3DGStream: On-the-Fly Training of 3D Gaussians for Efficient Streaming of Photo-Realistic Free-Viewpoint Videos

Jiakai Sun,   Han Jiao,   Guangyuan Li,   Zhanjie Zhang,   Lei Zhao,*   Wei Xing*

Zhejiang University

{csjk, csjh, cslgy, cszzj, cszhl, wxing}@zju.edu.cn
https://sjojok.github.io/3dgstream

Abstract

Constructing photo-realistic Free-Viewpoint Videos (FVVs) of dynamic scenes from multi-view videos remains a challenging endeavor. Despite the remarkable advancements achieved by current neural rendering techniques, these methods generally require complete video sequences for offline training and are not capable of real-time rendering. To address these constraints, we introduce 3DGStream, a method designed for efficient FVV streaming of real-world dynamic scenes. Our method achieves fast on-the-fly per-frame reconstruction within 12 seconds and real-time rendering at 200 FPS. Specifically, we utilize 3D Gaussians (3DGs) to represent the scene. Instead of the naïve approach of directly optimizing 3DGs per-frame, we employ a compact Neural Transformation Cache (NTC) to model the translations and rotations of 3DGs, markedly reducing the training time and storage required for each FVV frame. Furthermore, we propose an adaptive 3DG addition strategy to handle emerging objects in dynamic scenes. Experiments demonstrate that 3DGStream achieves competitive performance in terms of rendering speed, image quality, training time, and model storage when compared with state-of-the-art methods.

1. Introduction

Constructing Free-Viewpoint Videos (FVVs) from videos captured by a set of known-poses cameras from multiple views remains a frontier challenge within the domains of computer vision and graphics. The potential value and application prospects of this task in the VR/AR/XR domains have attracted much research. Traditional approaches predominantly fall into two categories: geometry-based methods that explicitly reconstruct dynamic graphics primitives [15, 17], and image-based methods that obtain new views through interpolation [7, 76]. However, these conventional methods struggle to handle real-world scenes charac-

(a) **I-NGP** [40]: Per-frame training   (b) **HyperReel** [1]: Offline training

(c) **StreamRF** [29]: Online training   (d) **Ours**: Online training

Figure 1. **Comparison on the *flame steak* scene of the N3DV dataset [31].** The training time, requisite storage, and PSNR are computed as averages over the whole video. Our method stands out by the ability of fast online training and real-time rendering, standing competitive in both model storage and image quality.

terized by complex geometries and appearance.

In recent years, Neural Radiance Fields (NeRFs) [36] has garnered significant attention due to its potent capabilities in synthesizing novel views as a 3D volumetric representation. A succession of NeRF-like works [19, 29, 31–33, 43–46, 48, 61, 68] further propelled advancements in constructing FVVs on dynamic scenes. Nonetheless, the vast majority of NeRF-like FVV construction methods encountered two primary limitations: (1) they typically necessitate complete video sequences for time-consuming offline training, meaning they can replay dynamic scenes but are unable to stream them, and (2) they generally fail to achieve real-time rendering, thereby hindering practical applications.

Recently, Kerbl *et al.* [26] have achieved real-time radiance field rendering using 3D Gaussians (3DGs), thus enabling the instant synthesis of novel views in static scenes
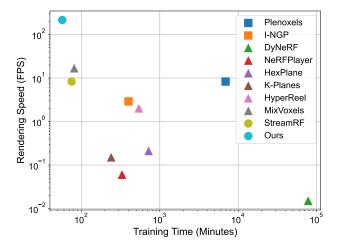
---

*Corresponding authors.

1

Figure 2. **Comparison of our method with other methods on the N3DV dataset [31].** □ denotes training from scratch per frame, △ represents offline training on complete video sequences, and ◯ signifies online training on video streams. While achieving online training , our method reaches state-of-the-art performance in both rendering speed and overall training time.

with just minutes of training. Inspired by this breakthrough, we propose 3DGStream, a method that utilizes 3DGs to construct Free-Viewpoint Videos (FVVs) of dynamic scenes. Specifically, we first train the initial 3DGs on the multi-view frames at timestep 0. Then, for each timestep i, we use the 3DGs of previous timestep $i - 1$ as initialization and pass it to a two-stage pipeline. (1) In Stage 1, we train a Neural Transformation Cache (NTC) to model the transformations of 3DGs. (2) Then in the Stage 2, we use an adaptive 3DG addition strategy to handle emerging objects by spawning frame-specific additional 3DGs near these objects and optimize them along with periodic splitting and pruning. After the two-stage pipeline concludes, we use both the 3DGs transformed by the NTC and the additional 3DGs for rendering at the current timestep i, with only the former carrying over for initialization of the subsequent timestep. This design significantly reduces the storage requirements for the FVV, as we only need to store the per-frame NTCs and frame-specific additions, rather than all 3DGs for each frame.

3DGStream is capable of rendering photo-realistic FVVs at megapixel resolution in real-time, boasting exceptionally rapid per-frame training speeds and limited model storage requirements. As illustrated in Figs. 1 and 2, compared with static reconstruction methods that train from scratch per-frame and dynamic reconstruction methods that necessitate offline training across the complete video sequences, our approach excels in both training speed and rendering speed, maintaining a competitive edge in image quality and model storage. Furthermore, our method outperforms StreamRF [29], a state-of-the-art technique tackling the exactly same task, in all the relevant aspects.

To summarize, our contributions include:
- We propose 3DGStream, a method for on-the-fly construction of photo-realistic, real-time renderable FVV on video streams, eliminating the necessity for lengthy offline training on the entire video sequences.
- We utilize NTC for modeling the transformations of 3DGs, in conjunction with an adaptive 3DG addition strategy to tackle emerging objects within dynamic scenes. This combination permits meticulous manipulation of 3DGs, accommodating scene alterations with limited performance overhead.
- We conduct extensive experiments to demonstrate 3DGStream's competitive edge in rendering quality, training time, and requisite storage, as well as its superior rendering speed, compared to existing state-of-the-art dynamic scene reconstruction methods.

