MoDec-GS: Global-to-Local Motion Decomposition and Temporal Interval Adjustment for Compact Dynamic 3D Gaussian Splatting

Supplementary Material

A. Project Page and Demo Video

Please refer to our project page: https://kaist-viclab.github.io/MoDecGS-site/. The project page provides a summarized description of our method, an interactive visual comparison demo, and demo videos. In the demo video (https://youtu.be/5L6gzc5-cw8), we demonstrated subjective quality comparisons for HyperNeRF's interp-cut-lemon, interp-torchocolate, misc-espresso, misc-tampling, vrig-peel-banana. We compared four framworks which are SC-GS [21], Deformable 3DGS [63], 4DGS [60], and MoDec-GS (Ours). The videos are concatenated in a 2×2 or 4×1 format depending on the shape of the video. Additionally, the code will be released through a GitHub repository: https://github.com/skwak-kaist/MoDec-GS.

B. Implementation Details

MoDec-GS is implemented using PyTorch and built upon 4DGS [60] and Scaffold-GS [40] codebases. Similar to 4DGS [60] we adopt a hexplane-based deformation method to represent video content, while using an anchor-based representation [40] for the canonical 3D Gaussians. The key hyperparameters for the anchor repesentation include n_offset=20, voxel_size=0.01, feat_dim=32, and appearance_dim=16, with no feature bank utilized. Iterations are set as follows: 3,000 for the Global stage, and between 20,000 and 60,000 for the Local stage depending on the sequence length. The global and local hexplanes are set to $[32, 32, 32, 10]$ and $[64, 64, 64, 100]$ with a two-level multi-resolution, respectively, for all test cases. The parameters for TIA are as follows: $T^{\text{TIA}}_{\text{from}}$ is set to 500, $T^{\text{TIA}}_{\text{until}}$ is set to 10,000 or 20,000 depending on the total number of iterations, and $T^{\text{TIA}}_{\text{period}}$ is set to 1,000, also depending on the iterations. τ_{TIA} is set to either 1.0 or 1.5, and the s_{TIA} is chosen within the range of 0.01 to 0.1. For the comparison methods, Deformable-3DGS and SC-GS were compared in the same local experimental environment and retrained on all datasets. For the Deformable-3DGS, the option for 6-degrees of freedom deformation is turned on for better rendering quality. We set the number of node and the dimension of hyper coordinates for the SC-GS at 2048 and 8, respectively. No mask images for background separation were used.

C. Preliminaries

C.1. Anchor-based representation

Lu *et al.* [40] proposed Scaffold-GS, a structured anchor-based 3DGS representation approach, designed to improve efficiency, robustness and scalability in novel view synthesis. Unlike traditional 3DGS, which often results in redundant Gaussians due to excessive fitting to training views, this approach introduces a hierarchical and structured representation. The method begins by initializing a sparse set of anchor points from Structure-from-Motion (SfM) points and distributing neural Gaussians around these anchors. As shown in Fig. 6, each anchor point is associate with learnable offsets and a scaling factor, allowing local Gaussians to be dynamically placed and adapted to varying viewpoints. Therefore, instead of allowing Gaussians to drift freely like in 3DGS [26], Scaffold-GS constrains their placement using anchor points. Given an anchor at position x_v, the position of k derived Gaussians are computed as:

$$\{\mu_0, \cdots, \mu_{k-1}\} = x_v + \{O_0, \cdots, O_{k-1}\} \cdot l_v, \tag{15}$$

where O_i are learnable offsets and l_v is a scaling factor. Their opacity values α, colors c, and other attributes are decoded through MLPs based on the local context feature and view-dependent information:

$$\{\alpha_0, \cdots, \alpha_{k-1}\} = F_\alpha(f_v, \delta_{vc}, \mathbf{d}_{vc}), \tag{16}$$

where $\delta_v c$ and $\mathbf{d}_{vc}$ represent the distance and direction from the camera to the anchor. The attributes of these Gaussians - position, opaicty, color and scale - are predicted on-the-fly based on local context feature assigned on the anchor and the viewing direction. This view-adaptive mechanism prevents excessive redundancy and enhances robustness to complex scene structures. To further refine the representation, Scaffold-GS employs a growing and pruning strategy. Growing introduces new anchors in underrepresented areas where the gradient magnitude of neural Gaussians exceeds a predefined threshold. Pruning removes anchors that consistently produce low-opacity Gaussians, ensuring an efficient representation. At inference time, only Gaussians within the view frustum and with significant opacity contribute to rendering, maintaining real-time performance.

