PanoGRF: Generalizable Spherical Radiance Fields for Wide-baseline Panoramas

Zheng Chen[1]*, **Yan-Pei Cao**[2], **Yuan-Chen Guo**[1], **Chen Wang**[1], **Ying Shan**[2], **Song-Hai Zhang**[1]
[1]Tsinghua University [2]ARC Lab, Tencent PCG
China
`{chenz20,guoyc19}@mails.tsinghua.edu.cn` `caoyanpei@gmail.com`
`cw.chenwang@outlook.com` `yingsshan@tencent.com` `shz@tsinghua.edu.cn`

Abstract

Achieving an immersive experience enabling users to explore virtual environments with six degrees of freedom (6DoF) is essential for various applications such as virtual reality (VR). Wide-baseline panoramas are commonly used in these applications to reduce network bandwidth and storage requirements. However, synthesizing novel views from these panoramas remains a key challenge. Although existing neural radiance field methods can produce photorealistic views under narrow-baseline and dense image captures, they tend to overfit the training views when dealing with *wide-baseline* panoramas due to the difficulty in learning accurate geometry from sparse 360° views. To address this problem, we propose PanoGRF, Generalizable Spherical Radiance Fields for Wide-baseline Panoramas, which construct spherical radiance fields incorporating 360° scene priors. Unlike generalizable radiance fields trained on perspective images, PanoGRF avoids the information loss from panorama-to-perspective conversion and directly aggregates geometry and appearance features of 3D sample points from each panoramic view based on spherical projection. Moreover, as some regions of the panorama are only visible from one view while invisible from others under wide baseline settings, PanoGRF incorporates 360° monocular depth priors into spherical depth estimation to improve the geometry features. Experimental results on multiple panoramic datasets demonstrate that PanoGRF significantly outperforms state-of-the-art generalizable view synthesis methods for wide-baseline panoramas (e.g., OmniSyn) and perspective images (e.g., IBRNet, NeuRay).

1 Introduction

The rise of 360° cameras and virtual reality headsets has fueled the popularity of 360° images among photographers and tourists. Commercial VR platforms, such as Matterport [2], enable users to experience virtual walks in 360° scenes by interpolating between panoramas [16]. Wide-baseline panoramas are frequently employed on these platforms for capture and network transmission to reduce storage space and bandwidth requirements. Consequently, synthesizing novel views from wide-baseline panoramas is an essential task for providing a seamless six degrees of freedom (6DoF) experience to users.

However, synthesizing novel views from a pair of wide baseline panoramas encounters two primary challenges. *First*, the input views are sparse, causing existing state-of-the-art methods such as NeRF [19] to struggle in learning accurate geometry due to the shape-radiance ambiguity [34], leading to overfitting of the input views. *Second*, certain regions in the scene may only be visible

*work done during internship at ARC Lab, Tencent PCG
[2]Matterport: https://matterport.com

Preprint. Under review.

in one view but not in another, making it challenging to provide a correct geometry prior to NeRF using only multi-view stereo. State-of-the-art generalizable NeRF methods [32, 29, 18] address the overfitting problem by incorporating scene priors into NeRF; however, they are designed for perspective images and require conversion of panoramas into perspective views when applied, resulting in information loss and suboptimal performance due to the limited field of view. Although spherical radiance fields [10] have been proposed to render panoramic images based on spherical projection, they still suffer from the overfitting problem without considering the 360° scene prior. Other approaches designed for 360° view synthesis, such as multi-sphere images (MSI) [1] and mesh-based methods [16], exhibit limited expressiveness and rendering quality compared to NeRF due to the use of discretized scene representations.

In this work, we present a method that addresses the overfitting issues in spherical radiance fields by incorporating 360° scene priors. Unlike existing generalizable methods that require panorama-to-perspective conversion, our approach retains the panoramic representation. Furthermore, we incorporate 360° monocular depth to alleviate the view occlusion problem.

To address the first challenge, we present a solution in the form of *generalizable spherical radiance fields*. To render the panorama at a new position, spherical radiance fields (Spherical NeRF) cast a ray for each pixel using spherical projection, sample points along the ray, and aggregate the colors of these points based on their density values following NeRF [19]. In our method, named PanoGRF, we incorporate 360° scene priors into Spherical NeRF. Specifically, we extract appearance and geometry features from input panoramas and estimated spherical depths through convolutions, respectively. PanoGRF accurately aligns the queried 3D points with the corresponding pixels in panoramas using spherical projection. This alignment strategy leverages the full field-of-view characteristic of panoramas and eliminates the information loss of panorama-to-perspective conversion. The local appearance and geometry features at these corresponding pixels are then aggregated and serve as the conditional input for Spherical NeRF.

To tackle the second challenge, we enhance the accuracy of depth estimation in 360° multi-view stereo by integrating a 360° monocular depth estimation network. In 360° multi-view stereo, depth inaccuracies can arise due to view inconsistencies caused by occluded regions. Inspired by the perspective method proposed in [2], we sample depth candidates around the estimated 360° monocular depth, assuming a Gaussian distribution. These depth candidates are then utilized in sphere sweeps during 360° multi-view matching. By incorporating monocular depth candidates, we can improve the accuracy of depth estimation in regions with view inconsistencies. This improvement contributes to the robustness of the geometry features employed in PanoGRF and ultimately leads to superior view synthesis performance.

In summary, we make the following contributions: 1) We propose the design of generalizable spherical radiance fields to learn 360° scene priors that alleviate the overfitting problem when dealing with wide-baseline panoramas. 2) We incorporate 360° monocular depth into the spherical multi-view stereo to mitigate the view inconsistency problem caused by occluded regions, which results in more robust geometry features, ultimately improving the rendering performance. 3) We achieve the state-of-the-art novel view synthesis performance on Matterport3D [3], Replica [26] and Residential [7] under the wide-baseline setting.

2 Related Work

In this section, we briefly review the methods of perspective view synthesis and 360° view synthesis.

2.1 Perspective View Synthesis

Recently, NeRF (neural radiance fields [19]) has become the mainstream scene representation due to its photo-realistic rendering quality for perspective view synthesis. However, given sparse input views, NeRF's per-scene optimization approach is prone to overfitting the training views and fails to learn the correct geometry due to the shape-radiance ambiguity, leading to significant floaters in novel views. Several subsequent methods [11, 5, 14, 20, 4, 31] have attempted to address this issue by incorporating depth constraints from COLMAP [24] or other regularization terms, but they still rely on per-scene optimization and lack the ability to generalize to new scenes. In contrast, PixelNeRF [32] and IBRNet [29] condition NeRF rendering with pixel-aligned appearance features to learn scene

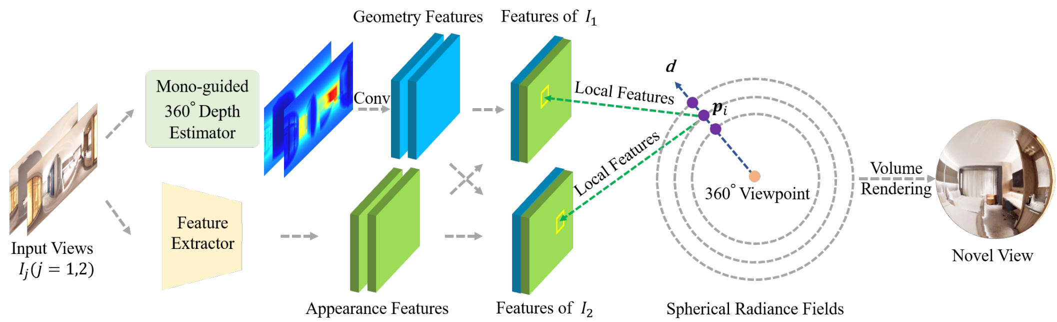

Figure 1: The overview of PanoGRF. Initially, we employ convolutional neural networks (CNNs) to extract appearance features and geometry features from the input views. The geometry feature is obtained from the predicted spherical depth generated by our proposed Mono-guided 360° Depth Estimator (Sec. 3.3). Next, we cast rays based on spherical projection to render a novel 360° view (Sec. 3.1). Along each ray, every 3D sample point $\boldsymbol{p}_i$ is projected onto the corresponding panoramic grid coordinate (u, v). The local geometry feature $\boldsymbol{g}_{i,j}$ at (u, v) is decoded into $v_{i,j}$ (the visible probability of $\boldsymbol{p}_i$ for the j-th view). The local appearance feature $\boldsymbol{f}_{i,j}$ at (u, v) is aggregated into $\boldsymbol{f}_i$ along with $v_{i,j}$ (Sec. 3.2). Subsequently, the aggregated feature $\boldsymbol{f}_i$ is decoded to determine the color and density of $\boldsymbol{p}_i$. Finally, a novel 360° view is synthesized through volume rendering (Sec. 3.1).

priors from large-scale datasets, enabling direct inference with input images. NeuRay [18] aggregates visibility information (obtained from multi-view stereo) and appearance features from each view to generate the density and color of 3D sample points. However, when dealing with panoramas, these methods require converting the panoramas to perspective images, leading to information loss and suboptimal performance due to limited field-of-view in each perspective view. In contrast, PanoGRF operates directly on panoramas using spherical projection, eliminating the need for the panorama-to-perspective conversion.