2. Related Work

2.1. Novel View Synthesis for Static Scenes

Synthesizing novel views from a set of images of static scenes is a time-honored problem in the domains of computer vision and graphics. Traditional methods such as Lumigraph [8, 22] or Light-Field [10, 16, 28, 50] achieve new view synthesis through interpolation. In recent years, Neural Radiance Fields (NeRF) [36] has achieved photorealistic synthesizing results by representing the radiance field using a multi-layer perceptron (MLP). A series of subsequent works enhance NeRF's performance in various aspects, such as accelerating training speeds [12, 13, 20, 25, 40, 53], achieving real-time rendering [14, 21, 23, 47, 65, 73], and improving synthesis quality on challenging scenes [2–4, 35, 37, 57] or sparse inputs [11, 41, 54, 64, 67, 70, 74]. Since the vanilla NeRF employs costly volume rendering, necessitating neural network queries for rendering, subsequent approaches faced trade-offs in training time, rendering speed, model storage, image quality, and applicability. To address these challenges, Kerbl *et al.* [26] propose 3D Gaussian Splatting (3DG-S), which integrates of 3DGs with differentiable point-based rendering. 3DG-S enables real-time high-fidelity view synthesis in large-scale unbounded scenes after brief training periods with modest storage requirements. Inspired by this work, we extend its application to the task of constructing FVVs of dynamic scenes. Taking it a step further, we design a on-the-fly training framework to achieve efficient FVV streaming.

2.2. Free-Viewpoint Videos of Dynamic Scenes

Constructing FVVs from a set of videos of dynamic scenes is a more challenging and applicable task in the domains of computer vision and graphics. Earlier attempts to address this task pivoted around the construction of dynamic primitives [15, 17] or resorting to interpolation [7, 76]. With the
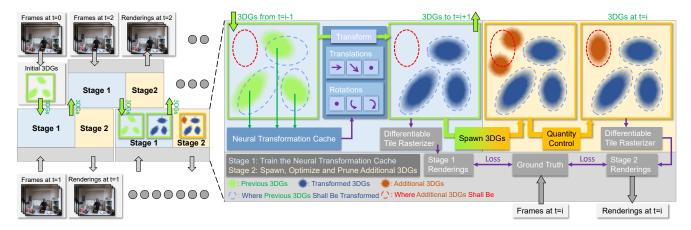
Figure 3. **Overview of 3DGStream.** Given a set of multi-view video streams, 3DGStream aims to construct high-quality FVV stream of the captured dynamic scene on-the-fly. Initially, we optimize a set of 3DGs to represent the scene at timestep 0. For each subsequent timestep i, we use the 3DGs from timestep $i-1$ as an initialization and then engage in a two-stage training process: **Stage 1**: We train the Neural Transformation Cache (NTC) to model the translations and rotations of 3DGs. After training, the NTC transforms the 3DGs, preparing them for the next timestep and the next stage in the current timestep. **Stage 2**: We spawn frame-specific additional 3DGs at potential locations and optimize them along with periodic splitting and pruning. After the two-stage process concludes, both transformed and additional 3DGs are used to render at the current timestep i, with only the transformed ones carried into the next timestep.

success of NeRF-like methods in novel view synthesis for static scenes, a series of works [1, 9, 19, 29–34, 42, 44–46, 48, 52, 56, 58, 60, 62, 63, 69, 75] attempt to use NeRF for constructing FVVs in dynamic scenes. These works can typically be categorized into five types: prior-driven, flow-based, warp-based, those using spatio-temporal inputs, and per-frame training.

Prior-driven methods [27, 30, 63, 69, 75] leverage parametric models or incorporate additional priors, such as skeletons, to bolster performance on the reconstruction of specific dynamic objects, *e.g.*, humans. However, their application is limited and not generalizable to broader scenes.

Flow-based methods [32, 33] primarily focus on constructing FVVs from monocular videos. By estimating the correspondence of 3D points in consecutive frames, they achieve impressive results. Nonetheless, the intrinsic ill-posedness of monocular reconstructions in intricate dynamic scenes frequently calls for supplementary priors like depth, optical flow, and motion segmentation masks.

Warp-based methods [1, 42, 44, 46, 52, 56, 62] assumpt that the dynamics of the scene arise from the deformation of static structures. These methods warp the radiance field of each frame onto one or several canonical frames, achieving notable results. However, the strong assumptions they rely on often prevent them from handling topological variations.

Methods that use **spatio-temporal inputs** [9, 19, 31, 45, 48, 58, 59] enhance radiance fields by adding a temporal dimension, enabling the querying of the radiance field using spatio-temporal coordinates. While these techniques showcase a remarkable ability to synthesize new viewpoints in dynamic scenes, the entangled scene parameters can constrain their adaptability for downstream applications.

Per-frame training methods [29, 34, 60] adapt to changes in the scene online by leveraging per-frame training, a paradigm we have also adopted. To be specific, StreamRF [29] employs Plenoxels [20] for scene representation and achieves rapid on-the-fly training with minimal storage requirements through techniques like narrow band tuning and difference-based compression. ReRF [60] uses DVGO [53] for scene representation and optimize motion grid and residual grid frame by frame to model inter-frame discrepancies, enabling high-quality FVV streaming and rendering. Dynamic3DG [34] optimizes simplified 3DGs and integrates physically-based priors for high-quality novel view synthesis on dynamic scenes.

Among the aforementioned works, only NeRF-Player [52], ReRF [60], StreamRF [29], and Dynamic3DG [34] are able to stream FVVs. NeRFPlayer achieves FVV streaming through a decomposition module and a feature streaming module, but it is only able to stream pre-trained models. ReRF and Dynamic3DG are limited to processing scenes with few objects and foreground mask, necessitating minute-level per-frame training times. StreamRF stands out by requiring only a few seconds for each frame's training to construct high-fidelity FVVs on challenging real-world dynamic scenes with compressed model storage. However, it falls short in rendering speed. Contrarily, our approach matches or surpasses StreamRF in training speed, model storage, and image quality, all while achieving real-time rendering at 200 FPS.

2.3. Concurrent Works

Except for Dynamic3DG, several concurrent works have extended 3DG-S to represent dynamic scenes. De-

formable3DG [71] employs an MLP to model the deformation of 3DGs, while [66] introduces a hexplane-based encoder to enhance the efficency of deformation query. Meanwhile, [18, 72] lift 3DG to 4DG primitives for dynamic scene representation. However, these approaches are limited to offline reconstruction and lack streamable capabilities, whereas our work aims to achieve efficient streaming of FVVs with an online training paradigm.

3. Background: 3D Gaussian Splatting

3D Gaussian Splatting (3DG-S) [26] employs anisotropic 3D Gaussians as an explicit scene representation. Paired with a fast differentiable rasterizer, 3DGs achieves real-time novel view synthesis with only minutes of training.

