MotionGS: Exploring Explicit Motion Guidance for Deformable 3D Gaussian Splatting

Ruijie Zhu[*]   **Yanzhe Liang**[*]   **Hanzhi Chang**   **Jiacheng Deng**   **Jiahao Lu**
Wenfei Yang   **Tianzhu Zhang**[†]   **Yongdong Zhang**

University of Science and Technology of China / Deep Space Exploration Lab

{ruijiezhu, yzliang, changhz, dengjc, lujiahao}@mail.ustc.edu.cn,
{yangwf, tzzhang, zhyd73}@ustc.edu.cn

Abstract

Dynamic scene reconstruction is a long-term challenge in the field of 3D vision. Recently, the emergence of 3D Gaussian Splatting has provided new insights into this problem. Although subsequent efforts rapidly extend static 3D Gaussian to dynamic scenes, they often lack explicit constraints on object motion, leading to optimization difficulties and performance degradation. To address the above issues, we propose a novel deformable 3D Gaussian splatting framework called MotionGS, which explores explicit motion priors to guide the deformation of 3D Gaussians. Specifically, we first introduce an optical flow decoupling module that decouples optical flow into camera flow and motion flow, corresponding to camera movement and object motion respectively. Then the motion flow can effectively constrain the deformation of 3D Gaussians, thus simulating the motion of dynamic objects. Additionally, a camera pose refinement module is proposed to alternately optimize 3D Gaussians and camera poses, mitigating the impact of inaccurate camera poses. Extensive experiments in the monocular dynamic scenes validate that MotionGS surpasses state-of-the-art methods and exhibits significant superiority in both qualitative and quantitative results. Project page: https://ruijiezhu94.github.io/MotionGS_page/.

1   Introduction

Dynamic scene reconstruction aims to model the 3D structure and appearance of time-evolving scenes, enabling novel-view synthesis at arbitrary timestamps. It is a crucial task in the field of 3D computer vision, attracting widespread attention from the research community and finding important applications in areas such as virtual/augmented reality and 3D content production. In comparison to static scene reconstruction, dynamic scene reconstruction remains a longstanding open challenge due to the difficulties arising from motion complexity and topology changes.

In recent years, a plethora of dynamic scene reconstruction methods [1, 2, 3, 4, 5, 6, 7, 8] have been proposed based on Neural Radiance Fields (NeRF) [9], driving rapid advancements in this field. While these methods exhibit impressive visual quality, their substantial computational overhead impedes their applications in real-time scenarios. Recently, a novel approach called 3D Gaussian Splatting (3DGS) [10], has garnered widespread attention in the research community. By introducing explicit 3D Gaussian representation and efficient CUDA-based rasterizer, 3DGS has achieved unprecedented high-quality novel-view synthesis with real-time rendering. Subsequent methods [11, 12, 13, 14,

---
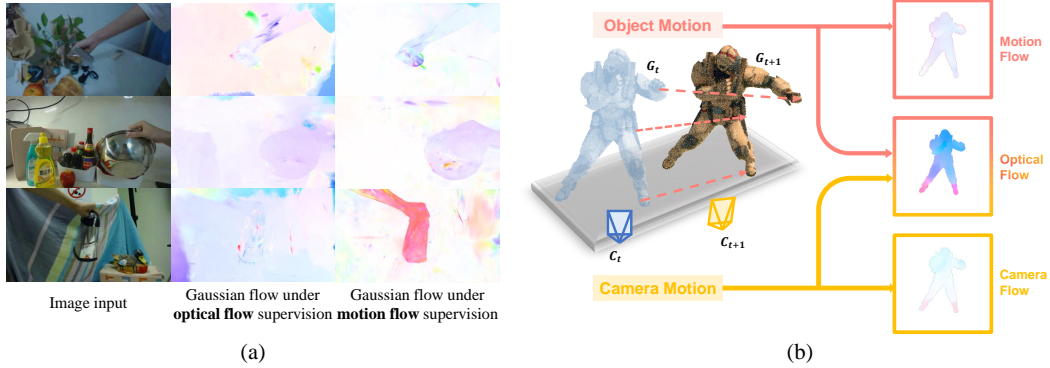
[*]Equal contribution

[†]Corresponding author

Figure 1: **(a) Gaussian flow under different supervision.** We model Gaussian flow under the supervision of optical flow and motion flow respectively. The latter can produce a more direct description of object motion, thereby effectively guiding the deformation of 3D Gaussians. **(b) The decoupling of optical flow.** We decouple the optical flow into motion flow which is only related to object motion and camera flow which is only related to camera motion.

15, 16, 17] rapidly extend 3DGS to dynamic scenes, also named 4D scenes. Initially, D-3DGS [11] proposes to iteratively reconstruct the scene frame by frame, but it incurs significant memory overhead. The more straightforward approaches [13, 17] utilize a deformation field to simulate the motion of objects by moving the 3D Gaussians to their corresponding positions at different time steps. Besides, some methods [12, 18] do not independently model motion but treat space-time as a whole to optimize. While these methods effectively extend 3DGS to dynamic scenes, they rely solely on appearance to supervise dynamic scene reconstruction, lacking explicit motion guidance on Gaussian deformation. When object motion is irregular (e.g., sudden movements), the model may encounter optimization difficulties and fall into local optima.

Based on the above discussions, we argue that explicit motion guidance is indispensable for the deformation of 3D Gaussians. Benefiting from the advancements in optical flow estimation [19, 20], a natural solution is to utilize an off-the-shelf optical flow network to provide 2D motion priors [21, 22]. However, the formation of optical flow is affected by both camera motion and object motion, which is not conducive to explicit modeling of object motion. Therefore, it is necessary to separate the optical flow related only to the moving object (*i.e.*, motion flow) to guide Gaussian deformation more efficiently. As shown in Figure 1(a), directly using optical flow (column 2) to supervise the Gaussian deformation will inevitably include the contribution of static objects to the optical flow, while using motion flow as supervision (column 3) can easily avoid this. Besides, the estimated camera pose in dynamic scenes is not always accurate. Due to the lack of geometric consistency between adjacent frames for moving objects, using point correspondences on dynamic objects to calculate the camera pose can lead to erroneous offsets, thereby affecting the optimization of 3DGS.

To address the above issues, we propose a novel deformable 3D Gaussian Splatting framework called MotionGS, which explicitly constrains the deformation of 3D Gaussians by extracting the motion priors from optical flow. Our method includes an optical flow decoupling module and a camera pose refinement module. In the optical flow decoupling module, we decouple the 2D optical flow into camera flow and motion flow, as shown in Figure 1(b). The camera flow comes from the camera ego-motion, while the motion flow comes from the motion of dynamic objects. We use the motion flow to directly constrain the deformation of 3D Gaussians (*i.e.*, Gaussian flow). Since the calculation of Gaussian flow is directly implemented in the CUDA-based rasterizer, this process is differentiable and efficient. In the camera pose refinement module, we first fix the 3D Gaussians and then utilize photometric consistency loss to backpropagate gradients to camera poses, thereby alternately optimizing 3D Gaussians and camera poses to further enhance the rendering quality.

To sum up, our main contributions are as follows:

- We propose a novel deformable 3D Gaussian framework called MotionGS, which provides explicit motion guidance for deformable 3DGS and achieves high-quality dynamic scene reconstruction with real-time rendering.

- The proposed optical flow decoupling module effectively separates the flow caused solely by object motion, thereby efficiently supervising the deformation of 3D Gaussians. The

proposed pose refinement module alternately optimizes 3DGS and camera poses, reducing reliance on accurate camera poses and further boosting rendering quality.

- Extensive experiments have demonstrated the effectiveness of the proposed method. Results on the NeRF-DS and HyperNeRF datasets validate the state-of-the-art performance of our approach in dynamic scene reconstruction.

2 Related Work

2.1 Novel-View Synthesis (NVS)

Novel view synthesis has been a hot research topic in the field of computer vision and graphics in recent years. NeRF [9], which represents 3D scene by neural radiance fields, first achieves high-resolution photorealistic results in this field. Despite many subsequent works [23, 24, 25, 26, 27, 28, 29, 30, 31] have been proposed to improve its efficiency and quality, NeRF-based methods still struggle to render high-quality images with real-time rendering speed. Recently, by modeling 3D scenes using a set of anisotropic 3D Gaussian with an efficient rasterizer, 3D Gaussian Splatting (3DGS) [10] has shown remarkable performance with real-time rendering. Compared to NeRF, 3DGS is an explicit 3D scene representation method with better scalability and editability. Therefore, it has been rapidly extended to other 3D vision tasks, including sparse-view reconstruction [32, 33, 34, 35], 3D generation [36, 37, 38, 39, 40], scene editing [41, 42, 43] and SLAM [44, 45, 46, 47].

