MVSGaussian: Fast Generalizable Gaussian Splatting Reconstruction from Multi-View Stereo

Tianqi Liu[1], Guangcong Wang[2,3], Shoukang Hu[2], Liao Shen[1], Xinyi Ye[1], Yuhang Zang[4], Zhiguo Cao[1*], Wei Li[2†], and Ziwei Liu[2]

[1] School of AIA, Huazhong University of Science and Technology
[2] S-Lab, Nanyang Technological University
[3] Great Bay University
[4] Shanghai AI Laboratory
{tq_liu,zgcao}@hust.edu.cn
https://mvsgaussian.github.io/

Abstract. We present MVSGaussian, a new generalizable 3D Gaussian representation approach derived from Multi-View Stereo (MVS) that can efficiently reconstruct unseen scenes. Specifically, 1) we leverage MVS to encode geometry-aware Gaussian representations and decode them into Gaussian parameters. 2) To further enhance performance, we propose a hybrid Gaussian rendering that integrates an efficient volume rendering design for novel view synthesis. 3) To support fast fine-tuning for specific scenes, we introduce a multi-view geometric consistent aggregation strategy to effectively aggregate the point clouds generated by the generalizable model, serving as the initialization for per-scene optimization. Compared with previous generalizable NeRF-based methods, which typically require minutes of fine-tuning and seconds of rendering per image, MVSGaussian achieves real-time rendering with better synthesis quality for each scene. Compared with the vanilla 3D-GS, MVSGaussian achieves better view synthesis with less training computational cost. Extensive experiments on DTU, Real Forward-facing, NeRF Synthetic, and Tanks and Temples datasets validate that MVSGaussian attains state-of-the-art performance with convincing generalizability, real-time rendering speed, and fast per-scene optimization.

Keywords: Generalizable Gaussian Splatting · Multi-View Stereo · Neural Radiance Field · Novel View Synthesis

1 Introduction

Novel view synthesis (NVS) aims to produce realistic images at novel viewpoints from a set of source images. By encoding scenes into implicit radiance fields, NeRF [29] has achieved remarkable success. However, this implicit representation

* Corresponding author
† Project lead

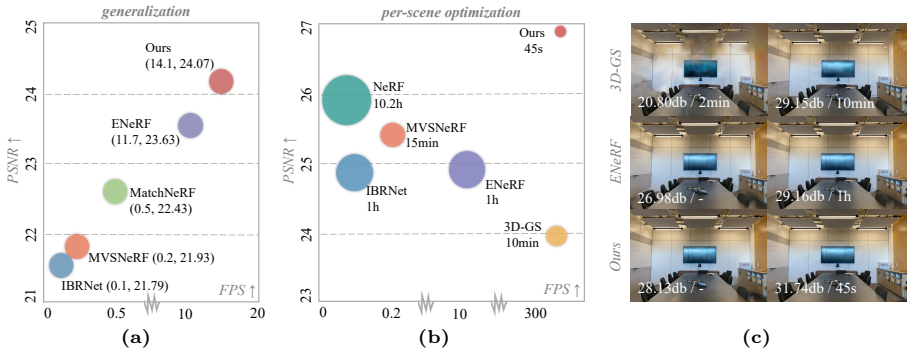


Fig. 1: Comparison with existing methods. (a) We present the generalizable results on the Real Forward-facing dataset [28]. Compared with other competitors, our method achieves better performance at a faster inference speed. (b) The results after per-scene optimization, where circle size represents optimization time. Our method achieves optimal performance in just 45 seconds. (c) We illustrate a scene ("room"), showcasing the (PSNR/optimization time) of synthesized views, with "-" indicating results from direct inference using the generalizable model.

is time-consuming due to the necessity of querying dense points for rendering. Recently, 3D Gaussian Splatting (3D-GS) [19] utilizes anisotropic 3D Gaussians to explicitly represent scenes, achieving real-time and high-quality rendering through a differentiable tile-based rasterizer. However, 3D-GS relies on per-scene optimization for several minutes, which limits its applications.

To remedy this issue, some initial attempts have been made to generalize Gaussian Splatting to unseen scenes. Generalizable Gaussian Splatting methods directly regress Gaussian parameters in a feed-forward manner instead of per-scene optimization. The general paradigm involves encoding features for 3D points in a scene-agnostic manner, followed by decoding these features to obtain Gaussian parameters. PixelSplat [4] leverages an epipolar Transformer [37] to address scale ambiguity and encode features. However, it focuses on image pairs as input, and the introduction of Transformers results in significant computational overhead. GPS-Gaussian [56] draws inspiration from stereo matching by first performing epipolar rectification on input image pairs, followed by disparity estimation and feature encoding. However, it focuses on human novel view synthesis and requires ground-truth depth maps. Splatter Image [35] introduces a single-view reconstruction approach based on Gaussian Splatting but focuses on object-centric reconstruction rather than generalizing to unseen scenes.

Due to the inefficiency of existing methods and their limitation to object-centric reconstruction, in this paper, we aim to develop an efficient generalizable Gaussian Splatting framework for novel view synthesis in unseen general scenes, which faces several critical challenges: ***First***, unlike NeRFs that use an implicit representation, 3D-GS is a parameterized explicit representation that uses millions of 3D Gaussians to overfit a scene. When applying the pre-trained

3D-GS to an unseen scene, the parameters of 3D Gaussians, such as locations and colors, are significantly different. It is a non-trivial problem to design a generalizable representation to tailor 3D-GS. ***Second***, previous generalizable NeRFs [6, 9, 22, 40, 54] have achieved impressive view synthesis results through volume rendering. However, the generalization capability of splatting remains unexplored. During splatting, each Gaussian contributes to multiple pixels within a certain region in the image, and each pixel's color is determined by the accumulated contributions from multiple Gaussians. The color correspondence between Gaussians and pixels is a more complex many-to-many mapping, which poses a challenge for model generalization. ***Third***, generalizable NeRFs show that further fine-tuning for specific scenes can greatly improve the synthesized image quality but requires lengthy optimization. Although 3D-GS is faster than NeRF, it still remains time-consuming. Designing a fast optimization approach based on the generalizable 3D-GS model is promising.