C.2. HexPlane-based deformation encoder

In this work, we employ the HexPlane which is widely adopted for the deformation fields in dynamic scene repre-

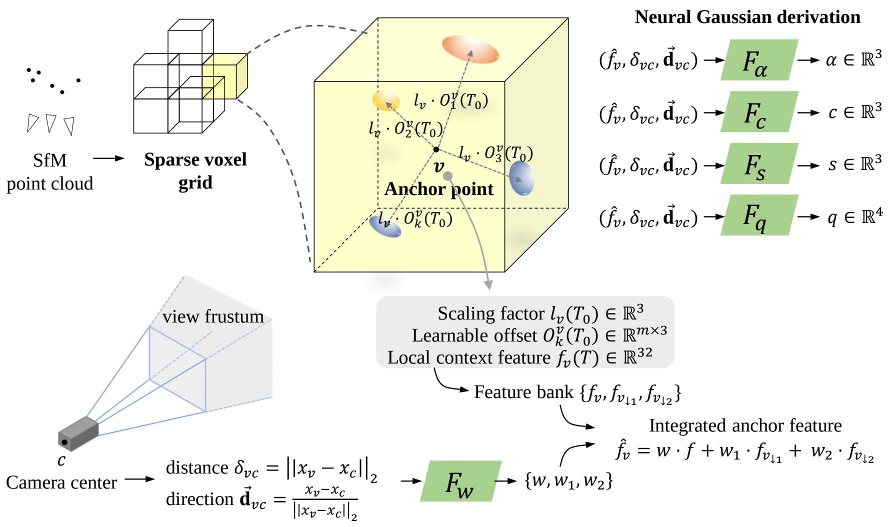
Neural Gaussian derivation
$(\hat{f}_v, \delta_{vc}, \vec{\mathbf{d}}_{vc}) \rightarrow F_\alpha \rightarrow \alpha \in \mathbb{R}^3$
$(\hat{f}_v, \delta_{vc}, \vec{\mathbf{d}}_{vc}) \rightarrow F_c \rightarrow c \in \mathbb{R}^3$
$(\hat{f}_v, \delta_{vc}, \vec{\mathbf{d}}_{vc}) \rightarrow F_s \rightarrow s \in \mathbb{R}^3$
$(\hat{f}_v, \delta_{vc}, \vec{\mathbf{d}}_{vc}) \rightarrow F_q \rightarrow q \in \mathbb{R}^4$
SfM point cloud
Sparse voxel grid
$l_v \cdot O_1^v(T_0)$
$l_v \cdot O_2^v(T_0)$
$l_v \cdot O_3^v(T_0)$
$l_v \cdot O_k^v(T_0)$
v
Anchor point
Scaling factor $l_v(T_0) \in \mathbb{R}^3$
Learnable offset $O_k^v(T_0) \in \mathbb{R}^{m \times 3}$
Local context feature $f_v(T) \in \mathbb{R}^{32}$
Feature bank $\{f_v, f_{v_{\downarrow 1}}, f_{v_{\downarrow 2}}\}$
Integrated anchor feature
$\hat{f}_v = w \cdot f + w_1 \cdot f_{v_{\downarrow 1}} + w_2 \cdot f_{v_{\downarrow 2}}$
view frustum
c
Camera center
distance $\delta_{vc} = ||x_v - x_c||_2$
direction $\vec{\mathbf{d}}_{vc} = \frac{x_v - x_c}{||x_v - x_c||_2}$
F_w
$\{w, w_1, w_2\}$

Figure 6. **Overview of the anchor-based 3DGS representation process** [40].

sentations [14, 60]. The spatio-temporal flow can be predicted by inputting the feature vector that corresponds to each spatio-temporal coordinate (x, y, z, t), the resulting hexplane feature vector H_L can be expressed as follows:

$$H_L(x, y, z, t) = \bigcup_{s \in S} \prod_{p \in P} f(x, y, z, t)_p^s, \tag{17}$$

where $f(x, y, z, t)_c^s$ denote the interpolated feature corresponding to the queried four-dimensional coordinate, p and s are the indices for the two-dimensional planes and the grid scales, respectively.

D. Concurrent Works

Very recently, some concurrent works [5, 17], have been proposed. Cho *et al.* [5] proposed a framework for extending 3D scaffolds into 4D space, aiming to efficiently represent 4D Gaussians through the introduction of neural velocity-based time-variant Gaussians and temporal opacity. Another work, Relay-GS [17], proposed a framework for effectively handling large-scale complex motion by modeling motion within temporal segments. It utilizes a learnable mask to separate the dynamic foreground and employs pre-generated pseudo-views, where semi-transparent Gaussians—*Relay Gaussians*—are placed along the trajectory. Both studies propose compact and efficient dynamic Gaussian representations for real-world scenes. However, they share a common limitation: neither can handle *casual monocular data*, which is closer to real-world settings.

E. Comparison to NeRF-extension Methods

Recent trends in NVS are driven by 3DGS [26] and its extensions [21, 28, 38, 56, 60, 63]. However, NeRF-based methods that utilize differentiable volume rendering are still being actively researched and demonstrate strong performance in terms of visual quality [14, 27, 36, 39, 45]. Although our primary target application focuses on being *compact* without losing *real-time rendering* capabilities, we also provide a comparison with NeRF-based approaches for reference information using results taken from [60]. Tab. 4 presents the comparison results. In the table, the numbers for [11, 19, 26, 48, 49] are sourced from [60] and those of the other are generated in our local environment. We confirmed that the performances of 4DGS reported in [60] is nearly reproduced in our side, and note that there are differences of GPU environment ([60]: RTX 3090, Ours: RTX A6000 - due to the time limitation, the speed comparison measured on the same machine has not prepared, but it is generally known that RTX A6000 is slower than RTX 3090. We plan to fairly measure the training time and rendering speed on the same GPU in the future). Through the comparison, we confirmed that our method achieved the lowest storage requirement at only 52% of the second-best [11], while maintaining the highest visual quality scores and high-speed rendering performance exceeding 20 fps. Additionally, to visualize the comparison results with other frameworks, we present a performance comparison graph, as shown in Fig. 7 where the x-axis represents rendering speed (FPS), the y-axis denotes PSNR, and the bubble size (MB) indicates the model's storage size. Our method achieves an exceptionally small storage size while maintaining the highest level of visual quality performance.