2.2 360° View Synthesis

Multi-sphere images (MSI) are widely adopted for 360° view synthesis. It is introduced by [1] to represent scenes with the input of omnidirectional stereo video. SOMSI [7] improved the expressive ability of MSI by incorporating high-dimensional feature layers and achieved real-time performance. However, MSI's capability is limited to narrow-baseline scenarios due to the back surface rendering issues as discussed in [17], which attempted to expand the renderable viewpoint range of MSI through the interpolation of two MSI instances. But their pure convolutional architecture only works at low resolutions. OmniSyn [16] constructs two meshes for input panoramas using 360° multi-view stereo. It warps the meshes to the target view and fuses the colors attached to the meshes with an in-painting network. This approach frequently produces ghosting artifacts due to surface-based warping and blending. 360° Roam [10] divides NeRF into multiple blocks to achieve real-time rendering of panoramas, akin to KiloNeRF [21]. But it fits a single scene without considering scene priors, making it unsuitable for wide-baseline panoramas. In contrast to the aforementioned methods, we introduce generalizable spherical radiance fields to learn both geometric and appearance priors from 360° datasets, enabling better generalization to unseen 360° scenes. Furthermore, PanoGRF leverages a 360° monocular depth network to guide the construction of 360° cost volume, which enhances the quality of geometry features and improves the rendering performance.

3 Method

PanoGRF facilitates novel view synthesis from wide-baseline panoramas as shown in Fig. 1. Before introducing PanoGRF, we first review NeRF and its spherical variant in Sec. 3.1. We demonstrate the two important parts of PanoGRF, the generalizable spherical radiance fields, and the mono-guided spherical depth estimator in Sec. 3.2 and Sec. 3.3 respectively.

3.1 Spherical Radiance Fields (Spherical NeRF)

To render a pixel, NeRF [19] casts a ray from a given viewpoint and parameterizes the ray as $\boldsymbol{p}(t) = \boldsymbol{o} + t\boldsymbol{d}, t \in \mathbb{R}^{+}$, where $\boldsymbol{o}$ is the camera center and $\boldsymbol{d}$ is the ray direction. N points are sampled along the ray with increasing t, we denote $\boldsymbol{p}_i \equiv \boldsymbol{p}(t_i), i = 1, ..., N$. NeRF utilizes an MLP to map the position $\boldsymbol{p}_i$ together with viewing direction $\boldsymbol{d}$ into color $\boldsymbol{c}_i$ and density σ_i. The pixel color of the ray $\boldsymbol{p}$ can then be computed by:

$$\boldsymbol{c} = \sum_{i=1}^{N} w_i \boldsymbol{c}_i, \tag{1}$$

where $\boldsymbol{c} \in \mathbb{R}^3$ is the rendered color of $\boldsymbol{p}$, and $\boldsymbol{c}_i \in \mathbb{R}^3$ is the color of the sampled point $\boldsymbol{p}_i$. w_i is the blending weight of the point $\boldsymbol{p}_i$, which is the accumulated transmittance that indicates the probability that a ray travels from the origin to t_i without hitting any particle. It is computed by $w_i = \prod_{k=1}^{i-1}(1 - \alpha_k)\alpha_i$, where α_i is the alpha values in the depth range (t_i, t_{i+1}), $\alpha_i = 1 - \exp(-\sigma_i \delta_i)$ and $\delta_i = t_{i+1} - t_i$. The color $\boldsymbol{c}$ of a ray $\boldsymbol{p}$ is optimized by a photometric loss:

$$\mathcal{L} = \sum \|\boldsymbol{c} - \boldsymbol{c}_{gt}\|^2, \tag{2}$$

where $\boldsymbol{c}_{gt}$ represents the ground truth color.

Ray-casting based on spherical projection The panorama uses the panoramic pixel grid coordinate system, while the coordinate systems in NeRF are all Cartesian. By employing *spherical projection* [7, 10], a pixel (u, v) in the panorama is firstly transformed into spherical polar coordinate (ϕ, θ) and subsequently into cartesian coordinate (x, y, z). Detailed transformation formulas can be seen in the supplementary material. The ray direction emitted from the pixel (u, v) can be computed by $\boldsymbol{d} = \mathbf{R}[x, y, z]^T$, where $\mathbf{R} \in \mathbb{R}^{3\times3}$ is the panorama's camera-to-world rotation matrix.

3.2 Generalizable Spherical Radiance Fields

Under a wide baseline, Spherical NeRF tends to overfit training views and struggle to generate plausible novel views. To address this limitation, we draw inspiration from the generalizable NeRF approaches [32, 29, 18] designed for perspective images. In our work, we propose to incorporate $360°$ scene priors into Spherical NeRF by aggregating local features from input panoramas.

Alignment based on spherical projection Panoramas can be converted into perspective images, and then generalizable NeRF methods for perspective images can be applied to novel $360°$ view synthesis. However, When projecting the 3D sample point $\boldsymbol{p}_i$ onto the source perspective view, it may fall outside the image borders or even be located behind the source perspective camera (with a z-depth < 0) due to the limited field-of-view. This can introduce errors in the aggregation of features and result in poor rendering results. To overcome this problem, we directly align $\boldsymbol{p}_i$ with the corresponding pixels in panoramas using spherical projection, enabling us to leverage the full field-of-view characteristic of panoramas. We first transform $\boldsymbol{p}_i$ into the camera's Cartesian coordinate system (x, y, z) of the j-th panoramic view I_j. Subsequently, (x, y, z) is converted into spherical polar coordinate (θ, ϕ, t) ($t \in \mathbb{R}^{+}$ indicates the spherical depth of $\boldsymbol{p}_i$ for I_j) and finally transformed into the panoramic grid coordinate (u, v). The specific formulas for these transformations can be found in the supplementary material.

Appearance and geometry feature aggregation We follow NeuRay [18] to aggregate appearance and geometry features. But differently, we take panoramas and spherical depth (radial distance from the camera center) as input instead of perspective images and z-depth. PanoGRF respectively extracts appearance feature $\mathbf{W}_j$ from the j-th panoramic view ($j = 1, 2$) and geometry feature $\mathbf{G}_j$ from the spherical depth using convolutions. Details for getting the spherical depth will be introduced in Sec. 3.3. For a given 3D sample point $\boldsymbol{p}_i$, we project it onto the panoramic grid coordinate (u, v), and extract the local appearance feature $\boldsymbol{f}_{i,j} = \mathbf{W}_j(u, v)$ and the local geometry feature $\boldsymbol{g}_{i,j} = \mathbf{G}_j(u, v)$ at (u, v). The local geometry feature $\boldsymbol{g}_{i,j}$ is decoded into the visibility $v_{i,j}$ (demonstrated below) using an MLP. The local appearance feature $\boldsymbol{f}_{i,j}$ and visibility $v_{i,j}$ of the j-th panoramic view are aggregated for $\boldsymbol{p}_i$ with an aggregation network $\mathcal{F}$:

$$\boldsymbol{f}_i = \mathcal{F}(\{\boldsymbol{f}_{i,j}, v_{i,j} | j = 1, 2\}). \tag{3}$$

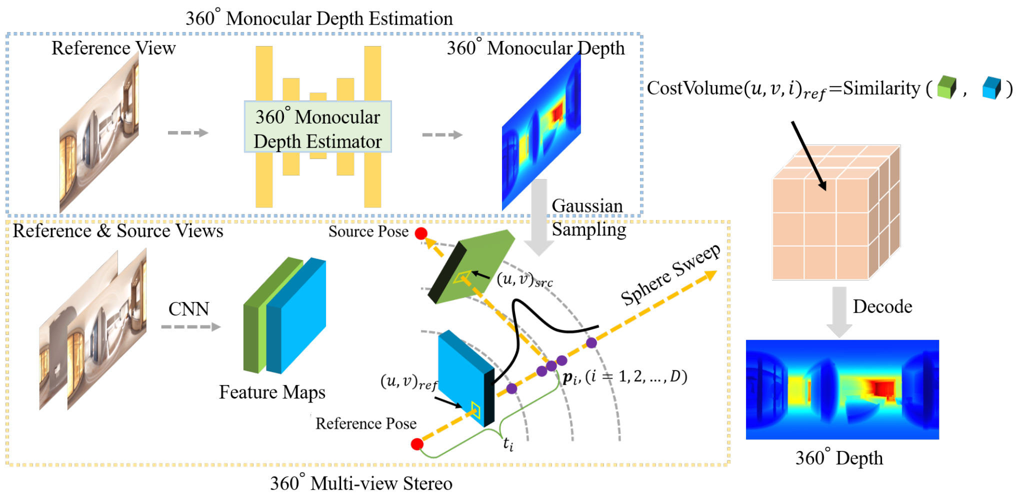

Figure 2: The process of mono-guided 360° depth estimation. We first extract the image features of reference and source views with convolutions. Using spherical projection, we determine the corresponding pixel $(u, v)_{src}$ in the source view for each pixel $(u, v)_{ref}$ of the reference view, with the depth hypothesis of t_i. The similarity between the local features at $(u, v)_{ref}$ and $(u, v)_{src}$ is computed as the value of the cost volume at (u, v, i). Depth sampling of t_i is guided by the 360° monocular depth using a Gaussian distribution assumption. The cost volume is obtained after D sphere sweeps. Lastly, the cost volume is decoded into 360° depth using convolutions.

