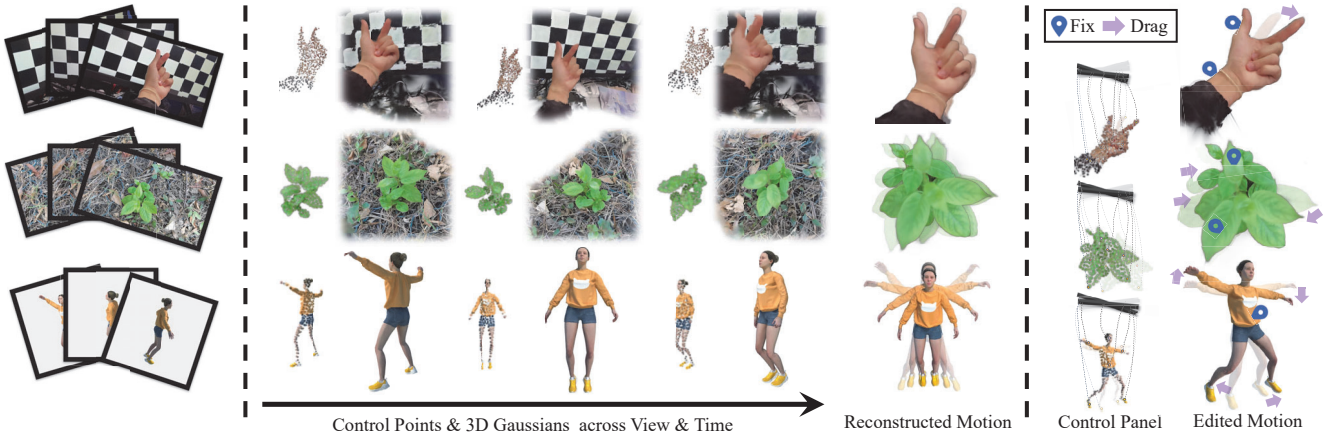
SC-GS: Sparse-Controlled Gaussian Splatting for Editable Dynamic Scenes

Yi-Hua Huang[1#*] **Yang-Tian Sun**[1#*] **Ziyi Yang**[3*] **Xiaoyang Lyu**[1] **Yan-Pei Cao**[2†] **Xiaojuan Qi**[1†]
[1] The University of Hong Kong [2] VAST [3] Zhejiang University



(a) Input Images **(b) Dynamic View Synthesis** **(c) Motion Editing**

Figure 1. Given **(a)** an image sequence from a monocular dynamic video, we propose to represent the motion with a set of sparse control points, which can be used to drive 3D Gaussians for high-fidelity rendering. Our approach enables both **(b)** dynamic view synthesis and **(c)** motion editing due to the motion representation based on sparse control points.

Abstract

Novel view synthesis for dynamic scenes is still a challenging problem in computer vision and graphics. Recently, Gaussian splatting has emerged as a robust technique to represent static scenes and enable high-quality and real-time novel view synthesis. Building upon this technique, we propose a new representation that explicitly decomposes the motion and appearance of dynamic scenes into sparse control points and dense Gaussians, respectively. Our key idea is to use sparse control points, significantly fewer in number than the Gaussians, to learn compact 6 DoF transformation bases, which can be locally interpolated through learned interpolation weights to yield the motion field of 3D Gaussians. We employ a deformation MLP to predict time-varying 6 DoF transformations for each control point, which reduces learning complexities, enhances learning abilities, and facilitates obtaining temporal and spatial coherent motion patterns. Then, we jointly learn the 3D Gaussians, the canonical space locations of control points, and the deformation MLP to reconstruct the appearance, geometry, and dynamics of 3D scenes. During learning, the location and number of control points are adaptively adjusted to accommodate varying motion complexities in different regions, and an ARAP loss following the principle of as rigid as possible is developed to enforce spatial continuity and local rigidity of learned motions. Finally, thanks to the explicit sparse motion representation and its decomposition from appearance, our method can enable user-controlled motion editing while retaining high-fidelity appearances. Extensive experiments demonstrate that our approach outperforms existing approaches on novel view synthesis with a high rendering speed and enables novel appearance-preserved motion editing applications.

1. Introduction

Novel view synthesis from a monocular video is a crucial problem with many applications in virtual reality, gaming, and the movie industry. However, extracting scene geometry and appearance from limited observations [30, 31, 49] is challenging. Concurrently, real-world scenes often contain dynamic objects, which pose additional challenges in representing object movements accurately to reflect real-world dynamics [18, 19, 33, 34, 37]. Recent advancements in this area are primarily driven by neural radiance fields (NeRF) [19, 30, 37, 66], which utilizes an implicit function to simultaneously learn scene geometry [26, 29] and

#This work is in collaboration with VAST.
*Equal Contribution †Corresponding Author

textures [12, 57] from multi-view images. Despite significant progress, NeRF-based representations still struggle with low rendering speeds and high memory usage. This issue is particularly evident when rendering at high resolutions [6, 31, 62], as they necessitate sampling hundreds of query points along each ray to predict color and opacity.

Most recently, Gaussian splatting [13] has shown remarkable performance in terms of rendering quality, resolution, and speed. Utilizing a point-based [2, 10, 14, 15, 46, 53, 67] scene representation, this method rasterizes 3D Gaussians to render images from specified views. It enables fast model training and real-time inference, achieving state-of-the-art (SOTA) visual quality. However, its existing formulation only applies to static scenes. It remains a challenge to incorporate object motion into the Gaussian representation without compromising rendering quality and speed. An intuitive method is to learn a flow vector for each 3D Gaussian. However, it will incur a significant time cost for training and inference. Moreover, it also leads to noisy trajectories and poor generalization in novel views, as demonstrated in Fig. 6 (a).

