	**0.024**	**0.015**	0.979	0.284	0.509	**0.170**	**0.233**	0.795	2.816	4.123	0.186	2.216	0.757
VGGT + NKSR [36]	0.150	0.348	0.052	0.051	0.914	0.908	1.187	0.407	0.709	0.453	5.701	6.879	0.441	4.386	0.379
Ours *w/o refine*	0.137	0.215	0.053	0.021	0.976	0.350	0.511	0.299	0.430	0.734	1.622	3.144	0.183	1.929	0.865
Ours	0.064	**0.165**	0.027	**0.015**	**0.981**	**0.254**	**0.458**	0.232	0.429	**0.822**	**1.233**	**2.977**	**0.144**	2.012	0.908

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 main contribution of our paper has been clearly declared in the abstract and introduction.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We include a limitation section.

   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: We are proposing a comprehensive framework, not including theoretical results.

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: A detailed description of our framework is provided, and the code will be released upon acceptance.

   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]

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We discuss all the experiment details both in the Experiment section and in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We take the experiments with random initialization on several different cases and take the average.

   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: We provide sufficient information on the computer resources in the Appendix.

   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 follow the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We discussed the potential positive impact of our work, as we outlined some of its application in the introduction section of our main text. As for negative social impacts, we have not yet identified significant negative implications.

    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: This paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We have properly cited all the assets we used in the paper.

    Guidelines:

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

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: This paper does not release new assets.

    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 nor research with human subjects.

    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 crowdsourcing nor research with human subjects.

    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: The core method development is totally finished by human, without any reliance on LLMs.

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.