3.1. 3D Gaussians as Scene Representation

A 3DG is defined by a covariance matrix Σ centered at point (*i.e.*, mean) μ:

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

To ensure positive semi-definiteness during optimization, the covariance matrix Σ is decomposed into a rotation matrix R and a scaling matrix S:

$$\Sigma = RSS^T R^T. \tag{2}$$

Rotation is conveniently represented by a unit quaternion, while scaling uses a 3D vector. Additionally, each 3DG contains a set of spherical harmonics (SH) coefficients of to represent view-dependent colors, along with an opacity value α, which is used in α-blending (Eq. (4)).

3.2. Splatting for Differentiable Rasterization

For novel view synthesis, 3DG-S [26] project 3DGs to 2D Gaussian (2DG) splats [77]:

$$\Sigma' = JW\Sigma W^T J^T. \tag{3}$$

Here, Σ' is the covariance matrix in camera coordinate. J is the Jacobian of the affine approximation of the projective transformation, and W is the viewing transformation matrix. By skipping the third row and third column of Σ', we can derive a 2×2 matrix denoted as Σ_{2d}. Furthermore, projecting the 3DG's mean, μ, into the image space results in a 2D mean, μ_{2d}. Consequently, this allows us to define the 2DG in the image space as $G_{2d}(x; \mu_{2d}, \Sigma_{2d})$.

Using Σ', the color C of a pixel can be computed by blending the N ordered points overlapping the pixel:

$$C = \sum_{i \in N} c_i \alpha'_i \prod_{j=1}^{i-1}(1 - \alpha'_j). \tag{4}$$

Here, c_i denotes the view-dependent color of the i-th 3DG. α'_i is determined by multiplying the opacity α_i of the i-th 3DG G with the evaluation of the corresponding 2DG G_{2d}.

Leveraging a highly-optimized rasterization pipeline coupled with custom CUDA kernels, the training and rendering of 3DG-S are remarkably fast. For instance, for megapixel-scale real-world scenes, just a few minutes of optimization allows 3DGs to achieve photo-realistic visual quality and rendering speeds surpassing 100 FPS.

4. Method

3DGStream constructs photo-realistic FVV streams from multi-view video streams on-the-fly using a per-frame training paradigm. We initiate the process by training 3DGs [26] at timestep 0. For subsequent timesteps, we employ the previous timestep's 3DGs as an initialization and pass them to a two-stage pipeline. Firstly (Sec. 4.1), a Neural Transformation Cache (NTC) is trained to model the transformation for each 3DG. Once the training is finished, we transform the 3DGs and carry the transformed 3DGs to the next timestep. Secondly (Sec. 4.2), we employ an adaptive 3DG addition strategy to handle emerging objects. For each FVV frame, we render views at the current timestep using both the transformed 3DGs and additional 3DGs, while the latter are not passed to the next timestep. Note that we only need to train and store the parameters of the NTC and the additional 3DGs for each subsequent timestep, not all the 3DGs. We depict an overview of our approach in Fig. 3.

4.1. Neural Transformation Cache

For NTC, we seek a structure that is compact, efficient, and adaptive to model the transformations of 3DGs. Compactness is essential to reduce the model storage. Efficiency enhances training and inference speeds. Adaptivity ensures the model focuses more on dynamic regions. Additionally, it would be beneficial if the structure could consider certain priors of dynamic scenes [5, 24, 55], such as the tendency for neighboring parts of an object to have similar motion.

Inspired by Neural Radiance Caching [39] and I-NGP [40], we employ multi-resolution hash encoding combined with a shallow fully-fused MLP [38] as the NTC. Specifically, following I-NGP, we use multi-resolution voxel grids to represent the scene. Voxel grids at each resolution are mapped to a hash table storing a d-dimensional learnable feature vector. For a given 3D position $x \in \mathbb{R}^3$, its hash encoding at resolution l, denoted as $h(x; l) \in \mathbb{R}^d$, is the linear interpolation of the feature vectors corresponding to the eight corners of the surrounding grid. Consequently, its multi-resolution hash encoding $h(x) = [h(x; 0), h(x; 1), ..., h(x; L-1)] \in \mathbb{R}^{Ld}$, where L represents the number of resolution levels. The multi-resolution hash encoding addresses all our requirements for the NTC:
- **Compactness**: Hashing effectively reduces the storage space needed for encoding the whole scene.
- **Efficiency**: Hash table lookup operates in $O(1)$, and is highly compatible with modern GPUs.

- **Adaptivity**: Hash collisions occur in hash tables at finer resolutions, allowing regions with larger gradients—representing dynamic regions in our context—to drive the optimization.
- **Priors**: The combination of linear interpolation and the voxel-grid structure ensures the local smoothness of transformations. Additionally, the multi-resolution approach adeptly merges global and local information.

Furthermore, to enhance the NTC's performance with minimal overhead, we utilize a shallow fully-fused MLP [38]. This maps the hash encoding to a 7-dimensional output: the first three dimensions indicate the translation of the 3DG; the remaining dimensions represent the rotation of the 3DG using quaternions. Given multi-resolution hash encoding coupled with MLP, our NTC is formalized as:

$$d\mu, dq = MLP(h(\mu)), \qquad (5)$$

where μ denotes the mean of the input 3DG. We transform the 3DGs based on $d\mu$ and dq. Specifically, the following parameters of the transformed 3DGs are given as:
- **Mean**: $\mu' = \mu + d\mu$, where μ' is the new mean and $+$ represents vector addition.
- **Rotation**: $q' = norm(q) \times norm(dq)$, where q' is the new roation, $\times$ indicates quaternion multiplication and $norm$ denotes normalization.
- **SH Coefficients**: Upon rotating the 3DG, the SH coefficients should also be adjusted to align with the rotation of the 3DG. Leveraging the rotation invariance of SH, we directly employ SH Rotation to update SHs. Please refer to the supplementary materials (Suppl.) for details.

In Stage 1, we transform the 3DGs from the previous frame by NTC and then render with them. The parameters of the NTC is optimized by the loss between the rendered image and the ground truth. Following 3DG-S [26], the loss function is L_1 combined with a D-SSIM term:

$$L = (1 - \lambda)L_1 + \lambda L_{D-SSIM}, \qquad (6)$$

where $\lambda = 0.2$ in all our experiments. It should be noted that during the training process, the 3DGs from the previous frame remain frozen and do not undergo any updates. This implies that the input to the NTC remains consistent.