Motivated by the observation that real-world motions are often *sparse*, *spatially continuous*, and *locally rigid*, we propose to drive 3D Gaussians with learnable sparse control points (≈512) compared to the number of Gaussians (≈100K), in a much more compact space for modeling scene dynamics. These control points are associated with time-varying 6 DoF transformations parameterized as rotation using quaternion and translation parameters, which can be locally interpolated through learned interpolation weights to yield the motion field of dense Gaussians. These 6 DoF parameters on control points are predicted by an introduced MLP conditioned on time and location. Then, we jointly learn the canonical space 3D Gaussian parameters, locations, and radius of sparse control points at canonical space and the MLP for dynamic novel view synthesis. During learning, we introduce an adaptive strategy to adaptively change the number of sparse points to accommodate motion complexities in different regions and employ an ARAP loss that encourages the learned motions to be locally rigid.

Owing to the effective motion and appearance representations, our approach simultaneously enables high-quality dynamic view synthesis and motion editing, as shown in Fig. 1. We perform extensive experiments and ablation studies on benchmark datasets, demonstrating that our model surpasses existing methods both quantitatively and qualitatively while maintaining high rendering speeds. Furthermore, by learning a control graph from the scene motion, our control points-based motion representation allows for convenient motion editing, a feature not present in existing methods [1, 5, 11, 37, 38]. More results for motion editing are included in Fig. 5 and the supplementary material. Our contributions can be summarized as follows:

- We introduce sparse control points together with an MLP for modeling scene motion, based on the insight that motions within a scene can be represented by a compact subspace with a sparse set of bases.
- We employ adaptive learning strategies and design a regularization loss based on rigid constraints to enable effective learning of appearances, geometry, and motion from a monocular video.
- Thanks to the sparse motion representation, our approach enables motion editing by manipulating the learned control points while maintaining high-fidelity appearances.
- Extensive experiments show our approach achieves SOTA performance quantitatively and qualitatively.

2. Related Work

Dynamic NeRF. Novel view synthesis has been a prominent topic in the academic field for several years. NeRF [30] models static scenes implicitly with MLPs, and many works [11, 19, 33, 34, 37, 45, 52, 66] have expanded its usage to dynamic scenes via a deformation field. Some methods [7, 18, 35] represent dynamic scenes as 4D radiance fields but face extensive computational costs due to ray point sampling and volume rendering. Several acceleration approaches have been used for dynamic scene modeling. DeVRF [25] introduces a grid representation, and IBR-based methods [20, 22, 23, 55] use multi-camera information for quality and efficiency. Other methods used primitives [27], predicted MLP maps [36], or grid/plane-based structures [1, 5, 38, 40, 47, 48] for speed and performance in various dynamic scenes. However, hybrid models underperform with high-rank dynamic scenes due to their low-rank assumption.

Dynamic Gaussian Splatting. Gaussian Splatting [13, 51] offers improved rendering quality and speed for radiance fields. Several concurrent works have adapted 3D Gaussians for dynamic scenes. Luiten *et al*. [28] utilizes frame-by-frame training, suitable for multi-view scenes. Yang *et al*. [58] separate scenes into 3D Gaussians and a deformation field for monocular scenes but face slow training due to an extra MLP for learning Gaussian offsets. Following [58], Wu *et al*. [50] replaced the MLP with multi-resolution hexplanes [1] and a lightweight MLP. Yang *et al*. [59] include time as an additional feature in 4D Gaussians but face quality issues compared to constraints in canonical space. Our work proposes using sparse control points to drive the deformation of 3D Gaussians, which enhances rendering quality and reduces MLP query overhead. The learned control point graph can also be used for motion editing.

3D Deformation and Editing. Traditional deformation methods in computer graphics are typically based on Laplacian coordinates [8, 24, 41–43], Poisson equation [63] and cage-based approaches [61, 69]. These methods primarily

focus on preserving the geometric details of 3D objects during the deformation process. In recent years, there have been other approaches [21, 54, 64, 65, 70] that aim to edit the scene geometry learned from 2D images. These methods prioritize the rendering quality of the edited scene. Our approach falls into this category. However, instead of relying on the implicit and computationally expensive NeRF-based approach, our method employs an explicit point-based control graph deformation strategy and Gaussian rendering, which is more intuitive and efficient.

3. Preliminaries

Gaussian splatting represents a 3D scene using colored 3D Gaussians [13]. Each Gaussian G has a 3D center location μ and a 3D covariance matrix Σ,

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

The covariance matrix Σ is decomposed as $\Sigma = RSS^TR^T$ for optimization, with R as a rotation matrix represented by a quaternion $q \in \mathbf{SO}(3)$, and S as a scaling matrix represented by a 3D vector s. Each Gaussian has an opacity value σ to adjust its influence in rendering and is associated with sphere harmonic (SH) coefficients sh for view-dependent appearance. A scene is parameterized as a set of Gaussians $\mathcal{G} = \{G_j : \mu_j, q_j, s_j, \sigma_j, sh_j\}$.

Rendering an image involves projecting these Gaussians onto the 2D image plane and aggregating them using fast α-blending. The 2D covariance matrix and center are $\Sigma' = JW\Sigma W^TJ^T$ and $\mu' = JW\mu$. The color $C(u)$ of a pixel u is rendered using a neural point-based α-blending as,

$$C(u) = \sum_{i\in N} T_i\alpha_i\mathcal{SH}(sh_i, v_i), \text{ where } T_i = \Pi_{j=1}^{i-1}(1-\alpha_j). \quad (2)$$

Here, $\mathcal{SH}$ is the spherical harmonic function and v_i is the view direction. α_i is calculated by evaluating the corresponding projected Gaussian G_i at pixel u as,

$$\alpha_i = \sigma_i e^{-\frac{1}{2}(p-\mu_i')^T\Sigma_i'(p-\mu_i')}, \quad (3)$$

where μ_i' and Σ_i' are the center point and covariance matrix of Gaussian G_i, respectively. By optimizing the Gaussian parameters $\{G_j : \mu_j, q_j, s_j, \sigma_j, c_j\}$ and adjusting Gaussian density adaptively, high-quality images can be synthesized in real-time. We further introduce sparse control points to adapt Gaussian splatting for dynamic scenes while maintaining rendering quality and speed.