2.2 Dynamic Scene Reconstruction

In recent years, various dynamic scene reconstruction approaches have been proposed, which can be broadly categorized into NeRF-based and 3DGS-based methods. NeRF-based works [48, 1, 49, 2, 4, 50, 51, 5] usually map dynamic scenes to a canonical space and render images based on this 3D canonical space. This kind of 4D scene representation is intuitive but requires a well-reconstructed canonical space. Other works propose to use time-varying NeRFs [6, 3, 52, 53, 54] or explicit representations [55, 56, 7, 8, 57, 58, 59] to represent and render dynamic scenes. However, all these NeRF-based methods require frequent point sampling or MLP queries, suffering from long training and rendering time. With the proposal of 3DGS, many works [16, 21, 15, 11, 13, 17, 12, 60, 14, 61, 62, 63] use 3DGS as the fundamental model for 4D scene representation. For instance, D-3DGS [11] models dynamic scenes by allowing the positions and rotation matrixes of 3DGS to change over time. Deformable 3DGS [17] uses an MLP to model a deformation field based on time and the canonical Gaussian space. SC-GS [15] bounds dense 3DGS with sparse control points, calculating the movement of Gaussians in a coarse-to-fine manner. Despite they have performed impressive rendering quality in some dynamic scenes, they lack explicit motion guidance to constrain the movement of Gaussian, resulting in degraded performance in more complex dynamic scenes. Recent works [21, 22] compose the movement of 3D points through their corresponding Gaussians, using 2D flow priors to supervise the deformation of 3DGS. Inspired by them, we decompose the optical flow to obtain more direct motion supervision, thus achieving higher rendering quality.

2.3 NVS with Pose Optimization

Several NVS works [64, 65, 66, 67, 68, 69, 70] have noticed that it is difficult to derive precise camera poses of input images in the real world, so they address novel view synthesis together with camera pose optimization. i-NeRF [64] initially estimates camera poses by matching the input images. Other methods such as NeRFmm [65] and Nope-NeRF [69] use monocular depth priors as guidance to do the joint optimization of NeRF and camera poses. Recently, CF-3DGS [70] proposes progressive reconstruction and leverages photometric loss to learn the affine transformation of Gaussians to optimize the camera pose. However, these methods are mostly effective only for static scenes and lack support for dynamic scenes. Motivated by these methods, we aim to extend 3DGS to dynamic scenes with pose optimization, thus boosting the rendering quality and robustness.

3 Preliminary

In this section, we briefly introduce the modeling and rendering of 3DGS in Section 3.1 and the deformable extension of 3DGS towards dynamic scene reconstruction in Section 3.2.

3.1 3D Gaussian Splatting

As an explicit 3D representation similar to point clouds, 3DGS models the scene with a set of 3D Gaussians. However, different from point clouds, each 3D Gaussian in the scene has its own opacity $o \in [0, 1]$, center position $\mu \in \mathbb{R}^{3 \times 1}$, and covariance matrix $\Sigma \in \mathbb{R}^{3 \times 3}$. These properties determine the contribution and influence range of 3D Gaussians on rendering. For a position $x \in \mathbb{R}^{3 \times 1}$ in 3D space, the corresponding contribution of a 3D Gaussian on it can be formulated as:

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

For differentiable optimization, the covariance matrix Σ can be decomposed into a scaling matrix $\mathbf{S}$ and a rotation matrix $\mathbf{R}$: $\Sigma = \mathbf{R}\mathbf{S}\mathbf{S}^T\mathbf{R}^T$, where scaling matrix $\mathbf{S} = \text{diag}([s_x, s_y, s_z])$ and rotation matrix $\mathbf{R}$ can be transformed from a quaternion $[r_w, r_x, r_y, r_z]$. Then the 3D Gaussians can be splatted to a 2D camera plane through differential gaussian splatting. Specially, given a viewing transform matrix W and the Jacobian matrix J of the affine approximation of the projective transformation, we can obtain the 2D covariance matrix Σ_{2D} through: $\Sigma_{2D} = JW\Sigma W^T J^T$. Similarly, we can obtain the 2D center position μ_{2D} of 3D Gaussians in camera plane. Therefore, given a 2D pixel p, the rendering contribution of a 3D Gaussian on the viewpoint W can be obtained through a 2D version of (1). To model the appearance of 3D Gaussians, spherical harmonics (SH) are introduced to define the color c. Finally, for each pixel, the rendering results of 3DGS can be derived by calculating the color contribution of all the related Gaussians. This process is known as α-blending:

$$C = \sum_i^N c_i \alpha_i \prod_{j=1}^{i-1}(1 - \alpha_j), \tag{2}$$

where c_i, α_i represent the color and density computed from the i-th 3D Gaussian.

3.2 Deformable 3D Gaussian Splatting

To extend 3DGS to dynamic scenes, an intuitive approach is to utilize a learnable deformation field to fit the movement of objects in the real world through Gaussian deformation. This idea originates from NeRF-based methods such as D-NeRF [1] and has been effectively applied to 3DGS in recent works [17, 13]. In these deformable 3DGS methods, a deformation network $\mathcal{D}$ is typically used to model the movement of the center position of 3D Gaussians. Additionally, due to the inherent properties of 3D Gaussians, the deformation network $\mathcal{D}$ may also consider the rotation and scaling factors of 3D Gaussians as they vary over time. Therefore, the deformation of 3D Gaussians can be formulated as:

$$(\mu + \Delta\mu, r + \Delta r, s + \Delta s) = \mathcal{D}(\mu, r, s, t), \tag{3}$$

where t is the timestamp, μ, r, s are the center position, rotation quaternion and scaling factors of 3D Gaussians, and $\Delta\mu, \Delta r, \Delta s$ are their residuals, respectively. Due to the various implementations of deformable 3DGS, in this paper we focus solely on the deformation aspect without discussing the other designs and specific differences in these works. We select method [17] as our baseline, leveraging explicit motion guidance and camera pose refinement to further enhance the rendering quality and the robustness in dynamic scenes.

4 Methodology

In this section, we first introduce the overall architecture of our approach in Section 4.1. Then the optical flow decoupling module is introduced to derive motion guidance for Gaussian deformation in Section 4.2. The camera pose refinement module is introduced to alternately optimize 3D Gaussians and camera poses in Section 4.3. Finally, the overall loss function is introduced in Section 4.4.

4.1 Overall Architecture

The overall architecture of our method is illustrated in Figure 2. Our method primarily focuses on the reconstruction of monocular dynamic scenes. Firstly, following 3DGS [10], we initialize camera poses and 3D Gaussians using COLMAP [71]. Given two adjacent frames I_t and I_{t+1}, we compute forward optical flow $F_{t \rightarrow t+1}$ using an off-the-shelf flow estimation network. Meanwhile, we can obtain the rendered depth map D_t of frame I_t at time t through the rasterizer. By feeding the depth
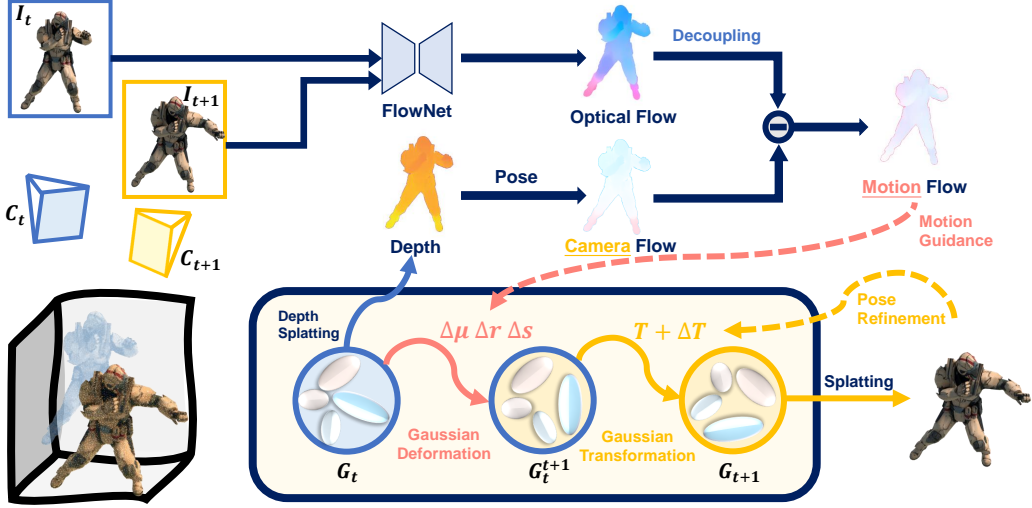
Figure 2: **The overall architecture of MotionGS.** It can be viewed as two data streams: (1) The 2D data stream utilizes the optical flow decoupling module to obtain the motion flow as the 2D motion prior; (2) The 3D data stream involves the deformation and transformation of Gaussians to render the image for the next frame. During training, we alternately optimize 3DGS and camera poses through the camera pose refinement module.

map I_t, camera poses C_t, C_{t+1}, and optical flow priors $F_{t \rightarrow t+1}$ into the optical flow decoupling module, we can calculate the motion flow $F^M_{t \rightarrow t+1}$ solely related to object movement. After predicting the deformation of Gaussians through the deformation network $\mathcal{D}$, we obtain the state of 3D Gaussians at time $t + 1$ and render the Gaussian flow $F^G_{t \rightarrow t+1}$ from time t to $t + 1$ under the assumption of a stationary camera viewpoint for the frame I_t. The motion flow should be consistent with the Gaussian flow, thus providing explicit motion guidance to Gaussian deformation. Additionally, since the initialized camera poses may be inaccurate, we add a small residual ΔT to the relative camera pose T. Leveraging the proposed camera pose refinement module, we cleverly backpropagate gradients to the camera poses, achieving refinement of the camera poses. During training, we alternately optimize 3D Gaussian and camera poses to enhance the rendering quality and robustness in dynamic scenes.