Additionally, to ensure training stability, we initialize the NTC with warm-up parameters. The loss employed during the warm-up is defined as:

$$L_{warm-up} = ||d\mu||_1 - cos^2(norm(dq), Q), \qquad (7)$$

where Q is the identity quaternion. The first term uses the L_1 norm to ensure the estimated translation approaches zero, while the second term, leveraging cosine similarity, ensures the estimated rotation approximates no rotation. However, given the double-covering property of the unit quaternions, we use the square of the cosine similarity. For each scene, we execute the warm-up solely after the training at timestep 0, using noise-augmented means of the initial 3DGs as input. After 3000 iterations of training (roughly 20 seconds), the parameters are stored and used to initialize the NTCs for all the following timesteps.

4.2. Adaptive 3DG Addition

Relying solely on 3DGs transformations adequately cover a significant portion of real-world dynamic scenes, with translations effectively managing occlusions and disappearances in subsequent timesteps. However, this approach falters when faced with objects not present in the initial frame, such as transient objects like flames or smoke, and new persistent objects like the liquid poured out of a bottle. Since 3DG is an unstructured explicit representation, it's essential to add new 3DGs to model these emerging objects. Considering constraints related to model storage requirements and training complexities, it's not feasible to generate an extensive number of additional 3DGs nor allow them to be used in subsequent frames, as this would cause 3DGs to accumulate over time. This necessitates a strategy for swiftly generating a limited number of frame-specific 3DGs to model these emerging objects precisely and thereby enhance the completeness of the scene at the current timestep.

Firstly, we need to ascertain the locations for the emerging objects. Inspired by 3DG-S [26], we recognized the view-space positional gradients of 3DGs as a key indicator. We observed that for emerging objects, the 3DGs in proximity exhibited large view-space positional gradients. This is attributed to the optimization attempting to 'masquerade' the emerging object by transforming the 3DGs. However, since we prevent the colors of the 3DGs from being updated in Stage 1, this attempt falls short. Nonetheless, they are still transformed to appropriate positions, with large view-space positional gradients.

Based on the aforementioned observations, we deem it appropriate to introduce additional 3DGs around these high-gradient regions. Moreover, to exhaustively capture every potential location where new objects might emerge, we adopt an **adaptive** 3DG spawn strategy. Specifically, we track view-space positional gradient during the final training epoch of Stage 1. Once this stage concludes, we select 3DGs that have an average magnitude of view-space position gradients exceeding a relatively low threshold $\tau_{grad} = 0.00015$. For each selected 3DG, the position of the additional 3DG is sampled from $X \sim \mathcal{N}(\mu, 2\Sigma)$, where μ and Σ is the mean and the convariance matrix of the selected 3DG. While we avoid assumptions about the other attributes of the additional 3DGs, improper initializations of SH coefficients and scaling vectors tend to result in an optimization preference for reducing opacity over adjusting these parameters. This causes additional 3DGs to quickly become transparent, thereby failing to capture the emerging objects. To mitigate
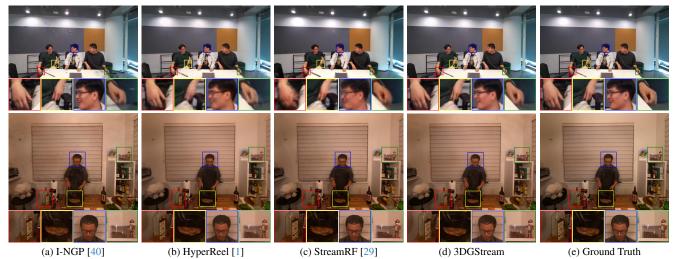
|(a) I-NGP [40] | (b) HyperReel [1] | (c) StreamRF [29] | (d) 3DGStream | (e) Ground Truth |

Figure 4. **Qualitative comparisons** on the *discussion* scene of the Meet Room dataset and the *sear steak* scene of the N3DV dataset.

Category	Method	PSNR↑ (dB)	Storage↓ (MB)	Train↓ (mins)	Render↑ (FPS)	Streamable
Static	Plenoxels [20]	30.77	4106	23	8.3	✓
	I-NGP [40]	28.62	48.2	1.3	2.9	✓
	3DG-S [26]	32.08	47.1	8.3	390	✓
Offline	DyNeRF [31]	29.58†	0.1	260	0.02	×
	NeRFPlayer [52]	30.69	17.1	1.2	0.05	✓
	HexPlane [9]	31.70	0.8	2.4	0.21	×
	K-Planes [48]	31.63	1.0	0.8	0.15	×
	HyperReel [1]	31.10	1.2	1.8	2.00	×
	MixVoxels [58]	30.80	1.7	0.27	16.7	×
Online	StreamRF [29]	30.68	17.7/31.4*	0.25	8.3	✓
	Ours	31.67	7.6/7.8*	0.20	215	✓

Table 1. **Quantitative comparison** on the N3DV dataset. The training time, required storage and PSNR are averaged over the whole 300 frames for each scene. †DyNeRF [31] only report metrics on the *flame salmon* scene. *Considering the initial model.

Method	PSNR↑ (dB)	Storage↓ (MB)	Train↓ (mins)	Render↑ (FPS)
Plenoxels [20]	27.15	1015	14	10
I-NGP [40]	28.10	48.2	1.1	4.1
3DG-S [26]	31.31	21.1	2.6	571
StreamRF [29]	26.72	5.7/9.0*	0.17	10
Ours	30.79	4.0/4.1*	0.10	288

Table 2. **Quantitative comparison** on the Meet Room dataset. Note that the training time, required storage and PSNR are averaged over the whole 300 frames. *Considering the initial model.

this issue, the SH coefficients and scaling vectors of these 3DGs are derived from the selected ones, with rotations set to the identity quaternion q = [1, 0, 0, 0] and opacity initialized at 0.1. After spawning, the 3DGs undergo optimization utilizing the same loss function (Eq. (6)) as Stage 1. Note that only the parameters of the additional 3DGs are optimized, while those of the transformed 3DGs remain fixed.

To guard against local minima and manage the number of additional 3DGs, we implement an **adaptive** 3DG quantity control strategy. Specifically, in Stage 2, we set a relatively high threshold, $\tau_\alpha = 0.01$, for the opacity value. At the end of each training epoch, for 3DGs with view-space position gradients exceeding τ_{grad}, we spawn additional 3DGs nearby to address under-reconstructed regions. These additional 3DGs inherit their rotations and SH coefficients from the original 3DG, but their scaling is adjusted to 80% of the original, mirroring the 'split' operation described by Kerbl et al. [26]. Subsequently, we discard any additional 3DGs with opacity values below τ_α to suppress the growth in the quantity of 3DGs.

5. Experiments

5.1. Datasets

We conduct experiments on two real-world dynamic scene datasets: N3DV dataset [31] and Meet Room dataset [29].

