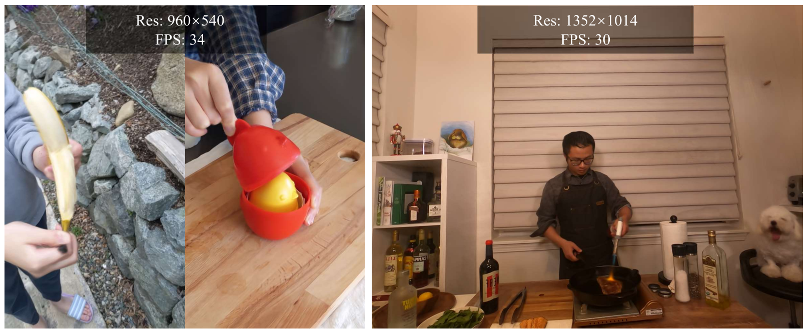
4D Gaussian Splatting for Real-Time Dynamic Scene Rendering

Guanjun Wu[1]*, Taoran Yi[2]*, Jiemin Fang[3]†, Lingxi Xie[3], Xiaopeng Zhang[3],
Wei Wei[1], Wenyu Liu[2], Qi Tian[3], Xinggang Wang[2]†‡
[1]School of CS, Huazhong University of Science and Technology
[2]School of EIC, Huazhong University of Science and Technology [3]Huawei Inc.
{guajuwu, taoranyi, weiw, liuwy, xgwang}@hust.edu.cn
{jaminfong, 198808xc, zxphistory}@gmail.com tian.qi1@huawei.com



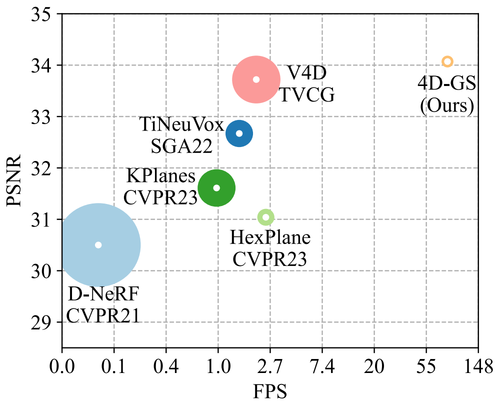


Figure 1. Our method achieves real-time rendering[‡] for dynamic scenes at high image resolutions while maintaining high rendering quality. The right figure is tested on synthetic datasets, where the radius of the dot corresponds to the training time. "Res": resolution.
[‡]The rendering speed not only depends on the image resolution but also the number of 3D Gaussians and the scale of deformation fields which are determined by the complexity of the scene.

Abstract

Representing and rendering dynamic scenes has been an important but challenging task. Especially, to accurately model complex motions, high efficiency is usually hard to guarantee. To achieve real-time dynamic scene rendering while also enjoying high training and storage efficiency, we propose 4D Gaussian Splatting (4D-GS) as a holistic representation for dynamic scenes rather than applying 3D-GS for each individual frame. In 4D-GS, a novel explicit representation containing both 3D Gaussians and 4D neural voxels is proposed. A decomposed neural voxel encoding algorithm inspired by HexPlane is proposed to efficiently build Gaussian features from 4D neural voxels and then a lightweight MLP is applied to predict Gaussian deformations at novel timestamps. Our 4D-GS method achieves real-time rendering under high resolutions, 82 FPS at an 800×800 resolution on an RTX 3090 GPU while maintaining comparable or better quality than previous state-of-the-art methods. More demos and code are available at https://guanjunwu.github.io/4dgs/.

[*]Equal contributions.
[†]Project Lead.
[‡]Corresponding author.

1. Introduction

Novel view synthesis (NVS) stands as a critical task in the domain of 3D vision and plays a vital role in many applications, *e.g.* VR, AR, and movie production. NVS aims at rendering images from any desired viewpoint or timestamp of a scene, usually requiring modeling the scene accurately from several 2D images. Dynamic scenes are quite common in real scenarios, rendering which is important but challenging as complex motions need to be modeled with both spatially and temporally sparse input.

NeRF [32] has achieved great success in synthesizing novel view images by representing scenes with implicit functions. The volume rendering techniques [7] are introduced to connect 2D images and 3D scenes. However, the original NeRF method bears big training and rendering costs. Though some NeRF variants [5, 8, 10, 11, 33, 44, 47] reduce the training time from days to minutes, the rendering process still bears a non-negligible latency.

Recent 3D Gaussian Splatting (3D-GS) [19] significantly boosts the rendering speed to a real-time level by representing the scene as 3D Gaussians. The cumbersome volume rendering in the original NeRF is replaced with efficient differentiable splatting [57], which directly projects 3D Gaussian onto the 2D image plane. 3D-GS not only enjoys real-time rendering speed but also represents the scene more explicitly, making it easier to manipulate the scene representation.

However, 3D-GS focuses on the static scenes. Extending it to dynamic scenes as a 4D representation is a reasonable, important but difficult topic. The key challenge lies in modeling complicated point motions from sparse input. 3D-GS holds a natural geometry prior by representing scenes with point-like Gaussians. One direct and effective extension approach is to construct 3D Gaussians at each timestamp [30] but the storage/memory cost will multiply especially for long input sequences. Our goal is to construct a compact representation while maintaining both training and rendering efficiency, *i.e.* 4D Gaussian Splatting (**4D-GS**). To this end, we propose to represent Gaussian motions and shape changes by an efficient Gaussian deformation field network, containing a temporal-spatial structure encoder and an extremely tiny multi-head Gaussian deformation decoder. Only one set of canonical 3D Gaussians is maintained. For each timestamp, the canonical 3D Gaussians will be transformed by the Gaussian deformation field into new positions with new shapes. The transformation process represents both the Gaussian motion and deformation. Note that different from modeling motions of each Gaussian separately [30, 55], the spatial-temporal structure encoder can connect different adjacent 3D Gaussians to predict more accurate motions and shape deformation. Then the deformed 3D Gaussians can be directly splatted for rendering the according-timestamp image. Our contributions can be summarized as follows.