We address these challenges point by point. ***First***, we propose leveraging MVS for geometry reasoning and encoding features for 3D points to establish pixel-aligned Gaussian representations. The point-wise features are aggregated from multi-view features, and the spatial awareness is enhanced through a 2D UNet, as each Gaussian contributes to multiple pixels. ***Second***, with the encoded point-wise features, we can decode them into Gaussian parameters through an MLP. Rather than solely relying on splatting, we propose adding a simple yet effective depth-aware volume rendering approach to enhance generalization. ***Third***, with the trained generalizable model, lots of 3D Gaussians can be generated from multiple views. These Gaussian point clouds can serve as an initialization for subsequent per-scene optimization. However, the generated 3D Gaussians from the generalizable model are not perfect. Directly concatenating such a large number of Gaussians as initialization for per-scene optimization leads to unexpected computational costs because these Gaussians further split and clone during optimization. One approach is to downsample the point cloud, such as voxel downsampling, which can reduce noise but also result in the loss of effective information. Therefore, we introduce a strategy to aggregate point clouds by preserving multi-view geometric consistency. Specifically, we filter out noisy points by computing the reprojection error of the depth of Gaussians from different viewpoints. This strategy can filter out noisy points while preserving effective ones, providing a high-quality initialization for subsequent optimization.

To summarize, we present a new fast generalizable Gaussian Splatting method. We evaluate our method on the widely-used DTU [1], Real Forward-facing [28], NeRF Synthetic [29], and Tanks and Temples [21] datasets. Extensive experiments show that our generalizable method outperforms other generalizable methods. After a short period of per-scene optimization, our method achieves performance comparable to or even better than other methods with longer optimization times, as shown in Fig. 1. On a single RTX 3090 GPU, compared with the vanilla 3D-GS, our proposed method achieves better novel view synthesis with similar rendering speed (300+ FPS) and 13.3× less training computational cost (45s). Our main contributions can be summarized as follows:

- We present MVSGaussian, a generalizable Gaussian Splatting method derived from Multi-View Stereo and a pixel-aligned Gaussian representation.
- We further propose an efficient hybrid Gaussian rendering approach to boost generalization learning.
- We introduce a consistent aggregation strategy to provide high-quality initialization for fast per-scene optimization.

2 Related Work

Multi-View Stereo (MVS) aims to reconstruct a dense 3D representation from multiple views. Traditional MVS methods [12, 13, 33, 34] rely on handcrafted features and similarity metrics, which limits their performance. With the advancement of deep learning in 3D perception, MVSNet [51] first proposes an end-to-end pipeline, with the key idea being the construction of a cost volume to aggregate 2D information into a 3D geometry-aware representation. Subsequent works follow this cost volume-based pipeline and make improvements from various aspects, *e.g.* reducing memory consumption with recurrent plane sweeping [47, 52] or coarse-to-fine architectures [10, 14, 49], optimizing cost aggregation [41, 43], enhancing feature representations [11, 25], and improving decoding strategy [31, 53]. As the cost volume encodes the consistency of multi-view features and naturally performs correspondence matching, in this paper, we develop a new generalizable Gaussian Spatting representation derived from MVS.

Generalizable NeRF. By implicitly representing scenes as continuous color and density fields using MLPs, Neural Radiance Fields (NeRF) achieve impressive rendering results with volume rendering techniques. Follow-up works [2, 5, 17, 18, 30, 38, 39, 45, 46] extend it to various tasks and achieve promising results. However, they all require time-consuming per-scene optimization. To address this issue, some generalizable NeRFs have been proposed. The general paradigm involves encoding features for each 3D point and then decoding these features to obtain volume density and radiance. According to the encoded features, generalizable NeRFs can be categorized into appearance features [54], aggregated multi-view features [22, 24, 36, 40], cost volume-based features [6, 22, 24, 26], and correspondence matching features [9]. Despite considerable progress, performance remains limited, with slow optimization and rendering speeds.

3D Gaussian Splatting (3D-GS) utilizes anisotropic Gaussians to explicitly represent scenes and achieves real-time rendering through differentiable rasterization. Motivated by this, several studies have applied it to various tasks, *e.g.* editing [3, 8], dynamic scenes [27, 44, 50], avatars [15, 16, 32] and others [7]. However, the essence of Gaussian Splatting still lies in overfitting the scene. To remedy this, a few concurrent works make initial attempts to generalize Gaussian Splatting to unseen scenes. The goal of Generalizable Gaussian Splatting is to predict Gaussian parameters in a feed-forward manner instead of per-scene optimization. PixelSplat [4] addresses scale ambiguity by leveraging an epipolar Transformer to encode features and subsequently decode them into Gaussian parameters. However, it focuses on image pairs as input and the Transformer incurs

significant computational costs. GPS-Gaussian [56] draws inspiration from stereo matching and performs epipolar rectification and disparity estimation on input image pairs. However, it focuses on human novel view synthesis and requires ground-truth depth maps. Spatter Image [35] introduces a single-view 3D reconstruction approach. However, it focuses on object-centric reconstruction rather than generalizing to unseen scenes. Overall, these methods are constrained by inefficiency, limited to object reconstruction, and restricted to either image pairs or a single view. To this end, in this paper, we aim to study an efficient generalizable Gaussian Splatting for novel view synthesis in unseen general scenes.

3 Preliminary

3D Gaussian Splatting represents a 3D scene as a mixture of anisotropic 3D Gaussians, each of which is defined with a 3D covariance matrix Σ and mean μ:

$$G(X) = e^{-\frac{1}{2}(X-\mu)^T \Sigma^{-1}(X-\mu)}\,. \tag{1}$$

The covariance matrix Σ holds physical meaning only when it is positive semi-definite. Therefore, for effective optimization through gradient descent, Σ is decomposed into a scaling matrix S and a rotation matrix R, as $\Sigma = RSS^TR^T$. To splat Gaussians from 3D space to a 2D plane, the view transformation W and the Jacobian matrix J representing the affine approximation of the projective transformation are utilized to obtain the covariance matrix Σ' in 2D space, as $\Sigma' = JW\Sigma W^T J^T$. Subsequently, a point-based alpha-blend rendering can be performed to obtain the color of each pixel:

$$C = \sum_i c_i \alpha_i \prod_{j=1}^{i-1}(1-\alpha_i)\,, \tag{2}$$

where c_i is the color of each point, defined by spherical harmonics (SH) coefficients. The density α_i is computed by the multiplication of 2D Gaussians and a learnable point-wise opacity. During optimization, the learnable attributes of each Gaussian are updated through gradient descent, including 1) a 3D position $\mu \in \mathbb{R}^3$, 2) a scaling vector $s \in \mathbb{R}^3_+$, 3) a quaternion rotation vector $r \in \mathbb{R}^4$, 4) a color defined by SH $c \in \mathbb{R}^k$ (where k is the freedom), and 5) an opacity $\alpha \in [0, 1]$. Additionally, an adaptive density control module is introduced to improve rendering quality, comprising mainly the following three operations: 1) split into smaller Gaussians if the magnitude of the scaling exceeds a threshold, 2) clone if the magnitude of the scaling is smaller than a threshold, and 3) prune Gaussians with excessively small opacity or overly large scaling magnitudes.