4.2 Optical Flow Decoupling Module

To provide explicit motion guidance for the deformation of Gaussians, we first utilize an off-the-shelf optical flow network to predict 2D motion priors. Since optical flow is influenced by both camera movement and object motion, we decompose it into camera flow and motion flow as illustrated in the Figure 1(b). Camera flow represents the optical flow caused solely by camera movement, assuming the objects in the scene remain stationary. In contrast, motion flow considers the camera as stationary, capturing only the movement of the objects. Essentially, optical flow can be viewed as the vector sum of these two components. By decoupling them, we can effectively isolate object motion, providing precise guidance for Gaussian deformation.

Camera flow and motion flow. We use a schematic diagram Figure 3 to illustrate the detailed calculation process. Camera flow can be directly computed from the camera poses and the depth of the current frame. Specifically, at the timestamp t, we obtain the depth map D_t corresponding to frame I_t directly from 3D Gaussians through the rasterizer. Given the intrinsics K_t and extrinsics T_t of camera C_t, we can reproject point p_t from frame I_t to 3D space using its depth D_t:

$$x_t = T_t^{-1} K_t^{-1} D_t \tilde{p}_t, \tag{4}$$

where $\tilde{p}_t$ is the homogeneous coordinate of p_t. Assuming x_t does not move, we can obtain the projection p_t^{t+1} of x_t on frame I_{t+1}:

$$p_t^{t+1} = \text{proj}(K_{t+1} T_{t+1} x_t), \tag{5}$$

where K_{t+1} and T_{t+1} are the intrinsics and extrinsics of camera C_{t+1}, proj() projects the 3D coordinates to 2D image planes by dividing the last dimension (depth). Then the camera flow can be
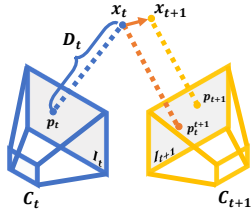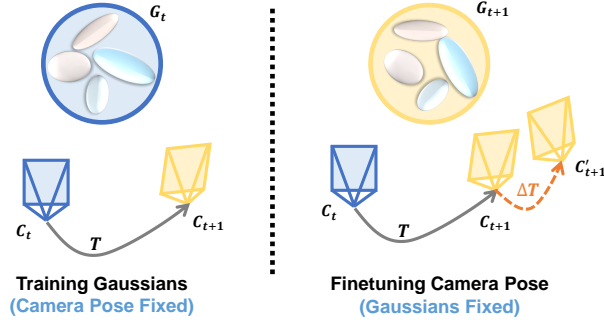
5

Figure 3: **Flow calculation.**

Figure 4: **Pose refinement on iterative training.**

defined as:

$$F_{t \to t+1}^C = p_t^{t+1} - p_t, \tag{6}$$

which indicates the flow caused solely by camera movement. As the point x_t moves over time, we denote its updated position as x_{t+1}. This new point x_{t+1} is then projected onto frame I_{t+1} as p_{t+1}. Thus, the optical flow $F_{t \to t+1}$ between two adjacent frame is defined as $p_{t+1} - p_t$. Finally, the motion flow $F_{t \to t+1}^M$ is derived by subtracting the camera flow from the optical flow:

$$F_{t \to t+1}^M = F_{t \to t+1} - F_{t \to t+1}^C = p_{t+1} - p_t^{t+1}, \tag{7}$$

which also corresponds to the optical flow caused by object movement at a fixed viewpoint.

Gaussion flow. To establish a correspondence between Gaussian deformation and motion flow, we need to splat the Gaussian deformation onto the 2D image plane, which is not implemented in the original 3DGS framework. Inspired by recent work [21], we introduce the concept of Gaussian flow, denoted as $F_{t \to t+1}^G$, to describe the 2D projection of Gaussian deformation, and implement it in the CUDA-based rasterizer. The core idea is to model the contribution of Gaussians to the optical flow by first transforming 3D Gaussians to canonical Gaussian space and then transforming them back to the state at the next time step. Please refer to Appendix A.1 for the specific derivation and modeling process of Gaussian flow. Gao *et al.* [21] computes the deformation of 3D Gaussians from time t to $t + 1$ under the transformation of the camera viewpoint from C_t to C_{t+1}, corresponding to optical flow. Different from it, our Gaussian flow is designed to match the motion flow, representing the deformation of 3D Gaussians from time t to $t + 1$ fixed under the camera viewpoint C_{t+1}.

Flow loss. To effective constrain the Gaussian deformation, we use a $\mathcal{L}_1$ loss between motion flow and Gaussian flow for simplicity:

$$\mathcal{L}_{\text{flow}} = \left\| sg(F_{t \to t+1}^M) - F_{t \to t+1}^G \right\|, \tag{8}$$

where $sg()$ means stop gradient. Note that we also stop the gradients of all variables at time t in the calculation of Gaussian flow for more efficient training.

Discussion. The benefits of decoupling the optical flow are evident. Since motion flow is only related to object motion, it can directly provide motion guidance. More importantly, in some previous works [4, 54, 6], an off-the-shelf segmentation network is often used to segment out the dynamic objects in the scene (such as humans, animals, cars, etc.). However, such masks are only used in their photometric loss to mask out dynamic regions. In contrast, our motion flow benefits from these dynamic masks more directly. By masking static objects with these masks, we can obtain a clear motion flow for supervising Gaussian deformation. If optical flow is used as motion guidance, this advantage will no longer exist because static objects can also contribute to the optical flow.

4.3 Camera Pose Refinement Module

In monocular dynamic scenes, due to the complexity of motion and sparsity of observations, even widely used methods like COLMAP [71] cannot accurately estimate camera poses. Since the optimization of 3DGS requires precise camera poses as input, it often performs poorly in complex

dynamic scenes. Existing 3DGS-based dynamic scene reconstruction methods rarely take this into account. Inspired by pose-free optimization methods for static scene reconstruction [72, 70], we design the camera pose refinement module. By alternately optimizing 3D Gaussian primitives and camera poses during training, we improve the rendering quality of 3DGS and its robustness in dynamic scenes.

Iterative training. Since the supervision of 3DGS primarily relies on photometric consistency loss, simultaneously optimizing camera parameters and 3DGS can be considered a chicken-and-egg problem. Therefore, similar to Bundle Adjustment, we adopt an alternating optimization strategy to train the model. Specifically, assuming G_t is the Gaussian at time t, we first predict the deformation of the Gaussian using the deformation field $\mathcal{D}$. We denote the deformed Gaussian as G_t^{t+1}. Since the observation viewpoint changes from time t to $t+1$, G_t^{t+1} needs to be transformed once again under the camera C_{t+1} to render frame I_{t+1}. We denote the transformed Gaussian as G_{t+1}. This transformation process actually corresponds to camera motion. To achieve differentiable optimization, we introduce a small residual ΔT into the relative pose T from camera viewpoint C_t to C_{t+1}, treating it as a learnable SE(3) transformation. With this small change, we enable gradients to backpropagate to the camera poses. During the optimization of camera poses, we freeze all attributes of 3D Gaussians to improve training stability and robustness. Then we update the camera poses initialized by COLMAP with the optimized relative camera poses, achieving global pose refinement.

Discussion. While several methods have been proposed for pose-free optimization in static scenes, dynamic scenes present greater challenges due to their inherently under-constrained nature. As a result, to ensure stable and robust optimization, our approach still leverages camera poses computed by COLMAP as an initialization step. This also necessitates the presence of sufficient static features in the scene. Fortunately, static features are commonly found in most real-world environments, particularly in background regions.

