Every Camera Effect, Every Time, All at Once: 4D Gaussian Ray Tracing for Physics-based Camera Effect Data Generation

Yi-Ruei Liu[1]*, **You-Zhe Xie**[2]*, **Yu-Hsiang Hsu**[3]*, **I-Sheng Fang**[4]†
Yu-Lun Liu[2], **Jun-Cheng Chen**[4]
[1]University of Illinois Urbana-Champaign
[2]National Yang Ming Chiao Tung University, [3]National Central University
[4]Research Center for Information Technology Innovation, Academia Sinica

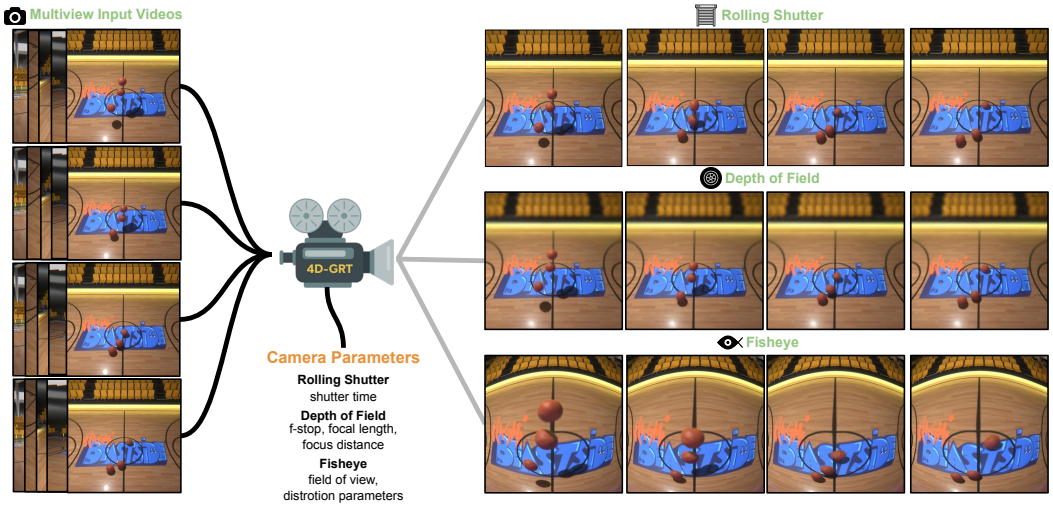

Figure 1: We propose 4D Gaussian Ray Tracing (4D-GRT), a novel framework for generating physically-accurate, controllable camera effects in dynamic scenes. (1) Given multi-view video input, our method reconstructs a dynamic scene using 4D Gaussian Splatting (4D-GS) and differentiable ray tracing. (2) We simulate various camera effects with controllable parameters using ray tracing, generating high-quality videos with controllable camera effects.

Abstract

Common computer vision systems typically assume ideal pinhole cameras but fail when facing real-world camera effects such as fisheye distortion and rolling shutter, mainly due to the lack of learning from training data with camera effects. Existing data generation approaches suffer from either high costs, sim-to-real gaps or fail to accurately model camera effects. To address this bottleneck, we propose 4D Gaussian Ray Tracing (4D-GRT), a novel two-stage pipeline that combines 4D Gaussian Splatting with physically-based ray tracing for camera effect simulation. Given multi-view videos, 4D-GRT first reconstructs dynamic scenes, then applies ray tracing to generate videos with controllable, physically accurate camera effects.

*Equal contribution. Work done at Academia Sinica as intern.
†Internship mentor.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: NeurIPS 2025 Workshop on Space in Vision, Language, and Embodied AI.

4D-GRT achieves the fastest rendering speed while performing better or comparable rendering quality compared to existing baselines. Additionally, we construct eight synthetic dynamic scenes in indoor environments across four camera effects as a benchmark to evaluate generated videos with camera effects. Project page: https://shigon255.github.io/4DGRT-project-page.

Figure 2: We evaluate several state-of-the-art video generation models by specifying camera parameters in prompts to generate videos with specific effects. The results show that these models fail to generate physically accurate videos, instead producing artifacts or incorrect effects. Please refer to the supplementary materials for details.

