Fully Explicit Dynamic Gaussian Splatting

Junoh Lee[1], Changyeon Won[2], Hyunjun Jung[2], Inhwan Bae[2], Hae-Gon Jeon[1,2]
[1]School of Electrical Engineering and Computer Science [2]AI Graduate School
Gwangju Institute of Science and Technology
{juno,cywon1997,hyunjun.jung,inhwanbae}@gm.gist.ac.kr, haegonj@gist.ac.kr

Abstract

3D Gaussian Splatting has shown fast and high-quality rendering results in static scenes by leveraging dense 3D prior and explicit representations. Unfortunately, the benefits of the prior and representation do not involve novel view synthesis for dynamic motions. Ironically, this is because the main barrier is the reliance on them, which requires increasing training and rendering times to account for dynamic motions. In this paper, we design Explicit 4D Gaussian Splatting (Ex4DGS). Our key idea is to firstly separate static and dynamic Gaussians during training, and to explicitly sample positions and rotations of the dynamic Gaussians at sparse timestamps. The sampled positions and rotations are then interpolated to represent both spatially and temporally continuous motions of objects in dynamic scenes as well as reducing computational cost. Additionally, we introduce a progressive training scheme and a point-backtracking technique that improves Ex4DGS's convergence. We initially train Ex4DGS using short timestamps and progressively extend timestamps, which makes it work well with a few point clouds. The point-backtracking is used to quantify the cumulative error of each Gaussian over time, enabling the detection and removal of erroneous Gaussians in dynamic scenes. Comprehensive experiments on various scenes demonstrate the state-of-the-art rendering quality from our method, achieving fast rendering of 62 fps on a single 2080Ti GPU.

1 Introduction

The recent flood of video content has encouraged view synthesis techniques for highly engaging, visually rich content creation to maintain viewer interest. However, even short form videos require a huge time for both data pre-processing, computing frame-wise 3D point clouds, and training time in novel view synthesis for dynamic motions. Furthermore, these techniques must be considered for running on mobile devices which have a limited computing power and storage space. In this aspect, it is crucial to not only achieve photorealistic rendering results, but also reduce the computational cost related to storage, memory and rendering pipeline. To achieve this, the spatio-temporal representation should be explicit and efficient to handle the complexity of dynamic motions in videos.

Recent methods for dynamic view synthesis are typically based on Neural Radiance Fields (NeRF) [1], which uses implicit multi-layer perceptron (MLP) with a combination of 5-DoF spatial coordinates and the additional time axis [2, 3, 4, 5, 6, 7, 8]. The MLP-based methods have shown high-fidelity rendering quality. However, the inevitable cost of decoding the implicit representation makes rendering critically slow. Various methods try to reduce the cost by adapting explicit representation, such as voxel [9, 10] and matrix decomposition [11, 12]. Nevertheless, since NeRF-based approaches require per-pixel ray sampling and dense sampling for each ray, it is hard to achieve real-time and high-resolution renderings.

Meanwhile, a promising alternative, 3D Gaussian Splatting (3DGS) [13], has emerged, which achieves photo-realistic rendering results with significantly faster training and rendering speeds.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Unlike NeRF-based approaches, 3DGS exploits fully explicit point-based primitives of 3D and employs a rasterization-based rendering pipeline. Recent advances [14, 15] attempt to extend 3DGS to 4D domain, handling the motion over time by storing the additional transformation information of 3D coordinates or 4D bases. However, these methods can only be trained under a restricted condition with dense point clouds. Moreover, these approaches borrow implicit representations for implementation, which lose the inherent advantage of 3DGS and make real-world applications more difficult. To become a more scalable model for real-world situations, it is important to be trained under more in-the-wild conditions (i.e., sparse point cloud) with more concise representation.

In this paper, we present Explicit 4D Gaussian Splatting (Ex4DGS), a keyframe interpolation-based 4D Gaussian splatting method that works well in a fully explicit on-time domain. Our key idea is to apply interpolation techniques under temporal explicit representation to realize the scalable 3DGS model. Temporal interpolation is a widely used technique in computer graphics [16] that only stores keyframe information in video and determines smooth transitions for the rest of the frames. We select keyframes with sparse time intervals and save the additional temporal information at each keyframe which includes each Gaussian's position, rotation and opacity. This information is fully explicit; it is stored without any encoding process, and continuous motion is calculated to have smooth temporal transitions between adjacent keyframes. Here, we use a polynomial basis interpolator for position, a spherical interpolator for rotation, and a simplified Gaussian mixture model for opacity. Specifically, the polynomial basis of cubic Hermite spline (CHip) [17] is used to effectively avoid overfitting or over-smoothing problems by spanning low-degree polynomials. For rotation, we introduce Spherical Linear Interpolation (Slerp) [18] to do a linear transition over angles. Lastly, we introduce a simplified Gaussian mixture model, which allows temporal opacity to handle the appearance and disappearance of objects.

For further optimization, we reduce the computational cost by isolating dynamic points from static points, and only storing additional temporal information of dynamic points. Here, we aim for this separation to be possible without any additional inputs such as object masks [19]. To tackle this, we introduce motion-based triggers to distinguish static and dynamic points in a scene. We first initialize all Gaussian points in a scene to be static which are assumed to move linearly. During training, static points with large movements are automatically classified as dynamic points. Next, we adopt a progressive training scheme to train our model even under sparse point cloud conditions. The progressive training, starting with a short duration and scaling up, can prevent falling into local minima by reducing the sudden appearance of objects. Lastly, an additional point backtracking technique is introduced to enhance rendering quality. Detecting redundant points in a dynamic scene is challenging because we need to consider all visible timestamps. To measure accumulated errors over time, we apply the point-backtracking technique that can track and prune high-error Gaussians in dynamic scenes.

The effectiveness of our method is validated through experiments on two major real-world video datasets: Neural 3D Video dataset [7] and Technicolor dataset [20]. Experimental results demonstrate that our approach significantly improves rendering results even with sparse 3D point clouds. Furthermore, benefiting from the proposed optimization scheme, our model only requires low storage and memory size without any auxiliary modules or tricks to encode lengthy temporal information. Finally, our model achieves 62 fps on a single 2080Ti GPU on 300-frame scenes with a 1352×1014 resolution.

2 Related Works

2.1 Novel View Synthesis

Photorealistic novel view synthesis can be achieved using a set of densely sampled images through image-based rendering [21, 22]. While the dense sampling of views is limited by memory constraints and results in aliased rendering, novel view synthesis has advanced with the development of neural networks. The ability of neural networks to process implicit information in images enables novel view synthesis with a sparse set of observations. Prior works on constructing multi-plane images (MPI)[23, 24, 25] use aggregated pixel information from correspondences in sparsely sampled views. MPI representation may fail with wide angles between the camera and depth planes. This can be mitigated using geometric proxies like depth information [26, 27], plane-sweep volumes [28, 29], and meshes [30, 31]. These methods risk unrealistic view synthesis with inaccurate proxy geometry.

Joint optimization of proxy geometry [32, 33, 34, 35] can help, but direct mesh optimization often gets stuck in local minima due to poor loss landscape conditioning.

In recent years, continuous 3D representation through neural networks has received widespread attention. NeRF [1] is the foundational work that started this trend. It implicitly learns the shape of an object as a density field, which makes optimization via gradient descent more tractable. The robustness of such geometric neural representations enables to reconstruct accurate geometry from a few given images [36, 37, 38, 39], large-scale geometry [40, 41, 42, 43, 44, 45], disentangled textures [46, 47, 48], and material properties [49, 50, 51, 52, 53]. However, one major issue on these neural representations is also slow rendering speeds coming from volume renderers. To resolve this issue, works in [54, 55] develop light field-inspired representations for single-pass pixel rendering. Both [56] and [57] introduce efficient voxel-based scene representations to improve the rendering speed. However, continuous representation inevitably combines with neural networks to implement its complex nature, and this limits rendering speed. As an alternative, 3DGS [13] is designed to construct an explicit radiance field through rasterization, not requiring MLP-based inference. This method leverages anisotropic 3D Gaussians as the scene representation and proposes a differentiable rasterizer to render them by splatting onto the image plane for static scenes.