4.4 Optimization

Thanks to the integration of optical flow rendering and camera pose gradient computation in our rasterization process, the overall training pipeline of our method is end-to-end differentiable. The overall training loss is given by:

$$\mathcal{L} = \mathcal{L}_{\text{baseline}} + \lambda \mathcal{L}_{\text{flow}}, \tag{9}$$

where $\mathcal{L}_{\text{baseline}}$ is the photometric loss used in our baseline [17], λ is the weight of our flow loss.

5 Experiment

5.1 Experimental Setup

To highlight the abilities of our method in handling complex dynamic scenes, we select two representative monocular dynamic scene datasets for evaluation: NeRF-DS [73] and HyperNeRF [49]. Our implementation is mainly based on PyTorch. We use a simple Adam [74] optimizer to adjust the rotation increment and translation increment of the camera, and the learning rates of the two are set to 3e-3 and 1e-1, respectively. The entire training process requires 20,000 iterations. We set λ to 0.5 for NeRF-DS and 0.1 for HyperNeRF scene. The rest of the settings are consistent with the baseline method [17]. All experiments are performed on a single Nvidia RTX 3090 GPU. For more implementation details, please refer to Appendix A.2.

5.2 Results

Following previous methods, we use metrics PSNR, SSIM, and LPIPS for evaluation. For more visualizations, please refer to Appendix A.4.

Results on the NeRF-DS dataset. Table 1 shows the performance comparison results with the state-of-the-art methods on the NeRF-DS dataset. In dynamic monocular scenes, especially in those with rapid movements and high complexity, our method significantly outperforms the baseline method. For example (see Figures 5 and 12), in the plate scene, our method accurately renders the reflections and sharp edges of the moving plate while significantly reducing visual distortions such as floating

Table 1: **Quantitative comparison on NeRF-DS dataset per-scene**. We highlight the `best` and the `second best` results in each scene. NeRF-DS and HyperNeRF employ MS-SSIM and LPIPS with the AlexNet [75], while other methods and ours use SSIM and LPIPS with the VGG [76] network.

Method	Sieve			Plate			Bell			Press		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
3D-GS [10]	23.16	0.8203	0.2247	16.14	0.6970	0.4093	21.01	0.7885	0.2503	22.89	0.8163	0.2904
TiNeuVox [59]	21.49	0.8265	0.3176	20.58	0.8027	0.3317	23.08	0.8242	0.2568	24.47	0.8613	0.3001
HyperNeRF [49]	25.43	0.8798	0.1645	18.93	0.7709	0.2940	23.06	0.8097	0.2052	26.15	0.8897	0.1959
NeRF-DS [73]	25.78	0.8900	0.1472	20.54	0.8042	0.1996	23.19	0.8212	0.1867	25.72	0.8618	0.2047
Deformable-3DGS [17]	25.14	0.8674	0.1502	18.82	0.7404	0.3554	25.42	0.8481	0.1570	25.41	0.8614	0.1918
Ours	26.17	0.8884	0.1502	21.01	0.8213	0.1907	26.33	0.8688	0.1490	26.63	0.8865	0.1955

Method	Cup			As			Basin			Mean		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
3D-GS [10]	21.71	0.8304	0.2548	22.69	0.8017	0.2994	18.42	0.7170	0.3153	20.29	0.7816	0.2920
TiNeuVox [59]	19.71	0.8109	0.3643	21.26	0.8289	0.3967	20.66	0.8145	0.2690	21.61	0.8234	0.2766
HyperNeRF [49]	24.59	0.8770	0.1650	25.58	0.8949	0.1777	20.41	0.8199	0.1990	23.45	0.8488	0.1990
NeRF-DS [73]	24.91	0.8741	0.1737	25.13	0.8778	0.1741	19.96	0.8166	0.1855	23.60	0.8494	0.1816
Deformable-3DGS [17]	24.76	0.8876	0.1544	26.08	0.8827	0.1832	19.61	0.7888	0.1871	23.61	0.8394	0.1970
Ours	24.97	0.8916	0.1556	26.56	0.8902	0.1757	20.11	0.8126	0.1865	24.54	0.8656	0.1719

Table 2: **Quantitative comparison on HyperNeRF's vrig dataset per-scene.**

Method	3D Printer		Chicken		Broom		Banana		Mean	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
HyperNeRF [49]	20.0	0.63	27.4	0.63	19.5	0.21	22.1	0.72	22.3	0.55
TiNeuVox [59]	22.8	0.73	28.2	0.79	21.3	0.31	24.4	0.64	24.2	0.62
Deformable 3DGS [17]	20.5	0.64	22.8	0.61	20.5	0.35	26.0	0.83	22.5	0.61
Ours	21.8	0.71	26.8	0.79	22.3	0.38	28.2	0.86	24.8	0.69

Table 3: **Ablations on the key components of our proposed framework.**

Setting	PSNR ↑	SSIM ↑	LPIPS ↓
Baseline	23.61	0.8394	0.1970
+ optical flow guidance	23.37	0.8333	0.2112
+ motion flow guidance	24.12	0.8609	0.1763
+ motion flow guidance + camera pose refinement	24.54	0.8656	0.1719

artifacts. Similarly, in the basin scene, our method effectively models the smooth surface of the basin, in contrast to other methods that result in a bumpy basin bottom. This is mainly because our proposed framework can provide accurate and effective motion guidance for Gaussian deformation.

Results on the HyperNeRF dataset. For scenes captured in the wild using smartphones, Table 2 summarizes the relevant performance comparison results. Our method also achieves consistent performance improvements in these scenarios. Qualitatively, our approach excels at accurately reconstructing scene geometry and appearance, even under irregular camera movements and inaccurate camera poses. For instance (see Figures 6 and 13), in the chicken scene, our method captures the subtle bumps on the red shell, while in the broom scene, it accurately renders the details of the broom in fast movement. This is mainly attributed to the motion guidance and camera pose refinement proposed by our method, both of which enhance the rendering performance of the baseline method.

5.3 Ablation Study

In this section, we conduct ablations on the NeRF-DS dataset to validate the effectiveness of the key components of our method, as shown in Table 3. For more ablations, please refer to Appendix A.3.

Effectiveness of the optical flow decoupling module. To illustrate the necessity of optical flow decoupling module, we conduct ablations using direct optical flow supervision instead of decoupled motion flow constraints. As shown in the row 2 of Table 3, directly using optical flow to supervise Gaussian motion even results in a performance decline compared to the baseline. This performance drop is likely due to the inherent ambiguity created by the mixed camera and object movements. When camera and object movements are not separated, the supervision signal becomes noisy and less

Figure 5: **Qualitative comparison on NeRF-DS dataset.** Refer to Figure 12 for more scenes.
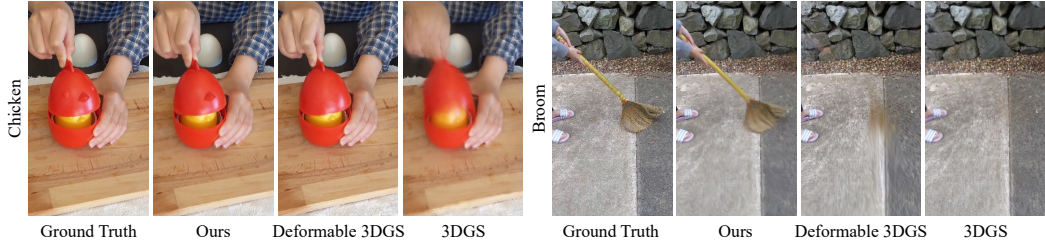

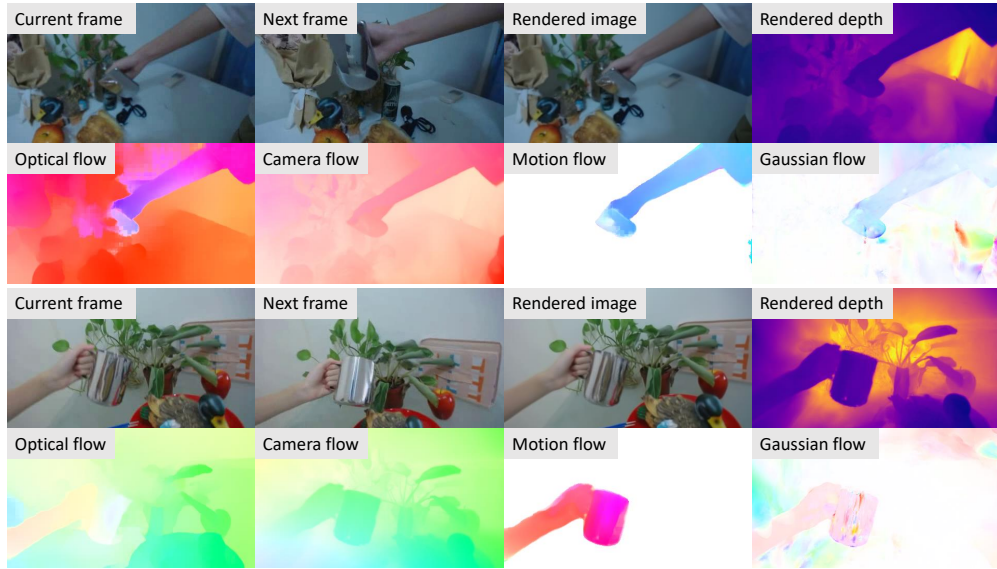Figure 6: **Qualitative comparison on HyperNeRF dataset.** Refer to Figure 13 for more scenes.