N3DV dataset [31] is captured using a multi-view system of 21 cameras, comprises dynamic scenes recorded at a resolution of 2704×2028 and 30 FPS. Following previous works [9, 29, 31, 48, 52, 58], we downsample the videos by a factor of two and follow the training and validation camera split provided by [31].

Meet Room dataset [29] is captured using a 13-camera multi-view system, comprises dynamic scenes recorded at a resolution of 1280×720 and 30 FPS. Following [29], we utilize 13 views for training and reserved 1 for testing.

5.2. Implementation

We implement 3DGStream upon the codes of 3D Gaussian Splatting (3DG-S) [26], and implement the Neural Transformation Cache (NTC) using tiny-cuda-nn [38]. For the training of initial 3DGs, we fine-tune the learning rates on the N3DV dataset based on the default settings of 3DG-S, and apply them to the Meet Room dataset. For all scenes,
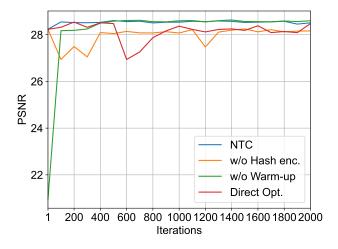
Figure 5. **Comparison of different approaches for modeling the transformation of 3DGs.** Conducted on the second frame of the *flame salmon* video, utilizing identical initial 3DGs.
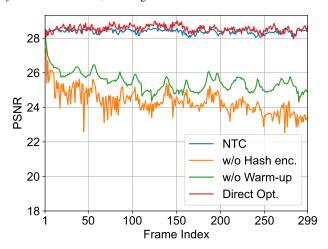

Figure 6. **Comparison of different approaches on the** *flame salmon* **scene.**

Variant	PSNR↑ (dB)	#Additional 3DGs↓
Baseline	28.39	0
Rnd. Spawn	28.39	971.9
w/o Quant. Ctrl.	28.43	8710.8
Full Model	28.42	477.7

Table 3. **Ablation study of the Adaptive 3DG Addition strategy on the** *flame salmon* **scene**. The metrics are averaged over the whole sequence.

we train the NTC for 150 iterations in Stage 1. and train the additional 3DGs for 100 iterations in Stage 2. Please refer to Suppl. for more details.

5.3. Comparisons

Quantitative comparisons. Our quantitative analysis involves benchmarking 3DGStream on the N3DV dataset and

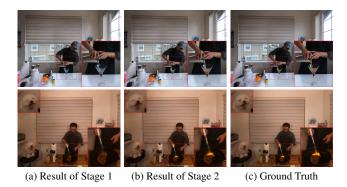(a) Result of Stage 1   (b) Result of Stage 2   (c) Ground Truth

Figure 7. **Quantitative results of the ablation study** conducted on the *flame steak* scene and the *coffee martini* scene.

Meet Room dataset, comparing it with a range of representative methods. We take Plenoxels [20], I-NGP [40], and 3DG-S [26] as representatives of fast static scene reconstruction methods, training them from scratch for each frame. StreamRF [29], Dynamic3DG [34], and ReRF [61] are designed for online training in dynamic scenes. Owing to the limitations of Dynamic3DG and ReRF, which necessitate foreground masks and are confined to scenes with fewer objects, and their minute-level per-frame training times, we select StreamRF selected as the representative for online training methods due to its adaptability and training feasibility on the N3DV and MeetRoom datasets. To demonstrate 3DGStream's competitive image quality, we drew comparisons with the quantitative results reported for the N3DV dataset in the respective papers of DyNeRF [31], NeRFPlayer [52], HexPlane [9], K-Planes [48], HyperReel [1], and MixVoxels [58], all of which are methods for reconstructing dynamic scenes through offline training on entire video sequences.

In Tab. 1, we present the averaged rendering speed, training time, required storage, and peak signal-to-noise ratio (PSNR) over all scenes of the N3DV dataset. For each scene, the latter three metrics are computed as averages over the whole 300 frames. Besides, we provide a breakdown of comparisons across all scenes within the N3DV dataset in the Suppl. To demonstrate the generality of our method, we conducted experiments on the MeetRoom dataset, as introduced by StreamRF [29], and performed a quantitative comparison against Plenoxels [20], I-NGP [40], 3DG-S [26], and StreamRF [29]. The results are presented in Tab. 2. As presented in Tabs. 1 and 2, our method demonstrates superiority through fast online training and real-time rendering, concurrently maintaining a competitive edge in terms of model storage and image quality. Furthermore, among the methods capable of streaming FVVs, our model requires the minimal model storage.

Qualitative comparisons. While our approach primarily aims to enhance the efficiency of online FVV construction, as illustrated in Tabs. 1 and 2, it still achieves competitive

Step	Overhead (ms)	FPS
Render *w/o* NTC	2.56	390
+ Query NTC	0.62	
+ Transformation	0.02	
+ SH Rotation	1.46	
Total	4.66	215

Table 4. **Rendering profilling** for the *flame salmon* scene at megapixel resolution. Note that *flame salmon* is the most time-consuming to render of all scenes in our experiments.

image quality. In Fig. 4, we present a qualitative comparison with I-NGP [40], HyperReel [1], and StreamRF [29] across scenes on the N3DV dataset [31] and the Meet Room dataset [29], with a special emphasis on dynamic objects such as faces, hands, and tongs, as well as intricate objects like labels and statues. It is evident that our method faithfully captures the dynamics of the scene without sacrificing the ability to reconstruct intricate objects. Please refer to our project page for more video results.

5.4. Evaluations

Neural Transformation Cache. We utilize distinct approaches to model the transformations of 3DGs from the first to the second frame within the *flame salmon* video of the N3DV dataset to show the effectiveness of NTC. Fig. 5 shows that, without multi-resolution hash encoding (*w/o* Hash enc.), the MLP faces challenges in modeling transformations effectively. Additionally, without the warm-up (*w/o* Warm-up), it takes more iterations for convergence. Besides, even when compared with the direct optimization of the previous frame's 3DGs (Direct Opt.), NTC demonstrate on-par performance. In Fig. 6, We present the results of different approaches applied across the entire *flame salmon* video, excluding the first frame (*i.e.*, Frame 0). *w/o* Hash enc. and *w/o* Warm-up. are not able to converge swiftly, resulting in accumulating errors as the sequence progresses. Direct Opt. yields the best outcomes but at the cost of inflated storage. Utilizing NTC, in contrast, delivers comparable results with substantially lower storage overhead by eliminating the need for saving all the 3DGs.