According to the local geometry feature $\boldsymbol{g}$ of an input panorama, we predict the visibility function $v(t)$ to indicate the probability that a point at spherical depth t (instead of z-depth in NeuRay) is visible for the input panorama. $v(t)$ ($v(t) \in [0, 1]$) is represented as $v(t) = 1 - o(t)$, where $o(t)$ is the occlusion probability. To parameterize $o(t)$, we employ a mixture of N_l logistic distributions:

$$o(t; \mu_k, \sigma_k, w_k) = \sum_{k}^{N_l} w_k S((t - \mu_k)/\sigma_k), \tag{4}$$

where μ_k, σ_k and w_k are the mean, standard variance, and blending weight of the k-th logistic distribution respectively. $\sum_i^{N_l} w_i = 1$ and $S(\cdot)$ denotes a sigmoid function. The parameters $[\mu_i, \sigma_i, w_i]$ are decoded from the local geometry feature vector $\boldsymbol{g}$ by an MLP. For each 3D sample point $\boldsymbol{p}_i$, we compute its spherical depth t_i for j-th input view based on spherical projection and obtain the visible probability $v_{i,j} = v_j(t_i)$ of $\boldsymbol{p}_i$ for the j-th panoramic view. Then we aggregate $v_{i,j}$ into $\boldsymbol{f}_i$ by Eq. 3 and decode $\boldsymbol{f}_i$ into the color and density of $\boldsymbol{p}_i$.

3.3 Mono-guided Spherical Depth Estimator

To obtain the visibility in Sec. 3.2, we need to predict the spherical depth of each input view. The process is illustrated in Fig. 2.

360° **multi-view stereo (MVS)** With the input of multi-view panoramas, we estimate the spherical depth of the reference view using 360° multi-view stereo similar to previous methods [16, 13]. Firstly, we extract the image features of the reference and source views with an image encoder. For each pixel $(u, v)_{ref}$ of the reference view, assuming its depth is t_i, we find its corresponding pixel $(u, v)_{src}$ in the source view according to spherical projection. We uniformly sample D depth candidates for t_i (with $i = 1, 2, ..., D$) without considering the monocular depth prior, which will be introduced later. Next, we compute the similarity between the local features at $(u, v)_{ref}$ and $(u, v)_{src}$, considering it as the value of the cost volume at (u, v, i). Assuming the length of the local feature vector is F, this process results in a 4D cost volume $\mathbf{V} \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4} \times D \times F}$ after D sphere sweeps, where $\frac{H}{4}$ and $\frac{W}{4}$ respectively represent the height and width of the feature maps for the reference view. Subsequently, the dimension of the cost volume is reduced to $\frac{H}{4} \times \frac{W}{4} \times D$ through a 3D CNN. Lastly, the processed cost volume is decoded into 360° depth using several convolution layers.

Mono-guided spherical depth sampling Under the wide-baseline setting, some areas might be visible from one view but occluded from another, in which case 360° MVS may struggle to produce accurate depth estimates. To address this challenge, we leverage the 360° monocular depth [12] to guide the depth sampling of 360° MVS, inspired by perspective methods [33, 28, 2].

For each pixel (u, v) in the panorama, we denote its estimated spherical depth from the monocular depth network as $\mu_{u,v}$. We generate depth candidates in the vicinity of the monocular depth using a Gaussian distribution assumption. Specifically, a search space for each pixel in the panorama is defined as $[\mu_{u,v} - \beta\sigma, \mu_{u,v} + \beta\sigma]$ where β and σ serve as hyperparameters and σ represents the standard deviation. This search space is divided into N_{mono} bins, ensuring that each bin has the same probability mass. We select the midpoint of each bin as the depth candidate. Consequently, the k-th depth candidate for pixel (u, v) is defined as $t_{u,v,k} = \mu_{u,v} + b_k\sigma, k = 1, 2, ..., N_{mono}$. b_k represents the offset corresponding to the k-th bin. Similar to [2], we calculate b_k as the following:

$$b_k = \frac{1}{2}\left[\Phi^{-1}(\frac{k-1}{N_{mono}}P^* + \frac{1-P^*}{2}) + \Phi^{-1}(\frac{k}{N_{mono}}P^* + \frac{1-P^*}{2})\right], \tag{5}$$

where $P^* = \mathrm{erf}(\frac{\beta}{\sqrt{2}})$ denotes the probability mass covered by the search space $[\mu_{u,v} - \beta\sigma, \mu_{u,v} + \beta\sigma]$, $\Phi^{-1}(\cdot)$ is the quantile function of standard normal distribution, erf is the error function. Considering errors may occur in the 360° monocular depth, we use a mixture of N_{mono} monocular depth candidates and N_{uni} uniform depth candidates sampled from a uniform distribution. By constructing $D = N_{uni} + N_{mono}$ sphere sweeps, we obtain a new spherical cost volume. The monocular depth guidance enables us to extract more reliable geometry features employed in PanoGRF.

4 Experiment

4.1 Metrics and Datasets

PSNR, SSIM [9], LPIPS [35] and WS-PSNR [27] are used as evaluation metrics. We conduct experiments on Matterport3D [3], Replica [26], and Residential [7]. For Matterport3D and Replica, we leverage HabitatAPI [23] to generate the 256×256 perspective images (representing the six sides of cube-maps) and stitch them into a 512×1024 panorama. We use panoramic sequences with a length of 3. The middle view is rendered based on the first and last views. We conduct comparative experiments on Matterport3D under fixed camera baselines of 1.0, 1.5, and 2.0 meters, where the camera baseline refers to the distance between the camera centers of the first and last views. The baselines used for Residential and Replica are approximately 0.3 and 1.0 meters, respectively.

4.2 Implementation Details

We set $N_{mono} = 5$, $N_{uni} = 59$, and $N_l = 2$. $D = N_{mono} + N_{uni}$ is 64. σ used in mono-guided depth sampling is set to 0.5 and β is set to 3. Additional details regarding network architecture can be found in the supplementary material.

4.3 Comparisons

We compared PanoGRF with Spherical NeRF (S-NeRF) [19], IBRNet [29], NeuRay [18], and OmniSyn [16]. We trained S-NeRF from scratch, as it is a per-scene optimization method. Other methods are pre-trained on Matterport3D and tested on unseen testing scenes. We fed the original cube maps of panoramas (in Matterport3D and Replica) to the perspective methods (IBRNet and NeuRay). As there are only panoramas in ERP (equirectangular projection) format for Residential, cube maps were converted from panoramas. To render a panoramic view for evaluation, we cast rays in IBRNet and NeuRay with spherical projection (refer to Sec. 3.1) and aggregate features with their original perspective projection. We use NeuRay* and IBRNet* to denote variants of NeuRay and IBRNet which render panoramas. We did not compare with methods based on multi-sphere images (MSI) [1, 7] as they can only render novel views within the smallest sphere of MSI, which are unsuitable for wide-baseline panoramas. In the supplementary materials, we provide additional analysis on per-scene fine-tuning of PanoGRF and also compare it with Cross Attention Renderer [6].

Analysis On Matterport3D and Replica, we quantitatively compared PanoGRF with the baseline methods. The results can be found in Table. 1 and Table. 2. Almost all the metrics show that PanoGRF

Table 1: Quantitative comparisons with baseline methods on Matterport3D. The best results are in bold.

baseline	1.0m			1.5m			2.0m		
method	WS-PSNR↑	SSIM↑	LPIPS↓	WS-PSNR↑	SSIM↑	LPIPS↓	WS-PSNR↑	SSIM↑	LPIPS↓
S-NeRF	15.25	0.579	0.546	14.16	0.563	0.580	13.13	0.523	0.607
OmniSyn	22.90	0.850	0.244	20.31	0.790	0.317	18.91	0.761	0.354
IBRNet*	25.72	0.855	0.258	21.69	0.751	0.382	20.04	0.706	0.431
NeuRay*	24.92	0.832	0.260	21.92	0.766	0.347	19.85	0.715	0.407
PanoGRF	**27.12**	**0.876**	**0.195**	**23.38**	**0.811**	**0.282**	**20.96**	**0.761**	**0.352**