- An efficient 4D Gaussian splatting framework with an efficient Gaussian deformation field is proposed by modeling both Gaussian motion and Gaussian shape changes across time.
- A multi-resolution encoding method is proposed to connect the nearby 3D Gaussians and build rich 3D Gaussian features by an efficient spatial-temporal structure encoder.
- 4D-GS achieves real-time rendering on dynamic scenes, up to 82 FPS at a resolution of 800×800 for synthetic datasets and 30 FPS at a resolution of 1352×1014 in real datasets, while maintaining comparable or superior performance than previous state-of-the-art (SOTA) methods and shows potential for editing and tracking in 4D scenes.

2. Related Works

In this section, we simply review the difference of dynamic NeRFs in Sec. 2.1, then discuss the point clouds-based neural rendering algorithm in Sec. 2.2.

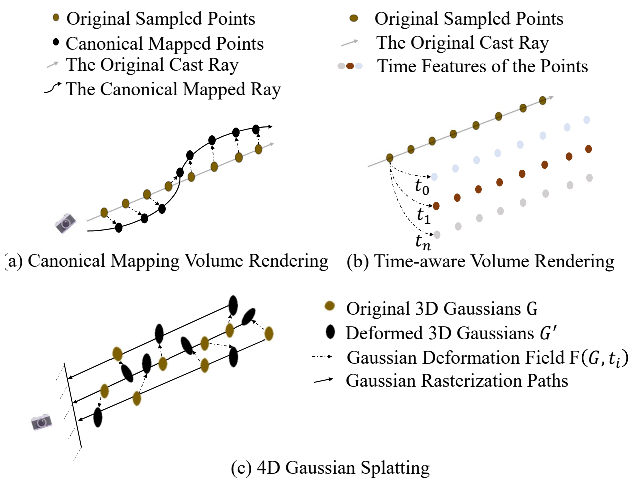
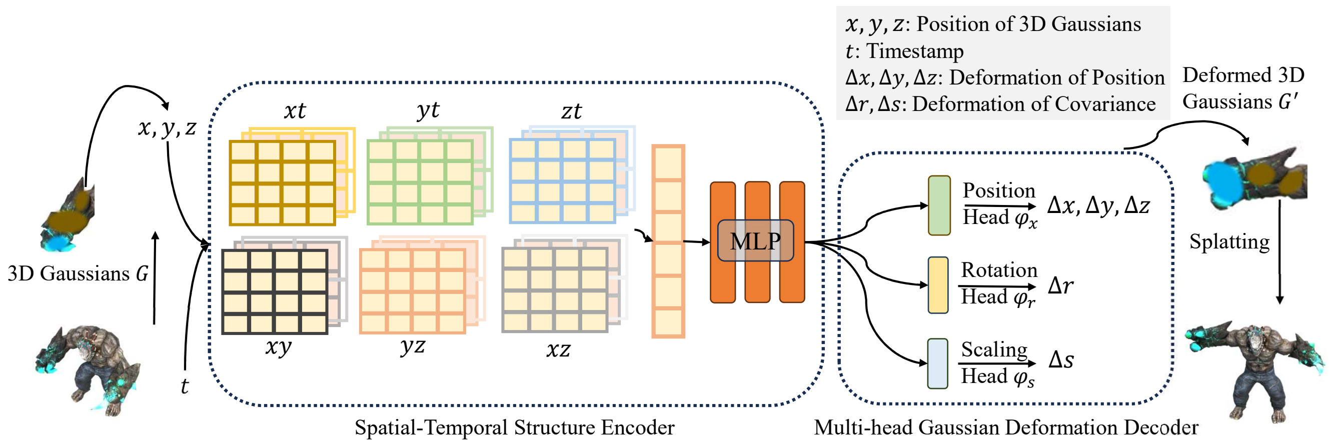

Original Sampled Points
Canonical Mapped Points
The Original Cast Ray
The Canonical Mapped Ray
(a) Canonical Mapping Volume Rendering
Original Sampled Points
The Original Cast Ray
Time Features of the Points
t_0
t_1
t_n
(b) Time-aware Volume Rendering
Original 3D Gaussians G
Deformed 3D Gaussians G'
Gaussian Deformation Field $F(G, t_i)$
Gaussian Rasterization Paths
(c) 4D Gaussian Splatting

Figure 2. Illustration of different dynamic scene rendering methods. (a) Points are sampled in the casted ray during volume rendering. The point deformation fields proposed in [8, 39] map the points into a canonical space. (b) Time-aware volume rendering computes the features of each point directly and does not change the rendering path. (c) The Gaussian deformation field converts original 3D Gaussians into another group of 3D Gaussians with a certain timestamp.

2.1. Novel View Synthesis

Novel view synthesis is a important and challenging task in 3D reconstruction. Much approaches are proposed to represent a 3D object and render novel views. Efficient representations such as light fields [4], mesh [6, 15, 24, 46], voxels [16, 18, 23], multi-planes [9] can render high quality image with enough supervisions. NeRF-based approaches [3, 32, 59] demonstrate that implicit radiance fields can effectively learn scene representations and synthesize high-quality novel views. [35, 36, 39] have challenged the static hypothesis, expanding the boundary of novel view synthesis for dynamic scenes. [8] proposes to use an explicit voxel grid to model temporal information, accelerating the learning time for dynamic scenes to half an hour and applied in [17, 29, 56]. The proposed deformation-based neural rendering methods are shown in Fig. 2 (a). Flow-based [13, 25, 29, 48, 61] methods adopting warping algorithm to synthesis novel views by blending nearby frames. [5, 11, 12, 22, 44, 49] represent further advancements in faster dynamic scene learning by adopting decomposed neural voxels. They treat sampled points in each timestamp individually as shown in Fig. 2 (b). [14, 27, 38, 50, 51, 53] are efficient methods to handle multi-view setups. The aforementioned methods though achieve fast training speed, real-time rendering for dynamic scenes is still challenging, especially for monocular input. Our method aims at constructing a highly efficient training and rendering pipeline in Fig. 2 (c), while maintaining the quality, even for sparse inputs.



x, y, z: Position of 3D Gaussians
t: Timestamp
$\Delta x, \Delta y, \Delta z$: Deformation of Position
$\Delta r, \Delta s$: Deformation of Covariance
Deformed 3D Gaussians G'
x, y, z
xt
yt
zt
xy
yz
xz
3D Gaussians G
t
MLP
Position Head φ_x
$\Delta x, \Delta y, \Delta z$
Rotation Head φ_r
Δr
Scaling Head φ_s
Δs
Splatting
Spatial-Temporal Structure Encoder
Multi-head Gaussian Deformation Decoder

Figure 3. The overall pipeline of our model. Given a group of 3D Gaussians $\mathcal{G}$, we extract the center coordinate of each 3D Gaussian $\mathcal{X}$ and timestamp t to compute the voxel feature by querying multi-resolution voxel planes. Then a tiny multi-head Gaussian deformation decoder is used to decode the feature and get the deformed 3D Gaussians $\mathcal{G}'$ at timestamp t. The deformed Gaussians are then splatted to the rendered image.