4. Method

Our goal is to reconstruct a dynamic scene from a monocular video. We represent the geometry and appearance of the dynamic scene using Gaussians in the canonical space while *modeling the motion through a set of control points together with time-varying 6DoF transformations predicted by an MLP*. These learned control points and corresponding transformations can be utilized to drive the deformation of Gaussians across different timesteps. The number of control points is significantly smaller than that of Gaussians, resulting in a set of *compact* motion bases for modeling scene dynamics and further facilitating *motion editing*. An overview of our method is shown in Fig. 2. In the following, we first present the sparse control points for representing compact motion bases in Sec. 4.1, followed by the dynamic scene rendering formulation in Sec. 4.2 and optimization process in Sec. 4.3.

4.1. Sparse Control Points

To derive a compact motion representation, we introduce a set of sparse control points $\mathcal{P} = \{(p_i \in \mathbb{R}^3, o_i \in \mathbb{R}^+)\}, i \in \{1, 2, \cdots, N_p\}$. Here, p_i denotes the learnable coordinate of control point i in the canonical space. o_i is the learnable radius parameter of a radial-basis-function (RBF) kernel that controls how the impact of a control point on a Gaussian will decrease as their distances increase. N_p is the total number of control points, which is considerably fewer than that of Gaussians.

For each control point k, we learn time-varying 6 DoF transformations $[R_i^t|T_i^t] \in \mathbf{SE}(3)$, consisting of a local frame rotation matrix $R_i^t \in \mathbf{SO}(3)$ and a translation vector $T_i^t \in \mathbb{R}^3$. Instead of directly optimizing the transformation parameters for each control point at different time steps, we employ an MLP Ψ to learn a time-varying transformation field and query the transformation of each control point p_k at each timestep t as:

$$\Psi : (p_i, t) \rightarrow (R_i^t, T_i^t). \quad (4)$$

Note that in practical implementations, R_i^t is represented equivalently as a quaternion r_i^t for more stable optimization and convenient interpolation for generating the motions of Gaussians in the follow-up steps.

4.2. Dynamic Scene Rendering

Equipped with the time-varying transformation parameters (R_i^t, T_i^t) for sparse control points which form a set of compact motion bases, the next step is to determine the transformation of each Gaussian at different time steps to derive the motion of the entire scene. We derive the dense motion field of Gaussians using linear blend skinning (LBS) [44] by locally interpolating the transformations of their neighboring control points. Specifically, for each Gaussian $G_j : (\mu_j, q_j, s_j, \sigma_j, sh_j)$, we use k-nearest neighbor (KNN) search to obtain its $K(=4)$ neighboring control points denoted as $\{p_k|k \in \mathcal{N}_j\}$ in canonical space. Then, the interpolation weights for control point p_k can be com-

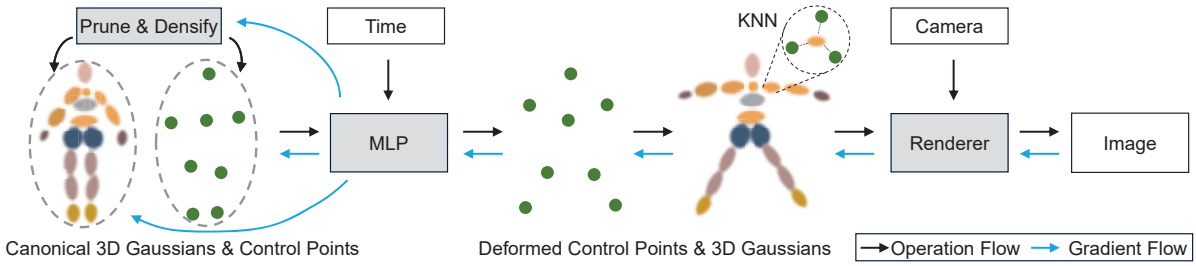


Figure 2. We present a novel method of employing sparse control points and a deformation MLP to direct 3D Gaussian dynamics. The MLP uses canonical control point coordinates and time to obtain per-control-point 6-DOF transformations, which guide 3D Gaussian deformation based on K nearest control points. Transformed Gaussians can then be rendered into images, and rendering loss calculated, before backpropagating gradients to optimize the Gaussians, control points, and MLP. Gaussian and control point density are adaptively managed during training.

puted with Gaussian-kernel RBF [4, 9, 32] as:

$$w_{jk} = \frac{\hat{w}_{jk}}{\sum_{k \in \mathcal{N}_j} \hat{w}_{jk}}, \text{ where } \hat{w}_{jk} = \exp(-\frac{d_{jk}^2}{2o_k^2}), \tag{5}$$

where d_{jk} is the distance between center of Gaussian G_j and the neighboring control point p_k, and o_k is the learned radius parameter of p_k. During training, these interpolation weights are adaptable to model complex motions by encouraging the learnable radius parameters to be optimized in a way that can accurately reconstruct the video frames.

Using the interpolation weights of neighboring control points, we can calculate a Gaussian motion field through interpolation. Following dynamic fusion works [4, 17, 32], we employ LBS [44] to compute the warped Gaussian μ_j^t and q_j^t as Eq. (6) and Eq. (7) for simplicity and efficiency:

$$\mu_j^t = \sum_{k \in \mathcal{N}_j} w_{jk} \left(R_k^t (\mu_j - p_k) + p_k + T_k^t \right), \tag{6}$$

$$q_j^t = (\sum_{k \in \mathcal{N}_j} w_{jk} r_k^t) \otimes q_j, \tag{7}$$

where $R_k^t \in \mathbb{R}^{3\times3}$ and $r_k^t \in \mathbb{R}^4$ are the matrix and quaternion representations of predicted rotation on control point k respectively. $\otimes$ is the production of quaternions, obtaining the quaternion of the composition of corresponding rotation transforms. Then, with the updated Gaussian parameters, we are able to perform rendering at time step t following Eq. (2) and Eq. (3).

4.3. Optimization

Our dynamic scene representation consists of control points $\mathcal{P}$ and Gaussians G in the canonical space and the deformation MLP Ψ. To stabilize the training process, we first pre-train $\mathcal{P}$ and Ψ to model the coarse scene motion with the Gaussians $\mathcal{G}$ fixed. The details are included in the supplementary material. Then, the whole model is optimized jointly. To facilitate learning, we introduce an ARAP loss to encourage the learned motion of control points to be locally rigid and employ an adaptive density adjustment strategy to adapt to varying motion complexities in different areas.

