Splatter-360: Generalizable 360° Gaussian Splatting for Wide-baseline Panoramic Images

Supplementary Material

7. Additional Quantitative Results

7.1. Comparisons with More Input Views

Table 4 presents a quantitative comparison of MVSplat and Splatter-360 using three-view inputs. Splatter-360 demonstrates superior performance to MVSplat in SSIM and LPIPS, while exhibiting comparable PSNR values. Additionally, we evaluate the estimated novel view depth using depth metrics used in [32]. Splatter-360 significantly outperforms MVSplat across all the used depth metrics. Our Splatter-360 exhibits robust general performance with three-view inputs, despite being trained on two-view inputs.

7.2. Comparisons under a Narrow-baseline

Table 5 presents a quantitative comparison between MVSplat and Splatter-360 under the narrow-baseline setting. In the main text, we sample an input pair with a frame interval of 100. Here, the frame interval of the input pair is further reduced to 50. Splatter-360 consistently outperforms MVSplat across all metrics. This result indicates that Splatter-360 exhibits superior performance under the narrow-baseline condition.

7.3. Additional Ablation Studies

We conduct additional ablation studies on the cost volume refinement U-Net and the depth refinement U-Net. Table 6 presents the statistical results obtained after removing these modules individually. In comparison to the complete model using all components, the model without depth refinement U-Net exhibits significantly degraded PSNR performance. Specifically, PSNR decreased by approximately 0.7 dB on Replica and by about 0.69 dB on HM3D. Upon removing the cost volume refinement U-Net, PSNR decreased by 0.119 dB on Replica and by 0.125 dB on HM3D.

7.4. The evaluation of 360° MVSNet within Splatter-360

Current depth estimation methods that utilize multiple wide-baseline panoramic views, such as [11, 12], depend on specialized camera setups that capture ultra-wide field-of-view imagery (e.g., 220°). In contrast, our work focuses on standard 360° cameras, making it difficult to fairly evaluate or compare these methods with ours. To thoroughly assess the depth quality of our method, we conduct a quantitative comparison with the current state-of-the-art method, PanoGRF, specifically designed for wide-baseline panoramic images. Importantly, our evaluation focuses on a direct comparison of the estimated depths for the input views, rather than on rendered depths for target views, which are typically emphasized in traditional depth estimation studies. As shown in Table 7, the depth estimation performance of Splatter-360 greatly exceeds that of 360° MVSNet within the PanoGRF.

Table 4. Quantitative comparison with three context views between MVSplat and Splatter-360 on the Replica and HM3D datasets.

Dataset	Metric	MVSplat	Splatter-360
Replica [34]	PSNR↑	**29.121**	29.109
	SSIM↑	0.908	**0.913**
	LPIPS↓	0.123	**0.116**
	Abs Diff↓	0.125	**0.103**
	Abs Rel↓	0.078	**0.060**
	RMSE↓	0.233	**0.193**
	$\delta < 1.25$↑	90.771	**94.367**
HM3D [29]	PSNR↑	27.858	**27.905**
	SSIM↑	0.861	**0.868**
	LPIPS↓	0.174	**0.168**
	Abs Diff↓	0.118	**0.095**
	Abs Rel↓	0.083	**0.067**
	RMSE↓	0.251	**0.209**
	$\delta < 1.25$↑	91.684	**94.545**

Table 5. Quantitative comparison under a narrow baseline between MVSplat and Splatter-360 on the Replica and HM3D datasets.

Dataset	Metric	MVSplat	Splatter-360
Replica [34]	PSNR↑	32.521	**33.282**
	SSIM↑	0.951	**0.957**
	LPIPS↓	0.064	**0.058**
	Abs Diff↓	0.109	**0.090**
	Abs Rel↓	0.057	**0.048**
	RMSE↓	0.214	**0.171**
	$\delta < 1.25$↑	94.257	**96.645**
HM3D [29]	PSNR↑	30.851	**31.493**
	SSIM↑	0.915	**0.925**
	LPIPS↓	0.109	**0.101**
	Abs Diff↓	0.102	**0.092**
	Abs Rel↓	0.060	**0.058**
	RMSE↓	0.228	**0.189**
	$\delta < 1.25$↑	94.802	**96.031**

Table 6. Additional ablation studies were conducted on the HM3D and Replica datasets. For simplicity, we use the following abbreviations: 'CVRU' for spherical cost volume refinement U-Net and 'DRU' for depth refinement U-Net.

Ablated	Replica [34]			HM3D [29]		
module	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
× DRU	28.306	0.905	0.129	26.800	0.846	0.189
× CVRU	29.002	0.912	0.115	27.362	0.856	0.175
Full	**29.121**	**0.914**	**0.111**	**27.487**	**0.860**	**0.171**

Table 7. Quantitative comparison of panoramic depth estimation between PanoGRF and Splatter-360, respectively.