2.2. Neural Rendering with Point Clouds

Effectively representing 3D scenes remains a challenging topic. The community has explored various neural representations [32], *e.g.* meshes, point clouds [54], voxels [10], and hybrid approaches [33, 47]. Point-cloud-based methods [28, 40, 41, 58] initially target at 3D segmentation and classification. A representative approach for rendering presented in [1, 54] combines point cloud representations with volume rendering, achieving rapid convergence speed even for dynamic novel view synthesis [34, 61]. [20, 21, 42] adopt differential point rendering technique for scene reconstructions.

Recently, 3D-GS [19] is notable for its pure explicit representation and differential point-based splatting methods, enabling real-time rendering of novel views. Dynamic3DGS [30] models dynamic scenes by tracking the position and variance of each 3D Gaussian at each timestamp t_i. An explicit table is utilized to store information about each 3D Gaussian at every timestamp, leading to a linear memory consumption increase, denoted as $O(t\mathcal{N})$, in which $\mathcal{N}$ is num of 3D Gaussians. For long-term scene reconstruction, the storage cost will become non-negligible. The memory complexity of our approach only depends on the number of 3D Gaussians and parameters of Gaussians deformation fields network $\mathcal{F}$, which is denoted as $O(\mathcal{N}+\mathcal{F})$. [55] adds a marginal temporal Gaussian distribution into the origin 3D Gaussians, which uplift 3D Gaussians into 4DHowever, it may cause each 3D Gaussian to only focus on their local temporal space. [26] track each 3D Gaussians individually. Our approach also models 3D Gaussian motions but with a compact network, resulting in highly efficient training efficiency and real-time rendering.

3. Preliminary

In this section, we simply review the representation and rendering process of 3D-GS [19] in Sec. 3.1 and the formula of dynamic NeRFs in Sec. 3.2.

3.1. 3D Gaussian Splatting

3D Gaussians [19] is an explicit 3D scene representation in the form of point clouds. Each 3D Gaussian is characterized by a covariance matrix Σ and a center point $\mathcal{X}$, which is referred to as the mean value of the Gaussian:

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

For differentiable optimization, the covariance matrix Σ can be decomposed into a scaling matrix $\mathbf{S}$ and a rotation matrix $\mathbf{R}$:

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

When rendering novel views, differential splatting [57] is employed for the 3D Gaussians within the camera planes. As introduced by [62], using a viewing transform matrix W and the Jacobian matrix J of the affine approximation of the projective transformation, the covariance matrix Σ' in camera coordinates can be computed as

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

In summary, each 3D Gaussian is characterized by the following attributes: position $\mathcal{X} \in \mathbb{R}^3$, color defined by spherical harmonic (SH) coefficients $\mathcal{C} \in \mathbb{R}^k$ (where k represents nums of SH functions), opacity $\alpha \in \mathbb{R}$, rotation factor $r \in \mathbb{R}^4$, and scaling factor $s \in \mathbb{R}^3$. Specifically, for each pixel, the color and opacity of all the Gaussians are computed using the Gaussian's representation Eq. 1. The blending of N ordered points that overlap the pixel is given by the formula:

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

Here, c_i, α_i represents the density and color of this point computed by a 3D Gaussian G with covariance Σ multiplied by an optimizable per-point opacity and SH color coefficients.

3.2. Dynamic NeRFs with Deformation Fields

All the dynamic NeRF algorithms can be formulated as:

$$c, \sigma = \mathcal{M}(\mathbf{x}, d, t, \lambda), \tag{5}$$

where $\mathcal{M}$ is a mapping that maps 8D space $(\mathbf{x}, d, t, \lambda)$ to 4D space (c, σ). Where x reveals to the spatial point, λ is the optional input as used to build topological and appearance changes in [36], and d stands for view-dependency.

As is shown in Fig. 2 (a), all the deformation NeRF based method which estimate the **world-to-canonical mapping** by a deformation network $\phi_t : (\mathbf{x}, t) \to \Delta\mathbf{x}$. Then a network is introduced to compute volume density and view-dependent RGB color from each ray. The formula for rendering can be expressed as:

$$c, \sigma = \text{NeRF}(\mathbf{x} + \Delta\mathbf{x}, d, \lambda), \tag{6}$$

where 'NeRF' stands for vanilla NeRF pipeline, λ is a frame-dependent code to model the topological and appearance changes [31, 36].

However, our 4D Gaussian splatting framework presents a novel rendering technique. We successfully compute the **canonical-to-world mapping** by a Gaussian deformation field network $\mathcal{F}$ at the time t directly and differential splatting [19] is followed, which enables the skill of computing backward flow and tracking for 3D Gaussians.

4. Method

Sec. 4.1 introduces the overall 4D Gaussian Splatting framework. Then, the Gaussian deformation field is proposed in Sec. 4.2. Finally, we describe the optimization process in Sec. 4.3.

4.1. 4D Gaussian Splatting Framework

As shown in Fig. 3, given a view matrix $M = [R, T]$, timestamp t, our 4D Gaussian splatting framework includes 3D Gaussians $\mathcal{G}$ and Gaussian deformation field network $\mathcal{F}$. Then a novel-view image $\hat{I}$ is rendered by differential splatting [57] $\mathcal{S}$ following $\hat{I} = \mathcal{S}(M, \mathcal{G}')$, where $\mathcal{G}' = \Delta\mathcal{G} + \mathcal{G}$.