ARAP Loss and Overall Optimization Objective. To avoid local minima and regularize the unstructured control points, we introduce an ARAP loss $\mathcal{L}_{\text{arap}}$ that encourages their motions to be locally rigid, following the principle of being as rigid as possible [41]. Before computing the ARAP loss for control points, it is necessary to identify the edges that connect them. To avoid linking unrelated points, we opt to connect the points that have closely aligned trajectories in the scene motion. Specifically, for a control point p_j, we firstly calculate its trajectory p_i^{traj} that includes its locations across $N_t(=8)$ randomly sampled time steps as:

$$p_i^{\text{traj}} = \frac{1}{N_t} p_i^{t_1} \oplus p_i^{t_2} \oplus \cdots \oplus p_i^{t_{N_t}}, \tag{8}$$

where $\oplus$ denotes vector concatenation operation. Based on the trajectories obtained, we perform ball queries and use all control points $\mathcal{N}_{c_i}$ within a pre-defined radius to define a local area. Then, to calculate $\mathcal{L}_{\text{arap}}$, we randomly sample two time steps t_1 and t_2. For each point p_k within the radius (*i.e.* $k \in \mathcal{N}_{c_i}$), its transformed locations with learned translation parameters $T_k^{t_1}$ and $T_k^{t_2}$ are: $p_k^{t_1} = p_k + T_k^{t_1}$ and $p_k^{t_2} = p_k + T_k^{t_2}$, thus the rotation matrix $\hat{R}_i$ can be estimated following a rigid motion assumption [41] as:

$$\hat{R}_i = \underset{R \in \mathbf{SO}(3)}{\arg\min} \sum_{k \in \mathcal{N}_{c_i}} w_{ik} ||(p_i^{t_1} - p_k^{t_1}) - R(p_i^{t_2} - p_k^{t_2})||^2. \tag{9}$$

Here w_{ik} is calculated similarly to w_{jk} in Eq. (5) by replacing Gaussian position μ_j with control point position p_i, which weights the contribution of different neighboring points p_k according to their impact on p_i. Eq. (9) can be easily solved using SVD decomposition according to [41]. Then, $\mathcal{L}_{\text{arap}}$ is designed as,

$$\mathcal{L}_{\text{arap}}(p_i, t_1, t_2) = \sum_{k \in \mathcal{N}_{c_i}} w_{ik} ||(p_i^{t_1} - p_k^{t_1}) - \hat{R}_i(p_i^{t_2} - p_k^{t_2})||^2, \tag{10}$$

which evaluates the degree to which the learned motion deviates from the assumption of local rigidity. By penalizing $\mathcal{L}_{\text{arap}}$, the learned motions are encouraged to be locally rigid. The rigid regularization significantly enhances the learned motion with visualizations shown in Fig. 6.

For optimization, besides L_{arap}, the rendering loss $\mathcal{L}_{\text{render}}$ is derived by comparing the rendered image at different time steps with ground truth reference images using a combination of $\mathcal{L}_1$ loss and D-SSIM loss following [13]. Finally, the overall loss is constructed as: $\mathcal{L} = \mathcal{L}_{\text{render}} + \mathcal{L}_{\text{arap}}$.

Adaptive Control Points. Following [13], we also develop an adaptive density adjustment strategy to add and prune control points, which adjusts their distributions for modeling varying motion complexities, *e.g.* areas that exhibit complex motion patterns typically require control points of high densities. **1)** To determine whether a control point p_i should be pruned, we calculate its overall impact $W_i = \sum_{j \in \tilde{\mathcal{N}}_i} w_{ji}$ on the set of Gaussians $j \in \tilde{\mathcal{N}}_i$ whose K nearest neighbors include p_i. Then, we prune p_i if W_i is close to zero, indicating little contribution to the motion of 3D Gaussians. **2)** To determine whether a control point p_k should be cloned, we calculate the summation of Gaussian gradient norm with respect to Gaussians in set $\tilde{\mathcal{N}}_k$ as:

$$g_i = \sum_{j \in \tilde{\mathcal{N}}_i} \tilde{w}_j || \frac{d\mathcal{L}}{d\mu_j} ||_2^2, \text{ where } \tilde{w}_j = \frac{w_{ji}}{\sum_{j \in \tilde{\mathcal{N}}_k} w_{ji}}. \tag{11}$$

A large g_k indicates poor reconstructions. Therefore, we clone p_k and add a new control point p'_k to the expected position of related Gaussians to improve the reconstruction:

$$p'_k = \sum_{j \in \tilde{\mathcal{N}}_k} \tilde{w}_i \mu_j; \ \sigma'_k = \sigma_k. \tag{12}$$

5. Motion Editing

Since our approach utilizes an explicit and sparse motion representation, it further allows for efficient and intuitive motion editing through the manipulation of control points. It is achieved by predicting the trajectory of each control point across different time steps, determining their neighborhoods, constructing a rigid control graph, and performing motion editing by graph deformation.

Control Point Graph. With the trained control points $\mathcal{P}$ and the MLP Ψ, we construct a control point graph $\mathcal{G}$ that connects control points based on their trajectories. For each vertex of the graph, i.e., control point p_i, we firstly calculate its trajectory p_i^{traj} derived from Eq. (8). Then, the vertex is connected with other vertices that fall within a ball of a pre-determined radius parameter based on p_i^{traj}. The edge weights w_{ij} between two connected vertices p_i and p_j are calculated using Eq. (5). Building the control graph based on point trajectory helps take into account the overall motion sequence instead of a single timestep, which avoids unreasonable edge connections. We demonstrate the advantage of this choice in the supplementary material.