2.2 Dynamic Novel View Synthesis

The time domain in dynamic scenes is typically parameterized with Plenoptic function [58]. Classical image-based rendering [21, 22] and novel view synthesis methods using explicit geometry [30, 31] are restricted to a limited memory space [21] when they extend to the time dimension. This is because they require additional optimization procedures for frame-by-frame dynamic view synthesis and storage for the parameterized space. Fueled by implicit representations, works in [2, 3, 8, 59, 4, 7, 5] handle challenging tasks using neural volume rendering. They learn dynamic scenes by optimizing deformation and canonical fields for object motions [2, 3, 6], using human body priors [60, 59, 61, 62, 63, 64, 65, 66], and decoupling dynamic parts [4, 5, 67, 68, 69, 70, 71, 8, 72]. These methods introduce additional neural fields or learnable variables to represent the time domain, taking more memory usage and rendering time as well.

Temporally extended 3DGS has been considered as a feasible solution to dynamic novel view synthesis. A work in [73] assigns parameters to 3DGS at each timestamp and imposes rigidity through a regularization. Another work in [74] leverages Gaussian probability to model density changes over time to explicitly represent dynamic scenes. However, they require many primitives to capture complex temporal changes. Concurrently, works in [14, 75, 76, 77, 78, 79] utilize MLPs to represent the temporal changes. These methods inherit the drawbacks of dynamic neural representations, resulting in slower rendering speeds. The others in [15, 80] explicitly parameterize the motion of dynamic 3D Gaussians to preserve the rendering speed of 3DGS by predicting their trajectory function. However, they only handle the motion as a continuous trajectory, and require multiple Gaussians for motions that disappear and reappear due to self-occlusion, increasing memory burden.

In contrast, our key idea for dynamic 3D Gaussian uses keyframes to minimize primitives and to devise a progressive optimization to cope with scenarios where face disappearing/reappearing objects. Thanks to our schemes, we can improve rendering speed, memory efficiency, and achieve impressive performance for dynamic novel view synthesis.

3 Preliminary: 3D Gaussian Splatting

Our model starts from the point-based differentiable rasterization of 3DGS [13]. 3DGS uses three-dimensional Gaussian as geometry primitive, which is composed of position (mean) μ, covariation $\mathbf{\Sigma}$, density σ and color $\boldsymbol{c}$. The 3D Gaussian is referred to as follows:

$$\mathcal{G}(\boldsymbol{x}) = e^{-\frac{1}{2}(\boldsymbol{x}-\mu)^\top \mathbf{\Sigma}^{-1}(\boldsymbol{x}-\mu)}. \tag{1}$$

We need to project the 3D Gaussian onto a 2D plane to render an image. In this process, the approximated graphics pipeline is used to render 2D Gaussians. The covariance matrix $\mathbf{\Sigma}'$ in camera coordinate is given as follows:

$$\mathbf{\Sigma}' = \boldsymbol{J}\,\boldsymbol{W}\,\mathbf{\Sigma}\,\boldsymbol{W}^\top \boldsymbol{J}^\top, \tag{2}$$

where $\boldsymbol{J}$ is the Jacobian of the affine approximation of the perspective projection and $\boldsymbol{W}$ is a viewing transformation. By skipping the third row and column of $\mathbf{\Sigma}'$, it is approximated to two-dimensional anisotropic Gaussian on the image plane.

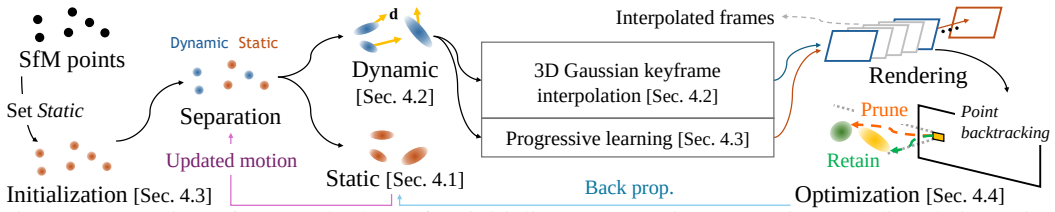


Figure 1: Overview of our method. We first initialize 3D Gaussians as static, modeling their motion linearly. During optimization, dynamic and static objects are separated based on the amount of predicted motion, and the 3D Gaussians between the selected keyframes are interpolated and rendered.

The covariance is a positive semi-definite which can be decomposed into a scale $\boldsymbol{S}$ and a rotation $\boldsymbol{R}$ as:

$$\boldsymbol{\Sigma} = \boldsymbol{R}\boldsymbol{S}\boldsymbol{S}^\top\boldsymbol{R}^\top. \tag{3}$$

Spherical harmonics coefficients are used to represent view-dependent color changes as proposed in [81]. A rendered color from the Gaussian uses point-based α blending similar to NeRF's volume rendering. For the interval between points along ray δ which can be obtained from $\mathcal{G}(\boldsymbol{x})$, the color of ray is

$$C = \sum_{i=1}^{N} T_i(1 - e^{-\sigma_i\delta_i})\boldsymbol{c}_i,\ T_i = e^{-\sum_{j=1}^{i-1}\sigma_j\delta_j}, \tag{4}$$

where N is the number of visible Gaussians along the ray, i and j denote the order of Gaussians by depth.

4 Methodology

To achieve both memory efficiency and rendering capacity, our scheme is two-fold: (1) Keyframe-based interpolation to span position and rotation of Gaussians over time; (2) Classification of static and dynamic Gaussian. These are described in Sections 4.1 and 4.2. After that, we introduce our progressive training scheme to handle a variety of running times in Section 4.3, and deal with details of the optimization process of our method in Section 4.4. The overview of our method is depicted in Figure 1.

4.1 Static Gaussians

Static Gaussian $\mathcal{G}_s$ is modeled as the same as the 3DGS model except for its position. $\mathcal{G}_s$ changes the position linearly over time, which can be formulated with the position μ at time t as below:

$$\mu(t) = \mathbf{x} + t'\mathbf{d}, t' = \frac{t}{l} \in [0, 1] \tag{5}$$

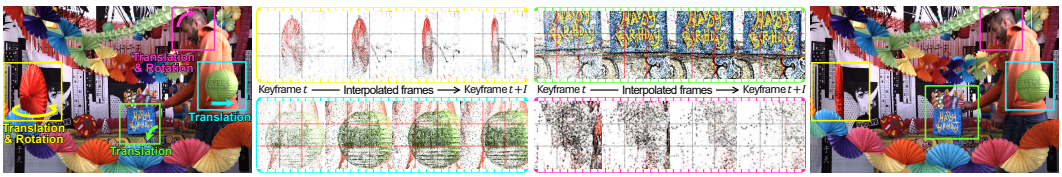
where $\mathbf{x}$ is a pivot position of $\mathcal{G}_s$ and $\mathbf{d}$ is a vector representing the translation, and l is the duration of a scene. We normalize t with l to prevent $\mathbf{d}$ from becoming too large.

4.2 Dynamic Gaussians

The dynamic Gaussian model is based on interpolations of keyframes, as visualized in Figure 2. Specifically, the state of the dynamic Gaussian $\mathcal{G}_d$ at an intermediate timestamp is synthesized from adjacent keyframes. In this work, we assume that the keyframe interval is uniform for simplicity. The keyframe is defined as $\mathcal{K} = \{t \mid t = nI, n \in \mathbb{Z}, t \in \mathcal{T}\}$ where I is its interval and $\mathcal{T}$ is a set of timestamps. $\mathcal{G}_d$ acquires position μ and rotation in quaternion r from keyframe information. We use different interpolators with different properties for smooth and continuous motion: CHip using polynomial bases is applied for positions, and a Slerp is used for rotations. We further adapt the Gaussian mixture model for temporal opacity to handle changes in the visibility of objects over time.