4 MVSGaussian

4.1 Overview

Given a set of source views $\{I_i\}_{i=1}^N$, NVS aims to synthesize a target view from a novel camera pose. The overview of our proposed generalizable Gaussian Splat-

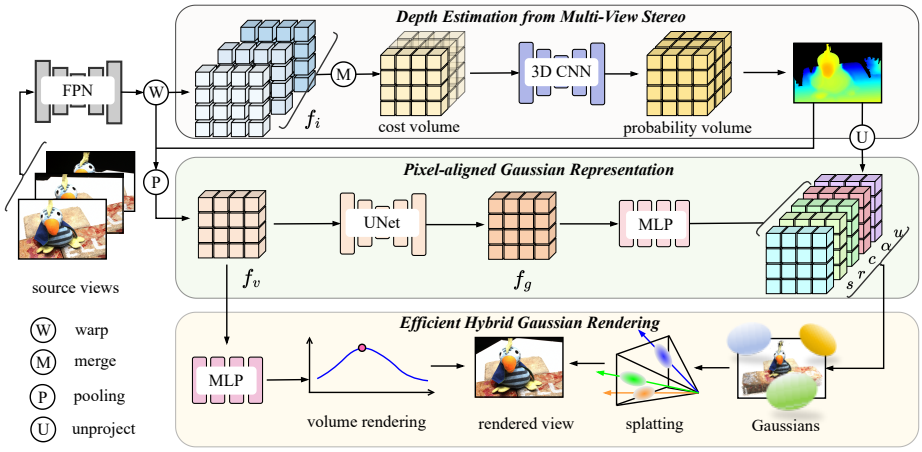


Fig. 2: Overview of MVSGaussian. We first extract features $\{f_i\}_{i=1}^N$ from input source views $\{I_i\}_{i=1}^N$ using FPN. These features are then aggregated into a cost volume, regularized by 3D CNNs to produce depth. Subsequently, for each 3D point at the estimated depth, we use a pooling network to aggregate warped source features, obtaining the aggregated feature f_v. This feature is then enhanced using a 2D UNet, yielding the enhanced feature f_g. f_g is decoded into Gaussian parameters for splatting, while f_v is decoded into volume density and radiance for depth-aware volume rendering. Finally, the two rendered images are averaged to produce the final rendered result.

ting framework is depicted in Fig. 2. We first utilize a Feature Pyramid Network (FPN) [23] to extract multi-scale features from source views. These features are then warped onto the target camera frustum to construct a cost volume via differentiable homography, followed by 3D CNNs for regularization to produce the depth map. Based on the obtained depth map, we encode features for each pixel-aligned 3D point by aggregating multi-view and spatial information. The encoded features can be then decoded for rendering. However, Gaussian Splatting is a region-based explicit representation and is designed for tile-based rendering, involves a complex many-to-many mapping between Gaussians and pixels, posing challenges for generalizable learning. To address this, we propose an efficient hybrid rendering by integrating a simple depth-aware volume rendering module, where only one point is sampled per ray. We render two views using Gaussian Splatting and volume rendering, then average these two rendered views into the final view. This pipeline is further constructed in a cascade structure, propagating the depth map and rendered view in a coarse-to-fine manner.

4.2 MVS-based Gaussian Splatting Representation

Depth Estimation from MVS. The depth map is a crucial component of our pipeline, as it bridges 2D images and 3D scene representation. Following learning-based MVS methods [51], we first establish multiple fronto-parallel planes at the

target view. Then, we warp the features of source views onto these sweeping planes using differentiable homography as:

$$H_i(z) = K_i R_i (I + \frac{(R_i^{-1} t_i - R_t^{-1} t_t) a^T R_t}{z}) R_t^{-1} K_t^{-1}\,, \tag{3}$$

where $[K_i, R_i, t_i]$ and $[K_t, R_t, t_t]$ are the camera intrinsic, rotation and translation of the source view I_i and target view, respectively. The a represents the principal axis of the target view camera, I denotes the identity matrix and z is the sampled depth. With the warped features from source views, a cost volume is constructed by computing their variance, which encodes the consistency of multi-view features. Then, the cost volume is fed into 3D CNNs for regularization to obtain the probability volume. With this depth probability distribution, we weight each depth hypothesis to obtain the final depth.

Pixel-aligned Gaussian Representation. With the estimated depth, each pixel can be unprojected to a 3D point, which is the position of the 3D Gaussian. The subsequent step is encoding features for these 3D points to establish a pixel-aligned Gaussian representation. Specifically, we first warp the features from source views to the target camera frustum using Eq. (3), and then utilize a pooling network ρ [22, 40] to aggregate these multi-view features into features $f_v = \rho(\{f_i\}_{i=1}^N)$. Considering the properties of splatting, each Gaussian contributes to the color values of pixels in a specific region of the image. However, the aggregated feature f_v only encodes multi-view information for individual pixels, lacking spatial awareness. Therefore, we utilize a 2D UNet for spatial enhancement, yielding f_g. With the encoded features, we can decode them to obtain Gaussian parameters for rendering. Specifically, each Gaussian is characterized by attributes $\{\mu, s, r, \alpha, c\}$ as described in Sec. 3. For the position μ, it can be obtained by unprojecting pixels according to the estimated depth as:

$$\mu = \Pi^{-1}(x, d)\,, \tag{4}$$

where Π^{-1} represents the unprojection operation. x and d represent the coordinates and estimated depth of the pixel, respectively. For scaling s, rotation r, and opacity α, they can be decoded from the encoded features, given by:

$$\begin{aligned} s &= Softplus(h_s(f_g))\,, \\ r &= Norm(h_r(f_g))\,, \\ \alpha &= Sigmoid(h_\alpha(f_g))\,, \end{aligned} \tag{5}$$

where h_s, h_r, and h_α represent the scaling head, rotation head, and opacity head, respectively, instantiated as MLPs. For the last attribute, color c, 3D Gaussian Splatting [19] utilizes spherical harmonic (SH) coefficients to define it. However, the generalization of learning SH coefficients from features is not robust (Sec. 5.4). Instead, we directly regress color from features as:

$$c = Sigmoid(h_c(f_g))\,, \tag{6}$$

where h_c represents the color head.