F. Detailed Experimental Results

F.1. Datasets and metrics

In this paper, the following three datasets are used: iPhone [16], HyperNeRF [49], and Nvidia [64]. All datasets

Methods	PSNR(dB)↑	MS-SSIM↑	Training times↓	Run times(FPS)↑	Storage(MB)↓
Nerfies [48]	22.2	0.803	$\sim$ hours	< 1	-
HyperNeRF [49]	22.4	0.814	32 hours	< 1	-
TiNeuVox-B [11]	24.3	0.836	**30 mins**	1	48
FFDNeRF [19]	24.2	**0.842**	-	0.05	440
V4D [15]	24.8	0.832	5.5 hours	0.29	377
3DGS [26]	19.7	0.680	40 mins	**55**	52
4DGS [60]	25.0	0.838	1.2 hour	24.9	61
MoDec-GS (Ours)	**25.0**	0.836	1.2 hour	23.8	**28**

Table 4. **Performance comparison with a NeRF-extension framework, including training and rendering speed.** Averaged over 536×960 HyperNeRF's vrig datasets [49]. The performance numbers of [11, 19, 26, 48, 49] are sourced from [60]. The training times and run times reported in [60] were measured on an NVIDIA RTX 3090 GPU, while our framework was tested on an RTX A6000 GPU. Please note that the A6000 GPU has approximately 20 % lower memory bandwidth compared to that of the RTX 3090.

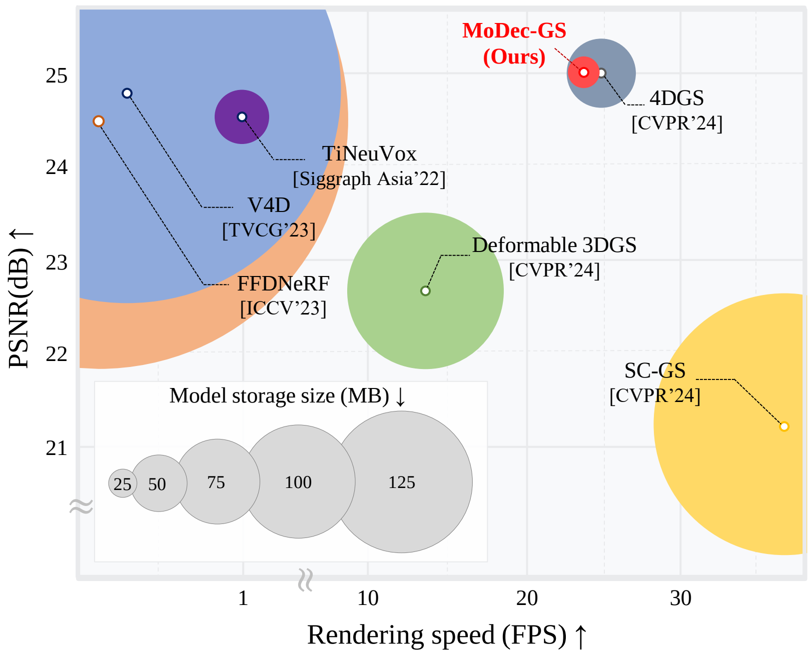

Figure 7. **Performance comparison visualization graph.** The x-axis represents rendering speed (FPS)↑, and the y-axis indicates PSNR↑. Each framework is depicted as a bubble, with the size of the bubble representing the model storage size (MB)↓.

were downloaded from their official repositories, and the COLMAP [53] data for iPhone and HyperNeRF datasets were directly generated using the script provided in [60] by ourselves. Note that the script is designed to subsample frames to ensure that the number of frames does not exceed 200 when obtaining the initial point cloud. For COLMAP inputs, we used the 2× version of the sequences for each dataset. Specifically, 360×480 for the iPhone dataset and 536×960 for the HyperNeRF dataset. For the Nvidia dataset, multi-view video frames were sampled sequentially at one frame per timestamp, resulting in a total of 192 *monocular* frames. To define the test frames, every 8th frame was excluded from the training views. This is one of the settings provided in [60], resulting in a total of 168 training views and 24 test views meaning the temporal interpolation, which is more challenging setting. Through validation on this setting of NVIDIA dataset, which features long-range time duration and high resolution (around FHD to 2K), we aimed to effectively verify the storage reduction capabilities of our model. Regarding the metrics, PSNR, SSIM [59], and LPIPS [65] metrics are calculated using the functions in [60], while masked metrics for the iPhone dataset are obtained by the functions and covisible masks provided by DyCheck [16]. For tOF [7], module form Teco-

GAN [6] is utilized. For model storage, it is calculated as the sum of the sizes of a single global CS ply, two deformation fields, per-attribute MLPs, and the canonical time list. Note that H_L is shared across temporal intervals, meaning that only a single pair of H_G and H_L exists.