Figure 7: **Visualization of all data flows.** Each example corresponds to two rows.

effective. This ambiguity hampers the optimization process of 3DGS, thereby reducing the motion modeling capabilities of the deformation field. In contrast, we use motion flow as supervision, which effectively provides explicit motion guidance for Gaussian deformation, thereby better modeling complex dynamic scenes. As shown in Figures 7 and 14, only the motion flow can clearly highlights movement information in dynamic regions. This explicit motion information efficiently constrains the Gaussian flow, ensuring that the motion guidance remains consistent and effective.

Effectiveness of the camera pose refinement module.   Leveraging the alternating optimization of 3DGS and camera poses, our approach adaptively corrects potential errors in camera poses. Furthermore, the updated camera poses contribute to more accurate camera flow, thus improving the accuracy of motion guidance. In row 4 of Table 3, our camera pose refinement module, built upon motion guidance, yields substantial performance gains for the model. This iterative optimization process enhances the robustness of our model in complex dynamic scenes. For instance, in the HyperNeRF
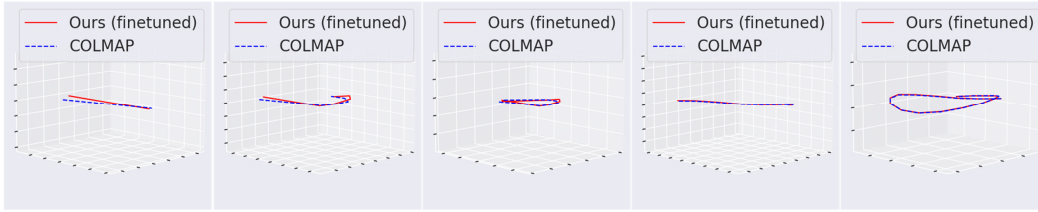
9

Figure 8: **Visualization of the camera trajectories optimized by our method and COLMAP.**

dataset, our method reconstructs more plausible results compared to the baseline approach. Unlike static scene datasets (e.g., Tanks & Temples) that use COLMAP to obtain the ground truth of camera poses, we assume that COLMAP may not provide accurate poses for dynamic scene datasets. In this setting, we lack ground truth for a direct quantitative evaluation for refined camera pose. Therefore, we provide visualizations of the pose refinement process in the Figure 8 as qualitative comparison.

6 Conclusion

In this paper, we propose MotionGS, a novel deformable 3D Gaussian Splatting framework for explicitly modeling and constraining object motion in dynamic scene reconstruction. The proposed framework includes two key modules: the optical flow decoupling module and the camera pose refinement module. The optical flow decoupling module decouples the motion flow related solely to object motion from the optical flow priors, providing explicit supervision for Gaussian deformation. The camera pose refinement module alternately optimizes 3DGS and camera poses, further enhancing the rendering quality and robustness of our model in dynamic scenes. Quantitative and qualitative results on the NeRF-DS and HyperNeRF datasets strongly demonstrate the contributions and effectiveness of our proposed method. More importantly, the proposed improvements are agnostic to specific network designs, which can be applied to similar deformation-based 3DGS methods. In future work, we aim to develop a 3DGS method that does not rely on camera pose inputs, thereby achieving robust high-quality reconstruction in dynamic scenes.

Acknowledgements

References

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

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

[3] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021.

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

[5] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. *Advances in Neural Information Processing Systems*, 35:33768–33780, 2022.

[6] 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*, pages 4273–4284, 2023.

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

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

[9] 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.

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

[11] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023.

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

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

[14] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *International Conference on Learning Representations (ICLR)*, 2024.

[15] 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. *arXiv preprint arXiv:2312.14937*, 2023.

[16] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. *arXiv preprint arXiv:2312.03431*, 2023.

[17] 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.

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

[19] Zhaoyang Huang, Xiaoyu Shi, Chao Zhang, Qiang Wang, Ka Chun Cheung, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer: A transformer architecture for optical flow. In *European conference on computer vision*, pages 668–685. Springer, 2022.

[20] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, and Dacheng Tao. Gmflow: Learning optical flow via global matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8121–8130, 2022.

[21] Quankai Gao, Qiangeng Xu, Zhe Cao, Ben Mildenhall, Wenchao Ma, Le Chen, Danhang Tang, and Ulrich Neumann. Gaussianflow: Splatting gaussian dynamics for 4d content creation. *arXiv preprint arXiv:2403.12365*, 2024.

[22] Zhiyang Guo, Wengang Zhou, Li Li, Min Wang, and Houqiang Li. Motion-aware 3d gaussian splatting for efficient dynamic scene reconstruction. *arXiv preprint arXiv:2403.11447*, 2024.

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

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

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

[26] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5438–5448, 2022.

[27] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2021.

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

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

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

[31] Ruijie Zhu, Jiahao Chang, Ziyang Song, Jiahuan Yu, and Tianzhu Zhang. Tiface: Improving facial reconstruction through tensorial radiance fields and implicit surfaces. *arXiv preprint arXiv:2312.09527*, 2023.

[32] Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization. *arXiv preprint arXiv:2403.06912*, 2024.

[33] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. *arXiv preprint arXiv:2403.14627*, 2024.

[34] David Charatan, Sizhe Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction, 2024.

[35] Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[36] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023.

[37] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22563–22575, 2023.

[38] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. *arXiv preprint arXiv:2402.05054*, 2024.

[39] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolific-dreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *arXiv preprint arXiv:2305.16213*, 2023.

[40] Xu Yinghao, Shi Zifan, Yifan Wang, Chen Hansheng, Yang Ceyuan, Peng Sida, Shen Yujun, and Wetzstein Gordon. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation, 2024.

[41] 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, 2023.

[42] Jiemin Fang, Junjie Wang, Xiaopeng Zhang, Lingxi Xie, and Qi Tian. Gaussianeditor: Editing 3d gaussians delicately with text instructions. In *CVPR*, 2024.

[43] Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suya You, Zhangyang Wang, and Achuta Kadambi. Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields. *arXiv preprint arXiv:2312.03203*, 2023.

[44] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. In *CVPR*, 2024.

[45] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat, track & map 3d gaussians for dense rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[46] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian Splatting SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[47] Huajian Huang, Longwei Li, Cheng Hui, and Sai-Kit Yeung. Photo-slam: Real-time simultaneous localization and photorealistic mapping for monocular, stereo, and rgb-d cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[48] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14304–14314. IEEE Computer Society, 2021.

[49] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021.

[50] Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking everything everywhere all at once. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19795–19806, 2023.

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

[52] 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*, pages 9421–9431, 2021.

[53] Chaoyang Wang, Ben Eckart, Simon Lucey, and Orazio Gallo. Neural trajectory fields for dynamic novel view synthesis. *arXiv preprint arXiv:2105.05994*, 2021.

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

[55] Kaichen Zhou, Jia-Xing Zhong, Sangyun Shin, Kai Lu, Yiyuan Yang, Andrew Markham, and Niki Trigoni. Dynpoint: Dynamic neural point for view synthesis. *Advances in Neural Information Processing Systems*, 36, 2023.

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

[57] 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.

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

[59] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022.

[60] Kai Katsumata, Duc Minh Vo, and Hideki Nakayama. An efficient 3d gaussian representation for monocular/multi-view dynamic scenes. *arXiv preprint arXiv:2311.12897*, 2023.

[61] Zhicheng Lu, Xiang Guo, Le Hui, Tianrui Chen, Ming 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*, 2024.

[62] Devikalyan Das, Christopher Wewer, Raza Yunus, Eddy Ilg, and Jan Eric Lenssen. Neural parametric gaussians for monocular non-rigid object reconstruction. *arXiv preprint arXiv:2312.01196*, 2023.

[63] 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. *arXiv preprint arXiv:2403.01444*, 2024.

[64] Lin Yen-Chen, Pete Florence, Jonathan T Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. inerf: Inverting neural radiance fields for pose estimation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1323–1330. IEEE, 2021.

[65] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf–: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021.

[66] Yitong Xia, Hao Tang, Radu Timofte, and Luc Van Gool. Sinerf: Sinusoidal neural radiance fields for joint pose estimation and scene reconstruction. *arXiv preprint arXiv:2210.04553*, 2022.