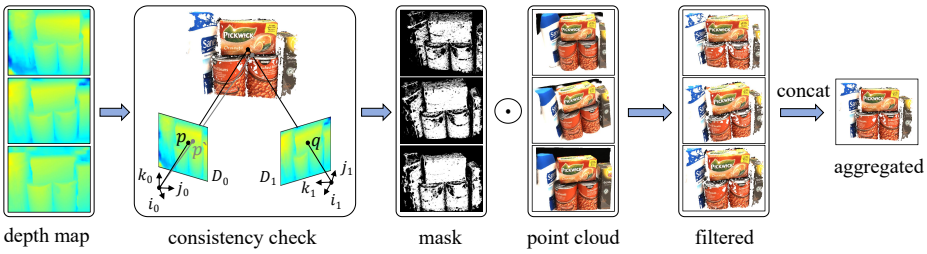


Fig. 3: Consistent aggregation. With depth maps and point clouds produced by the generalizable model, we first conduct geometric consistency checks on depths to derive masks for filtering out unreliable points. The filtered point clouds are then concatenated to obtain a point cloud, serving as the initialization for per-scene optimization.

Efficient Hybrid Gaussian Rendering. With the aforementioned Gaussian parameters, a novel view can be rendered using the splatting technique. However, the obtained view lacks fine details, and this approach exhibits limited generalization performance. Our insight is that the splatting approach introduces a complex many-to-many relationship between 3D Gaussians and pixels in terms of color contribution, which poses challenges for generalization. Therefore, we propose using a simple one-to-one correspondence between 3D Gaussians and pixels to predict colors for refinements. In this case, the splatting degenerates into the volume rendering with a single depth-aware sampling point. Specifically, following [22, 40], we obtain radiance and volume density by decoding f_v, followed by volume rendering to obtain a rendered view. The final rendered view is formed by averaging the views rendered through splatting and volume rendering.

4.3 Consistent Aggregation for Per-Scene Optimization

The generalizable model can reconstruct a reasonable 3D Gaussian representation for an unseen scene. We can further optimize this Gaussian representation for specific scenes using optimization strategies described in Sec. 3. Since the aforementioned generalizable model reconstructs Gaussian representations at several given novel viewpoints, the primary challenge is how to effectively aggregate these Gaussian representations into a single Gaussian representation for efficient rendering. Due to the inherent limitations of the MVS method, the depth predicted by the generalizable model may not be entirely accurate, leading to the presence of noise in the resulting Gaussian point cloud. Directly concatenating these Gaussian point clouds results in a significant amount of noise. Additionally, a large number of points slow down subsequent optimization and rendering speeds. An intuitive solution is to downsample the concatenated point cloud. However, while reducing noise, it also diminishes the number of effective points. Our insight is that a good aggregation strategy should minimize noisy points and retain effective ones as much as possible, while also ensuring that the total number of points is not excessively large. To this end, we introduce an aggregation strategy based on multi-view geometric consistency. The predicted depth for

the same 3D point across different viewpoints should demonstrate consistency. Otherwise, the predicted depth is considered unreliable. This geometric consistency can be measured by calculating the reprojection error between different views. Specifically, as illustrated in Fig 3, given a reference depth map D_0 to be examined and a depth map D_1 from a nearby viewpoint, we first project the pixel p in D_0 to the nearby view to obtain the projected point q as:

$$q = \frac{1}{d} \Pi_{0-1}(p, D_0(p)) \,, \tag{7}$$

where Π_{0-1} represents the transformation from D_0 to D_1, and d is the depth from projection. In turn, we back-project the obtained pixel q with estimated depth $D_1(q)$ onto the reference view to obtain the reprojected point p' as:

$$p' = \frac{1}{d'} \Pi_{1-0}(q, D_1(q)) \,, \tag{8}$$

where Π_{1-0} represents the transformation from D_1 to D_0, and d' is the depth of the reprojected pixel. Then, the reprojection errors are calculated by:

$$\begin{aligned} \xi_p &= \|p - p'\|_2 \,, \\ \xi_d &= \|D_0(p) - d'\|_1 / D_0(p) \,, \end{aligned} \tag{9}$$

The reference image will be compared pairwise with each of the remaining images to calculate the reprojection error. Inspired by [25, 48], we adopt the dynamic consistency checking algorithm to select the valid depth values. The main idea is that the estimated depth is reliable when it has very a low reprojection error in a minority of views or a relatively low error in the majority of views. It can be formulated as follows:

$$\xi_p < \theta_p(n) \,, \xi_d < \theta_d(n) \,, \tag{10}$$

where $\theta_p(n)$ and $\theta_d(n)$ represent predefined thresholds, whose values increase as the number of views n increases. The depth is reliable when there are n nearby views that meet the corresponding thresholds $\theta_p(n)$ and $\theta_d(n)$. We filter out noise points that do not meet the conditions and store the correctly reliable points.

4.4 Full Objective

Our model is trained end-to-end using only RGB images as supervision. We optimize the generalizable model with the mean squared error (mse) loss, SSIM loss [42], and perceptual loss [55], as follows:

$$L^k = L_{\text{mse}} + \lambda_s L_{\text{ssim}} + \lambda_p L_{\text{perc}} \,, \tag{11}$$

where L^k represents the loss for the k^{th} stage of the coarse-to-fine framework. λ_s and λ_p denote the loss weights. The overall loss is the sum of losses from each stage, given by:

$$L = \sum \lambda^k L^k \,, \tag{12}$$

Table 1: Quantitative results of generalization on the DTU test set [1]. FPS and Mem are measured under a 3-view input, while FPS* and Mem* are measured under a 2-view input. The best result is in **bold**, and the second-best one is in underlined.