4.2.1 Cubic Hermite Interpolator for Temporal Position

CHip uses a third-degree polynomial. It is commonly used to model dynamic motions or shapes [17]. The interpolator function can be defined with third-degree polynomials and four variables: position and tangent vector of the start and end points. On the unit interval $[0, 1]$, given a start point p_0 at $t = 0$ and an end point p_1 at $t = 1$ with start tangent m_0 at $t = 0$ and an end tangent m_1 at $t = 1$, CHip can be defined as:



(a) Rendered image at t (b) Interpolated dynamic points (c) Rendered image at $t+I$

Figure 2: Effectiveness of our keyframe interpolation.

$$\begin{aligned}\text{CHip}(\boldsymbol{p}_0, \boldsymbol{m}_0, \boldsymbol{p}_1, \boldsymbol{m}_1; t) &= (2t^3 - 3t^2 + 1)\boldsymbol{p}_0 + (t^3 - 2t^2 + t)\boldsymbol{m}_0 \\ &\quad + (-2t^3 + 3t^2)\boldsymbol{p}_1 + (t^3 - t^2)\boldsymbol{m}_1, \qquad \textit{where} \quad t \in [0, 1].\end{aligned} \tag{6}$$

Based on CHip, we compute the position μ of $\mathcal{G}_d$ at time t as follows:

$$\begin{gathered}\mu(t) = \text{CHip}(\boldsymbol{p}_n, \boldsymbol{m}_n, \boldsymbol{p}_{n+1}, \boldsymbol{m}_{n+1}; t'), \\ \textit{where} \;\; n = \left\lfloor \frac{t}{I} \right\rfloor, \;\; t' = \frac{t - nI}{I}, \;\; \boldsymbol{m}_n = \frac{\boldsymbol{p}_{n+1} - \boldsymbol{p}_{n-1}}{2I}, \;\; \boldsymbol{m}_{n+1} = \frac{\boldsymbol{p}_{n+2} - \boldsymbol{p}_n}{2I},\end{gathered} \tag{7}$$

where $\boldsymbol{p}_n$ is a position of Gaussian at $n-$th keyframe. We use tangent values that is calculated using the position of two neighbor keyframes. This design can reduce additional requirements for storing tangent values at each keyframe, while still keeping the representational power for complex motion.

Other interpolators such as linear interpolation or piecewise cubic Hermite interpolating polynomial can be alternative choices. In this work, the cubic Hermit interpolator is selected because we can approximate the complex movements of points without any additional computational cost or storage.

4.2.2 Spherical Linear Interpolation for Temporal Rotation

Slerp is typically used for interpolating rotations [18] because linear interpolation causes a bias problem when it interpolates angular value. On the unit interval $[0, 1]$, given the unit vector $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ which represent rotations at $t = 0$ and $t = 1$ each, Slerp is defined as follows:

$$\text{Slerp}(\boldsymbol{x}_0, \boldsymbol{x}_1; t) = \frac{\sin[(1-t)\Omega]}{\sin\Omega}\boldsymbol{x}_0 + \frac{\sin[(t)\Omega]}{\sin\Omega}\boldsymbol{x}_1, \quad \textit{where} \quad t \in [0, 1] \;\text{ and }\; \cos\Omega = \boldsymbol{x}_0 \cdot \boldsymbol{x}_1. \tag{8}$$

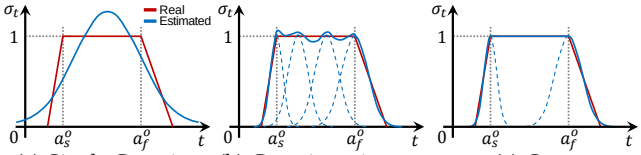
Slerp can be directly applied to quaternion rotations since it is independent of quaternions and dimensions. We thus have a rotation of intermediate frames in the quaternion of $\mathcal{G}_d$ at time t without any modification as follows:

$$q(t) = \text{Slerp}(\boldsymbol{r}_n, \boldsymbol{r}_{n+1}; t') \quad \textit{where} \quad n = \left\lfloor \frac{t}{I} \right\rfloor, \;\; t' = \frac{t - nI}{I}, \tag{9}$$

where $\boldsymbol{r}_n$ is the rotation of Gaussian at $n-$th keyframe.

4.2.3 Temporal Opacities

Modeling the temporal opacity is important because it is directly related to appearing/disappearing objects. A straightforward model for temporal opacity is to directly use a single Gaussian. However, there is a limitation to model diverse cases only using the single Gaussian, such as sudden emerging/slowly vanishing objects and objects disappearing in videos, as illustrated in Figure 3. We introduce the Gaussian mixture model to handle these situations. Since using too many Gaussians is impractical, we approximate the model with two Gaussians. We divide the temporal opacity into three cases: when an object appears, the object remains and the object disappears. One Gaussian handles the appearance of the object and the other manages disappearance. The interval between two Gaussians indicates the duration of the object when it is fully visible.



(a) Single Gaussian (b) Gaussian mixtures (c) **Ours**

Figure 3: Comparison between the single Gaussian, Gaussian mixture, and our model for temporal opacity modeling.

Let a Gaussian with a smaller mean value be g_s^o and the other is g_f^o where $a_s^o < a_f^o$. And, a_s^o, b_s^o, a_f^o and b_f^o are the mean and variance of g_s^o and the mean and variance of g_f^o each. The temporal opacity

σ_t at time t is defined as follows:

$$\sigma_t(t) = \begin{cases} e^{-\left(\frac{t-a_s^o}{b_s^o}\right)^2}, & for \quad t < a_s^o \\ 1, & for \quad a_s^o \leq t \leq a_f^o \\ e^{-\left(\frac{t-a_f^o}{b_f^o}\right)^2}, & for \quad t > a_f^o \ . \end{cases} \tag{10}$$

Using a single Gaussian may require multiple points to represent an object over a long duration. In contrast, our model can handle both the short and long temporal opacity of an object using two Gaussians.

4.3 Training Scheme

Progressive training scheme Our goal is to minimize both memory and computational costs of the entire pipeline, including preprocessing, not just reducing the representation of 3DGS model in a dynamic scene. To this end, we adopt a progressive training scheme that allows to learn over a long duration using only a small amount of point clouds obtained from the first frame, which is illustrated in Figure 4. To effectively handle objects moving or disappearing quickly, our model starts to learn a small part of an input video and gradually increases the video duration. The duration is incremented every specific step and made longer by the interval size.

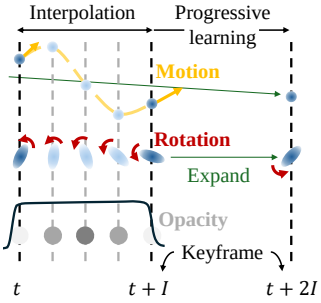


Figure 4: Progressive learning of dynamic Gaussians.

Expanding time duration As the time duration increases, the number of keyframes in the dynamic Gaussian obviously increases. We estimate the position and rotation of the Gaussian by linear regression using the last ρ frames when the number of keyframes increases so that the motion information of the previous frame can be shared with the next keyframe.

Extracting dynamic points from static points We want to separate dynamic points from static points without auxiliary information or supervision, such as masks or scribbles [70]. The separation is done based on the motion of the static points which is modeled to be movable, so we select the dynamic points based on the distance they moved. In particular, to avoid biased selection of distant points, we measure the motion in image space, normalizing the translation by the distance between points and the camera at the last timestamp. Therefore, if the distance to a point from the camera at the last timestamp is $\boldsymbol{\lambda}$, then the expression is $\frac{\|\mathbf{d}\|}{\|\boldsymbol{\lambda}\|^2}$. We sort points by the measured movement and convert the top-η percent of points (in this work, $\eta = 2$ is empirically set) into dynamic points. The position of the converted dynamic points is estimated at each keyframe using $\mathbf{x}$ and $\mathbf{d}$. The rotation is made to have the same value in all keyframes, and the opacity is initialized to be visible in all keyframes. We perform the extraction when we increase the duration or at specific iterations.

4.4 Optimization

Point backtracking for pruning Since it is difficult to filter out unnecessary dynamic points in a temporal context, we introduce a way to track errors in the image as points. Unlike contemporary works [82] that track points, we let our model track the value on a single backward pass. We use two measures, L1 distance and SSIM, whose formula is as follows:

$$\mathcal{E} = \frac{\sum_k \left(\sigma_i \times \prod_{j=1}^{i-1}(1-\sigma_j) \times q_k\right)}{\sum_k \left(\sigma_i \times \prod_{j=1}^{i-1}(1-\sigma_j)\right)}, \tag{11}$$

where q_k is the measured error in image space, k is a pixel index, and i and j are the order of Gaussian by depth which is visible at $k-\text{th}$ pixel. The accumulated error $\mathcal{E}_{total}$ is as follows:

$$\mathcal{E}_{total} = \frac{\sum_{v \in \mathcal{D}} \mathcal{E}_v}{\sum_{v \in \mathcal{D}} \mathbf{1}}, \tag{12}$$

where $\mathcal{D}$ is a set of training views. We prune the points over $\mathcal{E}_{total}$ at every pre-defined step.