Specifically, the deformation of 3D Gaussians $\Delta\mathcal{G}$ is introduced by the Gaussian deformation field network $\Delta\mathcal{G} = \mathcal{F}(\mathcal{G}, t)$, in which the spatial-temporal structure encoder $\mathcal{H}$ can encode both the temporal and spatial features of 3D Gaussians $f_d = \mathcal{H}(\mathcal{G}, t)$, and the multi-head Gaussian deformation decoder $\mathcal{D}$ can decode the features and predict each 3D Gaussian's deformation $\Delta\mathcal{G} = \mathcal{D}(f)$, then the deformed 3D Gaussians $\mathcal{G}'$ can be introduced.

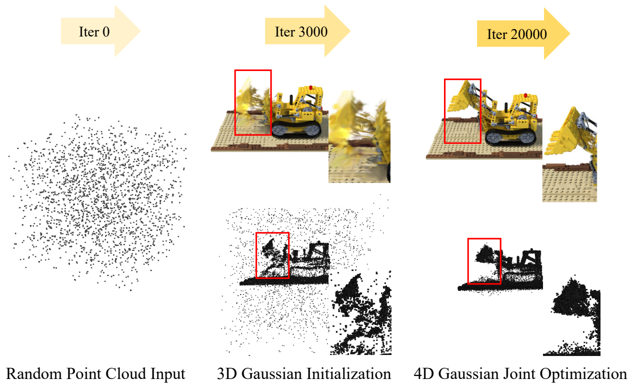


Figure 4. Illustration of the optimization process. With static 3D Gaussian initialization, our model can learn high-quality 3D Gaussians of the motion part.

The rendering process of our 4D Gaussian Splatting is depicted in Fig. 2 (c). Our 4D Gaussian splatting converts the original 3D Gaussians $\mathcal{G}$ into another group of 3D Gaussians $\mathcal{G}'$ given a timestamp t, maintaining the effectiveness of the differential splatting as referred in [57].

4.2. Gaussian Deformation Field Network

The network to learn the Gaussian deformation field includes an efficient spatial-temporal structure encoder $\mathcal{H}$ and a Gaussian deformation decoder $\mathcal{D}$ for predicting the deformation of each 3D Gaussian.

Spatial-Temporal Structure Encoder. Nearby 3D Gaussians always share similar spatial and temporal information. To model 3D Gaussians' features effectively, we introduce an efficient spatial-temporal structure encoder $\mathcal{H}$ including a multi-resolution HexPlane $R(i, j)$ and a tiny MLP ϕ_d inspired by [5, 8, 11, 44]. While the vanilla 4D neural voxel is memory-consuming, we adopt a 4D K-Planes [11] module to decompose the 4D neural voxel into 6 planes. All 3D Gaussians in a certain area can be contained in the bounding plane voxels and Gaussian's deformation can also be encoded in nearby temporal voxels.

Specifically, the spatial-temporal structure encoder $\mathcal{H}$ contains 6 multi-resolution plane modules $R_l(i, j)$ and a tiny MLP ϕ_d, *i.e.* $\mathcal{H}(\mathcal{G}, t) = \{R_l(i, j), \phi_d | (i, j) \in \{(x, y), (x, z), (y, z), (x, t), (y, t), (z, t)\}, l \in \{1, 2\}\}$. The position $\mu = (x, y, z)$ is the mean value of 3D Gaussians $\mathcal{G}$. Each voxel module is defined by $R(i, j) \in \mathbb{R}^{h \times lN_i \times lN_j}$, where h stands for the hidden dim of features, N denotes the basic resolution of voxel grid and l equals to the upsampling scale. This entails encoding information of the 3D

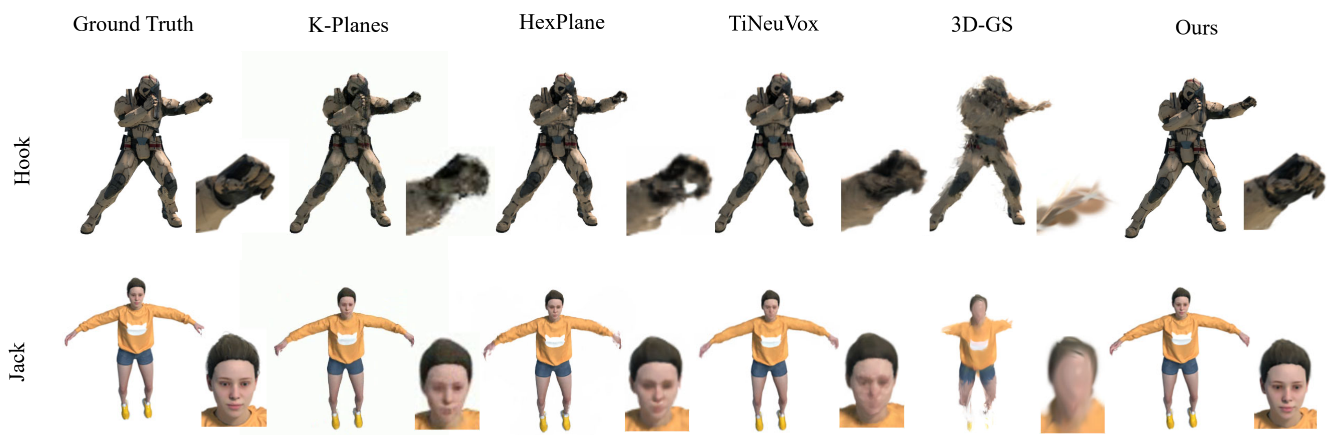


Figure 5. Visualization of synthesized datasets compared with other models [5, 8, 11, 17, 19, 49]. The rendering results of [11] are displayed with a default green background. We have adopted their rendering settings.

Gaussians within the 6 2D voxel planes while considering temporal information. The formula for computing separate voxel features is as follows:

$$\begin{gathered} f_h = \bigcup_l \prod \operatorname{interp}(R_l(i,j)), \\ (i,j) \in \{(x,y),(x,z),(y,z),(x,t),(y,t),(z,t)\}. \end{gathered} \tag{7}$$

$f_h \in \mathbb{R}^{h*l}$ is the feature of neural voxels. 'interp' denotes the bilinear interpolation for querying the voxel features located at 4 vertices of the grid. The discussion of the production process is similar to [11]. Then a tiny MLP ϕ_d merges all the features by $f_d = \phi_d(f_h)$.