1 Introduction

In the real world, various camera effects are usual phenomena that fundamentally characterize how cameras serve as the visual interface between the physical world and the digital world. For example, fisheye cameras capture substantially more scene content than conventional perspective cameras, providing additional context across a wider range of viewing directions.

However, common computer vision systems only consider pinhole cameras, rather than leveraging the beneficial properties of camera effects, because of the lack of high-quality visual data paired with accurate camera-effect parameters. As a result, these systems could fail when tested with these common and unavoidable camera effects in real-world scenarios. [27, 19, 18, 73] This data scarcity issue becomes a significant obstacle for modeling these effects, particularly in dynamic scenes where they interact non-trivially with motion (e.g., rolling shutter depends on object and scene dynamics [91, 64]). To overcome this issue, a straightforward approach is to synthesize data with camera effects. Traditional methods rely on physics-based rendering engines such as Blender [7], but they are labor intensive, costly, and can suffer from a pronounced sim-to-real gap [55, 70]. More recently, world models that are largely built on pretrained video generation models [2, 30, 35, 36] have been promoted as a path to realistic, controllable data. Unfortunately, as shown in Fig. 2, off-the-shelf video generation models have limited understanding of camera effects and their corresponding parameters, often generating videos that violate physical principles. These findings suggest that such world models underfit visual data paired with accurate camera-effect parameters, creating a chicken-and-egg problem.

The failure of these approaches motivates us to generate such video data by direct 4D reconstruction. With a reconstructed 4D representation, we can eliminate both the high cost of real-world data collection and the sim-to-real gap. However, existing reconstruction-based methods present significant shortcomings for this task. While Dynamic Gaussian Splatting methods [50, 84, 77] excel at dynamic scene reconstruction, they lack the capability to simulate light transport necessary for camera effect modeling. Conversely, Dynamic NeRF approaches [60, 58, 25, 59, 23, 21, 10] can model light transport but suffer from prohibitively slow rendering speed, along with inferior reconstruction quality compared to Gaussian Splatting methods [40].

To address these limitations, we propose 4D Gaussian Ray Tracing (4D-GRT), a novel two-stage data generation pipeline that combines the reconstruction capabilities of Gaussian Splatting with physically-based ray tracing for camera effect simulation. To the best of our knowledge, 4D-GRT is the first work to perform ray tracing on dynamic scenes using a Gaussian scene representation. In the first stage, given multi-view video captured by pinhole cameras, we optimize a 4D Gaussian Splatting (4D-GS) representation using differentiable ray tracing. In the second stage, given camera effect parameters, we use ray tracing to simulate different camera effects on the reconstructed scene. This approach enables controllable and physically grounded camera effects generation, while achieving the fastest generation speed and better or comparable generation quality compared to baselines.

Our contributions are summarized as follows:

- **Problem Identification:** Current video generation approaches are limited in understanding or generating videos with accurate camera effects given numerical parameters.
- **4D-GRT Pipeline:** We propose a novel data generation pipeline that integrates 4D-GS with ray tracing to rapidly generate high-quality videos with controllable camera effects.
- **Benchmark Dataset:** We construct and release a comprehensive paired dataset containing 8 dynamic indoor scenes with multi-view videos under four camera effects.

2 Related Work

2.1 Generating Visual Data with Camera Effect

The generation of visual data with realistic camera effects has become increasingly crucial for training robust computer vision models. Unlike ideal pinhole camera models commonly assumed in computer vision, real cameras exhibit various optical phenomena causing different camera effects.

Synthetic Data Generation Approaches. Traditional approaches for generating camera effect data rely heavily on synthetic rendering pipelines using computer graphics software such as Blender [7]. These methods provide precise control over camera parameters, generating numerically labeled ground truth data. However, such methods face several fundamental limitations: (1) high labor costs for creating realistic scenes, (2) a sim-to-real gap that limits model generalization.

Generative Model Approaches. The emergence of powerful generative models, particularly diffusion-based methods [62, 15, 5, 81, 32, 11, 33, 13, 85], has opened new avenues for camera effect synthesis [19]. These approaches can generate diverse visual content with various camera effects. Recent representative works include Curved Diffusion [71], AKiRa [74], which enable control over optical distortions or camera effect parameters. However, generative methods suffer from several critical limitations for training data generation: (1) most focus on specific camera effects rather than general effect synthesis, (2) lack of precise physical constraints leading to unrealistic camera effects, (3) inability to accurately control camera parameters, and (4) difficulty in ensuring multi-view consistency or temporal consistency in dynamic scenes.