F.2. Detailed results

The full quantitative results on three datasets are presented in Tab 9. Our method achieves the best or second-best visual quality performance in almost all sequences while using significantly less storage. Regarding the average performance of HyperNeRF, not only in the *interp* results which are reported in the main paper, but also in *misc* and *vrig*, our method shows the highest PSNR/tOF and second-best SSIM performance, while using about 40% less storage compared to the second-best model from a storage perspective [60].

F.3. Generalization to additional datasets

We further evaluated our method's robustness using the D-NeRF [50] and PanopticSports [23] datasets, each representing synthetic and real-world complex motion characteristics, respectively. For D-NeRF, we referenced the results form Compact Dynamic 3DGS (C. D. 3DGS) [25]. For PanopticSports, the results are adopted from TC-3DGS [22]. As confirmed by the experimental results, our method demonstrate considerble rendering quality while maintaining low storage requirements, in both synthetic scenes and real-world complex motions.

Methods	PSNR(dB)↑	MS-SSIM↑	LPIPS↓	Storage(MB)↓
TiNeuVox-S [11]	30.75	0.96	0.07	**8**
TiNeuVox-B [11]	32.67	0.97	0.04	48
V4D [15]	**33.72**	0.98	**0.02**	1200
C.D.3DGS [25]	32.19	0.97	0.04	159
MoDec-GS (Ours)	33.25	**0.99**	**0.02**	**8**

Table 5. **Performance comparison on D-NeRF dataset.** The results were averaged over all sequences in the dataset, and the values for the comparison method were taken from [25].

Methods	PSNR(dB)↑	SSIM↑	LPIPS↓	Storage(MB)↓
Dynamic 3DGS [41]	**28.70**	0.91	0.17	2008
STG [32]	20.45	0.79	**0.10**	**19**
4DGS [60]	27.22	0.91	**0.10**	63
TC-3DGS [22]	27.81	0.89	0.20	49
MoDec-GS (Ours)	27.96	**0.95**	0.13	34

Table 6. **Performance comparison on PanopticSports dataset.** Results for the comparison method were sourced from [22].

G. Ablation Studies

G.1. Rendering Overhead by 2-stage Deformation

As shown in Tab. 4, our method experiences only a marginal drop in FPS compared to 4DGS [60], while maintaining real-time rendering capability. To further clarify the computational overhead introduced by the 2-stage deformation, we compared the rendering speed when using only a 1-stage deformation in our method. This corresponds to (b) in the ablation studies of Tab. 3. As in Tab. 4, the rendering speed comparison was conducted on the HyperNeRF's vrig dataset, and the results are presented in Tab. 7.

Method	Rendering speed (FPS)
Ours (1-stage)	24.7
Ours (2-stage)	23.8

Table 7. **Rendering speed comparison** between 1-stage and 2-stage deformation of our method.

G.2. Hyperparameter studies

We conducted an ablation study to assess the robustness of our framework and analyze the impact of hyperparameter variations. To align with the results in Tab. 3, we performed experiments by varying several key parameters on the iPhone [16] dataset. We conducted variation experiments on the local hexplane H_L size, voxel size ϵ, and the number of Gaussians per grid cell N_{offset} as shown in Tab. 8. The default settings are shown in the middle column of the table. We observed that a trade-off between quality and storage depending on the HexPlane/voxel grid resolution and the number of Gaussians per grid cell N_{offset}. The current setting provides a well-balanced compromise between these factors.

G.3. Visualization of GLMD

Our MoDec-GS is characterized by its ability to decompose global and local motion through a 2-stage deformation process. This technique, called GLMD, enables effective representation of complex motions even with a limited-size hexplane. To verify whether GLMD operates as our design intention, we visualize the individual rendering results of Global CS, Local CS, and the final deformed frame, which is shown in Fig. 9. For the cut-lemon scene in HyperNeRF, we rendered the Global CS directly, as shown in the topmost image. After the Global CS is deformed into each Local CS through GAD, we rendered each Local CS as shown in the central image and then measured the optical flow [55] between the two. As we can see in the rendered Local CS and the optical flow, it can be observed that a global motion with an overall similar direction is represented according to the movement of the knife cutting the

Params	Variation A				Default				Variation B			
	PSNR↑	SSIM↑	LPIPS↓	Storage↓	PSNR↑	SSIM↑	LPIPS↓	Storage↓	PSNR↑	SSIM↑	LPIPS↓	Storage↓
H_L size	[32, 32, 32, 50]				[64, 64, 64, 100]				[128, 128, 128, 150]			
	14.28	0.330	0.476	16.23	14.60	0.480	0.443	18.37	14.61	0.489	0.416	29.65
Voxel size	0.1				0.01				0.001			
	13.93	0.332	0.528	17.46	14.60	0.480	0.443	18.37	14.45	0.475	0.429	23.12
N_{offset}	5				10				20			
	13.80	0.322	0.513	15.15	14.60	0.480	0.443	18.37	14.54	0.486	0.422	23.00

Table 8. **Hyper-parameter variation experiments** on the local hexplane size, voxel size, and the number of Gaussians per grid cell. The default settings used in the main paper's experiments are shown in the middle column.