Motion Editing. In order to maintain the local rigidity, we perform ARAP [41] deformation on the control graph based on constraints specified by users. Mathematically, given a set of user-defined handle points $\{h_l \in \mathbb{R}^3 \ | l \in \mathcal{H} \subset \{1, 2, \cdots, N_p\}\}$, the control graph $\mathcal{P}'$ can be deformed by minimizing the APAR energy formulated as:

$$E(\mathcal{P}') = \sum_{i=1}^{N_p} \sum_{j \in \mathcal{N}_i} w_{ij} ||(p'_i - p'_j) - \hat{R}_i (p_i - p_j)||^2, \tag{13}$$

with the fixed position condition $p'_l = h_l$ for $l \in \mathcal{H}$. Here $\hat{R}_i$ is the rigid local rotation defined on each control point. This optimization problem can be efficiently solved by alternately optimizing local rotations $\hat{R}_i$ and deformed control point positions p'. We refer the readers to [41] for the specific optimization process. The solved rotation $\hat{R}_i$ and translation $\hat{T}_i = p'_i - p_i$ form a 6 DoF transformation for each control point, which is consistent with our motion representation. Thus, Gaussians can be warped by the deformed control points by simply replacing the transformation in Eq. (6) and Eq. (7), which can be rendered into high-quality edited images even for motion out of the training sequence. We visualize the motion editing results in Fig. 5.

6. Experiment

6.1. Datasets and Evaluation Metrics

To validate the superiority of our method, we conducted extensive experiments on D-NeRF [37] datasets and NeRF-DS [56] datasets. D-NeRF datasets contain eight dynamic scenes with 360° viewpoint settings, and the NeRF-DS datasets consist of seven captured videos with camera pose estimated using colmap [39]. The two datasets involve a variety of rigid and non-rigid deformation of various objects. The metrics we use to evaluate the performance are Peak Signal-to-Noise Ratio (PSNR), Structural Similarity(SSIM), Multiscale SSIM(MS-SSIM), and Learned Perceptual Image Patch Similarity (LPIPS) [68].

6.2. Quantitative Comparisons

D-NeRF Datasets. We compare our method against existing state-of-the-art methods: D-NeRF [37], TiNeuVox [5], Tensor4D [40], K-Planes [38], and FF-NVS [11] using the official implementations and follow the same data setting. Concurrent work 4D-GS [50] is also compared since the official code has been released. We also evaluate the baseline that directly applies estimated per-Gaussian transformation with a deformation MLP to demonstrate the effectiveness of control points. The comparisons are carried out on the resolution of 400x400, following the same approach as in previous methods [1, 5, 37]. We demonstrate the comparison results in Tab. 1. Our approach significantly outperforms others. The baseline method also achieves high synthesis quality thanks to the superiority of 3D Gaussians. However, without the regularization of compact motion bases,

Table 1. **Quantitative comparison on D-NeRF [37] datasets.** We present the average PSNR/SSIM/LPIPS (VGG) values for novel view synthesis on dynamic scenes from D-NeRF, with each cell colored to indicate the best, second best, and third best.

Methods	Hook			Jumpingjacks			Trex			BouncingBalls		
	PSNR(↑)	SSIM(↑)	LPIPS(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)
D-NeRF [37]	29.25	.968	.1120	32.80	.981	.0381	31.75	.974	.0367	38.93	.987	.1074
TiNeuVox-B [5]	31.45	.971	.0569	34.23	.986	.0383	32.70	.987	.0340	40.73	.991	.0472
Tensor4D [40]	29.03	.955	.0499	24.01	.919	.0768	23.51	.934	.0640	25.36	.961	.0411
K-Planes [38]	28.59	.953	.0581	32.27	.971	.0389	31.41	.980	.0234	40.61	.991	.0297
FF-NVS [11]	32.29	.980	.0400	33.55	.980	.0300	30.71	.960	.0400	40.02	.990	.0400
4D-GS [50]	30.99	.990	.0248	33.59	.990	.0242	32.16	.988	.0216	38.59	.993	.0267
Baseline	34.47	.990	.0195	35.74	.992	.0178	36.37	.994	.0103	41.45	.996	.0190
Ours	39.87	.997	.0076	41.13	.998	.0067	41.24	.998	.0046	44.91	.998	.0166

Methods	Hellwarrior			Mutant			Standup			*Average*		
	PSNR(↑)	SSIM(↑)	LPIPS(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)
D-NeRF [37]	25.02	.955	.0633	31.29	.978	.0212	32.79	.983	.0241	31.69	.975	.0575
TiNeuVox-B [5]	28.17	.978	.0706	33.61	.982	.0388	35.43	.991	.0230	33.76	.983	.0441
Tensor4D [40]	31.40	.925	.0675	29.99	.951	.0422	30.86	.964	.0214	27.62	.947	.0471
K-Planes [38]	25.27	.948	.0775	33.79	.982	.0207	34.31	.984	.0194	32.32	.973	.0382
FF-NVS [11]	27.71	.970	.0500	34.97	.980	.0300	36.91	.990	.0200	33.73	.979	.0357
4D-GS [50]	31.39	.974	.0436	35.98	.996	.0120	35.37	.994	.0136	34.01	.987	.0316
Baseline	39.07	.982	.0350	41.45	.998	.0045	41.04	.996	.0071	38.51	.992	.0162
Ours	42.93	.994	.0155	45.19	.999	.0028	47.89	.999	.0023	43.31	.997	.0063

Table 2. **Quantitative comparison on NeRF-DS [56] datasets.** We display the average PSNR/MS-SSIM/LPIPS (Alex) metrics for novel view synthesis on dynamic scenes from NeRF-DS, with each cell colored to indicate the best, second best, and third best.