Regularizations and losses We use regularization for large motions on both static and dynamic points. The regularization minimizes $\|\mathbf{d}\|$ for static points and $\|\boldsymbol{p}_{n+1} - \boldsymbol{p}_n\|$ for dynamic points. The optimization process follows 3DGS, which uses differentiable rasterization based on gradient backpropagation. Both L1 loss and SSIM loss, which measure the error between a rendered image and its ground truth, are used.

Table 1: Comparison of ours with the comparison methods on Neural 3D Video dataset [7]. Training time: Both preprocessing and the accumulated time of all subsequent training phases. Both the training time and FPS are measured under the same machine with an NVIDIA 4090 GPU for strictly fair comparisons. †: STG is done with an H100 GPU machine due to the memory issue. ‡: Trained using a dataset split into 150 frames.

Model	PSNR (dB)							MB	Frame/s	Hours
	Coffee Martini	Cook Spinach	Cut Roasted Beef	Flame Salmon	Flame Steak	Sear Steak	Average	Size	FPS	Training time
NeRFPlayer [72]	31.53	30.56	29.35	31.65	31.93	29.13	30.69	5130	0.05	6
HyperReel [86]	28.37	32.30	32.92	28.26	32.20	32.57	31.10	360	2	9
Neural Volumes [87]	N/A	N/A	N/A	22.80	N/A	N/A	22.80	N/A	N/A	N/A
LLFF [88]	N/A	N/A	N/A	23.24	N/A	N/A	23.24	N/A	N/A	N/A
DyNeRF [7]	N/A	N/A	N/A	29.58	N/A	N/A	29.58	28	0.015	1344
HexPlane [11]	N/A	32.04	32.55	29.47	32.08	32.39	31.71	200	N/A	12
K-Planes [12]	29.99	32.60	31.82	30.44	32.38	32.52	31.63	311	0.3	1.8
MixVoxels-L [89]	29.63	32.25	32.40	29.81	31.83	32.10	31.34	500	37.7	1.3
MixVoxels-X [89]	30.39	32.31	32.63	30.60	32.10	32.33	31.73	500	4.6	N/A
Im4D [90]	N/A	N/A	32.58	N/A	N/A	N/A	32.58	N/A	N/A	N/A
4K4D [19]	N/A	N/A	32.86	N/A	N/A	N/A	32.86	N/A	110	N/A
Dense COLMAP point cloud input										
STG‡ [15]	28.41	32.62	32.53	28.61	33.30	33.40	31.48	107	88.5	5.2†
4DGS [74]	28.33	32.93	33.85	29.38	34.03	33.51	32.01	6270	71.4	5.5
4DGaussians [14]	27.34	32.46	32.90	29.20	32.51	32.49	31.15	34	136.9	1.7
Sparse COLMAP point cloud input										
STG‡ [15]	27.71	31.83	31.43	28.06	32.17	32.67	30.64	109	101.0	1.3†
4DGS [74]	26.51	32.11	31.74	26.93	31.44	32.42	30.19	6057	72.0	4.2
4DGaussians [14]	26.69	31.89	25.88	27.54	28.07	31.73	28.63	34	146.6	1.5
3DGStream [91]	27.75	33.31	33.21	28.42	34.30	33.01	31.67	1200	-	-
Ours	28.79	33.23	33.73	29.29	33.91	33.69	32.11	115	120.6	0.6

4.5 Implementation Details

Our codebase is built upon 3DGS [13] and Mip-Splatting [83] and uses almost its hyperparameters. For initialization, our experiments use only COLMAP [84] point clouds from the first frame. The time interval and initial duration are both set to 10. We increment the duration by its interval every 400 iterations. Both static and dynamic regularization parameters are set to 0.0001. We employ the RAdam optimizer [85] for training.

5 Experiments

In this section, we conduct comprehensive experiments on two real-world datasets, Neural 3D Video [7] and Technicolor dataset [20] in Sections 5.1 and 5.2 each. We follow a conventional evaluation protocol in [86, 15], which uses subsequences divided from whole videos. We report PSNR, SSIM and LPIPS values. For SSIM, we use *scikit_image* library. Here, SSIM_1 and SSIM_2 are computed using *data_range* value of 1 and 2, respectively. We also measure frame-per-second (FPS) for rendering speed, and training time including preprocessing time. To compare the robustness of our method according to initial point clouds, we additionally test contemporary works on sparse point cloud initialization, which uses only the point cloud of the first frame in Sections 5.1 and 5.2. We also visualize the separation of static and dynamic points to show that our model can successfully distinguish them in Section 5.3. The ablation study shows the effectiveness of each component in our model in Section 5.4.

5.1 Neural 3D Video Dataset

Neural 3D Video dataset [7] provides six sets of multi-view indoor videos, captured with a range of 18 to 21 cameras with a 2704×2028 resolution and 300 frames. Following the conventional evaluation protocol, both training and evaluation procedures are performed at half the original resolution, and the center camera is held out as a novel view for evaluation. For a fair comparison, we train all models for all 300 frames including concurrent works, except for STG [15], NeRFPlayer [72] and HyperReel [86]. For NeRFPlayer and HyperReel, we directly borrow the results from [72, 86]. For STG, it is not possible to train for all 300 frames due to a GPU memory issue, so we report the results for only 150 frames, which is the maximum duration running on a single NVIDIA H100 80GB GPU.

(a) Ground Truth (b) 4DGS (c) STG† (d) 4DGaussians (e) **Ours**

Figure 5: Comparison of our Ex4DGS with other the state-of-the-art dynamic Gaussian splatting methods on Neural 3D Video [7] dataset.

(a) Ground Truth (b) Rendered static points (c) Rendered dynamic points (d) Rendered static&dynamic points

Figure 6: Visualization of our static and dynamic point separation on Coffee Martini, Flame Steak and Fabien scene in Neural 3D Video [7] and Technicolor [20] datasets.

As shown in Table 1, our model outperforms most of the contemporary models while maintaining the low computational cost. The example is displayed in Figure 5, which shows that our model produces high-quality rendering results over the comparison methods.

Comparison on sparse conditions We also carry out an experiment to check if concurrent methods work well with sparse point cloud initialization, which uses only it for the first frame. We report the result in Table 1. Interestingly, all the concurrent methods yield unsatisfactory results because motions in videos are learned by relying on the point clouds, not temporal changes of objects in the training phase. This implies that they require well-reconstructed 3D point clouds for proper initialization, while our method is free from the initial condition.

5.2 Technicolor Dataset

Technicolor light field dataset encompasses video recordings captured using a 4×4 camera array, wherein each camera is synchronized temporally, and the spatial resolution is 2048×1088. Adhering to the methodology introduced in HyperReel [86], we reserve the camera positioned at the intersection of the second row and second column for evaluation purposes. Evaluation is conducted on five distinct scenes (Birthday, Fabien, Painter, Theater, Trains) using their original full resolution. We retrain STG [15] using the COLMAP point cloud from the first frame, instead of the point cloud from every frame, for strictly fair comparison.

Table 2: Comparison results on the Technicolor dataset [20]. †: Trained with sparse point cloud input.

Model	PSNR	$SSIM_1$	$SSIM_2$	LPIPS
DyNeRF [7]	31.80	N/A	0.958	0.142
HyperReel [86]	32.73	0.906	N/A	0.109
STG† [15]	33.23	0.912	0.960	0.085
4DGS [74]	29.54	0.873	0.937	0.149
4DGaussians [14]	30.79	0.843	0.921	0.178
Ours	33.62	0.916	0.962	0.088

As shown in Table 2, Ex4DGS is comparable with the second-best model in the sparse COLMAP scenario. Although the Technicolor dataset contains various colorful objects, our model successfully synthesizes the novel view without dense prior or additional parameters. The reason why STG shows the impressive performance is that Technicolor dataset does not have rapid movements.