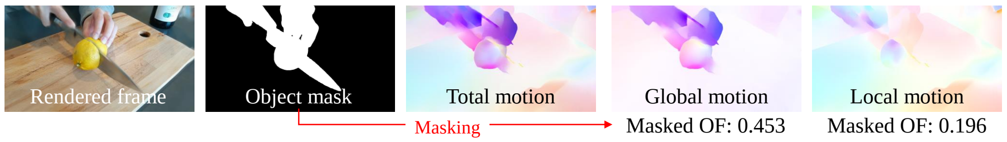

Figure 8. **Masked optical flow analysis for GLMD.**

lemon. Based on the optical flow color map, we visualized this by overlaying arrows on the rendered patch. The Local CS is then deformed into individual frames through LGD. We also rendered the frames at a fixed camera position during this process and observed the optical flow. As a result, various directional components of local motion were observed, which were also overlaid as arrows on the rendered patch. Through this detailed and intuitive visualization, we confirmed that the proposed GLMD effectively captures both global and local motions. Thanks to this capability, it achieves high scene representation for complex motions even with a smaller model size.

G.4. Analysis of Complex Motion through GLMD

In our work, global motion refers to *rigid transformations* within a time interval, while local motion captures *non-rigid deformations* between consecutive time steps. Complex motion is defined as a combination of these two types of motion. These characteristics can be observed in Fig. G.3. To further investigate this, we measured the average normalized optical flow magnitudes within an object mask [51] for global motion modeled by GAD, and local motion modeled by LGD. The results are shown in Fig . 8. As seen in the figure, GAD is primarily associated with object-centric rigid transformations, exhibiting a higher optical flow magnitude in the object mask regions on average. In contrast, LGD distributes the optical flow magnitude across the entire scene with relatively smaller values.

H. Limitations and Future Works

Fig. 10 illustrates a failure case on the HyperNeRF-broom dataset. In the challenging context of monocular video, representing thin and highly detailed textured objects using a finite number of 3D Gaussians remains a limitation. Consequently, neither the comparison methods [21, 60, 63] nor ours are able to effectively learn the scene.

To address this issue, previous studies have explored integrating traditional graphics techniques such as texture and alpha mapping into 3DGS [3], utilizing generalized exponential functions instead of 3D Gaussians [20], or incorporating hierarchical pyramid features to enhance detail representation [13]. As part of future work, we aim to enhance the Gaussian primitives used in MoDec-GS by building upon these prior studies, enabling robust expressivity even in scenes with intricate and highly detailed textures.