Methods	Bell			Sheet			Press			Basin		
	PSNR(↑)	MS-SSIM(↑)	LPIPS(↓)	PSNR(↑)	MS-SSIM(↑)	LPIPS(↓)	PSNR(↑)	MS-SSIM(↑)	LPIPS(↓)	PSNR(↑)	MS-SSIM(↑)	LPIPS(↓)
HyperNeRF [34]	24.0	.884	.159	24.3	.874	.148	25.4	.873	.164	20.2	.829	.168
NeRF-DS [56]	23.3	.872	.134	25.7	.918	.115	26.4	.911	.123	20.3	.868	.127
TiNeuVox-B [5]	23.1	.876	.113	21.1	.745	.234	24.1	.892	.133	20.7	.896	.105
Baseline	24.9	.917	.124	26.1	.903	.127	25.1	.884	.221	19.6	.852	.144
Ours	25.1	.918	.117	26.2	.898	.142	26.6	.901	.135	19.6	.846	.154

Methods	Cup			Sieve			Plate			*Average*		
	PSNR(↑)	MS-SSIM(↑)	LPIPS(↓)	PSNR(↑)	MS-SSIM(↑)	LPIPS(↓)	PSNR(↑)	MS-SSIM(↑)	LPIPS(↓)	PSNR(↑)	MS-SSIM(↑)	LPIPS(↓)
HyperNeRF [34]	20.5	.705	.318	25.0	.909	.129	18.1	.714	.359	22.5	.827	.206
NeRF-DS [56]	24.5	.916	.118	26.1	.935	.108	20.8	.867	.164	23.9	.898	.127
TiNeuVox-B [5]	20.5	.806	.182	20.1	.822	.205	20.6	.863	.161	21.5	.843	.162
Baseline	24.7	.919	.116	25.3	.917	.109	20.3	.842	.214	23.7	.891	.151
Ours	24.5	.916	.115	26.0	.919	.114	20.2	.837	.202	24.1	.891	.140

the baseline has difficulty in achieving global optima. We also report the rendering speed comparison in the supplementary material to show the efficiency of our method.

NeRF-DS Datasets. Although the datasets provide relatively accurate camera poses compared with [34], some inevitable estimation errors still exist. This resulted in a downgraded performance of our method. However, our approach still achieves the best visual quality compared with SOTA methods, as reported in Tab. 2. It's worth mentioning that NeRF-DS outperforms both our method and the baseline on certain datasets, as it employs a specialized design for modeling the specular parts of dynamic objects. Despite this, our approach, which doesn't employ any additional processes, still achieves a higher average performance.

6.3. Qualitative Comparison

We also conduct qualitative comparisons to illustrate the advantages of our method over SOTA methods. The comparisons on D-NeRF datasets are shown in Fig. 3, where zoom-in images show the details of synthesized images. Our approach produces results closest to the ground truths and has the best visual quality. Note that, in the Lego scene, the

Figure 3. Qualitative comparison of dynamic view synthesis on D-NeRF [37] datasets. We compare our method with state-of-the-art methods including D-NeRF [37], TiNeuVox-B [5], K-Planes [38], and 4D-GS [59]. Our method delivers a higher visual quality and preserves more details of dynamic scenes. Notably, in the Lego scene (bottom row), the train motion is inconsistent with the test motion.

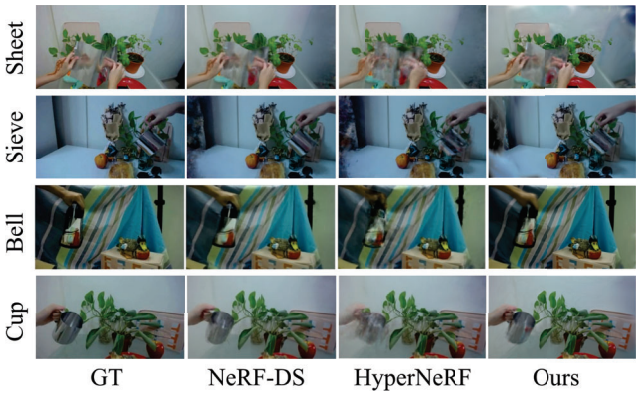


Figure 4. Qualitative comparisons of dynamic view synthesis on scenes from NeRF-DS [56]. Our method produces high-fidelity results even without specialized design for specular surfaces.

motion in the test set does not align with that in the training set, as indicated in the bottom row of the figure. The same observation can also be seen in [58]. The qualitative comparisons conducted on the NeRF-DS dataset are also demonstrated in Fig. 4. It is clear that our method is capable of producing high-fidelity novel views, even in the absence of a specialized design for specular surfaces.

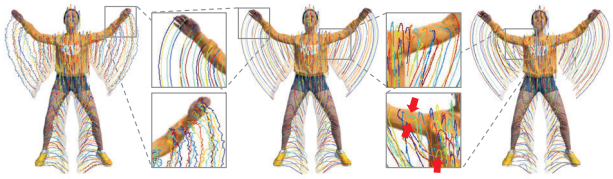
Table 3. We quantitatively evaluate the effect of control points and ARAP loss on D-NeRF [37] datasets.

Methods	PSNR(↑)	SSIM(↑)	LPIPS(↓)
w/o Control Points	38.512	0.9922	0.0162
w/o ARAP loss	42.617	0.9963	0.0067
Full	**43.307**	**0.9976**	**0.0063**

6.4. Ablation study

Control Points. Our motion representation driven by control points constructs a compact and sparse motion space, effectively mitigating overfitting on the training set. We quantitatively compare the novel view synthesis quality of

Figure 5. We visualize the reconstructed motion sequence from the dynamic scene (top) and edited motion sequence (bottom). Our approach generalizes well for motion out of the training set benefitting from the locally rigid motion space modeled by control points.



(a) w/o Control Points (b) Full (c) w/o ARAP Loss

Figure 6. We visualize the rendering results and Gaussian trajectories of (a) the baseline method without control points, (b) our full method, and (c) our method without ARAP loss.

our method with the baseline that does not utilize control points on both D-NeRF [37] and NeRF-DS [56] datasets, as presented in Tab. 1 and Tab. 2. To intuitively elucidate the effects of control points, we compare the results and visualize the trajectories of Gaussians driven either with or without control points in Fig. 6 (a) and (b). Clearly, directly predicting the motion of each Gaussian with an MLP leads to noise in Gaussian trajectories. While the baseline theoretically is more flexible in representing diverse motions, it tends to falter and descend into local minima during optimization, hindering it from achieving the global optimum.

ARAP Loss. Despite the control-point-driven motion representation providing effective regularization to Gaussian motions, there can be occasional breaches in rigidity. As evidenced in Fig. 6 (c), even though Gaussians achieve relatively smooth trajectories, some Gaussians on the arm move towards the girl's torso instead of moving alongside the ascending arm. This issue arises due to the lack of constraints on the inter-relation of control points' motions. By imposing ARAP loss on control points, such phenomena are eliminated, thus facilitating a robust motion reconstruction. Tab. 3 illustrates without ARAP loss, the performance of dynamic view synthesis on D-NeRF [37] slightly decreases.