5.3 Separation of Dynamic and Static Points

Ex4DGS has a capability to separate static and dynamic points during the learning process. To check how well they are separated, we render them individually. Figure 6 shows the separation result. The static and dynamic points are rendered on both the Neural 3D Video and Technicolor datasets. The results demonstrate that the dynamic points are successfully separated from the static points, even if they are trained in an unsupervised manner. As a result, view-dependent color-changing or reflective objects are also identified as dynamic parts. Furthermore, in Coffee Martini scene, Ex4DGS demonstrates the ability to detect dynamic fluid in the transparent glasses. It is also worth highlighting that the same object can have static and dynamic components, as shown in the dog's legs and head being classified as distinct points in the Flame Steak scene.

5.4 Ablation Studies

We conduct an extensive ablation study to check the effectiveness of the proposed technical components in Table 3.

Table 3: Ablation studies of the proposed methods.

Method	PSNR	$SSIM_1$	LPIPS	Size(MB)
w/ Linear position	31.12	0.9385	0.0524	204
w/o Temporal opacity	31.42	0.9394	0.0521	186
w/ Linear rotation	31.26	0.9392	0.0525	148
w/o Progressive growing	31.02	0.9389	0.0550	168
w/ Linear position&rotation	31.32	0.9394	0.0521	172
w/o Regularization	31.37	0.9395	0.0522	174
w/o Dynamic point extraction	28.58	0.9280	0.0756	58
w/o Point backtracking	31.40	0.9394	0.0529	169
Ours	32.11	0.9422	0.0478	115

We first examine the effectiveness of our interpolation method by changing them into linear models. The results show that linear modeling of the position and rotation reduces the quality of rendering. Interestingly, using different types of interpolations further exacerbates the performance. If an equal level of polynomial bases are not assigned to both attributes, one of them falls short of the representational capacity, resulting in overfitting or over-smoothing.

We also show how our dynamic point extraction affects the rendering quality. As expected, complex motions can only be handled with dynamic point modeling. We then evaluate the efficacy of our temporal opacity modeling, and observe the performance degradation when no temporal opacity changes. Points can only disappear by minimizing the size and hiding back to other Gaussian points, making them act as flutters without being removed properly.

We then check the effectiveness of our progressive growing strategy. Without this strategy, the optimization gets stuck in local minima. This is due to our approach using only the point cloud from the first frame, which results in a misalignment with their corresponding objects of future frames.

We evaluate our regularization terms for the temporal dynamics of both static and dynamic points within a scene. As expected, incorporating the additional regularization term into the learning process makes the dynamic scene representations better. This benefit comes from the reduction of accumulated motion errors, preventing the points from moving excessively and locating them at correct positions.

Lastly, we examine the effectiveness of our point backtracking approach for pruning step. As expected, the correct removal of the misplaced points mitigates the errors and leads to the best result.

6 Conclusion

We have proposed a novel parameterization of dynamic 3DGS by explicitly modeling 3D Gaussians' motions. To achieve this, we initially set keyframes and predict their position and rotation changes. Primitive parameters of the 3D Gaussians between keyframes are then interpolated. Our strategy for learning dynamic motions enables us to decouple static and dynamic parts of scenes, opening up more intuitive and interpretable representation in 4D novel view synthesis.

Limitations Although we achieve a memory-efficient explicit representation of dynamic scenes, two challenges remain. First, our reconstruction can get stuck in local minima for newly appeared objects that are not initialized with 3D points and have no relevant 3D Gaussians in neighboring frames. This issue could be mitigated by initializing new 3D points with an additional geometric prior such as depth information. Second, as 3DGS suffers from scale ambiguity, training on monocular videos is challenging. This is because every 3D Gaussians are treated as dynamic due to the lack of accurate geometric clues for objects at each timestamp. This challenge can be addressed by incorporating an additional semantic cue information like object mask and optical flow, which account for objects' motions more explicitly.

References

[1] B Mildenhall, PP Srinivasan, M Tancik, JT Barron, R Ramamoorthi, and R Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 1, 3

[2] 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 (CVPR)*, 2021. 1, 3

[3] 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 Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 3

[4] 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)*, 2021. 1, 3

[5] 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 (ICCV)*, 2021. 1, 3

[6] 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 Transactions on Graphics (TOG)*, 2021. 1, 3

[7] 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 (CVPR)*, 2022. 1, 2, 3, 7, 8, 18, 19

[8] Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Oztireli. Dˆ 2nerf: Self-supervised decoupling of dynamic and static objects from a monocular video. In *Proceedings of the Neural Information Processing Systems (NeurIPS)*, 2022. 1, 3

[9] Jia-Wei Liu, Yan-Pei Cao, Weijia Mao, Wenqiao Zhang, David Junhao Zhang, Jussi Keppo, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Devrf: Fast deformable voxel radiance fields for dynamic scenes. In *Proceedings of the Neural Information Processing Systems (NeurIPS)*, 2022. 1

[10] Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Ping Tan. Streaming radiance fields for 3d video synthesis. In *Proceedings of the Neural Information Processing Systems (NeurIPS)*, 2022. 1

[11] 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 (CVPR)*, 2023. 1, 7, 19

[12] 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 (CVPR)*, 2023. 1, 7, 19

[13] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 2023. 1, 3, 7, 15

[14] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2, 3, 7, 8, 16, 17, 18, 19

[15] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2, 3, 7, 8, 16, 17, 18, 19

[16] 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)*, 2004. 2

[17] Richard H Bartels, John C Beatty, and Brian A Barsky. *An introduction to splines for use in computer graphics and geometric modeling*. Morgan Kaufmann, 1995. 2, 4

[18] Ken Shoemake. Animating rotation with quaternion curves. *Proceedings of the 12th annual conference on Computer graphics and interactive techniques.*, 1985. 2, 5

[19] Zhen Xu, Sida Peng, Haotong Lin, Guangzhao He, Jiaming Sun, Yujun Shen, Hujun Bao, and Xiaowei Zhou. 4k4d: Real-time 4d view synthesis at 4k resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2, 7, 19

[20] Neus Sabater, Guillaume Boisson, Benoit Vandame, Paul Kerbiriou, Frederic Babon, Matthieu Hog, Tristan Langlois, Remy Gendrot, Olivier Bureller, Arno Schubert, and Valerie Allie. Dataset and pipeline for multi-view light-field video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, 2017. 2, 7, 8

[21] M LEVOY. Light field rendering. In *Proceedings of ACM SIGGRAPH*, 1996. 2, 3

[22] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of ACM SIGGRAPH*, 2001. 2, 3

[23] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *Proceedings of ACM SIGGRAPH*, 2018. 2

[24] 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 (CVPR)*, 2019. 2

[25] Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[26] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. In *Proceedings of ACM SIGGRAPH*, 2016. 2

[27] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[28] Inchang Choi, Orazio Gallo, Alejandro Troccoli, Min H Kim, and Jan Kautz. Extreme view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 2

[29] Zexiang Xu, Sai Bi, Kalyan Sunkavalli, Sunil Hadap, Hao Su, and Ravi Ramamoorthi. Deep view synthesis from sparse photometric images. In *Proceedings of ACM SIGGRAPH*, 2019. 2

[30] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2, 3

[31] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 3

[32] Ronghang Hu, Nikhila Ravi, Alexander C Berg, and Deepak Pathak. Worldsheet: Wrapping the world in a 3d sheet for view synthesis from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 3

[33] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *Proceedings of the Neural Information Processing Systems (NeurIPS)*, 2019. 3

[34] Kyle Genova, Forrester Cole, Aaron Maschinot, Aaron Sarna, Daniel Vlasic, and William T Freeman. Unsupervised training for 3d morphable model regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3

[35] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 3

[36] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

[37] Mijeong Kim, Seonguk Seo, and Bohyung Han. Infonerf: Ray entropy minimization for few-shot neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[38] 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 (CVPR)*, 2023. 3

[39] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 3

[40] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[41] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[42] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 3

[43] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv:2010.07492*, 2020. 3

[44] Junoh Lee, Hyunjun Jung, Jin-Hwi Park, Inhwan Bae, and Hae-Gon Jeon. Geometry-aware projective mapping for unbounded neural radiance fields. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024. 3