Multi-head Gaussian Deformation Decoder. When all the features of 3D Gaussians are encoded, we can compute any desired variable with a multi-head Gaussian deformation decoder $\mathcal{D} = \{\phi_x, \phi_r, \phi_s\}$. Separate MLPs are employed to compute the deformation of position $\Delta\mathcal{X} = \phi_x(f_d)$, rotation $\Delta r = \phi_r(f_d)$, and scaling $\Delta s = \phi_s(f_d)$. Then, the deformed feature $(\mathcal{X}', r', s')$ can be addressed as:

$$(\mathcal{X}', r', s') = (\mathcal{X} + \Delta\mathcal{X}, r + \Delta r, s + \Delta s). \tag{8}$$

Finally, we obtain the deformed 3D Gaussians $\mathcal{G}' = \{\mathcal{X}', s', r', \sigma, \mathcal{C}\}$.

4.3. Optimization

3D Gaussian Initialization. [19] shows that 3D Gaussians can be well-trained with structure from motion (SfM) [43] points initialization. Similarly, 4D Gaussians should also be fine-tuned in proper 3D Gaussian initialization. We optimize 3D Gaussians at initial 3000 iterations for warm-up and then render images with 3D Gaussians $\hat{I} = \mathcal{S}(M, \mathcal{G})$ instead of 4D Gaussians $\hat{I} = \mathcal{S}(M, \mathcal{G}')$. The illustration of the optimization process is shown in Fig. 4.

Loss Function. Similar to other reconstruction methods [8, 19, 39], we use the L1 color loss to supervise the training process. A grid-based total-variational loss [5, 8, 11, 47] $\mathcal{L}_{tv}$ is also applied.

$$\mathcal{L} = |\hat{I} - I| + \mathcal{L}_{tv}. \tag{9}$$

5. Experiment

In this section, we mainly introduce the hyperparameters and datasets of our settings in Sec. 5.1 and the results between different datasets will be compared with [2, 5, 8, 11, 19, 27, 45, 49, 50] in Sec. 5.2. Then, ablation studies are proposed to prove the effectiveness of our approaches in Sec. 5.3 and more discussion about 4D-GS in Sec. 5.4. Finally, we discuss the limitation of our proposed 4D-GS in Sec. 5.5.

5.1. Experimental Settings

Our implementation is primarily based on the PyTorch [37] framework and tested in a single RTX 3090 GPU, and we've fine-tuned our optimization parameters by the configuration outlined in the 3D-GS [19]. More hyperparameters will be shown in the appendix.

Synthetic Dataset. We primarily assess the performance of our model using synthetic datasets, as introduced by D-NeRF [39]. These datasets are designed for monocular settings, although it's worth noting that the camera poses for each timestamp are close to randomly generated. Each scene within these datasets contains dynamic frames, ranging from 50 to 200 in number.

Real-world Datasets. We utilize datasets provided by HyperNeRF [36] and Neu3D's [22] as benchmark datasets to evaluate the performance of our model in real-world scenarios. The Nerfies dataset is captured using one or two

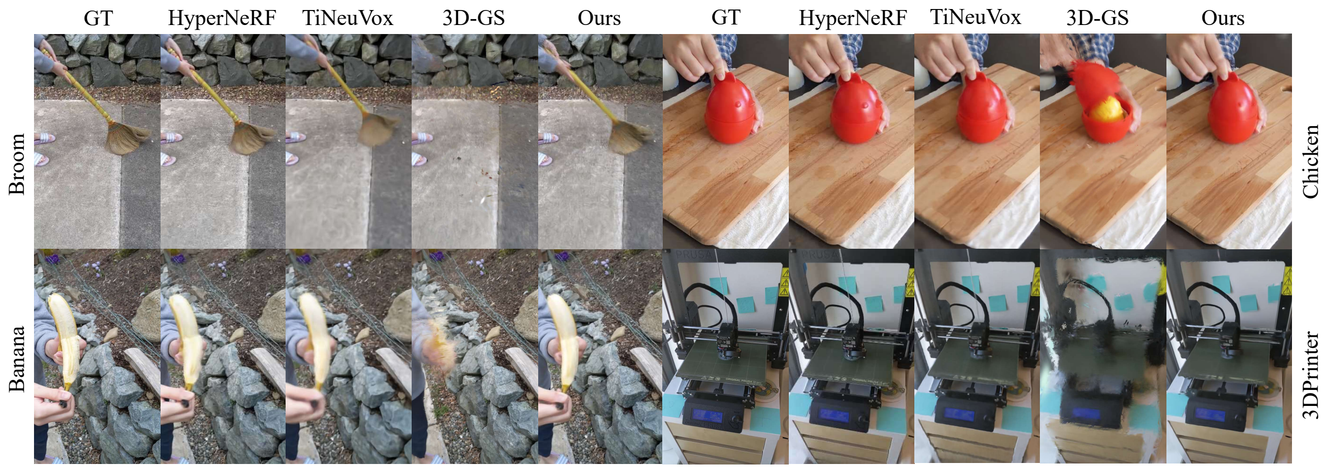


Figure 6. Visualization of the HyperNeRF [36] datasets compared with other methods [8, 17, 19, 36]. 'GT' stands for ground truth images.

Table 1. Quantitative results on the synthesis dataset. The best and the second best results are denoted by pink and yellow. The rendering resolution is set to 800×800. "Time" in the table stands for training times.