6.5. Motion Editing

Our method facilitates scene motion editing via the manipulation of control nodes, due to the explicit motion representation using control points. The learned correlation and weights between Gaussians and control points enable excellent generalization, even on motion beyond the training sequence. The reconstructed and edited motion sequences are demonstrated in Fig. 5.

7. Conclusion and Future Works

We present a method driving 3D Gaussians using control points and a deformation MLP, learnable from dynamic scenes. Our approach, combining a compact motion representation with adaptive learning strategies and rigid constraints, allows high-quality dynamic scene reconstruction and motion editing. Experiments showed our method outperforms existing approaches in the visual quality of synthesized dynamic novel views. However, limitations exist. The performance is prone to inaccurate camera poses, leading to reconstruction failures on datasets with inaccurate poses such as HyperNeRF [34]. The current approach also faces limitations in handling common specular effects, resulting in limited improvement on NeRF-DS [56] datasets. To address this, future work could focus on extending the method by incorporating Spec-Gaussian [60] with a specialized specular design. This enhancement would enable more accurate modeling of highlight and mirror effects. Furthermore, the presence of blurriness in videos with dynamic objects should be considered. To enhance the robustness of the proposed method, incorporating deblurring techniques [3, 16] for novel view synthesis can address this issue effectively.

Acknowledgement

This work has been supported by Hong Kong Research Grant Council - Early Career Scheme (Grant No. 27209621), General Research Fund Scheme (Grant No. 17202422), and RGC Matching Fund Scheme (RMGS). Part of the described research work is conducted in the JC STEM Lab of Robotics for Soft Materials funded by The Hong Kong Jockey Club Charities Trust.

References

[1] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *CVPR*, 2023. 2, 5

[2] Peng Dai, Yinda Zhang, Zhuwen Li, Shuaicheng Liu, and Bing Zeng. Neural point cloud rendering via multi-plane projection. In *CVPR*, 2020. 2

[3] Peng Dai, Yinda Zhang, Xin Yu, Xiaoyang Lyu, and Xiaojuan Qi. Hybrid neural rendering for large-scale scenes with motion blur. In *CVPR*, 2023. 8

[4] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, et al. Fusion4d: Real-time performance capture of challenging scenes. *ACM TOG*, 35(4):1–13, 2016. 4

[5] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *ACM SIGGRAPH ASIA*, 2022. 2, 5, 6, 7

[6] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 2

[7] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *ICCV*, 2021. 2

[8] Lin Gao, Yu-Kun Lai, Jie Yang, Ling-Xiao Zhang, Shihong Xia, and Leif Kobbelt. Sparse data driven mesh deformation. *IEEE TVCG*, 27(3):2085–2100, 2019. 2

[9] Wei Gao and Russ Tedrake. Surfelwarp: Efficient non-volumetric single view dynamic reconstruction. *Robotics: Science and Systems XIV*, 2018. 4

[10] Yiming Gao, Yan-Pei Cao, and Ying Shan. Surfelnerf: Neural surfel radiance fields for online photorealistic reconstruction of indoor scenes. In *CVPR*, 2023. 2

[11] Xiang Guo, Jiadai Sun, Yuchao Dai, Guanying Chen, Xiaoqing Ye, Xiao Tan, Errui Ding, Yumeng Zhang, and Jingdong Wang. Forward flow for novel view synthesis of dynamic scenes. In *ICCV*, 2023. 2, 5, 6

[12] Yi-Hua Huang, Yan-Pei Cao, Yu-Kun Lai, Ying Shan, and Lin Gao. Nerf-texture: Texture synthesis with neural radiance fields. In *ACM SIGGRAPH*, pages 1–10, 2023. 2

[13] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 42(4):1–14, 2023. 2, 3, 5

[14] Leonid Keselman and Martial Hebert. Approximate differentiable rendering with algebraic surfaces. In *ECCV*, 2022. 2

[15] Leonid Keselman and Martial Hebert. Flexible techniques for differentiable rendering with 3d gaussians. *arXiv preprint arXiv:2308.14737*, 2023. 2

[16] Byeonghyeon Lee, Howoong Lee, Xiangyu Sun, Usman Ali, and Eunbyung Park. Deblurring 3d gaussian splatting. *arXiv preprint arXiv:2401.00834*, 2024. 8

[17] Hao Li, Bart Adams, Leonidas J Guibas, and Mark Pauly. Robust single-view geometry and motion reconstruction. *ACM TOG*, 28(5):1–10, 2009. 4

[18] 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 *CVPR*, 2022. 1, 2

[19] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021. 1, 2

[20] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. In *CVPR*, 2023. 2

[21] Gao Lin, Liu Feng-Lin, Chen Shu-Yu, Jiang Kaiwen, Li Chunpeng, Yukun Lai, and Fu Hongbo. Sketchfacenerf: Sketch-based facial generation and editing in neural radiance fields. *ACM TOG*, 2023. 3

[22] Haotong Lin, Sida Peng, Zhen Xu, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Efficient neural radiance fields for interactive free-viewpoint video. In *ACM SIGGRAPH ASIA*, pages 1–9, 2022. 2

[23] 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 *ACM SIGGRAPH ASIA*, pages 1–9, 2023. 2

[24] Yaron Lipman, Olga Sorkine-Hornung, Marc Alexa, Daniel Cohen-Or, David Levin, Christian Rössl, and Hans-Peter Seidel. Laplacian framework for interactive mesh editing. *Int. J. Shape Model.*, 11:43–62, 2005. 2

[25] 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 *NeurIPS*, 2022. 2

[26] Yu-Tao Liu, Li Wang, Jie Yang, Weikai Chen, Xiaoxu Meng, Bo Yang, and Lin Gao. Neudf: Leaning neural unsigned distance fields with volume rendering. In *CVPR*, 2023. 1