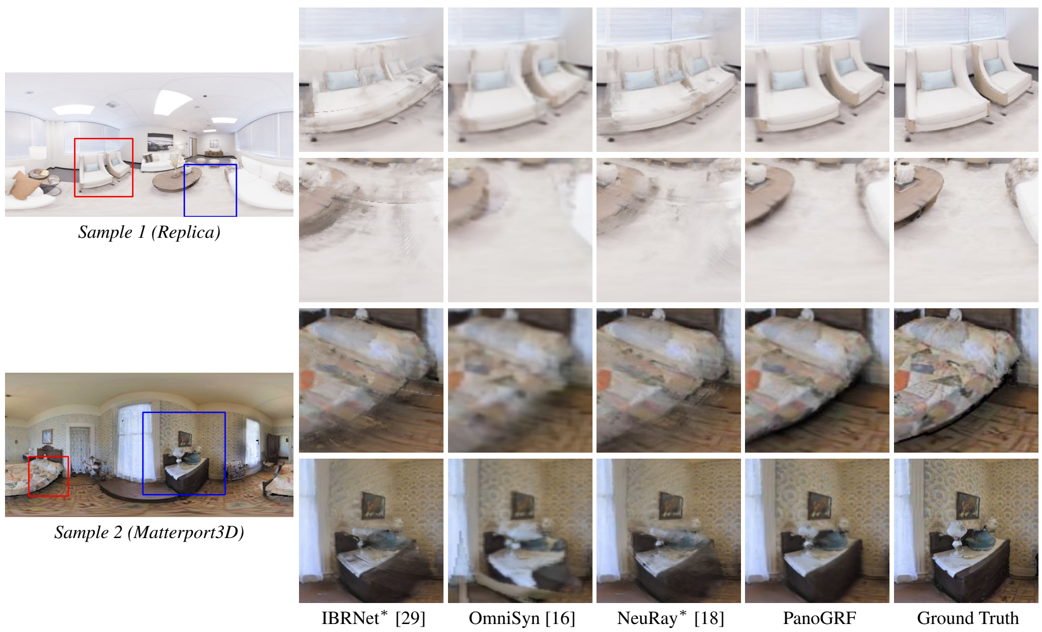

Figure 3: Qualitative comparison on Replica and Matterport3D between IBRNet*, OmniSyn, NeuRay* and PanoGRF.

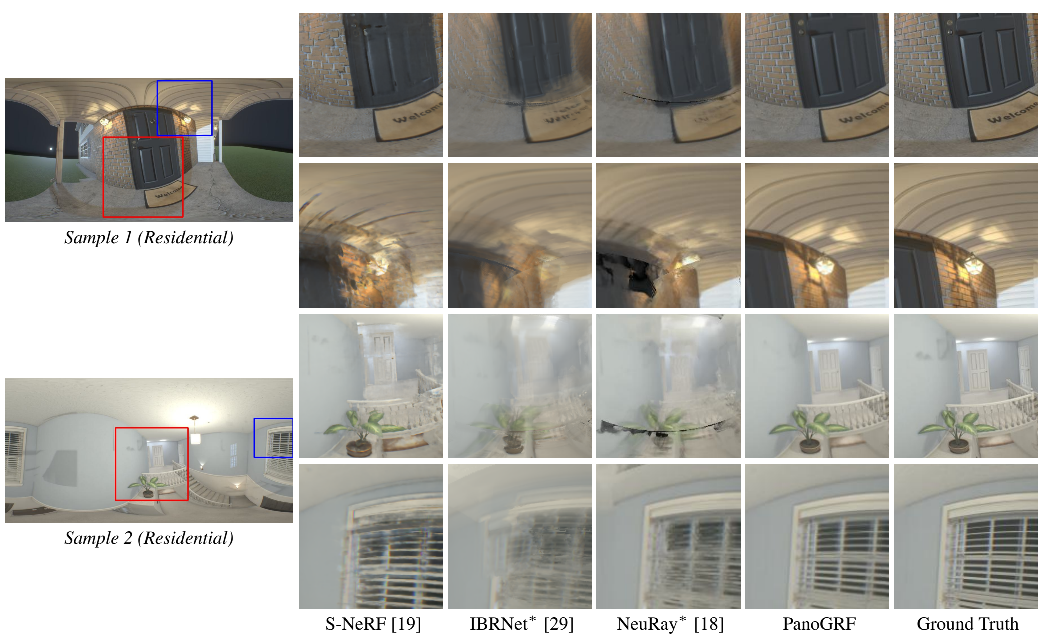

Figure 4: Qualitative comparison on Residential between S-NeRF, IBRNet*, NeuRay* and PanoGRF.

outperforms other methods significantly. We show the qualitative comparisons in Fig. 3. IBRNet* and NeuRay* aggregate features based on perspective projection. Due to the limited field of view, 3D sample points are often projected somewhere behind the source perspective camera (z-depth<0) or outside the image border during pixel alignment. In this case, the aggregated features are incorrect, causing bad rendering results. As for OmniSyn, its rendering outputs exhibit ghosting artifacts, particularly notable at the boundaries of the two sofas in Sample 1 of Fig. 3. Unlike these methods, PanoGRF is a generalizable spherical NeRF which is more appropriate for panoramic images due to the use of spherical coordinates. It achieves superior accuracy in synthesizing object boundaries and demonstrates better pixel alignment, as evident from the results showcasing the sofas in Sample 1 and the desk in Sample 2 of Fig. 3). On Residential, we also compared PanoGRF with S-NeRF, IBRNet*, NeuRay*. The results of S-NeRF show severe floaters (See the ceilings of Sample 1 in Fig. 4). As converting panoramas into perspective views brings information loss, the perspective generalization methods IBRNet* and NeuRay* have notable artifacts. In contrast, PanoGRF demonstrates excellent generalization performance on the Residential dataset, as shown in Fig. 4. Additional qualitative comparisons can be seen in the supplementary material.

Table 2: Quantitative comparison with baseline methods on Replica and Residential Dataset

Dataset	Replica (1.0m)				Residential (about 0.3m)			
method	PSNR↑	WS-PSNR↑	SSIM↑	LPIPS↓	PSNR↑	WS-PSNR↑	SSIM↑	LPIPS↓
S-NeRF	16.91	16.10	0.723	0.443	23.06	22.47	0.741	0.435
OmniSyn	23.91	23.17	0.898	0.189	-	-	-	-
IBRNet*	23.35	22.65	0.854	0.291	22.95	22.47	0.735	0.498
NeuRay*	26.62	25.90	0.899	0.187	23.01	22.38	0.753	0.427
PanoGRF	**30.08**	**29.22**	**0.937**	**0.134**	**31.64**	**31.03**	**0.909**	**0.207**

4.4 Ablation Studies

We conducted ablation studies to evaluate the effectiveness of the key components in mono-guided spherical depth estimator: 360° monocular depth and multi-view stereo. We provide qualitative comparisons of 360° depth estimation quality on Matterport3D in Fig. 5, and analyze the view synthesis quality on Replica in Table. 3. More results can be found in the supplementary material.

360° **monocular depth** We removed the 360° monocular depth guidance and kept the total number of depth candidates unchanged ($N_{uni} = 64$) in the depth sampling. Under three camera baselines, all the metrics dropped after removing the 360° monocular depth. Especially, under the camera baseline of 2.0 meters, WS-PSNR dropped by 1.33dB without monocular depth. This removal resulted in error-prone object boundaries, such as the bedside and wooden pillar (shown in Fig. 6). The guidance of monocular depth can give more accurate depth candidates and mitigates the view inconsistency problem, which 360° multi-view stereo cannot handle alone.

360° **multi-view stereo** We removed 360° multi-view stereo and used 360° monocular depth directly as the input of the geometry feature extraction in PanoGRF. In this case, WS-PSNR dropped more than 1.0 dB under the camera baselines of 1.5 and 2.0 meters. And LPIPS became 0.165 from 0.134 under the baseline of 1.0 meters. Without multi-view stereo, multi-view consistency of geometry is not guaranteed, degrading the quality of novel view synthesis results. In Fig. 6, we observed the presence of ghosting artifacts near the left wooden pillar when relying solely on 360° monocular depth.

5 Conclusion

In this paper, we propose PanoGRF, generalizable spherical radiance fields for wide-baseline panoramas. We aggregate the appearance and geometry features from input panoramas through spherical projection to avoid panorama-to-perspective conversion. To address the view inconsistency problem, we use 360° monocular depth to guide the spherical depth sampling and obtain a more accurate geometric prior for the spherical radiance fields. Experiments on multiple datasets verify that PanoGRF can render high-quality novel views given wide-baseline panoramas.

Table 3: Ablation studies on Replica

baseline	1.0m			1.5m			2.0m		
method	WS-PSNR↑	SSIM↑	LPIPS ↓	WS-PSNR↑	SSIM↑	LPIPS↓	WS-PSNR↑	SSIM↑	LPIPS↓
w/o Mono	28.71	0.935	0.136	25.53	0.898	0.188	23.33	0.864	0.249
w/o MVS	28.23	0.925	0.165	25.15	0.888	0.227	23.15	0.855	0.274
full	**29.22**	**0.937**	**0.134**	**26.38**	**0.903**	**0.187**	**24.48**	**0.885**	**0.223**

Limitations Similar to other generalizable radiance fields, PanoGRF suffers from the issue of rendering speed. Additionally, since we only train PanoGRF on indoor data due to the lack of large-scale outdoor 360° datasets, its generalization performance can be limited when applied to outdoor scenes with significantly different depth scales.

Societal impact This method may be used to synthesize fake or deceptive panoramas, combined with generative methods.