Model	PSNR(dB)↑	SSIM↑	LPIPS↓	Time↓	FPS ↑	Storage (MB)↓
TiNeuVox-B [8]	32.67	0.97	0.04	28 mins	1.5	48
KPlanes [11]	31.61	0.97	-	52 mins	0.97	418
HexPlane-Slim [5]	31.04	0.97	0.04	11m 30s	2.5	38
3D-GS [19]	23.19	0.93	0.08	10 mins	170	10
FFDNeRF [17]	32.68	0.97	0.04	-	< 1	440
MSTH [49]	31.34	0.98	0.02	6 mins	-	-
V4D [12]	33.72	0.98	0.02	6.9 hours	2.08	377
Ours	34.05	0.98	0.02	8 mins	82	18

cameras, following straightforward camera motion, while the Neu3D's dataset is captured using 15 to 20 static cameras, involving extended periods and intricate camera motions. We use the points computed by SfM [43] from the first frame of each video in Neu3D's dataset and 200 frames randomly selected in HyperNeRF's.

5.2. Results

We primarily assess our experimental results using various metrics, encompassing peak-signal-to-noise ratio (PSNR), perceptual quality measure LPIPS [60], structural similarity index (SSIM) [52] and its extensions including structural dissimilarity index measure (DSSIM), multiscale structural similarity index (MS-SSIM), FPS, training times and Storage.

To assess the quality of novel view synthesis, we conducted benchmarking against several state-of-the-art methods in the field, including [5, 8, 11, 12, 17, 19, 27, 49]. The results are summarized in Tab. 1. While current dynamic hybrid representations can produce high-quality results, they often come with the drawback of rendering speed. The lack of modeling dynamic motion part makes [19] fail to reconstruct dynamic scenes. In contrast, our method enjoys both the highest rendering quality within the synthesis dataset and exceptionally fast rendering speeds while keeping extremely low storage consumption and convergence time.

Additionally, the results obtained from real-world datasets are presented in Tab. 2 and Tab. 3. It becomes apparent that some NeRFs [2, 5, 45] suffer from slow convergence speed, and the other grid-based NeRF methods [5, 8, 11, 49] encounter difficulties when attempting to capture intricate object details. In stark contrast, our methods research comparable rendering quality, fast convergence, and excel in free-view rendering speed in indoor cases. Though [27] addresses the high quality in comparison to ours, the need for multi-cam setups makes it hard to model monocular scenes and other methods also limit free-view rendering speed and storage.

5.3. Ablation Study

Spatial-Temporal Structure Encoder. The explicit HexPlane encoder $R_l(i, j)$ possesses the capacity to retain 3D Gaussians' spatial and temporal information, which can reduce storage consumption in comparison with purely explicit methods [30]. Discarding this module, we observe

Table 2. Quantitative results on HyperNeRF's [36] vrig dataset. Rendering resolution is set to 960×540.

Model	PSNR(dB)↑	MS-SSIM↑	Times↓	FPS↑	Storage(MB)↓
Nerfies [35]	22.2	0.803	$\sim$ hours	< 1	-
HyperNeRF [36]	22.4	0.814	32 hours	< 1	-
TiNeuVox-B [8]	24.3	0.836	30 mins	1	48
3D-GS [19]	19.7	0.680	40 mins	55	52
FFDNeRF [17]	24.2	0.842	-	0.05	440
V4D [12]	24.8	0.832	5.5 hours	0.29	377
Ours	25.2	0.845	30 mins	34	61

Table 3. Quantitative results on the Neu3D's [22] dataset, rendering resolution is set to 1352×1014.

Model	PSNR(dB)↑	D-SSIM↓	LPIPS↓	Time ↓	FPS↑	Storage (MB)↓
NeRFPlayer [45]	30.69	0.034	0.111	6 hours	0.045	-
HyperReel [2]	31.10	0.036	0.096	9 hours	2.0	360
HexPlane-all* [5]	31.70	0.014	0.075	12 hours	0.2	250
KPlanes [11]	31.63	-	-	1.8 hours	0.3	309
Im4D [27]	32.58	-	0.208	28 mins	$\sim$5	93
MSTH [49]	32.37	0.015	0.056	20 mins	2(15‡)	135
Ours	31.15	0.016	0.049	40 mins	30	90

*: The metrics of the model are tested without "coffee martini" and resolution is set to 1024×768.
‡: The FPS is tested with fixed-view rendering.

that using only a shallow MLP ϕ_d falls short in modeling complex deformations across various settings. Tab. 4 demonstrates that, while the model incurs minimal memory costs, it does come at the expense of rendering quality.

Gaussian Deformation Decoder. Our proposed Gaussian deformation decoder $\mathcal{D}$ decodes the features from the spatial-temporal structure encoder $\mathcal{H}$. All the changes in 3D Gaussians can be explained by separate MLPs $\{\phi_x, \phi_r, \phi_s\}$. As is shown in Tab. 4, 4D Gaussians cannot fit dynamic scenes well without modeling 3D Gaussian motion. Meanwhile, the movement of human body joints is typically manifested as stretching and twisting of surface details in a macroscopic view. If one aims to accurately model these movements, the size and shape of 3D Gaussians should also be adjusted accordingly. Otherwise, there may be underfitting of details during excessive stretching, or an inability to correctly simulate the movement of objects at a microscopic level.

3D Gaussian Initialization. In some cases without SfM [43] points initialization, training 4D-GS directly may cause difficulty in convergence. Optimizing 3D Gaussians for warm-up enjoys: (a) making some 3D Gaussians stay in the dynamic part, which releases the pressure of large deformation learning by 4D Gaussians as shown in Fig. 4. (b) learning proper 3D Gaussians $\mathcal{G}$ and suggesting deformation fields paying more attention to the dynamic part. (c) avoiding numeric errors in optimizing the Gaussian deformation network $\mathcal{F}$ and keeping the training process stable.

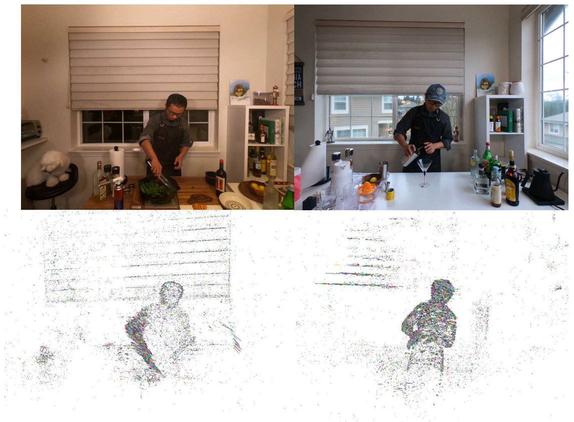

(a) Cook Spinach (b) Coffee Martini

Figure 7. Visualization of tracking with 3D Gaussians. Each line in the figure of second rows stands for trajectories of 3D Gaussians

Tab. 4 also shows that if we train our model without the warm-up coarse stage, the rendering quality will suffer.