Metrics	Abs Diff↓	Abs Rel↓	RMSE↓	$\delta < 1.25$↑
PanoGRF	0.1645	0.1014	0.2880	91.6294
Ours	**0.0504**	**0.0315**	**0.1585**	**98.5995**

7.5. Network Parameters and GPU Memory Usage Evaluation

Initially, we faced limitations in training resources, preventing us from training DepthSplat and HiSplat. During the short rebuttal period, we rented a server with 8× A800 (80G) GPUs to retrain both DepthSplat and HiSplat on HM3D dataset. The quantitative results for novel view synthesis are comprehensively presented in Table 8, providing a detailed performance comparison across all evaluated methods. Additionally, the model parameters and GPU memory consumption are summarized in Table 9, offering insights into the computational efficiency of each approach.

In this section, we conduct a detailed analysis of the experimental results obtained from the Replica dataset. DepthSplat (37.7M params.) achieved a PSNR of 28.600, an SSIM of 0.910, and an LPIPS of 0.116. HiSplat (150M params.) reached a PSNR of 29.195, an SSIM of 0.914, and an LPIPS of 0.107. In comparison, our Splatter-360 (38.8M params.) outperformed both, achieving a PSNR of 29.888, an SSIM of 0.924, and an LPIPS of 0.097. Importantly, the rented A800 allowed us to evaluate the peak memory consumption of all three methods. DepthSplat consumed 52.7 G, HiSplat used 78.8 G, while our Splatter-360 only required 31.4 G for training.

7.6. Splatter-360 vs. MVSplat with Spherical Cost Volume

We modified MVSplat with a spherical cost volume, trained it on HM3D dataset, and tested it on Replica dataset. For a fair comparison, we adopt the same ERP gaussian decoder part with Splatter-360 as a spherical cost volume needs a ERP feature decoder.

MVSplat with spherical cost volume achieves a PSNR of 27.859, SSIM of 0.901, and LPIPS of 0.122. In contrast, Splatter-360 exhibited superior performance, with a PSNR of 29.888, SSIM of 0.924, and LPIPS of 0.097. This further demonstrates that merely adding a spherical cost volume to MVSplat does not yield performance on par with our method. This finding emphasizes that our method is the outcome of careful design, rather than an incremental improvement by substituting MVSplat's planar cost volume with a spherical one. The PSNR of 27.859 achieved by MVSplat (with spherical cost volume) is significantly lower than the 28.399 achieved by MVSplat†(with planar cost volume), as shown in Table 1. This performance gap can be attributed to the inherent limitation of spherical cost volumes, which are optimized to perform best when paired with an Equirectangular Projection (ERP) image encoder rather than a perspective image encoder. The mismatch between the spherical cost volume and the perspective image encoder likely results in suboptimal feature extraction and reconstruction, thereby impacting the overall rendering quality.

8. More Implementation Details

8.1. Dataset Details

The datasets are built based on Replica [34] and HM3D [29] textured mesh dataset. In particular, we sample camera trajectories to render videos with AI-Habitat simulation tool [31]. Since AI-habitat only provides the API for capturing perspective views, we first get cube maps for each viewpoint and stitch them into panoramas.

For HM3D [29], we split the train and test set following their original split. HM3D contains 800 training scenes and 100 test scenes. We sample 5 camera trajectories for each scene. In total, we finally rendered 4000 training scenes and 500 test scenes.

For Replica, we use all the scenes for testing. Replica has 18 scenes in total, and we sample 5 camera trajectories for each scene. In total, we render 90 test scenes. We randomly sample 3 target views between the context image pair for testing.

8.2. Experiment Details

We implement our proposed method using PyTorch and conduct all experiments on a cluster of NVIDIA V100 GPUs, each equipped with 32GB of VRAM. Splatter-360 is trained with an RGB loss function that is a linear combination of mean squared error (MSE) and LPIPS loss, with weights set to 1.0 and 0.05, respectively. We also supervise the depth prediction, with the weight set to 0.1.

The code for depth rendering is the same as that used in MVSplat. We utilize perspective for supervision and stitch cubemaps into panoramas for holistic visualization. While Gaussian rendering is not the primary focus of our contribution, the generalizability of network training is, and our method is not restricted to rendering approaches. We will

Table 8. Quantitative comparison with baseline methods on the HM3D and Replica datasets. † indicates models that were trained by us on the panoramic dataset

Method	HM3D [29]			Replica [34]		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
HiSplat†	27.850	0.862	0.165	29.195	0.914	0.107
DepthSplat†	28.280	0.871	0.155	28.600	0.910	0.116
Splatter-360†	**28.293**	**0.875**	**0.155**	**29.888**	**0.924**	**0.097**

Table 9. Quantitative comparison with baseline methods on the HM3D and Replica datasets. † indicates models that were trained by us on the panoramic dataset

Method	Training GPU Memory (G)↓	Parameters (million)↓
HiSplat	78.8	150.0
DepthSplat	52.7	**37.7**
Splatter-360	**31.4**	38.8

consider updating our renderer in our future work.