The above limitations motivate the need for approaches that can generate physically accurate visual data with camera effects in dynamic scenes. Our method is designed to address these challenges.

2.2 Dynamic Neural Rendering

Neural rendering has revolutionized scene reconstruction. This progression provides the necessary foundation for physically-based camera effect simulation.

NeRF. Neural Radiance Fields (NeRF) [51] introduced a new paradigm for representing static scenes, modeling them as continuous volumetric functions with MLPs that map 3D coordinates and viewing directions to density and color for photorealistic view synthesis. Numerous extensions [3, 89, 87, 54, 69, 4, 68, 47] address NeRF's limitations, including adaptations for dynamic scenes [60, 58, 59, 25, 20, 49, 82, 67, 24, 10, 72, 12]. Among these, K-Planes [23], HexPlane [10], and MSTH [72] achieve state-of-the-art performance by leveraging plane- or grid-based representations to accelerate dynamic scene reconstruction while maintaining high rendering quality.

Gaussian Splatting. Despite NeRF's impressive results, its high computational cost and implicit representation limit practical use. 3D Gaussian Splatting (3D-GS)[40] addresses these issues with an explicit 3D Gaussian [39] representation and efficient rasterization, enabling real-time, high-quality

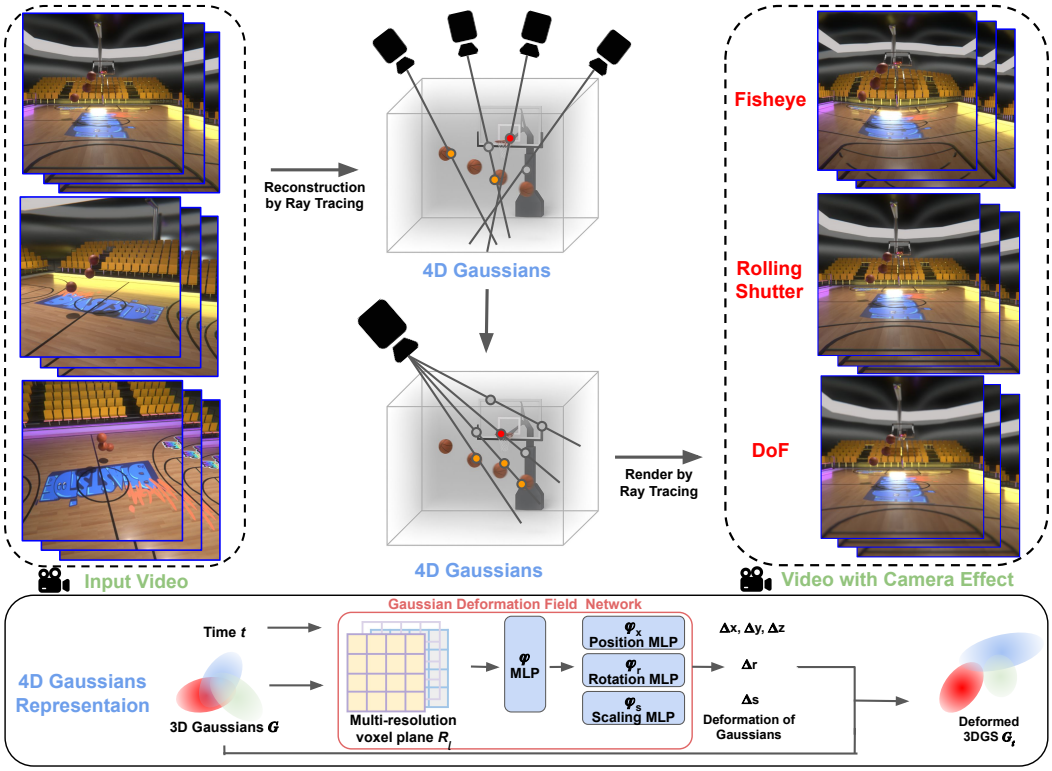

Figure 3: The overall pipeline of our model. Given multi-view videos, we optimize the 4D-GS representation through differentiable ray tracing. Then, given camera effect parameters, we can utilize ray tracing to render videos with physically-correct camera effects.