5.4. Discussions

Tracking with 3D Gaussians. Tracking in 3D is also a important task. [17] also shows tracking objects' motion in 3D. Different from dynamic3DGS [30], our methods even can present tracking objects in monocular settings with pretty low storage *i.e.* 10MB in 3D Gaussians $\mathcal{G}$ and 8 MB in Gaussian deformation field network $\mathcal{F}$. Fig. 7 shows the 3D Gaussian's deformation at certain timestamps.

Table 4. Ablation studies on synthetic datasets using our proposed methods.

Model	PSNR(dB)↑	SSIM↑	LPIPS↓	Time↓	FPS↑	Storage (MB)↓
Ours w/o HexPlane $R_l(i,j)$	27.05	0.95	0.05	4 mins	140	12
Ours w/o initialization	31.91	0.97	0.03	7.5 mins	79	18
Ours w/o ϕ_x	26.67	0.95	0.07	8 mins	82	17
Ours w/o ϕ_r	33.08	0.98	0.03	8 mins	83	17
Ours w/o ϕ_s	33.02	0.98	0.03	8 mins	82	17
Ours	34.05	0.98	0.02	8 mins	82	18

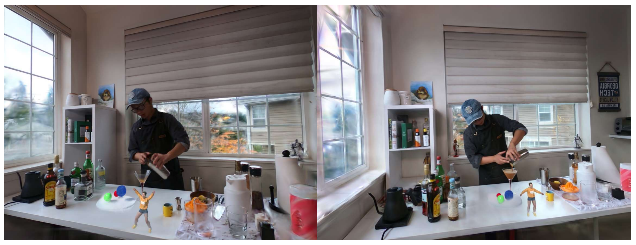


Figure 8. Visualization of composition with 4D Gaussians.

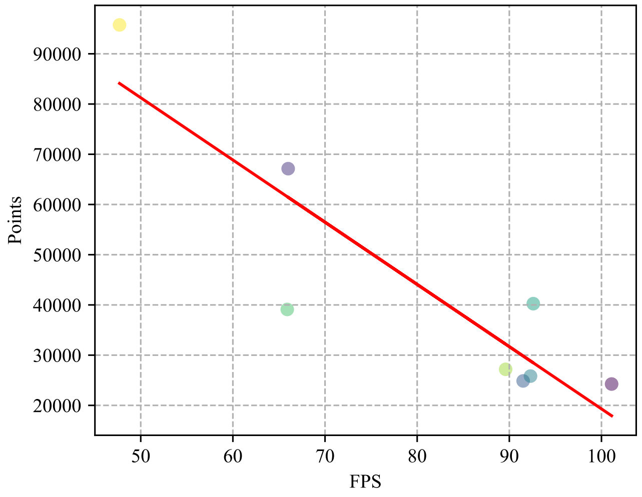


Figure 9. Visualization of the relationship between rendering speed and numbers of 3D Gaussians in the rendered screens. All the tests are finished in the synthesis dataset.

Composition with 4D Gaussians. Similar to dynamic3DGS [30], our proposed methods can also propose editing in 4D Gaussians in Fig. 8. Thanks to the explicit representation of 3D Gaussians, all the trained models can predict deformed 3D Gaussians in the same space following $\mathcal{G}' = \{\mathcal{G}'_1, \mathcal{G}'_2, ..., \mathcal{G}'_n\}$ and differential rendering [57] can project all the point clouds into viewpoints by $\hat{I} = \mathcal{S}(M, \mathcal{G}')$.

Analysis of Rendering Speed. As is shown in Fig. 9, we also test the relationship between points in the rendered screen and rendering speed at the resolution of 800×800. We found that if the rendered points are lower than 30000, the rendering speed can be up to 90. The config of Gaussian deformation fields are discussed in the appendix. To achieve render-time rendering speed, we should strike a balance among all the rendering resolutions, 4D Gaussians representation including numbers of 3D Gaussians, and the capacity of the Gaussian deformation field network and any other hardware constraints.

5.5. Limitations

Though 4D-GS can indeed attain rapid convergence and yield real-time rendering outcomes in many scenarios, there are a few key challenges to address. First, large motions, the absence of background points, and the unprecise camera pose cause the struggle of optimizing 4D Gaussians. What is more, it is still challenging to 4D-GS also cannot split the joint motion of static and dynamic Gaussiansparts under the monocular settings without any additional supervision. Finally, a more compact algorithm needs to be designed to handle urban-scale reconstruction due to the heavy querying of Gaussian deformation fields by huge numbers of 3D Gaussians.

6. Conclusion

This paper proposes 4D Gaussian splatting to achieve real-time dynamic scene rendering. An efficient deformation field network is constructed to accurately model Gaussian motions and shape deformations, where adjacent Gaussians are connected via a spatial-temporal structure encoder. Connections between Gaussians lead to more complete deformed geometry, effectively avoiding avulsion. Our 4D Gaussians can not only model dynamic scenes but also have the potential for 4D objective tracking and editing.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62376102). The authors would like to thank Haotong Lin for providing the quantitative results of Im4D [27].