Adaptive 3DG Addition. Tab. 3 presents the quantitative results of the ablation study conducted on the *flame salmon* scene, and more results are presented in Suppl. The base model without Stage 2, and a set of randomly spawned 3DGs (Rnd. Spawn) in equivalent quantities to our spawn strategy, both fail to capture emerging objects. The variant without our quantity control strategy (*w/o* Quant. Ctrl.) manages to model emerging objects but requires a significantly larger number of additional 3DGs. In contrast, our full model proficiently reconstructs emerging objects using a minimal addition of 3DGs. The ablation study illustrated in Fig. 7 qualitatively showcases the effect of the Adap-

tive 3DG Addition strategy, highlighting its ability to reconstruct the objects not present in the initial frame, such as coffee in a pot, a dog's tongue, and flames.

Real-time Rendering. Following 3DG-S [26], we employ the SIBR framework [6] to measure the rendering speed. Once all resources required are loaded onto the GPU, the additional overhead of our approach is primarily the time taken to query the NTC and transform the 3DGs. As detailed in Tab. 4, our method benefits from the efficiency of the multi-resolution hash encoding and the fully-fused MLP [38], which facilitate rapid NTC query. Notably, the most time-consuming step is the SH Rotation. However, our experiments indicate that the SH rotation has a minimal impact on the reconstruction quality, which may be attributed to the 3DGs modeling view-dependent colors through alternative mechanisms (*e.g.*, small 3DGs of varying colors surrounding the object) rather than SH coefficients. Nonetheless, we maintain SH rotation for theoretical soundness.

6. Discussion

The quality of 3DG-S [26] on the initial frame is crucial to 3DGStream. Therefore, we inherit the limitations of 3DG-S, such as high dependence on the initial point cloud. As illustrated in Fig. 7, there are obvious artifacts beyond the windows, attributable to COLMAP's [49] inability to reconstruct distant landscapes. Hence, our method stands to benefit directly from future enhancements to 3DG-S. Moreover, for efficient on-the-fly training, we limit the number of training iterations, which restricts modeling of drastic motion in Stage 1 and complex emerging objects in Stage 2.

7. Conclusion

We propose 3DGStream, an novel method for efficient Free-Viewpoint Video streaming. Based on 3DG-S [26], we utilizes an effective Neural Transformation Cache to capture the motion of objects. In addition, we propose an Adaptive 3DG Addition strategy to accurately model emerging objects in dynamic scenes. The two-stage pipeline of 3DGStream enables the online reconstruction of dynamic scenes in video streams. While ensuring photo-realistic image quality, 3DGStream achieves on-the-fly training (∼10s per-frame) and real-time rendering (∼200FPS) at megapixel resolution with moderate requisite storage.

8. Acknowledgement

3DGStream: On-the-Fly Training of 3D Gaussians for Efficient Streaming of Photo-Realistic Free-Viewpoint Videos

Supplementary Material

9. Implementation Details

We implement 3DGStream upon the codebase of 3D Gaussian Splatting (3DG-S) [26] and use tiny-cuda-nn [38] to implement Neural Transformation Cache (NTC). All experiments were conducted on an NVIDIA RTX 3090 GPU. In training of the initial frame, we let the densification of 3DG-S end at iteration 5000. For the scenes in the N3DV dataset, we use the 3DGs of iteration 15000 as the initial 3DGs, while for the scenes in the Meet Room dataset, we use the results of iteration 10000. For convenience, we set the maximum degree of spherical harmonics (SH) to 1, and all other hyperparameters are consistent with the 3DG-S.

Training NTC. We set the learning rate of NTC to 0.002. For the scenes in the N3DV dataset [31], the hash table size of the multi-resolution hash encoding is 2^{15}, the feature vector dimension is 4, and there are 16 resolution levels. For the Meet Room dataset [29], the hash table length is 2^{14}, with all other hyperparameters matching those specified for the N3DV dataset. For all scenes, our fully-fused MLP comprises 2 hidden layers with 64 neurons each, employing ReLu as the activation function. Given that the N3DV dataset and the Meet Room dataset both record indoor dynamic scenes, and multi-resolution hash encoding requires normalized coordinates for input, we create an axis-aligned bounding box that roughly encloses the house to normalize the 3D points and discard any points outside the bounding box to prevent distant landscapes from influencing the training.

Training the additional 3DGs. Compared to training on the initial frame, we increase the learning rate in the second stage for faster convergence. Specifically, the learning rates for the mean, SH coefficient, opacity value, scaling vector, and rotation quaternion of the 3DGs are set to 0.0024, 0.0375, 0.75, 0.075, and 0.015, respectively. Note that these learning rates were not individually fine-tuned; instead, their proportions are following the default settings of 3DG-S.

10. SH Rotation

In order to preserve theoretical soundness, we also rotate the SH after transforming the 3DGs. The zeroth-degree SH does not require rotation; therefore, we only need to rotate the first-degree SH coefficients.

We utilize the projection function [51] to project normal vectors onto the first-order SH. Given a rotation matrix R, we seek a matrix M that can rotate the first-degree SH. Be-

Method	Coffee Martini	Cook Spinach	Cut Beef	Flame Salmon	Flame Steak	Sear Steak	Mean
Plenoxels [20]†	27.65	31.73	32.01	28.68	32.24	32.33	30.77
I-NGP [40]†	25.19	29.84	30.73	25.51	30.04	30.40	28.62
3DG-S [26]†	27.78	34.10	34.03	28.66	34.41	33.48	32.08
DyNeRF [31]	–	–	–	29.58	–	–	29.58
NeRFPlayer [52]	31.53	30.58	29.35	31.65	31.93	29.13	30.69
HexPlane [9]	–	32.04	32.55	29.47	32.08	32.39	31.70
K-Planes [48]	29.99	32.60	31.82	30.44	32.38	32.52	31.63
HyperReel [1]	28.37	32.30	32.92	28.26	32.20	32.57	31.10
MixVoxels [58]	29.36	31.61	31.30	29.92	31.43	31.21	30.80
StreamRF [29]†	27.84	31.59	31.81	28.26	32.24	32.36	28.26
Ours	27.75	33.31	33.21	28.42	34.30	33.01	31.67

Table 5. **Quantitative comparison** of PSNR values across all scenes in the N3DV dataset, with the metric for each scene calculated as the average over 300 frames. †Obtained in our own experiments with the official codes.