Method	3-view			2-view			Mem (GB)↓	FPS↑
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓		
PixelNeRF [54]	19.31	0.789	0.382	-	-	-	-	0.019
IBRNet [40]	26.04	0.917	0.191	-	-	-	-	0.217
MVSNeRF [6]	26.63	0.931	0.168	24.03	0.914	0.192	-	0.416
ENeRF [22]	27.61	0.957	0.089	25.48	0.942	0.107	2.183	19.5
MatchNeRF [9]	26.91	0.934	0.159	25.03	0.919	0.181	-	1.04
PixelSplat [4]	-	-	-	14.01	0.662	0.389	11.827*	1.13*
Ours	**28.21**	**0.963**	**0.076**	**25.78**	**0.947**	**0.095**	0.876/0.866*	21.5/24.5*

where λ^k represents the loss weight for the k^{th} stage. During per-scene optimization, following [19], we optimize Gaussian point clouds using the L_1 loss combined with a D-SSIM term:

$$L_{\text{ft}} = (1 - \lambda_{\text{ft}})L_1 + \lambda_{\text{ft}} L_{\text{D-SSIM}}\,, \tag{13}$$

where λ_{ft} is the loss weight.

5 Experiments

5.1 Settings

Datasets. Following MVSNeRF [6], we train the generalizable model on the DTU training set [1] and evaluate it on the DTU test set. Subsequently, we conduct further evaluations on the Real Forward-facing [28], NeRF Synthetic [29], and Tanks and Temples [21] datasets. For each test scene, we select 20 nearby views, with 16 views comprising the working set and the remaining 4 views as testing views. The quality of synthesized views is measured by widely-used PSNR, SSIM [42], and LPIPS [55] metrics.

Baselines. We compare our method with state-of-the-art generalizable NeRF methods [6,9,22,40,54], as well as the recent generalizable Gaussian method [4]. For the generalization comparison, we follow the same experimental settings as [6,9,22] and borrow some results reported in [6,9]. For [22] and [4], we evaluate them using their officially released code and pre-trained models. For per-scene optimization experiments, we include NeRF [29] and 3D-GS [19] for comparison.

Implementation Details. Following [22], we employ a two-stage cascaded framework. For depth estimation, we sample 64 and 8 depth planes for the coarse and fine stages, respectively. We set $\lambda_s = 0.1$ and $\lambda_p = 0.05$ in Eq. (11), $\lambda^1 = 0.5$ and $\lambda^2 = 1$ in Eq. (12), and $\lambda_{ft} = 0.2$ in Eq. (13). The generalizable model is trained using the Adam optimizer [20] on four RTX 3090 GPUs. During the per-scene optimization stage, for fair comparison, our optimization strategy and hyperparameters settings remain consistent with the vanilla 3D-GS [19], except for the number of iterations. For the initialization of 3D-GS, we use COLMAP [33] to reconstruct the point cloud from the working set.

Table 2: Quantitative results of generalization on Real Forward-facing [28], NeRF Synthetic [29], and Tanks and Temples [21] datasets. Due to the significant memory consumption of PixelSplat [4], we conduct performance evaluation and comparison on low-resolution (512×512) images, denoted as PixelSplat* and Ours*. The best result is in **bold**, and the second-best one is in underlined.

Method	Settings	Real Forward-facing [28]			NeRF Synthetic [29]			Tanks and Temples [21]		
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
PixelNeRF [54]	3-view	11.24	0.486	0.671	7.39	0.658	0.411	-	-	-
IBRNet [40]		21.79	0.786	0.279	22.44	0.874	0.195	20.74	0.759	0.283
MVSNeRF [6]		21.93	0.795	0.252	23.62	0.897	0.176	20.87	0.823	0.260
ENeRF [22]		23.63	0.843	0.182	26.17	0.943	0.085	22.53	0.854	0.184
MatchNeRF [9]		22.43	0.805	0.244	23.20	0.897	0.164	20.80	0.793	0.300
Ours		**24.07**	**0.857**	**0.164**	**26.46**	**0.948**	**0.071**	**23.28**	**0.877**	**0.139**
MVSNeRF [6]	2-view	20.22	0.763	0.287	20.56	0.856	0.243	18.92	0.756	0.326
ENeRF [22]		22.78	0.821	0.191	24.83	0.931	0.117	22.51	0.835	0.193
MatchNeRF [9]		20.59	0.775	0.276	20.57	0.864	0.200	19.88	0.773	0.334
Ours		**23.11**	**0.834**	**0.175**	**25.06**	**0.937**	**0.079**	**22.67**	**0.844**	**0.162**
PixelSplat* [4]	2-view	22.99	0.810	0.190	15.77	0.755	0.314	19.40	0.689	0.223
Ours*		23.30	0.835	0.152	25.34	0.935	0.071	23.18	0.849	0.130

Table 3: Quantitative results after per-scene optimization. Time_{ft} represents the time for fine-tuning. The best result is in **bold**, and second-best one is in underlined.

Method	Optimization	Real Forward-facing [28]					NeRF Synthetic [29]					Tanks and Temples [21]				
		PSNR ↑	SSIM ↑	LPIPS ↓	Time_{ft} ↓	FPS ↑	PSNR ↑	SSIM ↑	LPIPS ↓	Time_{ft} ↓	FPS ↑	PSNR ↑	SSIM ↑	LPIPS ↓	Time_{ft} ↓	FPS ↑
NeRF [29]	Pipeline	25.97	0.870	0.236	10.2h	0.08	30.63	0.962	0.093	10.2h	0.07	21.42	0.702	0.558	10.2h	0.08
IBRNet [40]		24.88	0.861	0.189	1.0h	0.10	25.62	0.939	0.111	1.0h	0.10	22.22	0.813	0.221	1.0h	0.10
MVSNeRF [6]		25.45	0.877	0.192	15min	0.20	27.07	0.931	0.168	15min	0.19	21.83	0.841	0.235	15min	0.20
ENeRF [22]		24.89	0.865	0.159	1.0h	11.7	27.57	0.954	0.063	1.0h	10.5	24.18	0.885	0.145	1.0h	11.7
Ours		25.92	0.891	0.135	1.0h	14.1	27.87	0.956	0.061	1.0h	12.5	24.35	0.888	**0.125**	1.0h	14.0
3D-GS$_{7k}$ [19]	Gaussians	22.15	0.808	0.243	2min	370	32.15	0.971	0.048	1min15s	450	20.13	0.778	0.319	2min30s	320
3D-GS$_{30k}$ [19]		23.92	0.822	0.213	10min	350	31.87	0.969	0.050	7min	430	23.65	0.867	0.184	15min	270
Ours		**26.98**	**0.913**	**0.113**	45s	350	**32.20**	**0.972**	**0.043**	50s	470	**24.58**	**0.903**	0.137	90s	330