	(a) iPhone dataset							
Method	Apple		Block		Paper-windmill		Space-out	
SC-GS [21]	14.96 / 0.692 / 0.508 / 0.704	173.3	13.98 / 0.548 / 0.483 / 0.931	115.7	14.87 / **0.221** / 0.432 / 0.473	446.3	**14.79** / 0.511 / **0.440** / 0.411	114.2
Deformable 3DGS [63]	15.61 / 0.696 / **0.367** / 0.523	87.71	14.87 / 0.559 / **0.390** / 0.924	118.9	14.89 / 0.213 / **0.341** /0.519	160.2	14.59 / 0.510 / 0.450 / 0.562	42.01
4DGS [60]	15.41 / 0.691 / 0.524 / 0.591	61.52	13.89 / 0.550 / 0.539 / 1.095	63.52	14.44 / 0.201 / 0.445 / 0.375	123.9	14.29 / 0.515 / 0.473 / 0.331	52.02
MoDec-GS (Ours)	**16.48** / **0.699** / 0.402 / **0.459**	**23.78**	**15.57** / **0.590** / 0.478 / **0.852**	**13.65**	**14.92** / 0.220 / 0.377 / **0.357**	**17.08**	14.65 / **0.522** / 0.467 / **0.310**	**18.24**
	Spin		Teddy		Wheel		**Average**	
SC-GS [21]	14.32 / 0.407 / 0.445 / **1.191**	219.1	12.51 / 0.516 / **0.562** / 1.095	318.7	11.90 / 0.354 / 0.484 / 1.623	239.2	13.90 / 0.464 / 0.479 / 0.923	232.4
Deformable 3DGS [63]	13.10 / 0.392 / 0.490 / 1.482	133.9	11.20 / 0.508 / 0.573 / 1.460	117.1	11.79 / 0.345 / **0.394** / 1.732	106.1	13.72 / 0.461 / **0.430** / 1.029	109.4
4DGS [60]	14.89 / 0.413 / 0.441 / 1.362	71.80	12.31 / 0.509 / 0.605 / 1.156	80.44	10.83 / 0.339 / 0.538 / 2.007	96.50	13.72 / 0.460 / 0.509 / 0.988	78.54
MoDec-GS (Ours)	**15.53** / **0.433** / **0.366** / 1.265	**26.84**	**12.56** / **0.521** / 0.598 / **1.056**	**12.28**	**12.44** / **0.374** / 0.413 / **1.561**	**16.68**	**14.60** / **0.480** / 0.443 / **0.837**	**18.37**
	(b) Hypernerf dataset							
	interp - Aleks-teapot		*interp* - Chickchiken		*interp* - Cut-lemon		*interp* - Hand	
SC-GS [21]	24.86 / 0.854 / 0.186 / 5.406	426.0	26.05 / 0.781 / **0.239** / **4.176**	101.2	29.63 / 0.862 / 0.182 / 2.469	130.8	28.97 / 0.859 / 0.192 / **4.206**	404.3
Deformable 3DGS [63]	20.13 / 0.625 / 0.479 / 11.00	108.0	25.89 / 0.782 / 0.272 / 4.539	50.77	28.61 / 0.792 / 0.269 / 3.936	82.65	28.91 / 0.855 / 0.191 / 4.574	144.6
4DGS [60]	**26.99** / 0.853 / 0.193 / 3.309	105.6	**26.88** / **0.797** / 0.336 / 7.036	50.34	30.17 / 0.776 / 0.325 / 5.598	56.05	**29.87** / 0.847 / 0.223 / 4.928	85.26
MoDec-GS (Ours)	26.72 / **0.871** / **0.162** / **3.074**	**55.69**	26.65 / 0.793 / 0.271 / 4.884	**31.17**	**31.08** / **0.878** / **0.161** / **2.462**	**25.40**	29.65 / **0.867** / **0.187** / 4.355	**73.60**
	interp - Slice-banana		*interp* - Torchocolate		*misc* - Americano		*misc* - Cross-hands	
SC-GS [21]	24.57 / 0.641 / **0.323** / **7.697**	76.15	27.62 / 0.893 / 0.155 / **2.640**	217.0	30.84 / 0.928 / 0.101 / 3.055	271.4	**28.78** / **0.844** / **0.198** / **2.209**	222.1
Deformable 3DGS [63]	24.74 / 0.647 / 0.380 / 8.594	52.10	27.47 / 0.890 / 0.171 / 2.924	84.52	30.87 / 0.929 / **0.094** / **2.896**	141.6	27.70 / 0.813 / 0.246 / 2.683	142.8
4DGS [60]	**25.27** / **0.676** / 0.428 / 11.10	47.45	25.44 / 0.829 / 0.301 / 6.784	91.10	**31.30** / 0.917 / 0.137 / 3.706	85.72	28.06 / 0.763 / 0.350 / 6.644	62.10
MoDec-GS (Ours)	24.70 / 0.653 / 0.428 / 8.729	**31.74**	**27.86** / **0.896** / **0.136** / 2.657	**27.34**	30.55 / **0.932** / 0.100 / 2.934	**43.99**	28.39 / 0.821 / 0.253 / 4.545	**23.97**