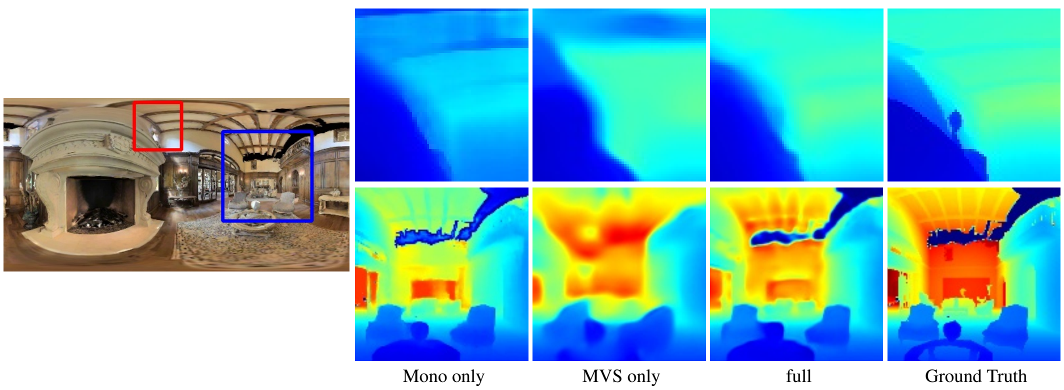

Figure 5: Qualitative results of ablation studies for depth estimation on Matterport3D. *Mono only* and *MVS only* respectively refer to the results obtained when using only 360° monocular depth and 360° multi-view stereo. The use of 360° monocular depth alone does not ensure multi-view consistency, resulting in potential discrepancies between the predicted scale and the ground truth in certain regions. Due to the occlusion problem, using only 360° MVS results in inaccurate and less detailed depth predictions, especially at the boundaries of objects.

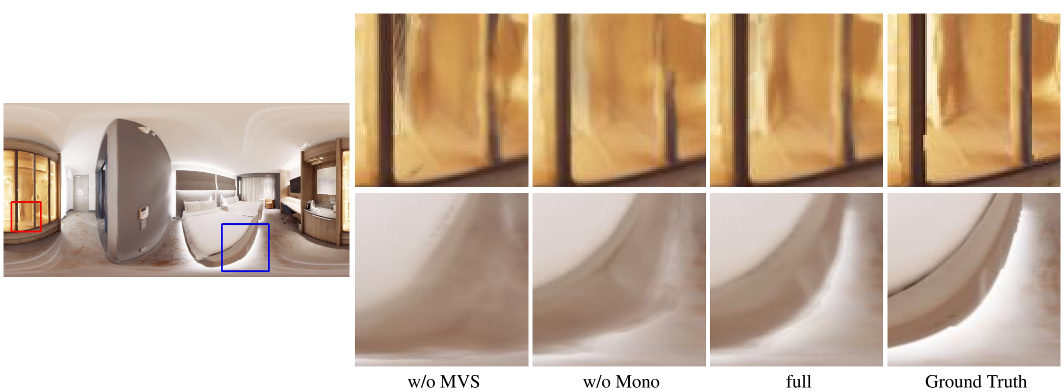

Figure 6: Qualitative results of ablation studies for novel view synthesis on Replica.

References

[1] Benjamin Attal, Selena Ling, Aaron Gokaslan, Christian Richardt, and James Tompkin. Matryodshka: Real-time 6dof video view synthesis using multi-sphere images. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, pages 441–459. Springer, 2020.

[2] Gwangbin Bae, Ignas Budvytis, and Roberto Cipolla. Multi-view depth estimation by fusing single-view depth probability with multi-view geometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2842–2851, 2022.

[3] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[4] Zheng Chen, Chen Wang, Yuan-Chen Guo, and Song-Hai Zhang. Structnerf: Neural radiance fields for indoor scenes with structural hints. *arXiv preprint arXiv:2209.05277*, 2022.

[5] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. *CoRR*, abs/2107.02791, 2021.

[6] Yilun Du, Cameron Smith, Ayush Tewari, and Vincent Sitzmann. Learning to render novel views from wide-baseline stereo pairs. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[7] Tewodros Habtegebrial, Christiano Gava, Marcel Rogge, Didier Stricker, and Varun Jampani. Somsi: Spherical novel view synthesis with soft occlusion multi-sphere images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15725–15734, 2022.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pages 2366–2369. IEEE, 2010.

[10] Huajian Huang, Yingshu Chen, Tianjian Zhang, and Sai-Kit Yeung. 360roam: Real-time indoor roaming using geometry-aware 360° radiance fields. *arXiv preprint arXiv:2208.02705*, 2022.

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

[12] Hualie Jiang, Zhe Sheng, Siyu Zhu, Zilong Dong, and Rui Huang. Unifuse: Unidirectional fusion for 360 panorama depth estimation. *IEEE Robotics and Automation Letters*, 6(2):1519–1526, 2021.

[13] Sameh Khamis, Sean Fanello, Christoph Rhemann, Adarsh Kowdle, Julien Valentin, and Shahram Izadi. Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 573–590, 2018.

[14] Mijeong Kim, Seonguk Seo, and Bohyung Han. Infonerf: Ray entropy minimization for few-shot neural volume rendering. *CoRR*, abs/2112.15399, 2021.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[16] David Li, Yinda Zhang, Christian Häne, Danhang Tang, Amitabh Varshney, and Ruofei Du. Omnisyn: Synthesizing 360 videos with wide-baseline panoramas. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 670–671. IEEE, 2022.

[17] Jisheng Li, Yuze He, Jinghui Jiao, Yubin Hu, Yuxing Han, and Jiangtao Wen. Extending 6-dof vr experience via multi-sphere images interpolation. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 4632–4640, 2021.

[18] Yuan Liu, Sida Peng, Lingjie Liu, Qianqian Wang, Peng Wang, Christian Theobalt, Xiaowei Zhou, and Wenping Wang. Neural rays for occlusion-aware image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7824–7833, 2022.

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

[20] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. *CoRR*, abs/2112.00724, 2021.

[21] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14335–14345, 2021.

[22] Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12892–12901, 2022.

[23] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[24] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 4104–4113, 2016.

[25] Stefan Schubert, Peer Neubert, Johannes Pöschmann, and Peter Protzel. Circular convolutional neural networks for panoramic images and laser data. In *2019 IEEE intelligent vehicles symposium (IV)*, pages 653–660. IEEE, 2019.

[26] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[27] Yule Sun, Ang Lu, and Lu Yu. Weighted-to-spherically-uniform quality evaluation for omnidirectional video. *IEEE signal processing letters*, 24(9):1408–1412, 2017.

[28] Guangcong Wang, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. *arXiv preprint arXiv:2303.16196*, 2023.

[29] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2021.

[30] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. SynSin: End-to-end view synthesis from a single image. In *CVPR*, 2020.

[31] Jiawei Yang, Marco Pavone, and Yue Wang. Freenerf: Improving few-shot neural rendering with free frequency regularization. *arXiv preprint arXiv:2303.07418*, 2023.

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

[33] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *arXiv preprint arXiv:2206.00665*, 2022.

[34] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

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

A More Results of Qualitative Comparisons

Qualitative comparisons with baseline methods on Replica and Matterport3D can be seen in Fig. 7 and Fig. 8, respectively.

Figure 7: Qualitative comparisons with baseline methods on Replica.

B Spherical Projection

Equirectangular-to-spherical The transformation from the equirectangular image coordinate system to the polar coordinate system is defined as:

$$\begin{aligned}\phi &= v/H * \pi \\ \theta &= u/W * 2\pi - 0.5\pi,\end{aligned} \tag{6}$$