[67] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5741–5751, 2021.

[68] Shin-Fang Chng, Sameera Ramasinghe, Jamie Sherrah, and Simon Lucey. Garf: Gaussian activated radiance fields for high fidelity reconstruction and pose estimation. *arXiv e-prints*, pages arXiv–2204, 2022.

[69] Wenjing Bian, Zirui Wang, Kejie Li, Jia-Wang Bian, and Victor Adrian Prisacariu. Nope-nerf: Optimising neural radiance field with no pose prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4160–4169, 2023.

[70] Yang Fu, Sifei Liu, Amey Kulkarni, Jan Kautz, Alexei A Efros, and Xiaolong Wang. Colmap-free 3d gaussian splatting. *arXiv preprint arXiv:2312.07504*, 2023.

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

[72] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian Splatting SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[73] Zhiwen Yan, Chen Li, and Gim Hee Lee. Nerf-ds: Neural radiance fields for dynamic specular objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8285–8295, 2023.

[74] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[75] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[76] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[77] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3), 2022.

[78] Lingtong Kong and Jie Yang. Mdflow: Unsupervised optical flow learning by reliable mutual knowledge distillation. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(2):677–688, 2022.

[79] Ruijie Zhu, Chuxin Wang, Ziyang Song, Li Liu, Tianzhu Zhang, and Yongdong Zhang. Scaledepth: Decomposing metric depth estimation into scale prediction and relative depth estimation. *arXiv preprint arXiv:2407.08187*, 2024.

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

A Appendix

A.1 Formulation of Gaussian Flow
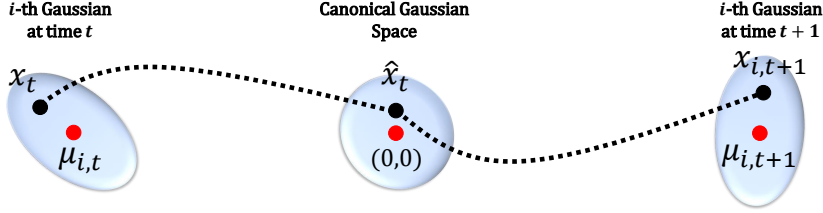
$$F^G_{i,t+1} = x_{i,t+1} - x_t$$

Figure 9: **The formulation of Gaussian flow.** We first project the point x_t corresponding to the i-th Gaussian at time t into the canonical Gaussian space, and then reproject this point from the canonical Gaussian space to the i-th Gaussian at time $t + 1$.

Motivated by [21], we formulate the Gaussian flow $F^G_{t \to t+1}$ to simulate the motion of dynamic object in the scene, as shown in Figure 9 Specifically, Gaussian flow corresponds to the deformation of 3D Gaussians from time t to $t + 1$ in the camera viewpoint C_{t+1}. For point x_t, we first transform it to the canonical space ^{3} corresponding to the Gaussian at time t:

$$\hat{x}_t = \Sigma^{-1}_{i,t}(x_t - \mu_{i,t}), \tag{10}$$

where $\mu_{i,t}$ and $\Sigma_{i,t}$ are the center position and covariance matrix of i-th Gaussian at the timestamp t. Then we transform $\hat{x}_t$ back to the Gaussian at the next time step $t + 1$:

$$x_{i,t+1} = \Sigma_{i,t+1}\hat{x}_t + \mu_{i,t}, \tag{11}$$

where $\mu_{i,t+1}$ and $\Sigma_{i,t+1}$ are the center position and covariance matrix of i-th Gaussian at the timestamp $t + 1$. Therefore, the flow contribution from i-th Gaussian to this point can be defined as:

$$F^G_{i,t \to t+1} = x_{i,t+1} - x_t. \tag{12}$$

Finally, all Gaussian flow contributions to the point can be accumulated in a similar way to α-blending:

$$F^G_{t \to t+1} = \sum_{i=1}^{K} w_i F^G_{i,t \to t+1} \tag{13}$$

$$= \sum_{i=1}^{K} w_i (x_{i,t+1} - x_t), \tag{14}$$

where w_i is the weight of α-blending. Note that since the computation of forward optical flow is referenced to frame I_t, the Gaussian flow should be consistently rendered under the camera viewpoint C_t, corresponding to the decoupled motion flow. It represents the 2D splatting of the Gaussian deformation field from time t to $t + 1$ when the camera viewpoint remains unchanged.

A.2 More Implementation Details

Datasets. The NeRF-DS dataset consists of eight stereo camera video sequences of daily scenes. These scenes contain high-speed moving high-gloss surface objects and changing camera poses, which pose challenges for dynamic scene modeling. The HyperNeRF dataset includes additional complications such as topological changes and inaccurate camera poses. For the NeRF-DS dataset, we use the default resolution 480×270 for all scenes for training and testing. We train the model using images from the left camera and test it on the right camera. For the HyperNeRF dataset, we select four sets of scenes in the vrig subset (3D Printer, Chicken, Broom, and Banana) for training and testing, with $2\times$ downsampling resolution of 536×960.

---

^{3}The canonical space mentioned here should not be confused with the canonical space in the Gaussian deformation field. It represents the standard Gaussian distribution space.

Table 4: **Training time comparison across different models.**

Training Time	As	Basin	Bell	Cup	Plate	Press	Sieve
Baseline	1h 1m	1h 11m	1h 42m	1h 3m	1h 0m	0h 51m	0h 57m
Ours (w/o pose refinement)	1h 8m	1h 15m	1h 53m	1h 13m	1h 6m	0h 58m	1h 3m
Ours	1h 33m	1h 46m	2h 4m	1h 34m	1h 30m	1h 17m	1h 25m
NeRF-DS	6h 43m	6h 48m	6h 49m	6h 50m	6h 53m	6h 48m	6h 47m

Table 5: **Max GPU memory usage comparison across different models.**

Max GPU Memory (GB)	As	Basin	Bell	Cup	Plate	Press	Sieve
Baseline	15.67	13.61	15.97	15.29	9.66	10.65	12.17
Ours	16.61	14.52	17.73	15.70	10.62	11.62	12.97

Table 6: **FPS, number of 3D Gaussians and storage on the NeRF-DS dataset per scene.**

Scene	FPS	Nums	Storage
AS	50	178K	49M
Basin	29	250K	70M
Bell	19	379K	97M
Cup	35	200K	54M
Plate	32	220K	58M
Press	45	197K	53M
Sieve	35	200K	54M

Implementation details. To achieve the differentiable Gaussian flow and differentiable camera pose, we integrate the forward and backward processes into our rasterizer. To provide reliable optical flow, we choose GMFlow [20] as the default optical flow network. In order to stabilize the initialization process of the scene, we introduce motion constraints only after the Gaussian starts to deform and move. In the NeRF-DS dataset, we set the weight of the flow loss λ to 0.5; while in the HyperNeRF scene, it is set to 0.1. Camera pose optimization is also only activated during the Gaussian deformation stage.

Data sampling mechanism. We adopt the same data sampling strategy as the baseline method, *i.e.*, reading image sequences in a randomly shuffled order. For an N-frame video, the frames are shuffled and then read sequentially. In each iteration, we read two frames and calculate the optical flow between them. To enhance efficiency, the second image from the last iteration is used as the first image in the current iteration. Thus, except for the first iteration, only one new image is read in each subsequent iteration. Consequently, there are $N - 1$ iterations per epoch, with optical flow computed once in each iteration. This strategy balances the introduction of accurate motion priors with maintaining training efficiency. During the first epoch of training, we calculate the optical flow for all adjacent frame pairs, resulting in a total of $N - 1$ optical flow maps. In subsequent epochs, we do not reshuffle the image sequence, allowing us to reuse the optical flow maps calculated in the first epoch. This effectively eliminates the need to recompute optical flow maps in each epoch, significantly reducing computational overhead.

Training time and GPU memory. For model training, we list the training time per scene and peak memory usage on the NeRF-DS dataset as shown in Tables 4 and 5, providing a comprehensive assessment of resource usage during training. Compared to our baseline, our approach incurs increased training time and peak memory usage. This is primarily due to the additional rendering of Gaussian flow and the refinement of camera poses, which are necessary for our method.

FPS, number of 3D Gaussians and storage. We provide statistics of FPS and number of Gaussians on NeRF-DS dataset, as shown in Table 6. In most scenes of NeRF-DS, our method MotionGS achieves real-time rendering (FPS>30).

Table 7: **Ablations on other choices of our proposed framework.** For fair comparison, we do not activate the proposed camera pose refinement module during training.