	misc - Espresso		*misc* - Keyboard		*misc* - Oven-mitts		*misc* - Split-cookie	
SC-GS [21]	**26.52** / **0.910** / **0.167** / **5.162**	160.4	28.47 / 0.904 / **0.129** / **3.980**	229.4	27.54 / 0.830 / 0.182 / 3.483	88.63	**33.01** / **0.940** / **0.087** / 2.529	255.1
Deformable 3DGS [63]	25.47 / 0.899 / 0.179 / 5.513	60.93	28.15 / 0.900 / 0.137 / 4.190	97.77	27.51 / **0.832** / **0.175** / **3.396**	39.83	32.63 / 0.937 / 0.087 / 2.417	107.9
4DGS [60]	25.82 / 0.899 / 0.191 / 5.732	72.93	28.64 / 0.895 / 0.177 / 4.762	62.57	**27.99** / 0.801 / 0.316 / 6.241	45.73	32.64 / 0.919 / 0.147 / 3.362	67.00
MoDec-GS (Ours)	26.16 / 0.905 / 0.170 / 5.808	**25.06**	**28.68** / **0.906** / 0.136 / 4.230	**25.63**	27.78 / 0.820 / 0.220 / 4.630	**20.03**	32.84 / 0.935 / 0.093 / **2.400**	**45.88**
	misc - Tamping		*vrig* - 3dprinter		*vrig* - Broom		*vrig* - Chicken	
SC-GS [21]	23.10 / 0.781 / **0.326** / 6.352	259.4	18.79 / 0.613 / 0.269 / 15.17	101.7	18.66 / 0.269 / **0.505** / 14.12	122.6	21.85 / 0.616 / 0.257 / 11.83	111.2
Deformable 3DGS [63]	23.95 / 0.804 / 0.331 / 6.409	**17.92**	20.33 / 0.666 / 0.306 / 14.11	40.33	21.00 / 0.306 / 0.646 / 13.12	181.8	22.66 / 0.642 / 0.276 / 11.12	63.25
4DGS [60]	24.15 / 0.801 / 0.342 / 6.656	78.26	21.97 / 0.704 / 0.328 / 14.92	55.82	**21.85** / **0.365** / 0.559 / **9.279**	51.13	28.53 / 0.807 / 0.295 / 8.137	46.11
MoDec-GS (Ours)	**24.33** / **0.809** / 0.339 / **6.329**	24.77	**22.00** / **0.706** / **0.265** / **13.06**	**26.60**	21.04 / 0.303 / 0.666 / 13.50	**30.83**	**28.77** / **0.834** / **0.197** / **4.936**	**23.22**
	vrig - Peel-banana		**Average - *interp***		**Average - *misc***		**Average - *vrig***	
SC-GS [21]	25.49 / 0.806 / 0.215 / 4.568	519.9	26.95 / 0.815 / **0.213** / 4.432	226.0	28.32 / **0.876** / **0.170** / 3.824	212.3	21.19 / 0.575 / **0.311** / 11.42	231.9
Deformable 3DGS [63]	26.93 / 0.851 / 0.193 / 4.386	268.0	25.96 / 0.766 / 0.294 / 5.929	87.13	28.04 / 0.873 / 0.178 / 3.929	86.97	22.72 / 0.616 / 0.355 / 10.68	138.3
4DGS [60]	27.66 / 0.847 / 0.206 / 4.179	93.02	27.44 / 0.797 / 0.302 / 6.459	72.65	28.37 / 0.857 / 0.237 / 5.301	67.76	25.00 / **0.680** / 0.347 / 9.131	61.52
MoDec-GS (Ours)	**28.25** / **0.873** / **0.171** / **3.801**	**29.80**	**27.78** / **0.827** / 0.219 / **4.360**	**40.82**	**28.39** / 0.875 / 0.187 / **4.411**	**29.90**	**25.01** / 0.679 / 0.324 / **8.827**	**27.61**
	(c) Nvidia monocular							
Method	Balloon1		Balloon2		Jumping		dynamicFace	
4DGS [60]	25.46 / 0.856 / 0.198 / -	67.43	27.12 / 0.842 / 0.151 / -	58.36	22.43 / 0.842 / 0.264 / -	46.19	27.32 / 0.935 / 0.121/ -	123.8
MoDec-GS (Ours)	**26.35** / **0.884** / **0.173** / -	**38.67**	**27.18** / **0.875** / **0.101** / -	**41.37**	**23.14** / **0.858** / **0.226** / -	**29.09**	**29.65** / **0.955** / **0.094** / -	**46.57**
	Playground		Skating		Truck		Umbrella	
4DGS [60]	22.17 / 0.743 / 0.215 / -	81.94	28.94 / 0.932 / 0.195 / -	42.08	28.28 / 0.889 / 0.234 / -	53.69	24.80 / 0.714 / 0.297 / -	65.96
MoDec-GS (Ours)	**23.35** / **0.817** / **0.149** / -	**49.41**	**29.31** / **0.942** / **0.155** / -	**25.27**	**29.21** / **0.911** / **0.184** / -	**37.68**	**25.04** / **0.762** / **0.223** / -	**49.08**