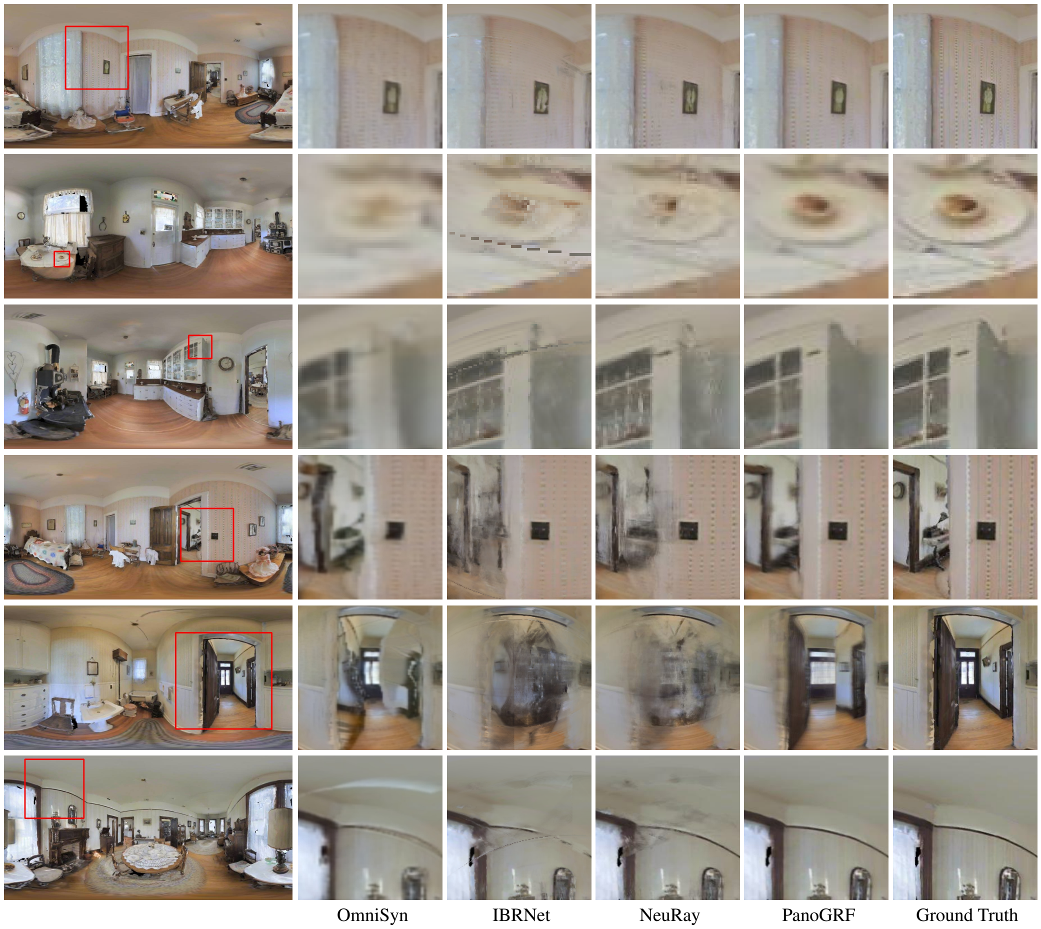

Figure 8: Qualitative comparisons with baseline methods on Matterport3D.

Table 4: Ablation studies for depth estimation on Matterport3D

setting	$L_1\downarrow$	$L_2\downarrow$	RMSE↓	WS-L_1↓	WS-L_2↓	WS-RMSE↓
MVS only	0.1731	0.5048	0.5831	0.1984	0.2806	0.4731
Mono only	0.2452	0.3175	0.4731	0.2445	0.2729	0.4522
full	**0.1441**	**0.2047**	**0.3877**	**0.1502**	**0.1624**	**0.3546**

Table 5: The impact of different backbones for depth estimation on Matterport3D

setting(backbone)	$L_1\downarrow$	$L_2\downarrow$	RMSE↓	WS-L_1↓	WS-L_2↓	WS-RMSE↓	parameters(M)
MVS(resnet-18)	0.1731	0.5048	0.5831	0.1984	0.2806	0.4731	29.75
MVS(resnet-34)	0.1654	0.4820	0.5676	0.1844	0.2577	0.4493	39.39
MVS(resnet-50)	0.1598	0.4725	0.5630	0.1822	0.2446	0.4442	47.10
MVS(resnet-101)	0.1642	0.4994	0.5717	0.1835	0.2519	0.4441	65.22
MVS(resnet-18)+Mono	**0.1441**	**0.2047**	**0.3877**	**0.1502**	**0.1624**	**0.3546**	58.95
MVS(resnet-34)+Mono	0.1549	0.2231	0.4120	0.1684	0.1878	0.3844	68.59
MVS(resnet-50)+Mono	0.1519	0.2379	0.4188	0.1675	0.1955	0.3887	76.30
MVS(resnet-101)+Mono	0.1735	0.2673	0.4452	0.1831	0.2129	0.4023	94.42

Table 6: The impact of different numbers of depth candidates N_{mono} for depth estimation on Matterport3D

N_{mono}	$L_1 \downarrow$	$L_2 \downarrow$	RMSE↓	WS-L_1↓	WS-L_2↓	WS-RMSE↓
5	**0.1441**	**0.2047**	**0.3877**	**0.1502**	**0.1624**	**0.3546**
16	0.1596	0.2379	0.4188	0.1675	0.1955	0.3887
32	0.1735	0.2673	0.4452	0.1831	0.2129	0.4023
48	0.1689	0.2618	0.4333	0.1776	0.2047	0.3941
64	0.1604	0.2432	0.4162	0.1669	0.2004	0.3853

Table 7: The impact of different values of σ for depth estimation on Matterport3D

σ	$L_1 \downarrow$	$L_2 \downarrow$	RMSE↓	WS-L_1↓	WS-L_2↓	WS-RMSE↓
0.1	0.1544	0.2515	0.4261	0.1672	0.1915	0.3830
0.5	0.1441	**0.2047**	**0.3877**	**0.1502**	**0.1624**	**0.3546**
1.0	0.1689	0.2686	0.4424	0.1803	0.2124	0.4029
1.5	**0.1426**	0.2457	0.4254	0.1563	0.1759	0.3723

where ϕ, θ represent the latitude and longitude of the sphere, u, v represent the rows and columns of the panorama, H and W represent the height and width of the panorama respectively.

Spherical-to-cartesian The transformation from the polar coordinate system to the 3D Cartesian coordinate system is:

$$\begin{aligned} x &= sin(\phi) * cos(\theta) \\ y &= cos(\phi) \\ z &= sin(\phi) * sin(\theta). \end{aligned} \tag{7}$$

Cartesian-to-spherical The camera cartesian coordinate (x, y, z) is transformed into the polar coordinate (θ, ϕ, t) ($t \in \mathbb{R}^+$ denotes its spherical depth in view I_j) by:

$$\begin{aligned} t &= \sqrt{x^2 + y^2 + z^2} \\ \theta &= arctan(\frac{z}{x}) \\ \phi &= arccos(\frac{y}{t}). \end{aligned} \tag{8}$$

Spherical-to-equirectangular The spherical polar coordinate (θ, ϕ, t) is turned into the equirectangular image coordinate (u, v) by the inverse process of Eq. 6.

C Experiments for Mono-guided Spherical Depth Estimator

C.1 Ablation Studies for Mono-guided Spherical Depth Estimator

To further validate the effectiveness of the key components, namely 360° multi-view stereos and 360° monocular depth, we conducted ablation studies specifically focused on spherical depth estimation. For evaluation purposes, we selected three commonly used metrics: L_1, L_2, and RMSE. Additionally,

Table 8: Quantitative comparison with Cross Attention Renderer [6] on Matterport3D and Replica

Dataset	Matterport3D (1.0m)			Replica(1.0m)		
method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
CAR [6]	22.87	0.7679	0.3108	23.60	0.8594	0.2515
PanoGRF	**27.78**	**0.8158**	**0.2444**	**29.87**	**0.9046**	**0.1604**