rendering [34, 88]. Recent works[84, 78, 45, 37, 48, 42, 83, 17] extend 3D-GS to dynamic scenes. For example, Deformable 3D Gaussians [84] introduce deformation modeling, while 4D Gaussian Splatting (4D-GS) [78] employs a spatial-temporal encoder composed of multi-resolution voxel planes and MLPs to model the deformation of 3D Gaussians. These advances preserve the efficiency of Gaussian Splatting while enabling high-quality dynamic scene reconstruction, establishing a strong foundation for camera effect simulation.

2.3 Camera effect rendering in neural scene representations

Traditional renderers simulate camera effects through ray tracing [14], reproducing physical optics. In neural scene representations, NeRF can achieve similar effects [80, 56] due to its ray-casting–based rendering, but NeRF-based methods are computationally expensive and lack efficient rendering. In contrast, 3D-GS offers fast rasterization but is restricted to the pinhole camera model, requiring specialized designs and complex Jacobian computations to support effects such as fisheye distortion [46], rolling shutter with motion blur [65], and depth of field [43, 75, 66]. 3DGRT [53] overcomes this limitation by enabling direct ray tracing over 3D Gaussians, allowing physically accurate and flexible camera effect simulation, while 3DGUT [79] achieves similar capabilities by approximating 3D Gaussians with sigma points via the unscented transform and projecting them through different camera models. However, both are restricted to static scenes and cannot simulate camera effects in dynamic settings. To address this, our method integrates 4D-GS with 3DGRT, leveraging ray tracing to enable camera effect rendering in dynamic scenes.

3 Method

As illustrated in Fig.3, our proposed 4D Gaussian Ray Tracing (4D-GRT) framework generates physically correct training data in two stages: dynamic scene reconstruction (Section 3.2) and camera-effect rendering (Section 3.3). In the first stage, we reconstruct a dynamic scene from synchronized

multi-view videos using a 4D-GS [77] model with differentiable ray tracing [52]. In the second stage, we render the scene with physically based camera models, including fisheye distortion, depth of field, and rolling shutter, producing physically accurate videos for effect-aware vision training.

3.1 Preliminary

3D Gaussian Splatting (3D-GS). 3D-GS [40] represents a static 3D scene as a collection of 3D Gaussians, each characterized by its mean position (x, y, z), a covariance matrix $\mathbf{\Sigma}$ decomposed into scaling s and rotation r parameters, opacity σ, and spherical harmonic coefficients $\mathcal{C}$. Rendering is done by differentiable splatting [86], where $\mathbf{\Sigma}$ is projected into 2D as $\mathbf{\Sigma}' = \mathbf{JW\Sigma W}^T\mathbf{J}^T$, with $\mathbf{J}$ and $\mathbf{W}$ denoting the Jacobian of the projective transformation and the viewing transformation, respectively. Pixel colors are then computed via alpha compositing of the projected Gaussians from front to back, $C = \sum_{i\in\mathcal{N}} T_i\alpha_i c_i$, where $T_i = \prod_{j=1}^{i-1}(1-\alpha_j)$ is the transmittance, $\alpha_i = \sigma_i \exp(-\frac{1}{2}\boldsymbol{x}^T\mathbf{\Sigma}'\boldsymbol{x})$ is the opacity, and c_i is the view-dependent color derived from spherical harmonics.

3.2 Dynamic Scene Reconstruction

As shown in Fig. 3, we reconstruct scene dynamics from multi-view videos $\{\hat{C}_{v,t} \mid v \in \mathcal{V}, t \in \mathcal{T}\}$, where $\mathcal{V}$ denotes the set of camera views and $\mathcal{T}$ the set of time indices. Following 4D-GS [77], we represent the scene with a set of 3D Gaussians G and a Gaussian deformation field network, which consists of a spatio-temporal structure encoder and a multi-head Gaussian deformation decoder to model the per-frame deformations of 3D Gaussians. Specifically, to compute the deformation of 3D Gaussians G at time t, we first employ a 4D voxel planes R_l to extract voxel features based on each Gaussian's mean position and time step t. These features are first fused by a lightweight MLP φ, and then passed to separate MLPs $(\varphi_x, \varphi_r, \varphi_s)$ that predict the residuals of the Gaussian attributes $((\Delta x, \Delta y, \Delta z), \Delta r, \Delta s)$. Finally, the attribute residuals are applied to G to obtain the deformed Gaussians G_t. For details of the Gaussian deformation field network, please refer to the supplementary material.