Splatter-360 is trained and evaluated at 1024×512, consistent with PanoGRF. The test set comprises average sequences of 304 frames, with a baseline of 1.59 meters. The model contains 38.8 million parameters, achieving inference and rendering times of 0.1786 s and 0.0048 s per frame, respectively, on an A800 GPU.

In the comparisons of the main paper, for HiSplat, MVSplat, and DepthSplat, we utilize their model pre-trained models on RE10K [57] for evaluation. We set $near = 0.5$ and $far = 10$ for these models as these parameters are relatively close to their training setting. We match features under the resolution of $\frac{1}{4}H \times \frac{1}{4}W$, where H and W are the height and width of input images. We apply $near = 0.1$ for HiSplat, MVSplat, and DepthSplat, but the results get worse as $near = 0.1$ is much different from their training setting on RE10K [57].

For MVSplat† and Splatter-360† trained on HM3D, we set $near = 0.1$ and $far = 10$ to match our indoor dataset for the consideration of fairness. We perform cross view matching under the resolution of $\frac{1}{8}H \times \frac{1}{8}W$ due to GPU memory limits.

8.3. Network Details

We adopt the encoder of UniMatch [46] as our backbone. The first convolution layer downsamples images with a stride of 2. Next, we utilize six residual layers to extract features. The first two residual layers contain utilize the stride of 1. Subsequently, we downsample features in half after every two residual layers with a stride of 2. We then get $\frac{1}{8}H \times \frac{1}{8}W$ feature maps. The downsampled feature maps are fed into a cross-view transformer composed of six stacked transformer blocks. Each transformer block contains a self-attention and a cross-view attention layer. Similar to MVSplat [5], we utilize the local window attention of SwinTransformer [27]. We apply the network architecture for our ERP multi-view transformer and CP multi-view transformer.

For the cost volume refinement U-UNet, we adopt the U-Net from Stable Diffusion 1.5 [30] as our implementation with an unchanged feature channel of 128 throughout the network. We apply two times $2\times$ down-sampling and one self-attention layer at the $4\times$ down-sampled level. We flatten the feature map before feeding them to the attention module, to interact with the features among different views utilizing the multi-view attention similar to [33]. For the depth refinement U-Net which we omitted in the main text for simplicity, we apply 4 times $2\times$ down-sampling and add the multi-view attention at $16\times$ down-sampled level.

We set $D = 128$ in the depth sampling consistently with MVSplat [5]

9. Preliminary of 3DGS

The 3D Gaussian ellipsoid is formally defined as:

$$G(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})} \tag{11}$$

where $\boldsymbol{\mu} \in \mathbb{R}^3$ represents the spatial mean, and $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$ denotes the covariance matrix. To ensure numerical stability during optimization, the covariance matrix $\boldsymbol{\Sigma}$ is decomposed into a scaling matrix $\boldsymbol{S}$ and a rotation matrix $\boldsymbol{R}$ as follows:

$$\boldsymbol{\Sigma} = \boldsymbol{R}\boldsymbol{S}\boldsymbol{S}^\top\boldsymbol{R}^\top \tag{12}$$

During the rendering process, the 3D Gaussians are projected onto a 2D image plane. Using the intrinsic matrix $\boldsymbol{K}$ and extrinsic matrix $\boldsymbol{T}$, the 2D mean $\boldsymbol{\mu}'$ and covariance matrix $\boldsymbol{\Sigma}'$ are computed as:

$$\boldsymbol{\mu}' = \boldsymbol{K}[\boldsymbol{\mu}, 1]^\top, \quad \boldsymbol{\Sigma}' = \boldsymbol{J}\boldsymbol{T}\boldsymbol{\Sigma}\boldsymbol{T}^\top\boldsymbol{J}^\top \tag{13}$$

Here, $\boldsymbol{J}$ represents the Jacobian matrix of the affine approximation of the projective transformation. Each Gaussian is associated with an opacity value o and a view-dependent color $\boldsymbol{c}$, which is determined by a set of spherical harmonics coefficients. The pixel color $\boldsymbol{C}$ is computed via alpha-blending over the 2D Gaussians, sorted from front to back:

$$\boldsymbol{C} = \sum_{i \in N} T_i G_i \left(\boldsymbol{u} \mid \boldsymbol{\mu}', \boldsymbol{\Sigma}' \right) \sigma_i \boldsymbol{c}_i \tag{14}$$

where the transmittance T_i is defined as:

$$T_i = \prod_{j=1}^{i-1} \left(1 - G_i \left(\boldsymbol{u} \mid \boldsymbol{\mu}', \boldsymbol{\Sigma}' \right) \sigma_i \right) \tag{15}$$

10. Limitations and Future Work

Our method achieves state-of-the-art performance on several benchmarks but shares some limitations with existing approaches. It requires pose input, lacks generative capabilities, and is currently limited to indoor scenes due to the absence of a large-scale 360° outdoor dataset. In the future, we plan to create a comprehensive synthetic outdoor dataset and enhance our method with a video diffusion model. More importantly, we aim to explore pose-free 360° reconstruction in diverse outdoor environments.