Row	Setting	PSNR ↑	SSIM ↑	LPIPS ↓
1	Baseline	23.61	0.8394	0.1970
2	w/o Motion mask	23.13	0.8242	0.2249
3	Different depth choice (Midas [77])	23.58	0.8384	0.1969
4	Different optical flow network (FlowFormer [19])	23.97	0.8525	0.1893
5	Different optical flow network (MDFlow [78])	23.25	0.8308	0.2137
6	Self-supervised flow supervision loss	23.76	0.8474	0.1807
7	Lower flow loss weight ($\lambda = 0.2$)	23.46	0.8343	0.2042
8	Higher flow loss weight ($\lambda = 0.8$)	23.75	0.8470	0.1819
9	Ours (w/o camera pose refinement)	24.12	0.8609	0.1763

A.3   More Ablations

We summarize the ablations on other choices of our proposed framework in Table 7. For fair comparison, we do not activate the proposed camera pose refinement module during training, since it also influences the flow calculation. Our interpretation and analysis of the ablations are as follows.

Effectiveness of motion mask (row 2).   Introducing a motion mask allows the motion flow to focus on the motion of dynamic objects, thereby reducing interference from static areas. When the motion mask is removed, the performance declines. We attribute this degradation to inaccurate optical flow in the background areas, which introduces errors in the motion guidance and subsequently leads to incorrect Gaussian deformations.

Different depth choice (row 3).   Estimating the accurate depth maps for depth warping is a critical issue when calculating camera flow. We find that using depth prediction from offline estimator Midas [77] yields suboptimal results. This approach degrades the quality of subsequent motion flow, reducing the accuracy of motion constraints and ultimately impacting the reconstruction quality. We attribute this degradation to the inherent scale ambiguity [79] in the depth estimator, as shown in Figure 10. In contrast, using rendered depth by 3DGS ensures scale and geometric consistency and provides superior detail.

Figure 10: **Rendered depth from 3D Gaussian splatting (ours) and off-the-shelf monocular depth estimator (MiDas).** Our rendered depth has richer details and is scale-aligned with the scene. MiDas rendered depth is usually more smooth and suffers from scale ambiguity.

Different optical flow network (row 4-5).   Our method relies on existing 2D optical flow estimators to provide motion guidance for the 3D Gaussian fields. The choice of optical flow prior can lead to performance differences. When we replace GMFlow [20] with another supervised method FlowFormer [19], the performance deteriorates. This is mainly due to the fact that FlowFormer performs inadequately in the "plate" scene, resulting in an overall performance decrease. Additionally, when we replace GMFlow [20] with a self-supervised method MDFlow [78], the performance is

17

even worse. This phenomenon may also illustrates the importance of accurate motion priors, while erroneous or noisy motion constraints may even have a negative effect on the optimization.

Self-supervised flow supervision loss (row 6). Inspired by self-supervised optical flow estimation methods, we attempt to provide motion priors in a self-supervised manner. Specifically, we estimate the Gaussian flow corresponding to the optical flow and use it to warp the I_t frame. We then compute the photometric loss with the I_{t+1} frame. As shown in the table, this method outperforms our baseline but is less effective compared to our proposed method. We hypothesize that the discrepancy arises because the self-supervised loss may not provide accurate supervision in areas with similar colors. Nevertheless, it is evident that employing self-supervised optical flow loss can reduce dependence on off-the-shelf optical flow estimation. When an optical flow estimation network is either unavailable or inaccurate, this approach can serve as a valuable alternative to improve rendering quality.

Different flow loss weights (row 7-8). We compare the rendering performance under different flow loss weights. The results indicate that the selected weight ($\lambda = 0.5$) achieves the best rendering quality. We speculate that excessively large loss weights may disrupt the original optimization process based on rendering losses, while too small weights may result in insufficient motion guidance.

A.4 More Visualizations

Please refer to Figures 12, 13 and 14.

A.5 Limitation

| Ground Truth | Ours – Motion flow supervised | Ours – Optical flow supervised |

Figure 11: **Failure case in DyNeRF dataset.** Since the viewpoints are fixed and sparse, neither motion flow nor optical flow can help our method avoid floating artifacts.

During our experiments, we identify several unresolved issues. Specifically, when applying our method to the DyNeRF dataset [80], we encounter significant challenges as illustrated in Figure 11. Upon further analysis, we find that the fixed and sparse camera viewpoints in the DyNeRF dataset hinder accurate depth rendering, affecting subsequent camera flow calculations and leading to artifacts. The inaccuracies in motion flow primarily comes from the inaccuracy of the camera flow, rather than a failure of the optical flow estimation itself. It is also important to clarify that the DyNeRF dataset is not continuous monocular video but rather dynamic scenes with sparse viewpoints, which posed challenges to the canonical 3D Gaussian initialization. Moving forward, our focus will be on addressing these issues to further improve the robustness of our model in dynamic scene reconstruction. We aim to develop more stable and reliable motion priors and adapt our approach to handle scenarios with minimal object movement more effectively. By doing so, we hope to extend the applicability and reliability of our method across a wider range of dynamic scenes.

A.6 Broader Impacts

To the best of our knowledge, the proposed method will not have significant negative social impact. The proposed dynamic reconstruction method can be used to reconstruct and render some daily dynamic scenes. Users can use the video shot by their mobile phones as input to obtain an explicit 3D asset represented by a 3D Gaussian and a deformation field. This 3D asset can be used for subsequent editing, development, secondary creation for entertainment.

A.7 Data Availability

The datasets that support the findings of this study are available in the following repositories: NeRF-DS [73] at `https://github.com/JokerYan/NeRF-DS/releases/tag/v0.1-pre-release` under Apache-2.0 license, HyperNeRF [49] at `https://github.com/google/hypernerf/releases/tag/v0.1` under Apache-2.0 license. The code of our baseline [17] is available at `https://github.com/ingra14m/Deformable-3D-Gaussians` under MIT license.
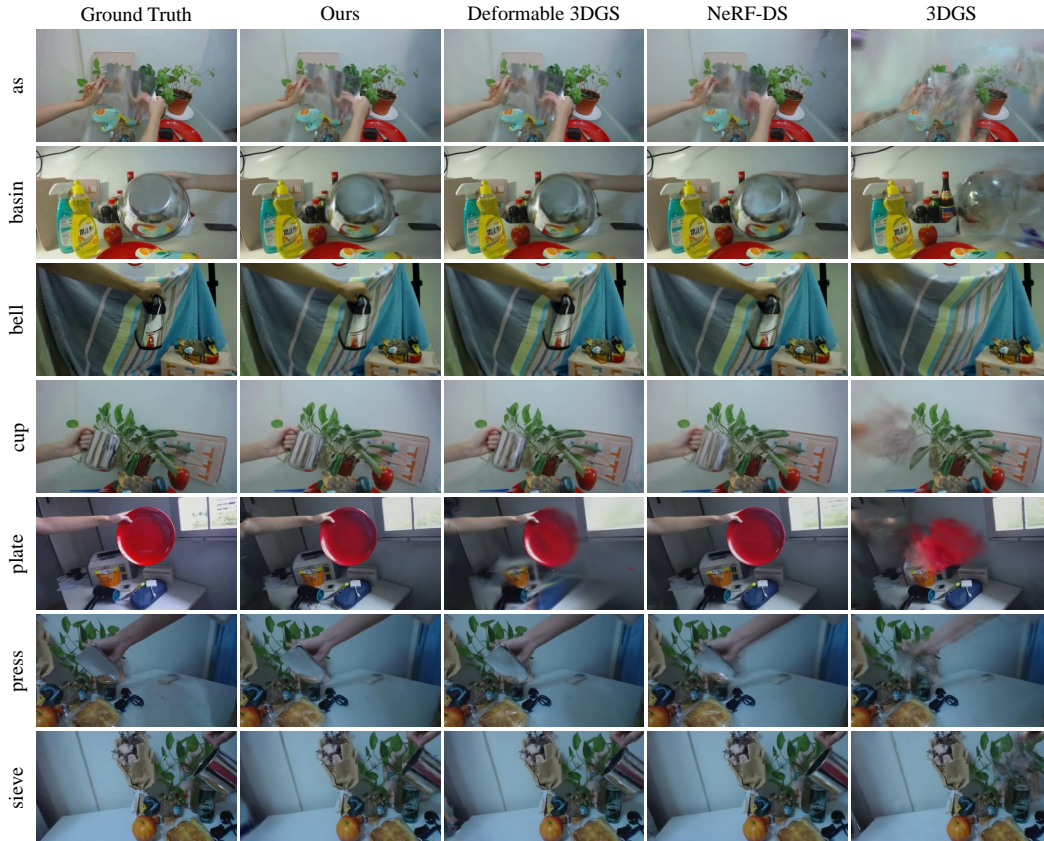