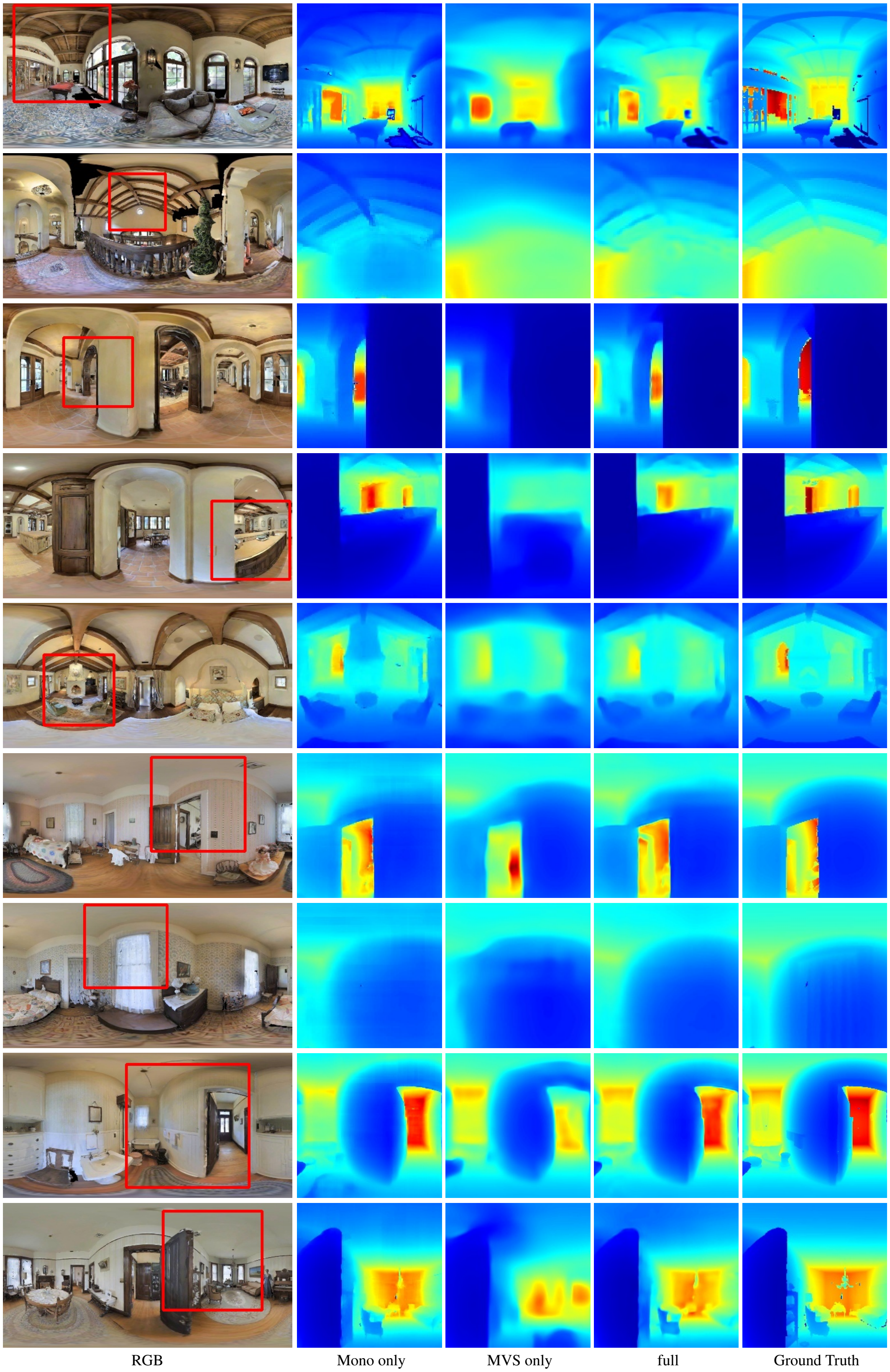

Figure 9: Qualitative results of ablation studies for depth estimation on Matterport3D. *Mono only* and *MVS only* respectively refer to the results obtained when using only 360° monocular depth and 360° multi-view stereo.

Cross Attention Renderer PanoGRF Ground Truth

Figure 10: Qualitative comparisons with Cross Attention Renderer on Replica

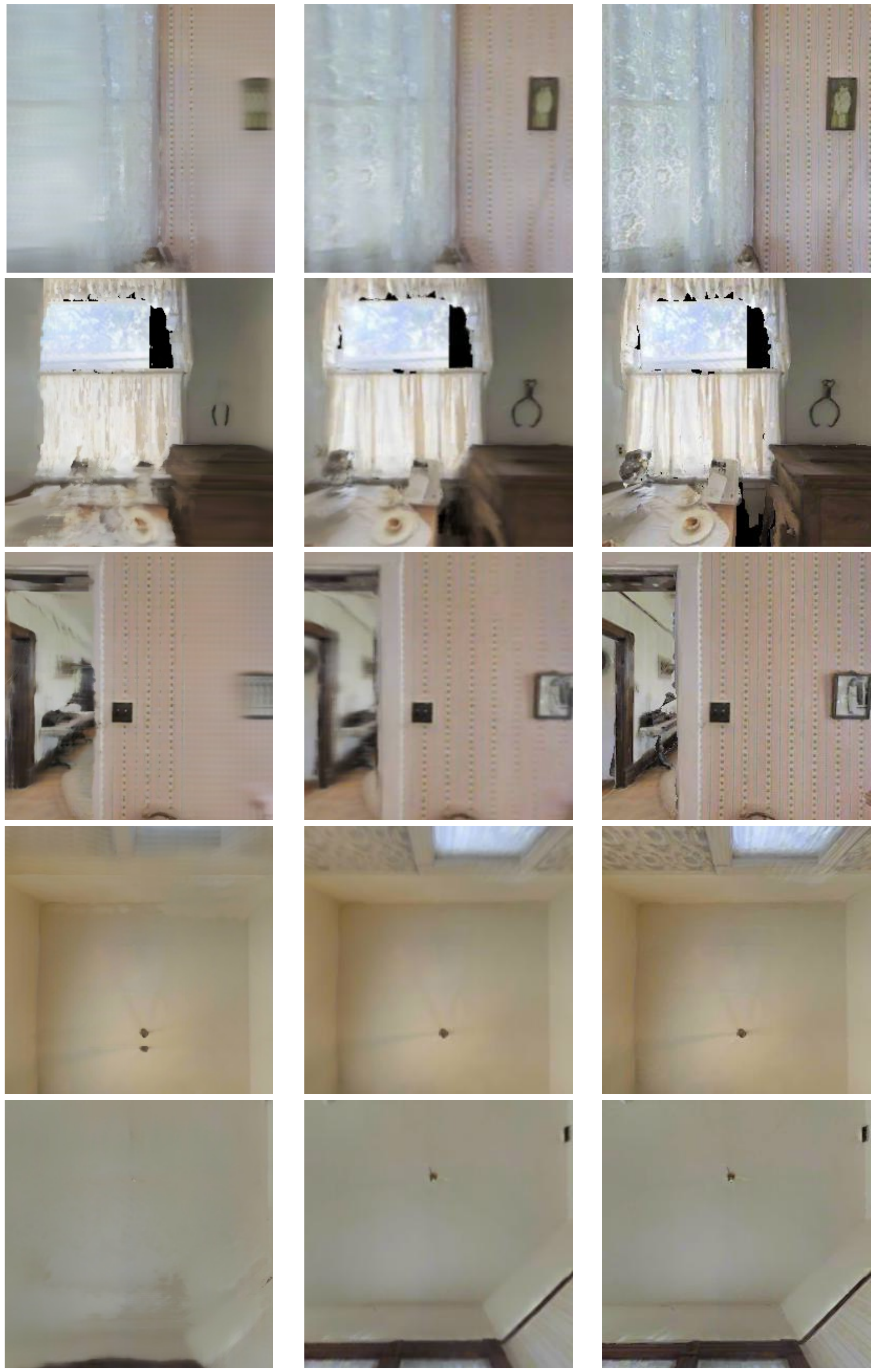

Figure 11: Qualitative comparisons with Cross Attention Renderer on Matterport3D

we also used WS-L_1, WS-L_2, and WS-RMSE as metrics, which incorporate weighted latitudes of equirectangular images to simulate WS-PSNR [27]. This approach aims to mitigate the impact of equirectangular projection distortion. We selected the first 1000 panorama pairs from Matterport3D as our test data, with a camera baseline of 1 meter. For the evaluation, we considered depth values within the range of $[0.1, 10]$ as valid.

The quantitative results of our ablation studies are presented in Table 4, while the qualitative results can be observed in Fig. 9. The experiments clearly demonstrate the importance of each module in achieving accurate depth estimation. Removing either the $360°$ multi-view stereo or the $360°$ monocular depth significantly reduces the depth accuracy. Using only $360°$ monocular depth does not guarantee multi-view consistency, resulting in potential discrepancies between the predicted scale and the ground truth in certain regions. The occlusion problem poses a challenge for using only $360°$ MVS, particularly at the boundaries of objects. Consequently, the depth predictions in these regions are inaccurate and lack detail.

C.2 Different Backbones for $360°$ Multi-view Stereo

Introducing $360°$ monocular depth to $360°$ multi-view stereo does result in an increase in the number of model parameters. However, it is important to note that the improvement in depth accuracy is not attributed to the increase in parameters. In our experiments, we increased the model size of $360°$ MVSNet by using larger backbones, specifically ResNet [8].

From Table. 5, we found that L_2 and RMSE of $360°$ MVSNet(ResNet-101) are still far inferior to those of $360°$ MVSNet(ResNet-18) together with the $360°$ monocular depth network [12]. This observation suggests that simply increasing the model size does not effectively address the view inconsistency problem in the wide-baseline setting. On the other hand, the introduction of $360°$ monocular depth provides a qualitative improvement to the accuracy of $360°$ multi-view stereo. By incorporating monocular depth information, the model gains additional cues that help mitigate the view inconsistency issue and improve depth estimation performance. Furthermore, when larger backbones were used as replacements, it was found that the performances of $360°$ MVS+Mono deteriorated. This could be attributed to the excessive model parameters leading to overfitting.

C.3 Different Hyperparameters for Mono-guided Spherical Depth Estimator

We conducted evaluations to assess the impact of different values for σ (standard deviation) and N_{mono} (number of monocular depth candidates) on mono-guided spherical depth sampling. The results are presented in Table 6 and Table 7. It can be seen that the results under $\sigma = 0.5$ and $N_{mono} = 5$ are the best.

D Comparisons with Cross Attention Renderer [6]

The Cross Attention Renderer (CAR) [3] is a method that operates on a wide-baseline perspective pair. We divided the two panoramas into cube maps and utilized the corresponding sides of the cube maps as inputs for CAR. Specifically, we rendered the corresponding side of the cube maps at the middle viewpoint. For example, we input the left side of the cube maps and generate the left side of the cube maps at the intermediate viewpoint as the output. We repeated this process for all six corresponding pairs of cube maps. In contrast, our method directly takes two panoramas as input and performs ray casting based on perspective projection. We then render the results for each side of the cube maps at the intermediate viewpoint, preserving the panoramic nature of the input.

We conducted quantitative comparative experiments on Matterport3D and Replica. The qualitative comparisons with CAR on Replica and Matterport3D can be seen in Fig. 10 and Fig. 11. The results clearly demonstrate that PanoGRF significantly outperforms CAR in terms of rendering quality. CAR suffers from limitations associated with its input field-of-view (FoV), particularly in edge regions that are only visible from a single perspective view. CAR relies on a pure stereo-matching method for geometric estimation, which leads to suboptimal rendering performance, as evidenced by the results presented in Table. 8. In contrast, PanoGRF is specifically designed to handle full FoV inputs, and it mitigates the issue of view inconsistency by incorporating the $360°$ monocular depth network.