For rendering, although 3D-GS [40] rasterization provides efficient and high-quality results, it is inherently limited in its ability to simulate physically accurate camera effects. In particular, phenomena that require explicit ray-based modeling, such as complex lens distortions or light-transport-dependent optical effects, cannot be faithfully reproduced. To address these limitations, we adopt the differentiable ray tracing framework of 3DGRT [52], which traces rays directly through 3D Gaussian primitives instead of projecting them onto the image plane. By combining a k-buffer hit-based marching scheme with the hardware-accelerated NVIDIA OptiX interface, this approach achieves both the accuracy required for realistic camera-effect rendering and the efficiency necessary for large-scale data generation.

By integrating the dynamic scene model with differentiable ray tracing, we reconstruct dynamic scenes end-to-end via training with a color loss. At each training iteration, we randomly sample a view $v \in \mathcal{V}$ and a time $t \in \mathcal{T}$, obtain the deformed Gaussians G_t, and render them via ray tracing from view v to obtain the image $C_{v,t}$. We then compute the L1 loss between $C_{v,t}$ and the ground-truth image $\hat{C}_{v,t}$, denoted by $\mathcal{L}_1$. Additionally, we apply a grid-based total variation loss $\mathcal{L}_{\text{TV}}$ following [77, 10]. The overall loss function is defined as

$$\mathcal{L} = \mathcal{L}_1(C_{v,t}, \hat{C}_{v,t}) + \mathcal{L}_{\text{TV}}. \tag{1}$$

3.3 Camera Effects Rendering

After reconstructing the dynamic scene, we synthesize common camera effects by integrating physical camera models into a ray-tracing renderer. Specifically, we implement three representative effects: fisheye distortion, depth of field, and rolling shutter.

Fisheye. Fisheye lenses enable extremely wide-angle capture but introduce strong radial distortion. To simulate this effect, we follow Blender [6] and adopt a 4th-degree polynomial radial distortion model, which provides a flexible and physically grounded representation of fisheye lens characteristics. The model is defined as:

$$\theta = k_0 + k_1 r + k_2 r^2 + k_3 r^3 + k_4 r^4, \tag{2}$$

where r is the radial distance from the principal point of the camera sensor, θ represents the polar angle between the optical axis and the direction of the incoming ray, and k_i are the distortion polynomial coefficients that define the specific fisheye lens characteristics.

Given the sensor dimensions and coefficients k_i, we convert each pixel's image coordinates into physical sensor coordinates (x, y) (in millimeters). We then compute the radial distance $r = \sqrt{x^2 + y^2}$, evaluate the polynomial to obtain the polar angle θ, and calculate the azimuthal angle $\phi = \arccos(x/r)$. These angles define the spherical ray direction, which is then passed to the ray tracer for rendering. Compared to prior generative methods, this parameter-based approach follows real lens behavior and preserves consistency across multiple views.

Depth of Field (DoF). DoF controls which parts of a scene appear sharp and which appear blurred, allowing cameras to emphasize subjects at specific depths while de-emphasizing background or foreground regions. We simulate this effect using the model introduced in [14]. For each pixel, given the focus distance f_z and the aperture radius r_a, we first compute the intersection point $\mathbf{p} = \mathbf{o} + f_z\mathbf{d}$ of the ideal pinhole ray with the focal plane at distance f_z. We then jitter the ray origin over a circular aperture of radius r_a by sampling a point $\boldsymbol{\ell} = (\ell_x, \ell_y)$ uniformly from the unit disk and mapping it to world space using the camera's lens-plane basis $(\mathbf{u}, \mathbf{v})$, where $\mathbf{u}$ and $\mathbf{v}$ are orthogonal unit vectors spanning the lens plane. The perturbed ray origin and direction are then computed as

$$\mathbf{o}' = \mathbf{o} + \ell_x\mathbf{u} + \ell_y\mathbf{v}, \qquad \mathbf{d}' = \frac{\mathbf{p} - \mathbf{o}'}{\|\mathbf{p} - \mathbf{o}'\|}. \tag{3}$$

Repeating this process for multiple samples per pixel and averaging their traced radiance yields physically accurate defocus blur, with the blur size determined by the aperture radius and the object's depth relative to the focal plane.