Table 9. **Quantitative results comparison on (a) iPhone [16], (b) HyperNeRF [49], (c) Nvidia [64] datasets**. **Red** and blue denote the best and second best performances, respectively. Each block element of 5-performance denotes (PSNR(dB)↑ / SSIM↑ [59] / LPIPS↓ [65] / tOF↓ [7] Storage(MB)↓). For iPhone dataset, the masked metrics are used. For Nvidia monocular dataset, tOF values are not computed since the test views are sparsely distributed along the temporal axis.

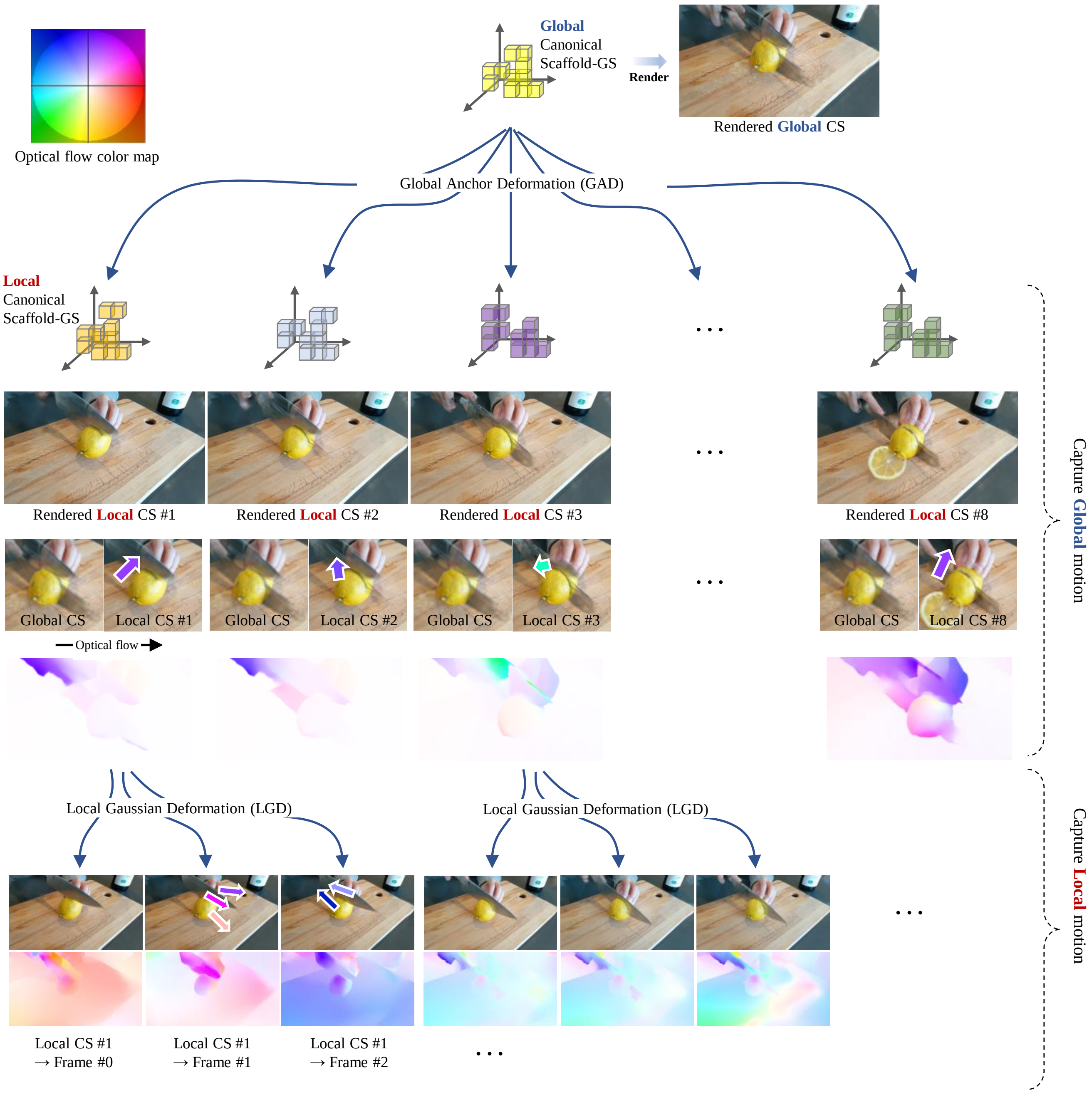

Figure 9. **Visualization of GLMD.** For *cut-lemon* scene in HyperNeRF [49] dataset, the rendered patch of Global CS, Local CS, and each time stamp are presented for a fixed camera viewpoint. We also illustrate the optical flow color map between those patches to observe the captured motion at each deformation stage. At GAD stage, deformation in mainly found near objects with dominant motion (e.g., the lemon and knife), and the overall color trends are similar, indicating a similar global motion direction. In contrast, at the LGD stage, motion is observed across the entire scene, with relatively more diverse range of motion directions.

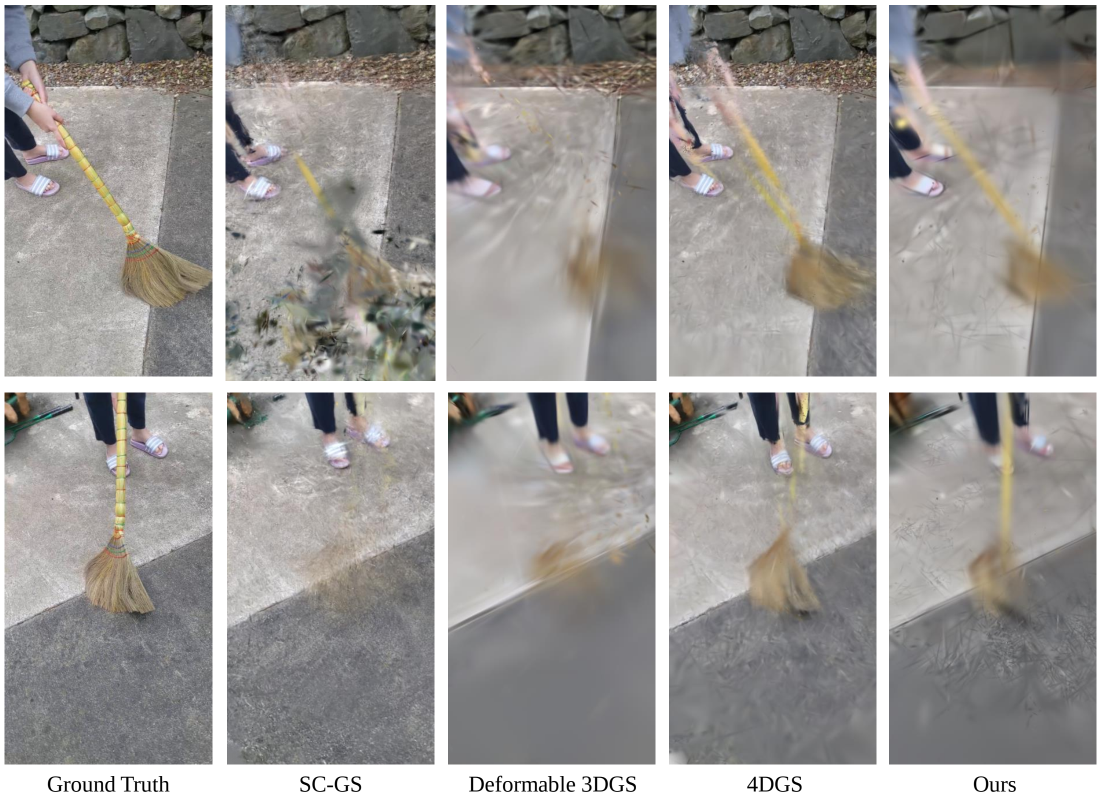

Figure 10. **Failure case: HyperNeRF-broom.** In the face of challenges in reconstructing dynamic scenes from monocular video, there are limitations in adequately representing thin and highly intricate textured objects.