[45] Peng Wang, Yuan Liu, Zhaoxi Chen, Lingjie Liu, Ziwei Liu, Taku Komura, Christian Theobalt, and Wenping Wang. F2-nerf: Fast neural radiance field training with free camera trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[46] Bangbang Yang, Chong Bao, Junyi Zeng, Hujun Bao, Yinda Zhang, Zhaopeng Cui, and Guofeng Zhang. Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 3

[47] Petr Kellnhofer, Lars C Jebe, Andrew Jones, Ryan Spicer, Kari Pulli, and Gordon Wetzstein. Neural lumigraph rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

[48] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *Proceedings of the Neural Information Processing Systems (NeurIPS)*, 2020. 3

[49] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (TOG)*, 2021. 3

[50] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

[51] Wenqi Yang, Guanying Chen, Chaofeng Chen, Zhenfang Chen, and Kwan-Yee K Wong. Ps-nerf: Neural inverse rendering for multi-view photometric stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 3

[52] Mark Boss, Andreas Engelhardt, Abhishek Kar, Yuanzhen Li, Deqing Sun, Jonathan Barron, Hendrik Lensch, and Varun Jampani. Samurai: Shape and material from unconstrained real-world arbitrary image collections. In *Proceedings of the Neural Information Processing Systems (NeurIPS)*, 2022. 3

[53] Weicai Ye, Shuo Chen, Chong Bao, Hujun Bao, Marc Pollefeys, Zhaopeng Cui, and Guofeng Zhang. Intrinsicnerf: Learning intrinsic neural radiance fields for editable novel view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 3

[54] Vincent Sitzmann, Semon Rezchikov, Bill Freeman, Josh Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *Proceedings of the Neural Information Processing Systems (NeurIPS)*, 2021. 3

[55] Huan Wang, Jian Ren, Zeng Huang, Kyle Olszewski, Menglei Chai, Yun Fu, and Sergey Tulyakov. R2l: Distilling neural radiance field to neural light field for efficient novel view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 3

[56] 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 (ICCV)*, 2021. 3

[57] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 3

[58] Edward H Adelson, James R Bergen, et al. *The plenoptic function and the elements of early vision*, volume 2. Vision and Modeling Group, Media Laboratory, Massachusetts Institute of Technology, 1991. 3

[59] 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 (CVPR)*, 2022. 3

[60] Hongyi Xu, Thiemo Alldieck, and Cristian Sminchisescu. H-nerf: Neural radiance fields for rendering and temporal reconstruction of humans in motion. In *Proceedings of the Neural Information Processing Systems (NeurIPS)*, 2021. 3

[61] Thiemo Alldieck, Hongyi Xu, and Cristian Sminchisescu. imghum: Implicit generative models of 3d human shape and articulated pose. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 3

[62] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM Transactions on Graphics (TOG)*, 2015. 3

[63] Tianjian Jiang, Xu Chen, Jie Song, and Otmar Hilliges. Instantavatar: Learning avatars from monocular video in 60 seconds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[64] ShahRukh Athar, Zexiang Xu, Kalyan Sunkavalli, Eli Shechtman, and Zhixin Shu. Rignerf: Fully controllable neural 3d portraits. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[65] Yunpeng Bai, Yanbo Fan, Xuan Wang, Yong Zhang, Jingxiang Sun, Chun Yuan, and Ying Shan. High-fidelity facial avatar reconstruction from monocular video with generative priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[66] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Animatable neural radiance fields for modeling dynamic human bodies. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 3

[67] 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)*, 2021. 3

[68] Jiakai Zhang, Xinhang Liu, Xinyi Ye, Fuqiang Zhao, Yanshun Zhang, Minye Wu, Yingliang Zhang, Lan Xu, and Jingyi Yu. Editable free-viewpoint video using a layered neural representation. *ACM Transactions on Graphics (TOG)*, 2021. 3

[69] Vadim Tschernezki, Diane Larlus, and Andrea Vedaldi. Neuraldiff: Segmenting 3d objects that move in egocentric videos. In *2021 International Conference on 3D Vision (3DV)*, 2021. 3

[70] 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. 3, 6

[71] Wei Jiang, Kwang Moo Yi, Golnoosh Samei, Oncel Tuzel, and Anurag Ranjan. Neuman: Neural human radiance field from a single video. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 3

[72] 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*, 2023. 3, 7, 19

[73] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *Proceedings of the International Conference on 3D Vision*, 2024. 3

[74] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023. 3, 7, 8, 16, 17, 18, 19

[75] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3

[76] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3

[77] Zhiyin Qian, Shaofei Wang, Marko Mihajlovic, Andreas Geiger, and Siyu Tang. 3dgs-avatar: Animatable avatars via deformable 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3

[78] Shoukang Hu and Ziwei Liu. Gauhuman: Articulated gaussian splatting from monocular human videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3

[79] Zhicheng Lu, Xiang Guo, Le Hui, Tianrui Chen, Min Yang, Xiao Tang, Feng Zhu, and Yuchao Dai. 3d geometry-aware deformable gaussian splatting for dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3

[80] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3

[81] 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 (CVPR)*, 2022. 4

[82] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 6

[83] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 7

[84] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 7

[85] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. 7

[86] 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 (CVPR)*, 2023. 7, 8, 18, 19

[87] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: learning dynamic renderable volumes from images. *ACM Transactions on Graphics (TOG)*, 2019. 7, 19

[88] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019. 7, 19

[89] 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 (ICCV)*, 2023. 7, 19

[90] 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 *Proceedings of SIGGRAPH Asia*, 2023. 7, 19

[91] Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao, and Wei Xing. 3dgstream: On-the-fly training of 3d gaussians for efficient streaming of photo-realistic free-viewpoint videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 7

A Overview

Within the appendix, we provide additional experiments in Appendix B, additional comparisons in Appendix C and per-scene breakdown of quantitative comparisons in Appendix D.

B Additional Experiments

In this section, we conduct experiments to further illustrate the behavior of Ex4DGS. In Appendix B.1, we present an experiment where changes in color alone are not treated as dynamic points. In Appendix B.2, we demonstrate how Ex4DGS behaves when objects reappear. In Appendix B.3 and Appendix B.4, we present additional ablation studies on various keyframe interval selection and dynamic point conversion rates.

B.1 Without Handling Color Components

Figure 7: Comparison between (b) handling color changes without dynamic points and (c) our complete model.

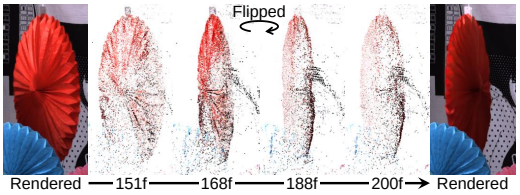


Figure 8: Visualization of the rotating decoration in the Technicolor Birthday scene.

We conduct experiments on the Coffee Martini scene from the Neural 3D Video dataset, focusing on scenarios where objects remain stationary but change color. We mask only the moving parts of the objects and train static Gaussians on the remaining regions using the original 3DGS model. The qualitative and quantitative results are presented in Figure 7 and Table 4. As shown in Figure 7, static points cannot handle changes such as shadows. However, Ex4DGS effectively manages these changes using dynamic points. In Table 4, "3DGS" denotes the unmodified 3DGS results. "3DGS + Our dynamic" denotes the results when the dynamic regions are replaced with Ex4DGS. Even when only color changes occur without any movement, significant performance loss occurs if these changes are not treated as dynamic points.

Table 4: Comparison results between without handling color changes and our complete model.

Model	PSNR	$SSIM_1$	LPIPS
3DGS [13]	21.69	0.851	0.126
3DGS + **Our dynamic**	26.07	0.891	0.089
Ours	28.79	0.912	0.070

B.2 Reappearing Objects

We visualize how Ex4DGS handles when objects disappear and reappear in Figure 8. Figure 8 illustrates the points associated with the decoration in the Birthday scene from the Technicolor dataset. It shows that the Gaussians corresponding to the part of the decoration that disappears at frame #168 and reappears at frame #188 have different distributions after the decoration flips that suggests that reappearing objects are regarded as new objects.