Rolling Shutter. The rolling shutter effect arises because image rows are not captured simultaneously but sequentially over time. As a result, objects in motion or a moving camera can cause geometric distortions in the recorded image. To simulate this effect, we cast a single ray for each pixel through the deformed 3D Gaussians at its corresponding capture time. This approach resembles the motion blur technique of [14], but it differs in that we assume no motion blur: instead of integrating multiple rays over the shutter interval of a row, we trace only one ray per pixel at the corresponding sensing time. Concretely, for a pixel at row r, we first compute the sensing time t_r, then evaluate the deformed 3D Gaussians G_{t_r}. A ray is traced through the deformed 3D Gaussians G_{t_r} to determine the pixel's radiance. We currently assume a static camera; however, the method can be extended to a moving camera by interpolating the camera trajectory at time t_r and casting the ray from the position.

A limitation of this approach is that sensing times differ across rows, preventing full parallelization of ray tracing and thus limiting rendering speed. To address this, we adopt an approximation strategy. Specifically, we divide the rows into chunks of size N_c. Assuming the scene motion is moderate and the shutter time is relatively short, we approximate the sensing time of all rows in a chunk by their average. For each chunk, we first compute the deformed Gaussian at this average sensing time and then trace the rays within the chunk in parallel. This strategy greatly improves rendering efficiency, though an excessively large chunk size may introduce blocking artifacts due to the approximation.

4 Experiments

Datasets. In real-world scenarios, simultaneously capturing dynamic scenes with different camera effects is extremely challenging. Existing public datasets lack multi-view data for the same dynamic scene under different camera effects. To address this limitation, we constructed our own dataset using Blender 4.5 [7], which enabled us to: (1) simulate realistic physics, and (2) consistently reproduce identical dynamic scenes while applying various camera effects.

We constructed 8 distinct dynamic scenes across 4 indoor environments (basketball court, warehouse, living room and bathroom) using high-quality models from BlenderKit [8] and other asset platforms. Each scene features diverse material properties and physical interactions, including bouncing balls, mechanical animations, and wind-influenced motions. For each scene, we established 50 camera viewpoints rendered under four different camera effects (pinhole, fisheye, rolling shutter, depth of field), generating 50-frame videos at 512×512 resolution. Camera placement and point cloud generation were automated using the PlenoBlenderNeRF plugin [29, 61], ensuring consistent geometric

Table 1: Quantitative comparison on pinhole camera rendering.

Methods	PSNR(dB)↑	SSIM↑	LPIPS↓	FPS↑
HexPlane [10]	23.1124	0.7956	0.2942	0.20
MSTH [72]	29.431	**0.9023**	0.1139	9.38
4D-GRT (Ours)	**32.801**	0.8898	**0.1018**	**36.56**

Table 2: Quantitative comparison on depth-of-field effect rendering.

Methods	PSNR(dB)↑	SSIM↑	LPIPS↓	FPS↑
HexPlane [10]	18.3722	0.7343	0.5056	0.01
MSTH [72]	28.4692	0.9009	0.1540	0.57
4D-GRT (Ours)	**31.2475**	**0.9124**	**0.1210**	**3.44**

Table 3: Quantitative comparison of fisheye distortion rendering. Metrics with "_m" indicate that the metric is computed within the pre-defined mask.

Methods	PSNR(dB)↑	SSIM↑	LPIPS↓	PSNR_m (dB)↑	SSIM_m↑	LPIPS_m↓	FPS↑
HexPlane [10]	15.6489	0.6678	0.4142	21.6869	0.7480	0.2726	0.21
MSTH [72]	**24.4222**	**0.8163**	0.1764	26.7930	**0.8571**	**0.1153**	9.38
4D-GRT (Ours)	24.1322	0.8162	**0.1678**	**28.8927**	0.8555	0.1259	**41.53**

Table 4: Quantitative comparison on rolling shutter effect.