References

[1] Jad Abou-Chakra, Feras Dayoub, and Niko Sünderhauf. Particlenerf: Particle based encoding for online neural radiance fields in dynamic scenes. *arXiv preprint arXiv:2211.04041*, 2022. 3

[2] 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. 5, 6, 7

[3] 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

[4] 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. 2

[5] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023. 1, 2, 4, 5, 6, 7

[6] 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):1–13, 2015. 2

[7] Robert A Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. *ACM Siggraph Computer Graphics*, 22(4): 65–74, 1988. 1

[8] 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*, pages 1–9, 2022. 1, 2, 4, 5, 6, 7

[9] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2367–2376, 2019. 2

[10] 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. 1, 3

[11] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023. 1, 2, 4, 5, 6, 7

[12] Wanshui Gan, Hongbin Xu, Yi Huang, Shifeng Chen, and Naoto Yokoya. V4d: Voxel for 4d novel view synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 2023. 2, 6, 7

[13] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5712–5721, 2021. 2

[14] Xiangjun Gao, Jiaolong Yang, Jongyoo Kim, Sida Peng, Zicheng Liu, and Xin Tong. Mps-nerf: Generalizable 3d human rendering from multiview images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 2

[15] Kaiwen Guo, Feng Xu, Yangang Wang, Yebin Liu, and Qionghai Dai. Robust non-rigid motion tracking and surface reconstruction using l0 regularization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3083–3091, 2015. 2

[16] Kaiwen Guo, Peter Lincoln, Philip Davidson, Jay Busch, Xueming Yu, Matt Whalen, Geoff Harvey, Sergio Orts-Escolano, Rohit Pandey, Jason Dourgarian, et al. The relightables: Volumetric performance capture of humans with realistic relighting. *ACM Transactions on Graphics (ToG)*, 38(6):1–19, 2019. 2

[17] Xiang Guo, Jiadai Sun, Yuchao Dai, Guanying Chen, Xiaoqing Ye, Xiao Tan, Errui Ding, Yumeng Zhang, and Jingdong Wang. Forward flow for novel view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16022–16033, 2023. 2, 5, 6, 7

[18] Tao Hu, Tao Yu, Zerong Zheng, He Zhang, Yebin Liu, and Matthias Zwicker. Hvtr: Hybrid volumetric-textural rendering for human avatars. In *2022 International Conference on 3D Vision (3DV)*, pages 197–208. IEEE, 2022. 2

[19] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14, 2023. 1, 3, 4, 5, 6, 7

[20] Leonid Keselman and Martial Hebert. Approximate differentiable rendering with algebraic surfaces. In *European Conference on Computer Vision*, pages 596–614. Springer, 2022. 3

[21] Leonid Keselman and Martial Hebert. Flexible techniques for differentiable rendering with 3d gaussians. *arXiv preprint arXiv:2308.14737*, 2023. 3

[22] 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. 2, 5, 7

[23] Zhong Li, Yu Ji, Wei Yang, Jinwei Ye, and Jingyi Yu. Robust 3d human motion reconstruction via dynamic template construction. In *2017 International Conference on 3D Vision (3DV)*, pages 496–505. IEEE, 2017. 2

[24] Zhong Li, Minye Wu, Wangyiteng Zhou, and Jingyi Yu. 4d human body correspondences from panoramic depth maps.

In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2877–2886, 2018. 2

[25] 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*, pages 6498–6508, 2021. 2

[26] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. *arXiv preprint arXiv:2312.16812*, 2023. 3

[27] Haotong Lin, Sida Peng, Zhen Xu, Tao Xie, Xingyi He, Hujun Bao, and Xiaowei Zhou. High-fidelity and real-time novel view synthesis for dynamic scenes. In *SIGGRAPH Asia Conference Proceedings*, 2023. 2, 5, 6, 7, 8

[28] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9246–9255, 2019. 3

[29] Yu-Lun Liu, Chen Gao, Andreas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang. Robust dynamic radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13–23, 2023. 2

[30] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 2, 3, 6, 7, 8

[31] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021. 4

[32] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 1, 2, 3

[33] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 1, 3

[34] Byeongjun Park and Changick Kim. Point-dynrf: Point-based dynamic radiance fields from a monocular video. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3171–3181, 2024. 3

[35] 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. 2, 7

[36] 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. *arXiv preprint arXiv:2106.13228*, 2021. 2, 4, 5, 6, 7

[37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 5

[38] Sida Peng, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Representing volumetric videos as dynamic mlp maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4252–4262, 2023. 2

[39] 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. 2, 5

[40] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 3

[41] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 3

[42] Darius Rückert, Linus Franke, and Marc Stamminger. Adop: Approximate differentiable one-pixel point rendering. *ACM Transactions on Graphics (ToG)*, 41(4):1–14, 2022. 3

[43] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. 5, 6, 7

[44] Ruizhi Shao, Zerong Zheng, Hanzhang Tu, Boning Liu, Hongwen Zhang, and Yebin Liu. Tensor4d: Efficient neural 4d decomposition for high-fidelity dynamic reconstruction and rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16632–16642, 2023. 1, 2, 4

[45] 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. 5, 6, 7

[46] Zhuo Su, Lan Xu, Zerong Zheng, Tao Yu, Yebin Liu, and Lu Fang. Robustfusion: Human volumetric capture with data-driven visual cues using a rgbd camera. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pages 246–264. Springer, 2020. 2

[47] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469, 2022. 1, 3, 5

[48] Fengrui Tian, Shaoyi Du, and Yueqi Duan. Mononerf: Learning a generalizable dynamic radiance field from

monocular videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17903–17913, 2023. 2

[49] Feng Wang, Zilong Chen, Guokang Wang, Yafei Song, and Huaping Liu. Masked space-time hash encoding for efficient dynamic scene reconstruction. *Advances in neural information processing systems*, 2023. 2, 5, 6, 7

[50] 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. 2, 5

[51] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3295–3306, 2023. 2

[52] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 6

[53] Qingshan Xu, Weihang Kong, Wenbing Tao, and Marc Pollefeys. Multi-scale geometric consistency guided and planar prior assisted multi-view stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4945–4963, 2022. 2

[54] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022. 3

[55] Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642*, 2023. 2, 3

[56] Taoran Yi, Jiemin Fang, Xinggang Wang, and Wenyu Liu. Generalizable neural voxels for fast human radiance fields. *arXiv preprint arXiv:2303.15387*, 2023. 2

[57] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)*, 38(6):1–14, 2019. 2, 3, 4, 8

[58] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2790–2799, 2018. 3

[59] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2

[60] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 6

[61] Kaichen Zhou, Jia-Xing Zhong, Sangyun Shin, Kai Lu, Yiyuan Yang, Andrew Markham, and Niki Trigoni. Dynpoint: Dynamic neural point for view synthesis. *Advances in Neural Information Processing Systems*, 36, 2024. 2, 3

[62] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, 2001. 3