[3] Cross Attention Renderer: https://github.com/yilundu/cross_attention_renderer

Table 9: The quantitative results of fine-tuning of the general renderers. The best results are in bold.

baseline		1.0m				0.2m			
method	setting	PSNR↑	WS-PSNR↑	SSIM↑	LPIPS↓	PSNR↑	WS-PSNR↑	SSIM↑	LPIPS↓
NeuRay	gen	27.751	27.253	0.8470	0.2565	34.318	33.376	0.9409	0.1158
PanoGRF	gen	**28.818**	**28.487**	**0.8778**	**0.1996**	35.817	35.056	0.9554	0.0959
NeuRay(+Consist)	ft	26.755	26.250	0.8181	0.2820	33.354	32.359	0.9240	0.1304
PanoGRF	ft	24.341	23.941	0.8175	0.2834	**35.828**	**35.138**	**0.9578**	**0.0912**
PanoGRF (+Depth)	ft	27.399	27.202	0.8556	0.2446	33.691	33.066	0.9333	0.1342

E Fine-tuning of PanoGRF

We conducted per-scene fine-tuning for NeuRay [18] and PanoGRF on the first test scene of Matterport3D with baselines of 1.0 and 0.2 meters, respectively. The general renderers were fine-tuned for 10k iterations, and the quantitative results are presented in Table.9. Under the baseline of 1.0 meters, the general renderer of PanoGRF, denoted as PanoGRF-gen, achieved the best performance. NeuRay-ft, the fine-tuned renderer of NeuRay, underwent fine-tuning using RGB loss and depth consistency loss, following the methodology described in their original paper [18]. PanoGRF-ft was fine-tuned with only RGB loss. However, the results of NeuRay-ft and PanoGRF-ft were inferior to NeuRay-gen and PanoGRF-gen, respectively. This suggests that fine-tuning under a wide-baseline setting does not improve the performance of general renderers. It is likely that fine-tuning with a wide baseline leads to overfitting. Even with the introduction of a depth uncertainty loss [22] by supervising the renderer depth of PanoGRF with predicted spherical depths during the fine-tuning process, the results of PanoGRF-ft remained inferior to those of the general renderer of PanoGRF. On the other hand, the performance of PanoGRF-ft under the baseline of 0.2 meters was slightly better than that of PanoGRF-gen. However, adding the depth loss [22] was still unable to improve PanoGRF-gen when fine-tuning. Inaccurate predicted depths in certain regions may be the cause, misleading the estimation of NeRF's geometry and thereby reducing the rendering performance. Additionally, NeuRay-ft was still inferior to NeuRay-gen under the narrow baseline. This may be attributed to the limited field-of-view, which can result in the aggregation of incorrect features. When projecting 3D sample points onto other source perspective views (cube-maps), these points may fall outside the image borders of the source perspective view or be located behind the source perspective camera.

F Quantitative Comparisons with Baseline Methods for Narrow-baseline Panoramas

Table 10: Quantitative comparisons with baseline methods on Matterport3D under the baseline of 0.2 and 0.5 meters. The best results are in bold.

baseline	0.2m				0.5m			
method	PSNR↑	WS-PSNR↑	SSIM↑	LPIPS↓	PSNR↑	WS-PSNR↑	SSIM↑	LPIPS↓
S-NeRF	20.79	19.52	0.6967	0.3756	17.95	16.81	0.6278	0.4856
OmniSyn	28.95	28.26	0.9132	0.1804	26.59	26.07	0.8897	0.2005
IBRNet	30.53	29.63	0.9271	0.1363	28.22	27.26	0.8844	0.1987
NeuRay	33.54	32.33	0.9485	0.1074	30.88	29.81	0.9196	0.1536
PanoGRF	**34.29**	**33.27**	**0.9515**	**0.0977**	**31.41**	**30.46**	**0.9238**	**0.1318**

We compared PanoGRF with baseline methods under the baseline of 0.2 and 0.5 meters on Matterport3D. As shown in Table. 10, the quantitative results demonstrate that PanoGRF consistently outperforms all the baseline methods under the baseline of 0.2 and 0.5 meters. These findings indicate that our method is applicable to both wide-baseline and narrow-baseline panoramas. In comparison to generalizable methods designed for perspective views, our method is particularly well-suited for synthesizing panoramic views by leveraging the aggregated features based on spherical projection.

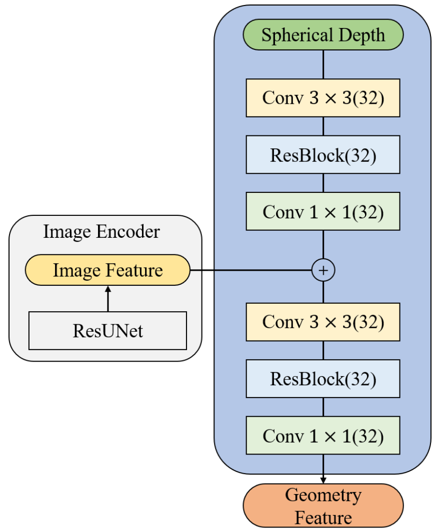

Figure 12: Architecture of geometry feature extractor.

G More Details of PanoGRF

G.1 Renderer

G.1.1 Training

PanoGRF employs the Adam optimizer [15] with an initial learning rate of 4.0e-4. The pre-training process of PanoGRF was conducted on an A100 GPU for 100k iterations, which required approximately two days. The learning rate is halved every 20k iterations, and a batch size of 512 was used during training.

G.1.2 Architecture

We adopted a coarse-to-fine sampling approach, similar to NeRF [19], and sampled 64 points in both phases. We followed a similar architecture as NeuRay [18] to build our renderers. The coarse and fine renderers share the same image encoder, geometry feature extractor, and visibility encoder. But they have different distribution decoders and aggregation networks $\mathcal{F}$, similar to NeuRay. During training, the weights of the 360° spherical depth estimator are fixed due to GPU memory limitations. Our image encoder, visibility encoder, distribution decoder, and aggregation network are implemented similarly to NeuRay, except for the padding mode. In the convolution layers of the image encoder and visibility encoder, we employ circular padding horizontally and zero padding vertically instead of direct zero padding. This is done to adapt to equirectangular image inputs and simulate Circular CNNs [25]. The circular padding helps account for the continuity of pixels at the leftmost and rightmost edges of equirectangular images, which are neighbors, while the top and bottom edges are not. The distribution network consists of 5 sub-networks, each with 3 fully connected layers. The appearance feature extractor is a ResNet [8] which contains 13 residual blocks and outputs the appearance feature map with 32 channels. The architecture details of the geometry feature extractor are shown in Fig. 12. The image encoder in the geometry feature extractor contains 14 residual blocks. The spherical depth input for the geometry feature extractor is first downsampled to 1/4 of its original resolution and then fed into the extractor to reduce GPU occupancy.

G.2 Spherical Depth Estimator

G.2.1 Training

We initially train the 360° monocular depth network and then freeze its weights when training the remaining components of 360° MVSNet. The Adam optimizer is employed with a fixed learning rate of 0.0001. Both the 360° monocular depth network and 360° MVSNet are trained for 100k iterations, which required approximately one day on a single V100 GPU. The batch size is set to 2.

G.2.2 Architecture

The 360° monocular depth network [12] and 360° MVSNet utilize ResNet-18 as the feature extractor. The 3D CNN regularization network is composed of 3 downsampling and 3 upsampling blocks, similar to [16]. The depth decoder consists of 2 convolution blocks. The feature map obtained from the middle layer in the 360° monocular depth network is concatenated with the regularized spherical cost volume and then decoded into 360° depth by the depth decoder. In the multi-view matching process, we compute the similarity by subtracting the feature vectors.

H Datasets

For Matterport3D, we split the training and testing set following SynSin [30]. The first 10 scenes of the test set are used for evaluation. In the case of the Replica dataset, we render a total of 18 scenes for evaluation. Additionally, we utilize the Residential dataset provided in [7], which comprises three scenes. For this dataset, we select the first and last panorama as the input views.

I Training Details of Baseline Methods

We trained NeuRay [18] and IBRNet [29] for 400k and 250k iterations, respectively. Spherical NeRF (S-NeRF) [19] underwent training for 2000 epochs, equivalent to approximately 256k iterations with the batch size of 4096. For OmniSyn [16], the in-painting network was trained for 50k iterations, which took approximately 3 days on a TitanRTX GPU. The depth estimator of OmniSyn was trained for 100k iterations. Due to limitations in GPU memory, OmniSyn was trained at a resolution of 256×256, and its output was resized to 512×1024 for evaluation purposes. CAR [6] was trained for 300k iterations.

Due to the significant difference between the spherical projection parameters used by OmniSyn and those provided by Residential, we could not run OmniSyn on Residential. But OmniSyn can use the parameters of the spherical projection provided by Matterport3D and Replica according to the code OmniSyn provides. The results of Tab. 1 and Tab. 2 have shown that the generalization ability of PanoGRF significantly surpasses OmniSyn.