[27] Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. Mixture of volumetric primitives for efficient neural rendering. *ACM TOG*, 40(4):1–13, 2021. 2

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

[29] Xiaoyang Lyu, Peng Dai, Zizhang Li, Dongyu Yan, Yi Lin, Yifan Peng, and Xiaojuan Qi. Learning a room with the occ-sdf hybrid: Signed distance function mingled with occupancy aids scene representation. In *ICCV*, 2023. 1

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

[31] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 41(4):1–15, 2022. 1, 2

[32] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *CVPR*, 2015. 4

[33] 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 *ICCV*, 2021. 1, 2

[34] 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. In *ACM TOG*, 2021. 1, 2, 6, 8

[35] Sungheon Park, Minjung Son, Seokhwan Jang, Young Chun Ahn, Ji-Yeon Kim, and Nahyup Kang. Temporal interpolation is all you need for dynamic neural radiance fields. In *CVPR*, 2023. 2

[36] Sida Peng, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Representing volumetric videos as dynamic mlp maps. In *CVPR*, 2023. 2

[37] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, 2021. 1, 2, 5, 6, 7, 8

[38] Sara Fridovich-Keil and Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *CVPR*, 2023. 2, 5, 6, 7

[39] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 5

[40] Ruizhi Shao, Zerong Zheng, Hanzhang Tu, Boning Liu, Hongwen Zhang, and Yebin Liu. Tensor4d: Efficient neural 4d decomposition for high-fidelity dynamic reconstruction and rendering. In *CVPR*, 2023. 2, 5, 6

[41] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry Processing*, pages 109–116. Citeseer, 2007. 2, 4, 5

[42] Olga Sorkine-Hornung. Laplacian mesh processing. In *Eurographics*, 2005.

[43] Olga Sorkine-Hornung, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian surface editing. In *Eurographics Symposium on Geometry Processing*, 2004. 2

[44] Robert W Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. In *ACM SIGGRAPH*, pages 80–es. 2007. 3, 4

[45] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *ICCV*, 2021. 2

[46] Cong Wang, Di Kang, Yan-Pei Cao, Linchao Bao, Ying Shan, and Song-Hai Zhang. Neural point-based volumetric avatar: Surface-guided neural points for efficient and photo-realistic volumetric head avatar. In *ACM SIGGRAPH ASIA*, pages 1–12, 2023. 2

[47] Feng Wang, Sinan Tan, Xinghang Li, Zeyue Tian, Yafei Song, and Huaping Liu. Mixed neural voxels for fast multi-view video synthesis. In *ICCV*, 2023. 2

[48] Liao Wang, Qiang Hu, Qihan He, Ziyu Wang, Jingyi Yu, Tinne Tuytelaars, Lan Xu, and Minye Wu. Neural residual radiance fields for streamably free-viewpoint videos. In *CVPR*, 2023. 2

[49] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 34, 2021. 1

[50] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Wang Xinggang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023. 2, 5, 6

[51] Tong Wu, Yu-Jie Yuan, Ling-Xiao Zhang, Jie Yang, Yan-Pei Cao, Ling-Qi Yan, and Lin Gao. Recent advances in 3d gaussian splatting. *arXiv preprint arXiv:2403.11134*, 2024. 2

[52] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *CVPR*, 2021. 2

[53] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *CVPR*, 2022. 2

[54] Tianhan Xu and Tatsuya Harada. Deforming radiance fields with cages. In *ECCV*, 2022. 3

[55] 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. *arXiv preprint arXiv:2310.11448*, 2023. 2

[56] Zhiwen Yan, Chen Li, and Gim Hee Lee. Nerf-ds: Neural radiance fields for dynamic specular objects. In *CVPR*, 2023. 5, 6, 7, 8

[57] Ziyi Yang, Yanzhen Chen, Xinyu Gao, Yazhen Yuan, Yu Wu, Xiaowei Zhou, and Xiaogang Jin. Sire-ir: Inverse rendering for brdf reconstruction with shadow and illumination removal in high-illuminance scenes. *arXiv preprint arXiv:2310.13030*, 2023. 2

[58] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023. 2, 7

[59] Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv 2310.10642*, 2023. 2, 7

[60] Ziyi Yang, Xinyu Gao, Yangtian Sun, Yihua Huang, Xiaoyang Lyu, Wen Zhou, Shaohui Jiao, Xiaojuan Qi, and Xiaogang Jin. Spec-gaussian: Anisotropic view-dependent appearance for 3d gaussian splatting. *arXiv preprint arXiv:2402.15870*, 2024. 8

[61] Wang Yifan, Noam Aigerman, Vladimir G. Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3D deformations. In *CVPR*, 2020. 2

[62] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 2

[63] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Mesh editing with Poisson-based gradient field manipulation. In *ACM SIGGRAPH*, pages 644–651. 2004. 2

[64] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. Nerf-editing: Geometry editing of neural radiance fields. *CVPR*, 2022. 3

[65] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, Leif Kobbelt, and Lin Gao. Interactive nerf geometry editing with shape priors. *IEEE TPAMI*, 2023. 3

[66] Raza Yunus, Jan Eric Lenssen, Michael Niemeyer, Yiyi Liao, Christian Rupprecht, Christian Theobalt, Gerard Pons-Moll, Jia-Bin Huang, Vladislav Golyanik, and Eddy Ilg. Recent trends in 3d reconstruction of general non-rigid scenes. In *Comput. Graph. Forum*. Blackwell-Wiley, 2024. 1, 2

[67] Qiang Zhang, Seung-Hwan Baek, Szymon Rusinkiewicz, and Felix Heide. Differentiable point-based radiance fields for efficient view synthesis. In *ACM SIGGRAPH ASIA*, pages 1–12, 2022. 2

[68] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. 2018. 5

[69] Yuzhe Zhang, Jianmin Zheng, and Yiyu Cai. Proxy-driven free-form deformation by topology-adjustable control lattice. *Computers & Graphics*, 89:167–177, 2020. 2

[70] Chengwei Zheng, Wenbin Lin, and Feng Xu. Editablenerf: Editing topologically varying neural radiance fields by key points. In *CVPR*, 2023. 3