5.2 Generalization Results

We train the generalizable model on the DTU training set and report quantitative results on the DTU test set in Table 1, and the quantitative results on three additional datasets in Table 2. Due to the MVS-based pixel-aligned Gaussian representation and the efficient hybrid Gaussian rendering, our method achieves optimal performance at a fast inference speed. Due to the introduction of the epipolar Transformer, PixelSplat [4] has slow speed and large memory consumption. Additionally, it focuses on natural scenes with image pairs as input, and its performance significantly decreases when applied to object-centric datasets [1, 29]. For NeRF-based methods, ENeRF [22] enjoys promising speeds by sampling only 2 points per ray, however, its performance is limited and consumes higher memory overhead. The remaining methods render images by sampling rays due to their high memory consumption, as they cannot process the entire image at once. The qualitative results are presented in Fig. 4. Our method produces high-quality views with more scene details and fewer artifacts.

Fig. 4: Qualitative comparison of rendering quality under generalization and 3-view settings with state-of-the-art methods [6, 9, 22].

Table 4: Ablation studies. The terms "gs" and "vr" represent Gaussian Splatting and volume rendering, respectively. PSNR_{dtu}, PSNR_{llff}, PSNR_{nerf}, and PSNR_{tnt} are the PSNR metrics for different datasets [1, 21, 28, 29].

	Cascade	Decoding	Color	PSNR_{dtu}	PSNR_{llff}	PSNR_{nerf}	PSNR_{tnt}
No.1	✗	gs	rgb	26.71	22.57	24.90	21.06
No.2	✓	gs	rgb	27.48	23.15	25.48	21.70
No.3	✓	vr	rgb	27.39	23.80	25.65	22.76
No.4	✓	gs+vr	rgb	28.21	24.07	26.46	23.28
No.5	✓	gs+vr	sh	28.19	23.74	24.27	22.70

5.3 Per-Scene Optimization Results

The quantitative results after per-scene optimization are reported in Table 3. For per-scene optimization, one strategy is to optimize the entire pipeline, similar to NeRF-based methods. Another approach is to optimize only the initial Gaussian point cloud provided by the generalizable model. When optimizing the entire pipeline, our method can achieve better performance with faster inference speeds compared to previous generalizable NeRF methods, and results comparable to NeRF, demonstrating the robust representation capabilities of our method. In contrast, optimizing only the Gaussians can significantly improve optimization and rendering speed because it eliminates the time-consuming feed-forward neural network. Moreover, performance can benefit from the adaptive density control module described in Sec. 3. Due to the excellent initialization provided

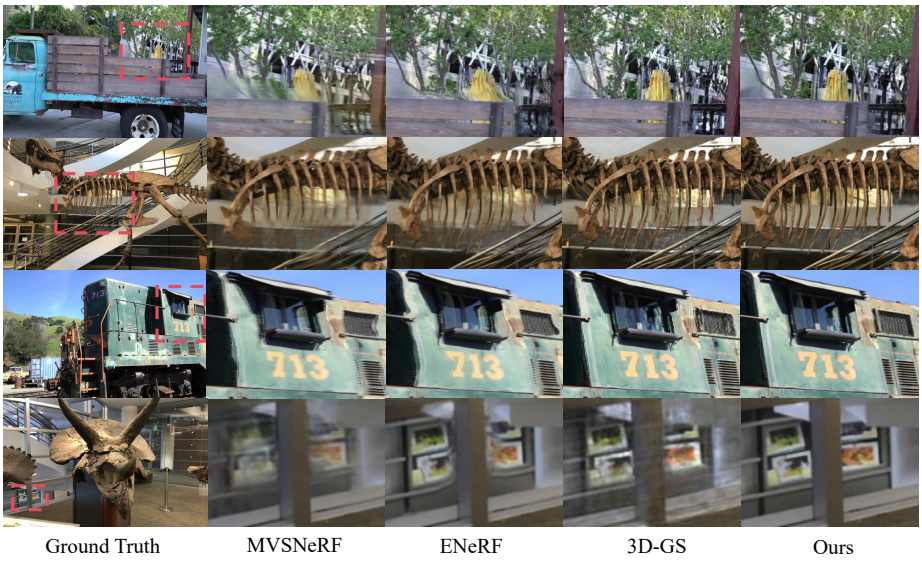
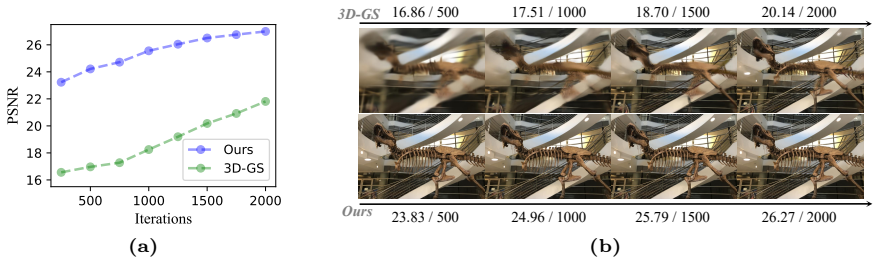


Fig. 5: Qualitative comparison of rendering quality with state-of-the-art methods [6, 19, 22] after per-scene optimization.

by the generalizable model and the effective aggregation strategy, we achieve optimal performance within a short optimization period, approximately one-tenth of that of 3D-GS. Especially on the Real Forward-facing dataset, our method achieves superior performance with only 45 seconds of optimization, compared to 10 minutes for 3D-GS and 10 hours for NeRF. Additionally, our method's inference speed is comparable to that of 3D-GS and significantly outperforms NeRF-based methods. As shown in Fig. 5, our method is capable of producing high-fidelity views with finer details.