B.3 Keyframe Interval Selections

In Table 5, we present results on the effects of keyframe intervals and motion magnitude on Cook Spinach scene from Neural 3D Video dataset. To simulate different motion speeds, we deliberately skip frames in the videos. As shown in Table 5, a keyframe interval of 10 generally yields good results under most conditions. Smaller keyframe intervals tend to perform poorly, especially when the motion speed is low (i.e., fewer frames are skipped). However, as the skipped frame size increases smaller keyframe intervals begin to show better performance. It also shows that the model size decreases as the keyframe interval increases.

Table 5: Ablation studies of keyframe interval selections and skipped frames.

Skipped frames	1				2				4			
Keyframe interval	PSNR	$SSIM_1$	LPIPS	Size(MB)	PSNR	$SSIM_1$	LPIPS	Size(MB)	PSNR	$SSIM_1$	LPIPS	Size(MB)
1	31.17	0.948	0.057	595	31.47	0.948	0.056	415	31.81	0.946	0.051	142
2	32.06	0.952	0.051	314	32.33	0.954	0.049	322	31.81	0.953	0.044	101
5	31.70	0.953	0.047	206	32.29	0.954	0.043	126	32.53	0.954	0.045	80
10	33.04	0.956	0.041	119	32.65	0.956	0.043	93	31.79	0.953	0.046	74
20	32.78	0.955	0.043	90	32.07	0.952	0.047	78	32.08	0.953	0.048	73
50	32.14	0.955	0.046	79	31.93	0.951	0.052	72	30.91	0.949	0.056	73

B.4 Different Dynamic Point Conversion Rates

We experiment with different dynamic point conversion rates on Cook Spinach scene from Neural 3D Video dataset in Table 6. Our results indicate that the best performance is achieved when the extraction percentage is set to 2%. If the percentage is too low, not enough dynamic points will be extracted; conversely, if it is too high, too many dynamic points may be extracted, leading to overfitting and degraded performance.

Table 6: Ablation studies of dynamic point conversion rate.

Percentage	PSNR	$SSIM_1$	LPIPS	Size(MB)
0.5	32.36	0.955	0.043	103
1	32.48	0.956	0.044	115
2	33.04	0.956	0.041	119
4	32.89	0.955	0.045	227
8	31.33	0.954	0.048	367

C Additional Comparisons

To assess the robustness of Ex4DGS, we sample different frame intervals from the Technicolor dataset. First, we experiment with occlusion scenarios in Appendix C.1. Next, we present results when a new object appears in Appendix C.2. Finally, we train on an extremely long-duration video in Appendix C.3.

C.1 Handling Occlusion

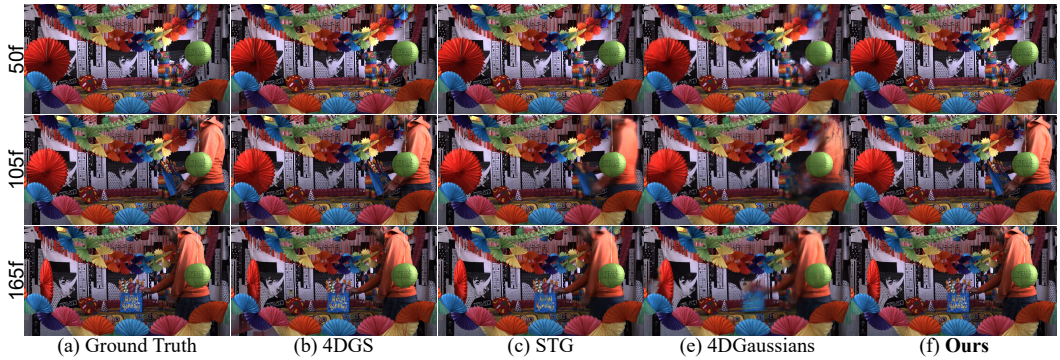
(a) Ground Truth (b) 4DGS (c) 4DGaussians (d) STG (e) Ours

Figure 9: Qualitative comparison of the repeatedly occluded objects in the Technicolor Train scene over a sequence of 100 frames (frame #170 to #269). All models are trained with the point cloud data from the frame #170.

We sample 100 frames (frames #170 to #269) from the Train scene in the Technicolor dataset containing occlusions of dynamic objects and compare Ex4DGS with other models. We use the point cloud prior of the first frame, which provides no information about the reappearing object after the occlusion. We compare the performance of STG, 4DGS and 4D Gaussians models in Table 7 and Figure 9. In these results, the explicit-based models, STG, 4DGS, and ours, perform significantly better. In the case of STG, the dynamic part is not well learned as the frames progress, and while 4DGS can render the dynamic part effectively, it struggles with the static part, negatively affecting the overall performance. In particular, 4D Gaussians, being an implicit model, fails to disentangle the static and dynamic components, resulting in missing renderings of the dynamic part. Our model, on the other hand, performs well and effectively learns both static and dynamic parts.

Table 7: Quantitative results of the repeatedly occluded objects in the Technicolor Train scene.

Model	PSNR	$SSIM_1$	LPIPS
STG [15]	32.17	0.940	0.035
4DGS [74]	29.11	0.877	0.119
4DGaussians [14]	23.31	0.657	0.385
Ours	32.24	0.941	0.044



(a) Ground Truth (b) 4DGS (c) STG (e) 4DGaussians (f) **Ours**

Figure 10: Qualitative comparison of the appearing objects in Technicolor Birthday scene over a sequence of 120 frames (frame #50 to #169). All models are trained with the point cloud data from the frame #50.

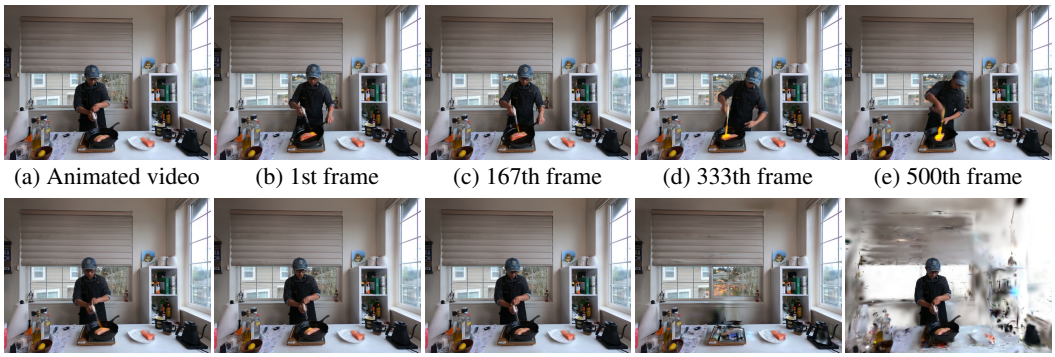
C.2 Handling Newly Appearing Objects

We conduct an experiment to determine whether Ex4DGS can learn about newly appearing objects that require the splitting of dynamic components. We sample 120 frames (frame #50 to #169) from the Birthday scene in the Technicolor dataset, during which a person appears. All models use a point cloud prior from a frame where the person is not yet visible. The numerical results are presented in Table 8, and the rendered images are shown in Figure 10. In contrast With the assertion made in the conclusion, the result is indeed feasible because Gaussians from neighboring objects can be utilized to facilitate the splitting process, even in the case of newly appearing objects. This is due to the effectiveness of the proposed splitting pipeline for static and dynamic Gaussians, which can handle newly appearing objects even when no initial Gaussian is provided.

Table 8: Quantitative results of the appearing objects in Technicolor Birthday scene.

Model	PSNR	$SSIM_1$	LPIPS
STG [15]	27.62	0.903	0.080
4DGS [74]	28.69	0.907	0.086
4DGaussians [14]	21.51	0.712	0.291
Ours	30.56	0.929	0.051

C.3 Extremely Long Duration



(a) Animated video (b) 1st frame (c) 167th frame (d) 333th frame (e) 500th frame

(f) 667th frame (g) 833th frame (h) 1000th frame (i) 1000th static (j) 1000th dynamic

Figure 11: Evaluation of the extremely long video on Flame Salmon scene in Neural 3D Video dataset. Best viewed at *Adobe Acrobat Reader*.

We conduct an experiment using a longer sequence of frames (1,000 frames, 20,000 images in total) on the Flame Salmon scene from the Neural 3D Video dataset. The results are presented in Table 9, and the rendered images are shown in Figure 11. The results of this experiment demonstrate that our model is capable of effective learning with reasonable storage requirements, even for extremely long videos. While

Table 9: Quantitative results of the extremely long video on Flame Salmon scene in Neural 3D Video dataset.