Methods	Chunk	PSNR(dB)↑	SSIM↑	LPIPS↓	FPS↑
HexPlane [10]	N/A	21.3468	0.7723	0.3162	0.21
MSTH [72]	N/A	28.7003	**0.8927**	0.1190	9.35
4D-GRT (Ours)	1 row	31.6144	0.8821	**0.1056**	0.76
4D-GRT (Ours)	4 rows	**31.6146**	0.8821	**0.1056**	4.99
4D-GRT (Ours)	16 rows	31.6082	0.8820	**0.1056**	**13.54**

Table 5: Comparison of training time and storage.

Methods	Time↓	Storage (MB)↓
HexPlane [10]	12 hours	**238**
MSTH [72]	**8 mins**	889
4D-GRT (Ours)	3 hours	364.875

foundations across different camera effects. Detailed scene descriptions and technical implementation are provided in the supplementary material.

Baselines. Our task of generating videos with camera effects from dynamic scenes is novel. For fair and meaningful evaluation, we compare against two state-of-the-art dynamic NeRF methods, HexPlane [10] and MSTH [72], selected for their capabilities in 4D scene reconstruction and ray tracing. To ensure consistency, we apply the same camera effect rendering module used in our method to these baselines, eliminating differences in effect generation. We conduct both quantitative and qualitative comparisons.

Evaluation Protocol. All methods are trained on multi-view videos rendered with a pinhole camera on GeForce RTX 4090. For evaluation, we render the same camera viewpoints with different camera effects (fisheye, rolling shutter, and depth of field) on GeForce RTX 4090, and then compute rendering FPS and the metrics against the corresponding ground-truth videos. For both training and evaluation, we use all camera viewpoints in all time frames. To ensure fairness, all methods, including ours, are trained on the identical set of input videos until convergence.

Evaluation Metrics. We employ a set of metrics to assess the visual quality of our rendered videos against the ground-truth data. Specifically, we use Peak Signal-to-Noise Ratio (PSNR) [38], Structural Similarity Index Measure (SSIM) [76], and Learned Perceptual Image Patch Similarity (LPIPS) [90]. We also report training time, inference speed (FPS), and storage for efficiency evaluation. For all metrics, we report the average score over all frames, camera views, and scenes. For the rolling-shutter rendering comparison, we evaluate our method with chunk sizes of $N_c = 1$, 4, and 16.

Quantitative comparison. Tables 1, 2, 3, 4, and 5 present the quantitative results. Across all camera effects including pinhole, fisheye, depth-of-field and rolling-shutter rendering, our method delivers the highest rendering speed. Besides this advantage, the rendering quality of ours remains competitive. For pinhole, depth-of-field, and rolling-shutter rendering, 4D-GRT achieves higher PSNR than both baselines, with SSIM and LPIPS comparable to MSTH. For fisheye rendering, it matches MSTH across all metrics while still outperforming HexPlane. The rendering speedup stems from our hardware-accelerated Gaussian ray tracing, while the quality gains follow from the continuous 4D-Gaussian representation.

In fisheye evaluation, the larger field of view reveals regions not visible in the pinhole cameras used for training. Since these regions are never observed, their appearance is unconstrained and leads to unreliable metrics. To address this, we adopt a masked evaluation strategy. Specifically, a