Method	Coffee Martini	Cook Spinach	Cut Beef	Flame Salmon	Flame Steak	Sear Steak	Mean
Baseline	27.68	33.19	33.10	28.39	33.54	32.79	31.45
Full Model	27.75	33.31	33.21	28.42	34.30	33.01	31.67

Table 6. **Ablation Study** of the Adaptive 3DG Addition strategy across all scenes in the N3DV dataset, with the metric for each scene calculated as the average over 300 frames. We take PSNR to measure the image quality.

cause rotating a vector before projecting it to SH produces the same outcome as projecting the vector first and then rotating the SH, we have the following relationship:

$$MP(N) = P(RN), \quad (8)$$

where N is a normal vector. For any three normal vectors N_0, N_1, and N_2 we denote $A = [P(N_0), P(N_1), P(N_2)]$. Consequently, we obtain:

$$MA = [P(RN_0), P(RN_1), P(RN_2)]. \quad (9)$$

And hence:

$$M = [P(RN_0), P(RN_1), P(RN_2)]A^{-1} \quad (10)$$

For computational convenience, we choose $N_0 = [1, 0, 0]^T$, $N_1 = [0, 1, 0]^T$, and $N_2 = [0, 0, 1]^T$.

11. More Results

11.1. Quantitative Results

We provide a quantitative comparison of image quality, measured by PSNR, across all scenes in the N3DV dataset

Step	Overhead (ms)	FPS
Render *w/o* NTC	1.75	571
+ Query NTC	+0.46	
+ Transformation	+0.02	
+ SH Rotation	+1.24	
Total	3.47	288

Table 7. **Rendering profiling** on the Meet Room dataset.

Dataset	NTC (KB)	New 3DGs (KB)	Total (KB)
N3DV	7781.5	49.1	7830.6
MeetRoom	3941.5	195.3	4136.8

Table 8. **Detailed "Storage" entry of our method** in Tabs. 1 and 2.

Scene	Stage 1		Stage 2	
	150	250	100	200
Flame Salmon	28.39	28.44	28.46	28.46
Flame Steak	33.54	33.81	34.44	34.46
Sear Steak	32.79	33.02	33.18	33.19
Cook Spinach	33.19	33.50	33.56	33.57
Cut Roasted Beef	33.10	33.39	33.44	33.44
Coffee Martini	27.68	27.77	27.83	27.83

Table 9. **Evaluation on the impact of training iterations** conducted on the N3DV dataset. The result of Stage 2 is is obtained after 250 iterations of optimization at Stage 1. We take PSNR to measure the image quality.
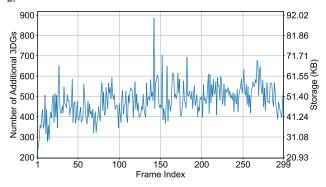

Figure 8. **Number of additional 3DGs and corresponding storage requirement** of each frame on the *flame salmon* scene.

in Tab. 5. Furthermore, we provide the quantitative result of the ablation study across all scenes in the N3DV dataset in Tab. 6, Additionally, we provide rendering profilling on the Meet Room Dataset in Tab. 7.

11.2. Qualitative Results

We provide videos to show the free view synthesis results on various scenes from the N3DV dataset in https://sjojok.github.io/3dgstream.

12. More Evaluations

12.1. Storage Requirements

Except the initial 3DGs, we only need to store per-frame NTCs and per-frame additional 3DGs for each FVV frame, as detailed in Tab. 8.

12.2. Quantity of 3DGs

In our experiments on the N3DV datasets, the quantity of initial 3DGs (*i.e.*, the transformed ones) is on the order of 10^5, while the quantity of frame-specific additional 3DGs is on the order of 10^2. We show how the number of the

frame-specific additional 3DGs changes as the frame number increases in Fig. 8.

12.3. Impact of Training Iterations

In the main text, we discuss the trade-off between training efficiency and reconstruction quality, noting that limiting the number of training iterations enables efficient on-the-fly training at the expense of reduced quality. To show this trade-off, we conduct experiments to evaluate the impact of training iterations, and show the quantitative results in Tab. 9. As shown in Tab. 9, increasing training iterations in Stage 1 significantly enhances the reconstruction quality. However, an additional 100 iterations result in an increment of 3 seconds in the per-frame training duration. Incrementing training iterations in the second stage has a minimal impact on quality, which can be attributed to the higher learning rate employed in this phase and the smaller number of additional 3DGs, facilitating rapid convergence.

References

[1] Benjamin Attal, Jia-Bin Huang, Christian Richardt, Michael Zollhoefer, Johannes Kopf, Matthew O'Toole, and Changil Kim. Hyperreel: High-fidelity 6-dof video with ray-conditioned sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16610–16620, 2023. 1, 3, 6, 7, 8, 9

[2] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 2

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

[4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P.

Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *ICCV*, 2023. 2

[5] Michael J Black and Paul Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer vision and image understanding*, 63(1):75–104, 1996. 4

[6] Sebastien Bonopera, Jerome Esnault, Siddhant Prakash, Simon Rodriguez, Theo Thonat, Mehdi Benadel, Gaurav Chaurasia, Julien Philip, and George Drettakis. sibr: A system for image based rendering, 2020. 8

[7] Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew Duvall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics (TOG)*, 39(4):86–1, 2020. 1, 2

[8] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *SIGGRAPH*, pages 425–432, 2001. 2

[9] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. *CVPR*, 2023. 3, 6, 7, 9

[10] Jin-Xiang Chai, Xin Tong, Shing-Chow Chan, and Heung-Yeung Shum. Plenoptic sampling. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 307–318, 2000. 2

[11] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14124–14133, 2021. 2

[12] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022. 2

[13] Anpei Chen, Zexiang Xu, Xinyue Wei, Siyu Tang, Hao Su, and Andreas Geiger. Dictionary fields: Learning a neural basis decomposition. *ACM Trans. Graph.*, 2023. 2

[14] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *The Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[15] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. High-quality streamable free-viewpoint video. *ACM Transactions on Graphics (TOG)*, 34(4):69, 2015. 1, 2

[16] Abe Davis, Marc Levoy, and Fredo Durand. Unstructured light fields. *Comput. Graph. Forum*, 31(2pt1):305–314, 2012. 2

[17] Mingsong Dou, Philip Davidson, Sean Ryan Fanello, Sameh Khamis, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, and Shahram Izadi. Motion2fusion: Real-time volumetric performance capture. *ACM Trans. Graph.*, 36(6):246:1–246:16, 2017. 1, 2

[18] Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 4d gaussian splatting: Towards efficient novel view synthesis for dynamic scenes. *arXiv preprint arXiv:2402.03307*, 2024. 4