Model	PSNR	$SSIM_1$	LPIPS	Size(MB)
4DGS [74]	26.26	0.897	0.115	6331
4DGaussians [14]	28.37	0.903	0.097	75
Ours	28.77	0.919	0.076	392

the 4D Gaussian model produces acceptable results, its performance declines in areas where new objects, such as flames, appear. This indicates that the rendering quality may vary depending on the presence or absence of newly appearing objects, as discussed in Appendix C.1 and Appendix C.2.

D Detailed Results

In this section, we report the scene breakdown results of PSNR, $SSIM_1$, $SSIM_2$, and LPIPS on the Technicolor dataset and $SSIM_1$, $SSIM_2$ and LPIPS on the Neural 3D Video dataset.

Table 10: Per-scene quantitative comparison on Technicolor dataset. †: Trained with sparse point cloud input.

Model	PSNR					
	Birthday	Fabien	Painter	Theater	Train	Average
DyNeRF [7]	29.20	32.76	35.95	29.53	31.58	31.80
HyperReel [86]	29.99	34.70	35.91	33.32	29.74	32.73
STG† [15]	31.96	34.53	36.47	30.54	32.65	33.23
4DGS [74]	28.01	26.19	33.91	31.62	27.96	29.54
4D Gaussians [14]	30.87	33.56	34.36	29.81	25.35	30.79
Ours	32.38	35.38	36.73	31.84	31.77	33.62

Model	$SSIM_1$					
	Birthday	Fabien	Painter	Theater	Train	Average
HyperReel [86]	0.922	0.895	0.923	0.895	0.895	0.906
STG† [15]	0.942	0.877	0.923	0.872	0.945	0.912
4DGS [74]	0.902	0.856	0.897	0.869	0.843	0.873
4D Gaussians [14]	0.904	0.854	0.884	0.841	0.730	0.843
Ours	0.943	0.889	0.929	0.880	0.937	0.916

Model	$SSIM_2$					
	Birthday	Fabien	Painter	Theater	Train	Average
DyNeRF [7]	0.952	0.965	0.972	0.939	0.962	0.958
STG† [15]	0.969	0.955	0.970	0.939	0.967	0.960
4DGS [74]	0.944	0.943	0.957	0.940	0.901	0.937
4D Gaussians [14]	0.950	0.946	0.951	0.925	0.832	0.921
Ours	0.970	0.961	0.972	0.944	0.961	0.962

Model	LPIPS					
	Birthday	Fabien	Painter	Theater	Train	Average
DyNeRF [7]	0.067	0.242	0.146	0.188	0.067	0.142
HyperReel [86]	0.053	0.186	0.117	0.115	0.072	0.109
STG† [15]	0.039	0.134	0.097	0.121	0.033	0.085
4DGS [74]	0.089	0.197	0.136	0.155	0.166	0.149
4D Gaussians [14]	0.087	0.186	0.161	0.187	0.271	0.178
Ours	0.044	0.123	0.091	0.129	0.052	0.088

Table 11: Per-scene quantitative comparison on Neural 3D Video dataset. ‡: Trained using a dataset split into 150 frames.

Model	$SSIM_1$						
	Coffee Martini	Cook Spinach	Cut Roasted Beef	Flame Salmon	Flame Steak	Sear Steak	Average
NeRFPlayer [72]	0.951	0.929	0.908	0.940	0.950	0.908	0.931
HyperReel [86]	0.892	0.941	0.945	0.882	0.949	0.952	0.927
Dense COLMAP point cloud input							
STG‡ [15]	0.916	0.952	0.954	0.918	0.960	0.961	0.944
4DGS [74]	N/A	N/A	0.980	0.960	N/A	N/A	0.970
4DGaussians [14]	0.905	0.949	0.957	0.917	0.954	0.957	0.940
Sparse COLMAP point cloud input							
STG‡ [15]	0.904	0.946	0.946	0.913	0.954	0.955	0.936
4DGS [74]	0.902	0.948	0.947	0.904	0.954	0.955	0.935
4DGaussians [14]	0.893	0.944	0.913	0.896	0.946	0.946	0.923
Ours	0.915	0.947	0.948	0.917	0.956	0.959	0.940

Model	$SSIM_2$						
	Coffee Martini	Cook Spinach	Cut Roasted Beef	Flame Salmon	Flame Steak	Sear Steak	Average
Neural Volumes [87]	N/A	N/A	N/A	0.876	N/A	N/A	0.876
LLFF [88]	N/A	N/A	N/A	0.848	N/A	N/A	0.848
DyNeRF [7]	N/A	N/A	N/A	0.960	N/A	N/A	0.960
HexPlane [11]	N/A	0.970	0.974	0.960	0.978	0.978	0.972
K-Planes [12]	0.953	0.966	0.966	0.953	0.970	0.974	0.964
MixVoxels-L [89]	0.951	0.968	0.966	0.949	0.971	0.976	0.964
MixVoxels-X [89]	0.954	0.968	0.971	0.953	0.973	0.976	0.966
Im4D [90]	N/A	N/A	0.970	N/A	N/A	N/A	0.970
4K4D [19]	N/A	N/A	0.972	N/A	N/A	N/A	0.972
Dense COLMAP point cloud input							
STG‡ [15]	0.949	0.974	0.976	0.950	0.980	0.981	0.968
4DGS [74]	N/A	N/A	0.980	0.960	N/A	N/A	0.972
Sparse COLMAP point cloud input							
STG‡ [15]	0.942	0.970	0.971	0.948	0.976	0.977	0.964
4DGS [74]	0.939	0.971	0.970	0.941	0.975	0.976	0.962
4DGaussians [14]	0.934	0.969	0.944	0.937	0.970	0.969	0.954
Ours	0.951	0.976	0.977	0.956	0.980	0.979	0.970

Model	LPIPS						
	Coffee Martini	Cook Spinach	Cut Roasted Beef	Flame Salmon	Flame Steak	Sear Steak	Average
NeRFPlayer [72]	0.085	0.113	0.144	0.098	0.088	0.138	0.111
HyperReel [86]	0.127	0.089	0.084	0.136	0.078	0.077	0.096
Neural Volumes [87]	N/A	N/A	N/A	0.295	N/A	N/A	0.295
LLFF [88]	N/A	N/A	N/A	0.235	N/A	N/A	0.235
DyNeRF [7]	N/A	N/A	N/A	0.083	N/A	N/A	0.083
HexPlane [11]	N/A	0.082	0.080	0.078	0.066	0.070	0.075
MixVoxels-L [89]	0.106	0.099	0.088	0.116	0.088	0.080	0.096
MixVoxels-X [89]	0.081	0.062	0.057	0.078	0.051	0.053	0.064
Dense COLMAP point cloud input							
STG‡ [15]	0.069	0.043	0.042	0.063	0.034	0.033	0.047
4DGS [74]	N/A	N/A	0.041	N/A	N/A	N/A	0.055
Sparse COLMAP point cloud input							
STG‡ [15]	0.087	0.056	0.060	0.074	0.046	0.046	0.062
4DGS [74]	0.079	0.041	0.041	0.078	0.036	0.037	0.052
4DGaussians [14]	0.095	0.056	0.104	0.095	0.050	0.046	0.074
Ours	0.070	0.042	0.040	0.066	0.034	0.035	0.048

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract concisely outlines the problem definition and proposed methodology.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations inherent to the proposed method in the conclusion section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: Our paper does not address theoretical results, thus it is unrelated to this question.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide implementation details for reproduction of results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide the source code in the supplemental materials along with its instructions.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide all training details, including the balancing values of loss terms.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We simply report and compare the evaluations without statistical analysis.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the information on the computer resources in the Implementation Details section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We follow the code of ethics mentioned in NeurIPS. We only use public datasets and we respect their licenses.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This work has no potential positive or negative societal impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not release pretrained models as our work focuses on per-scene optimization of primitives. Additionally, we only use publicly accessible datasets.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use only publicly available datasets, mark their names, and cite the papers that presented them.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not introduce any new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our work does not involve crowdsourcing experiments nor human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our work does not involve crowdsourcing experiments nor human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.