5.4 Ablations and Analysis

Ablation studies. As shown in Table 4, we conduct ablation studies to evaluate the effectiveness of our designs. Firstly, comparing No.1 and No.2, the cascaded structure demonstrates a significant role. Additionally, adopting the hybrid Gaussian rendering approach (No.4) notably enhances performance compared to utilizing splatting (No.2) or volume rendering (No.3) alone. Regarding color representation, we directly decode RGB values instead of spherical harmonic (SH) coefficients (No.5), as decoding coefficients may result in a degradation of generalization, especially notable on the NeRF Synthetic dataset.
Aggregation strategies. As shown in Table 5, we investigate the impact of different point cloud aggregation strategies, which provide varying qualities of initialization and significantly affect subsequent optimization. The direct concatenation approach leads to an excessively large initial point set, hindering optimization and rendering speeds. Downsampling the point cloud can mitigate



(a) (b)

Fig. 6: Analysis of the Optimization process. (a) The evolution of view quality (PSNR) on the Real Forward-facing [28] dataset during the first 2000 iterations of our method and 3D-GS [19]. (b) Qualitative comparison of our method (bottom) and 3D-GS (top) on the "trex" scene, where (PSNR/iteration number) is shown.

Table 5: Comparison of different aggregation strategies. We report the quantitative results obtained with different strategies on the Real Forward-facing dataset [28]. For downsampling aggregation, we employ widely-used voxel downsampling with a voxel size set to 2. The iteration number for all aggregation strategies is set to 2.5k.

Aggregation	PSNR ↑	SSIM ↑	LPIPS ↓	Time$_{ft}$ ↓	FPS ↑
direct concatenation	26.18	0.901	0.122	90s	220
downsampling	26.72	0.909	0.121	60s	340
consistency check	26.98	0.913	0.113	45s	350

this issue while also improving performance, as it reduces contamination from noisy points. However, performance remains limited as it also simultaneously reduces some valid points. Employing the consistency check strategy can further boost performance, as it filters out noisy points while preserving valid points.

Optimization process. We illustrate the optimization process in Fig. 6. Thanks to the excellent initialization provided by the generalizable model, our method quickly attains good performance and rapidly improves.

6 Conclusion

We present MVSGaussian, an efficient generalizable Gaussian Splatting approach. Specifically, we leverage MVS to infer depth, establishing a pixel-aligned Gaussian representation. To enhance generalization, we propose a hybrid rendering approach that integrates depth-aware volume rendering. Besides, thanks to high-quality initialization, our models can be fine-tuned quickly for specific scenes. Compared with generalizable NeRFs, which typically require minutes of fine-tuning and seconds of rendering per image, MVSGaussian achieves real-time rendering with superior synthesis quality. Moreover, compared with 3D-GS, MVSGaussian achieves better view synthesis with reduced training time.

Limitations. As our method relies on MVS for depth estimation, it inherits limitations from MVS, such as decreased depth accuracy in areas with weak textures or specular reflections, resulting in degraded view quality.

References

1. Aanaes, H., Jensen, R.R., Vogiatzis, G., Tola, E., Dahl, A.B.: Large-scale data for multiple-view stereopsis. IJCV **120**, 153–168 (2016)
2. Boss, M., Braun, R., Jampani, V., Barron, J.T., Liu, C., Lensch, H.: Nerd: Neural reflectance decomposition from image collections. In: ICCV. pp. 12684–12694 (2021)
3. Cen, J., Fang, J., Yang, C., Xie, L., Zhang, X., Shen, W., Tian, Q.: Segment any 3d gaussians. arXiv preprint arXiv:2312.00860 (2023)
4. Charatan, D., Li, S., Tagliasacchi, A., Sitzmann, V.: pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In: arXiv (2023)
5. Chen, A., Liu, R., Xie, L., Chen, Z., Su, H., Yu, J.: Sofgan: A portrait image generator with dynamic styling. ACM Trans. Graph. **41**(1), 1–26 (2022)
6. Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., Su, H.: Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In: ICCV. pp. 14124–14133 (2021)
7. Chen, G., Wang, W.: A survey on 3d gaussian splatting. arXiv preprint arXiv:2401.03890 (2024)
8. Chen, Y., Chen, Z., Zhang, C., Wang, F., Yang, X., Wang, Y., Cai, Z., Yang, L., Liu, H., Lin, G.: Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. arXiv preprint arXiv:2311.14521 (2023)
9. Chen, Y., Xu, H., Wu, Q., Zheng, C., Cham, T.J., Cai, J.: Explicit correspondence matching for generalizable neural radiance fields. arXiv preprint arXiv:2304.12294 (2023)
10. Cheng, S., Xu, Z., Zhu, S., Li, Z., Li, L.E., Ramamoorthi, R., Su, H.: Deep stereo using adaptive thin volume representation with uncertainty awareness. In: CVPR. pp. 2524–2534 (2020)
11. Ding, Y., Yuan, W., Zhu, Q., Zhang, H., Liu, X., Wang, Y., Liu, X.: Transmvsnet global context-aware multi-view stereo network with transformers. In: CVPR. pp. 8585–8594 (2022)
12. Fua, P., Leclerc, Y.G.: Object-centered surface reconstruction combining multi-image stereo and shading. IJCV **16**(ARTICLE), 35–56 (1995)
13. Galliani, S., Lasinger, K., Schindler, K.: Massively parallel multiview stereopsis by surface normal diffusion. In: ICCV. pp. 873–881 (2015)
14. Gu, X., Fan, Z., Zhu, S., Dai, Z., Tan, F., Tan, P.: Cascade cost volume for high-resolution multi-view stereo and stereo matching. In: CVPR. pp. 2495–2504 (2020)
15. Hu, L., Zhang, H., Zhang, Y., Zhou, B., Liu, B., Zhang, S., Nie, L.: Gaussianavatar: Towards realistic human avatar modeling from a single video via animatable 3d gaussians. arXiv preprint arXiv:2312.02134 (2023)
16. Hu, S., Liu, Z.: Gauhuman: Articulated gaussian splatting from monocular human videos. arXiv preprint arXiv: (2023)
17. Hu, S., Zhou, K., Li, K., Yu, L., Hong, L., Hu, T., Li, Z., Lee, G.H., Liu, Z.: Consistentnerf: Enhancing neural radiance fields with 3d consistency for sparse view synthesis. arXiv preprint arXiv:2305.11031 (2023)
18. Irshad, M.Z., Zakharov, S., Liu, K., Guizilini, V., Kollar, T., Gaidon, A., Kira, Z., Ambrus, R.: Neo 360: Neural fields for sparse view synthesis of outdoor scenes (2023)
19. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (2023)

20. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
21. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples benchmarking large-scale scene reconstruction. ACM Trans. Graph. **36**(4), 1–13 (2017)
22. Lin, H., Peng, S., Xu, Z., Yan, Y., Shuai, Q., Bao, H., Zhou, X.: Efficient neural radiance fields for interactive free-viewpoint video. In: SIGGRAPH Asia Conference Proceedings (2022)
23. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR. pp. 2117–2125 (2017)
24. Liu, T., Ye, X., Shi, M., Huang, Z., Pan, Z., Peng, Z., Cao, Z.: Geometry-aware reconstruction and fusion-refined rendering for generalizable neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7654–7663 (June 2024)
25. Liu, T., Ye, X., Zhao, W., Pan, Z., Shi, M., Cao, Z.: When epipolar constraint meets non-local operators in multi-view stereo. In: ICCV. pp. 18088–18097 (2023)
26. Liu, Y., Peng, S., Liu, L., Wang, Q., Wang, P., Theobalt, C., Zhou, X., Wang, W.: Neural rays for occlusion-aware image-based rendering. In: CVPR (2022)
27. Luiten, J., Kopanas, G., Leibe, B., Ramanan, D.: Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In: 3DV (2024)
28. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Trans. Graph. **38**(4), 1–14 (2019)
29. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020)
30. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. In: ICCV. pp. 5865–5874 (2021)
31. Peng, R., Wang, R., Wang, Z., Lai, Y., Wang, R.: Rethinking depth estimation for multi-view stereo a unified representation. In: CVPR. pp. 8645–8654 (2022)
32. Qian, Z., Wang, S., Mihajlovic, M., Geiger, A., Tang, S.: 3dgs-avatar: Animatable avatars via deformable 3d gaussian splatting. arXiv preprint arXiv:2312.09228 (2023)
33. Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: CVPR. pp. 4104–4113 (2016)
34. Schonberger, J.L., Zheng, E., Frahm, J.M., Pollefeys, M.: Pixelwise view selection for unstructured multi-view stereo. In: ECCV. pp. 501–518. Springer (2016)
35. Szymanowicz, S., Rupprecht, C., Vedaldi, A.: Splatter image: Ultra-fast single-view 3d reconstruction. In: arXiv (2023)
36. T, M.V., Wang, P., Chen, X., Chen, T., Venugopalan, S., Wang, Z.: Is attention all that neRF needs? In: ICLR (2023)
37. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
38. Wang, G., Chen, Z., Loy, C.C., Liu, Z.: Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. In: IEEE/CVF International Conference on Computer Vision (ICCV) (2023)
39. Wang, G., Wang, P., Chen, Z., Wang, W., Loy, C.C., Liu, Z.: Perf: Panoramic neural radiance field from a single panorama. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2024)

40. Wang, Q., Wang, Z., Genova, K., Srinivasan, P., Zhou, H., Barron, J.T., Martin-Brualla, R., Snavely, N., Funkhouser, T.: Ibrnet: Learning multi-view image-based rendering. In: CVPR (2021)
41. Wang, X., Zhu, Z., Huang, G., Qin, F., Ye, Y., He, Y., Chi, X., Wang, X.: Mvster epipolar transformer for efficient multi-view stereo. In: ECCV. pp. 573–591. Springer (2022)
42. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE TIP **13**(4), 600–612 (2004)
43. Wei, Z., Zhu, Q., Min, C., Chen, Y., Wang, G.: Aa-rmvsnet adaptive aggregation recurrent multi-view stereo network. In: ICCV. pp. 6187–6196 (2021)
44. Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., Xinggang, W.: 4d gaussian splatting for real-time dynamic scene rendering. arXiv preprint arXiv:2310.08528 (2023)
45. Xian, W., Huang, J.B., Kopf, J., Kim, C.: Space-time neural irradiance fields for free-viewpoint video. In: CVPR. pp. 9421–9431 (2021)
46. Xiang, F., Xu, Z., Hasan, M., Hold-Geoffroy, Y., Sunkavalli, K., Su, H.: Neutex: Neural texture mapping for volumetric neural rendering. In: CVPR. pp. 7119–7128 (2021)
47. Yan, J., Wei, Z., Yi, H., Ding, M., Zhang, R., Chen, Y., Wang, G., Tai, Y.W.: Dense hybrid recurrent multi-view stereo net with dynamic consistency checking. In: ECCV. pp. 674–689. Springer (2020)
48. Yan, J., Wei, Z., Yi, H., Ding, M., Zhang, R., Chen, Y., Wang, G., Tai, Y.W.: Dense hybrid recurrent multi-view stereo net with dynamic consistency checking. In: ECCV. pp. 674–689. Springer (2020)
49. Yang, J., Mao, W., Alvarez, J.M., Liu, M.: Cost volume pyramid based depth inference for multi-view stereo. In: CVPR. pp. 4877–4886 (2020)
50. Yang, Z., Gao, X., Zhou, W., Jiao, S., Zhang, Y., Jin, X.: Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. arXiv preprint arXiv:2309.13101 (2023)
51. Yao, Y., Luo, Z., Li, S., Fang, T., Quan, L.: Mvsnet depth inference for unstructured multi-view stereo. In: ECCV. pp. 767–783 (2018)
52. Yao, Y., Luo, Z., Li, S., Shen, T., Fang, T., Quan, L.: Recurrent mvsnet for high-resolution multi-view stereo depth inference. In: CVPR. pp. 5525–5534 (2019)
53. Ye, X., Zhao, W., Liu, T., Huang, Z., Cao, Z., Li, X.: Constraining depth map geometry for multi-view stereo: A dual-depth approach with saddle-shaped depth cells. In: ICCV. pp. 17661–17670 (2023)
54. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelNeRF: Neural radiance fields from one or few images. In: CVPR (2021)
55. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR. pp. 586–595 (2018)
56. Zheng, S., Zhou, B., Shao, R., Liu, B., Zhang, S., Nie, L., Liu, Y.: Gps-gaussian: Generalizable pixel-wise 3d gaussian splatting for real-time human novel view synthesis. arXiv (2023)