[19] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, 2022. 1, 3

[20] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 2, 3, 6, 7, 9

[21] Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14346–14355, 2021. 2

[22] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, page 43–54, New York, NY, USA, 1996. Association for Computing Machinery. 2

[23] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5855–5864, 2021. 2

[24] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981. 4

[25] Wenbo Hu, Yuling Wang, Lin Ma, Bangbang Yang, Lin Gao, Xiao Liu, and Yuewen Ma. Tri-miprf: Tri-mip representation for efficient anti-aliasing neural radiance fields. In *ICCV*, 2023. 2

[26] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. 1, 2, 4, 5, 6, 7, 8, 9

[27] Tobias Kirschstein, Shenhan Qian, Simon Giebenhain, Tim Walter, and Matthias Nießner. Nersemble: Multi-view radiance field reconstruction of human heads. *arXiv preprint arXiv:2305.03027*, 2023. 3

[28] Marc Levoy and Pat Hanrahan. Light field rendering. In *SIGGRAPH*, pages 31–42, 1996. 2

[29] Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Ping Tan. Streaming radiance fields for 3d video synthesis. In *NeurIPS*, 2022. 1, 2, 3, 6, 7, 8, 9

[30] Ruilong Li, Julian Tanke, Minh Vo, Michael Zollhöfer, Jürgen Gall, Angjoo Kanazawa, and Christoph Lassner. Tava: Template-free animatable volumetric actors. In *European Conference on Computer Vision*, pages 419–436. Springer, 2022. 3

[31] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5521–5531, 2022. 1, 2, 3, 6, 7, 8, 9

[32] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference*

on Computer Vision and Pattern Recognition (CVPR), pages 6498–6508, 2021. 3

[33] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 3

[34] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 3, 7

[35] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021. 2

[36] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020. 1, 2

[37] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul P Srinivasan, and Jonathan T Barron. Nerf in the dark: High dynamic range view synthesis from noisy raw images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16190–16199, 2022. 2

[38] Thomas Müller. tiny-cuda-nn, 2021. 4, 5, 6, 8, 9

[39] Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. Real-time neural radiance caching for path tracing. *ACM Transactions on Graphics (TOG)*, 40(4):1–16, 2021. 4

[40] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. 1, 2, 4, 6, 7, 8, 9

[41] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5480–5490, 2022. 2

[42] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5865–5874, 2021. 3

[43] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021. 1

[44] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.*, 40(6), 2021. 3

[45] Sungheon Park, Minjung Son, Seokhwan Jang, Young Chun Ahn, Ji-Yeon Kim, and Nahyup Kang. Temporal interpolation is all you need for dynamic neural radiance fields. In

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4212–4221, 2023. 3

[46] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 1, 3

[47] Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 42(4):1–12, 2023. 2

[48] Sara Fridovich-Keil and Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *CVPR*, 2023. 1, 3, 6, 7, 9

[49] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 8

[50] Heung-Yeung Shum and Li-Wei He. Rendering with concentric mosaics. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 299–306, 1999. 2

[51] Peter-Pike Sloan. Stupid spherical harmonics (sh) tricks. In *Game developers conference*, page 42, 2008. 9

[52] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 29(5):2732–2742, 2023. 3, 6, 7, 9

[53] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5449–5459, 2022. 2, 3

[54] Jiakai Sun, Zhanjie Zhang, Jiafu Chen, Guangyuan Li, Boyan Ji, Lei Zhao, and Wei Xing. Vgos: Voxel grid optimization for view synthesis from sparse inputs. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 1414–1422. International Joint Conferences on Artificial Intelligence Organization, 2023. Main Track. 2

[55] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International journal of computer vision*, 9:137–154, 1992. 4

[56] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12959–12970, 2021. 3

[57] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *CVPR*, 2022. 2

[58] Feng Wang, Sinan Tan, Xinghang Li, Zeyue Tian, Yafei Song, and Huaping Liu. Mixed neural voxels for fast multi-view video synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19706–19716, 2023. 3, 6, 7, 9

[59] Liao Wang, Jiakai Zhang, Xinhang Liu, Fuqiang Zhao, Yanshun Zhang, Yingliang Zhang, Minye Wu, Jingyi Yu, and Lan Xu. Fourier plenoctrees for dynamic radiance field rendering in real-time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13524–13534, 2022. 3

[60] Liao Wang, Qiang Hu, Qihan He, Ziyu Wang, Jingyi Yu, Tinne Tuytelaars, Lan Xu, and Minye Wu. Neural residual radiance fields for streamably free-viewpoint videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 76–87, 2023. 3

[61] Liao Wang, Qiang Hu, Qihan He, Ziyu Wang, Jingyi Yu, Tinne Tuytelaars, Lan Xu, and Minye Wu. Neural residual radiance fields for streamably free-viewpoint videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 76–87, 2023. 1, 7

[62] Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking everything everywhere all at once. In *International Conference on Computer Vision*, 2023. 3

[63] Chung-Yi Weng, Brian Curless, Pratul P Srinivasan, Jonathan T Barron, and Ira Kemelmacher-Shlizerman. Humannerf: Free-viewpoint rendering of moving people from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16210–16220, 2022. 3

[64] Felix Wimbauer, Nan Yang, Christian Rupprecht, and Daniel Cremers. Behind the scenes: Density fields for single view reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9076–9086, 2023. 2

[65] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8534–8543, 2021. 2

[66] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Wang Xinggang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023. 4

[67] Jamie Wynn and Daniyar Turmukhambetov. Diffusionerf: Regularizing neural radiance fields with denoising diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4180–4189, 2023. 2

[68] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9421–9431, 2021. 1

[69] Gengshan Yang, Minh Vo, Natalia Neverova, Deva Ramanan, Andrea Vedaldi, and Hanbyul Joo. Banmo: Building animatable 3d neural models from many casual videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2863–2873, 2022. 3

[70] Jiawei Yang, Marco Pavone, and Yue Wang. Freenerf: Improving few-shot neural rendering with free frequency regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8254–8263, 2023. 2

[71] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023. 4

[72] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. 2024. 4

[73] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021. 2

[74] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021. 2

[75] Fuqiang Zhao, Wei Yang, Jiakai Zhang, Pei Lin, Yingliang Zhang, Jingyi Yu, and Lan Xu. Humannerf: Efficiently generated human radiance field from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7743–7753, 2022. 3

[76] C Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *ACM transactions on graphics (TOG)*, 23(3):600–608, 2004. 1, 2

[77] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001. 4