Figure 12: **Qualitative comparison on NeRF-DS dataset per-scene.** Compared with the state-of-the-art methods, our method can render more reasonable details, especially on dynamic objects.
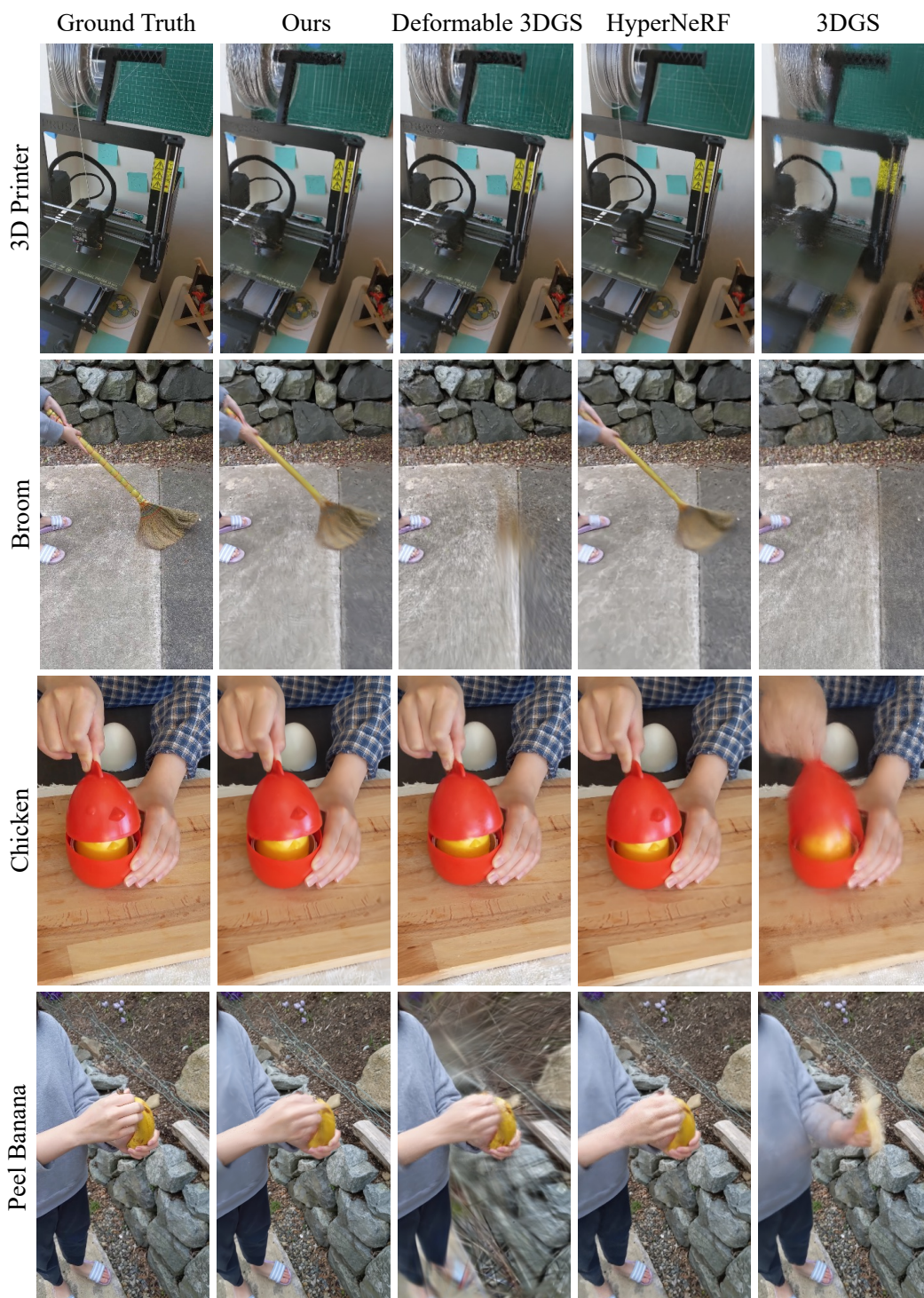
Figure 13: **Qualitative comparison on HyperNeRF dataset per-scene.** Compared with the state-of-the-art methods, our method is more robust in reconstructing dynamic scenes. Even if the input camera pose is not accurate on HyperNeRF dataset, our method can adaptively optimize the camera poses and produce reasonable rendering results.
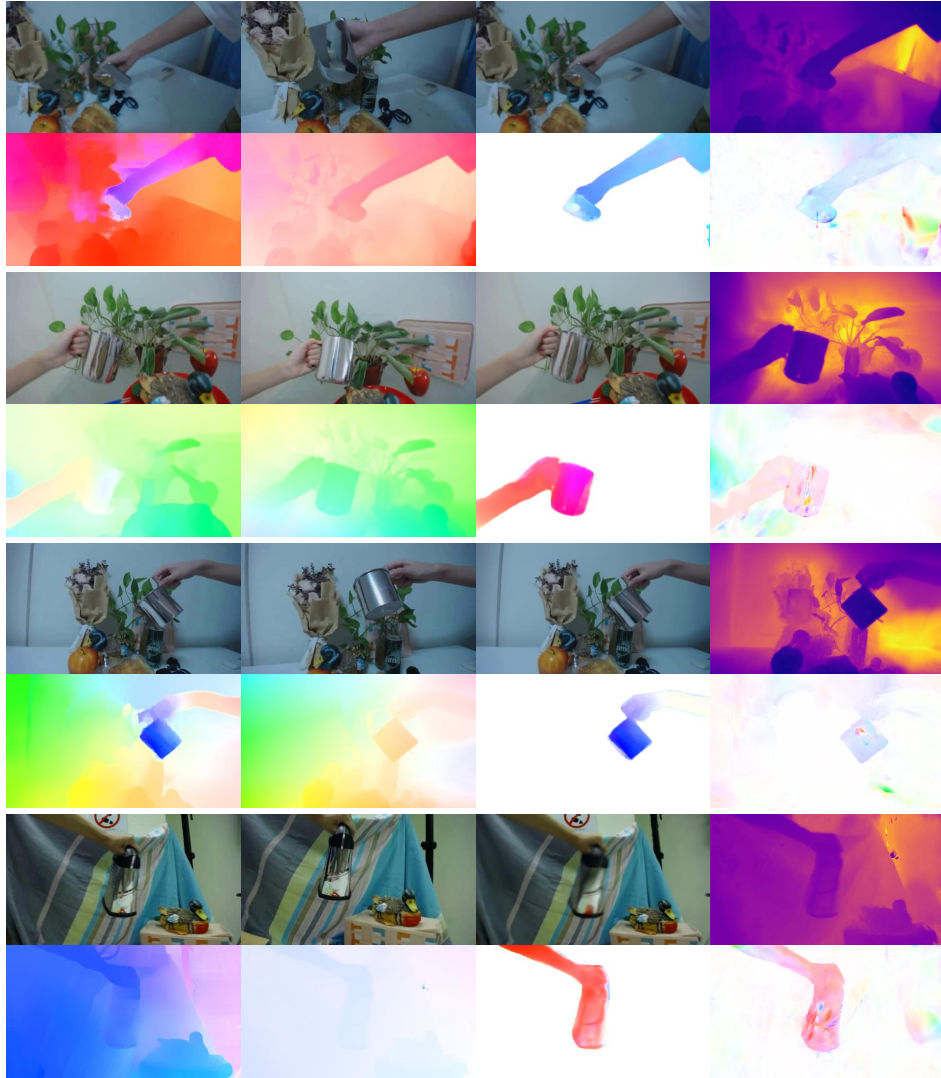
Figure 14: **Visualization of all data flows.** In order: ground truth of I_t, ground truth of I_{t+1}, rendered image of I_t, rendered depth of frame I_t, optical flow, camera flow, motion flow, Gaussian flow.

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: We state the main contributions of this paper and the differences from previous methods in both the abstract and introduction sections.

   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: Please refer to Appendix A.5 for the discussion of limitations and failure cases.

   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: [Yes]

Justification: We provide the necessary formulas and related explanations in the main text. Please see refer to Appendix A.1 for the derivation of Gaussian flow.

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 identify our baseline method and explain the implementation details where we differ from the baseline method in Section 5.1.

   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: [No]

Justification: The datasets and baseline methods in this paper are publicly available. Our source code will be openly accessible after the paper is accepted.

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 have provided necessary implementation details in Section 5.1.

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 follow the experimental settings of previous works to conduct experiments. Therefore, the comparison with previous works is fair.

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 have provided the information of our computer resources in Section 5.1.

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 ensure that our experiments comply with the NeurIPS Code of Ethics in all respects.

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: [Yes]

Justification: Please refer to our discussion on broader impacts in Appendix A.6.

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: The paper poses no such risks for misuse of released data or models.

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 have appropriately cited related work and abide by their licenses and terms. Please refer to Appendix A.7.

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: The paper does not release 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: The paper does not involve crowdsourcing nor research with 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: The paper does not involve crowdsourcing nor research with 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.