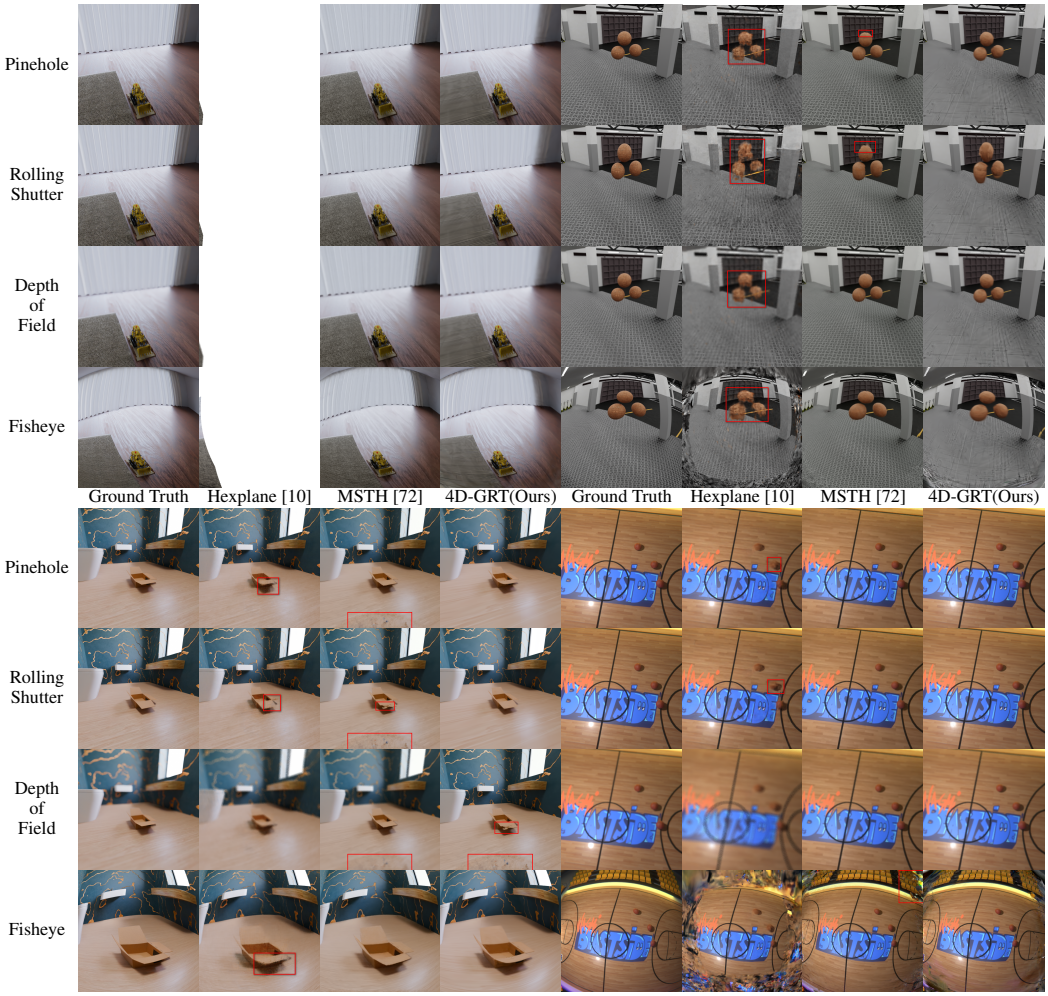

Figure 4: Qualitative comparison on our synthetic datasets. The artifacts are framed by red rectangles

circular mask centered at the principal point is applied to both the rendered and ground-truth frames, restricting the comparison to regions visible to the training cameras. We empirically set the diameter of the circle as 409.6. In this evaluation, as shown in Table 3, our method achieves substantially higher PSNR than the baselines, with comparable SSIM and LPIPS.

Regarding rolling-shutter rendering FPS, our method without approximation is slower than the baselines due to structural differences. Specifically, our approach models the dynamic scene using a set of canonical 3D Gaussians and a Gaussian deformation field network, which requires generating a separate deformed scene for each row in the rolling-shutter image, making parallel ray tracing infeasible. In contrast, the baselines employ plane-based or grid-based 4D representations, allowing direct spatio-temporal queries per ray, benefiting from parallelism for faster rendering. However, as shown in Table 4, we achieve higher FPS with only minimal quality loss with our approximation strategy. Larger chunk sizes yield higher FPS at the cost of slight image quality degradation, while smaller chunk sizes preserve higher rendering quality at the expense of longer rendering times.

In terms of efficiency, HexPlane is the most storage-efficient but slowest to train, while MSTH trains fastest but requires the most storage. Our method achieves the highest reconstruction and rendering quality with fast rendering speed, while keeping both training time and storage at reasonable levels.

Qualitative comparison. For qualitative evaluation, we present the rendering results of 4D-GRT (ours), MSTH [72], and Hexplane [10] in Fig. 4. Our method demonstrates higher-fidelity results compared to others. Specifically, the reconstruction quality of Hexplane [10] is obviously worse

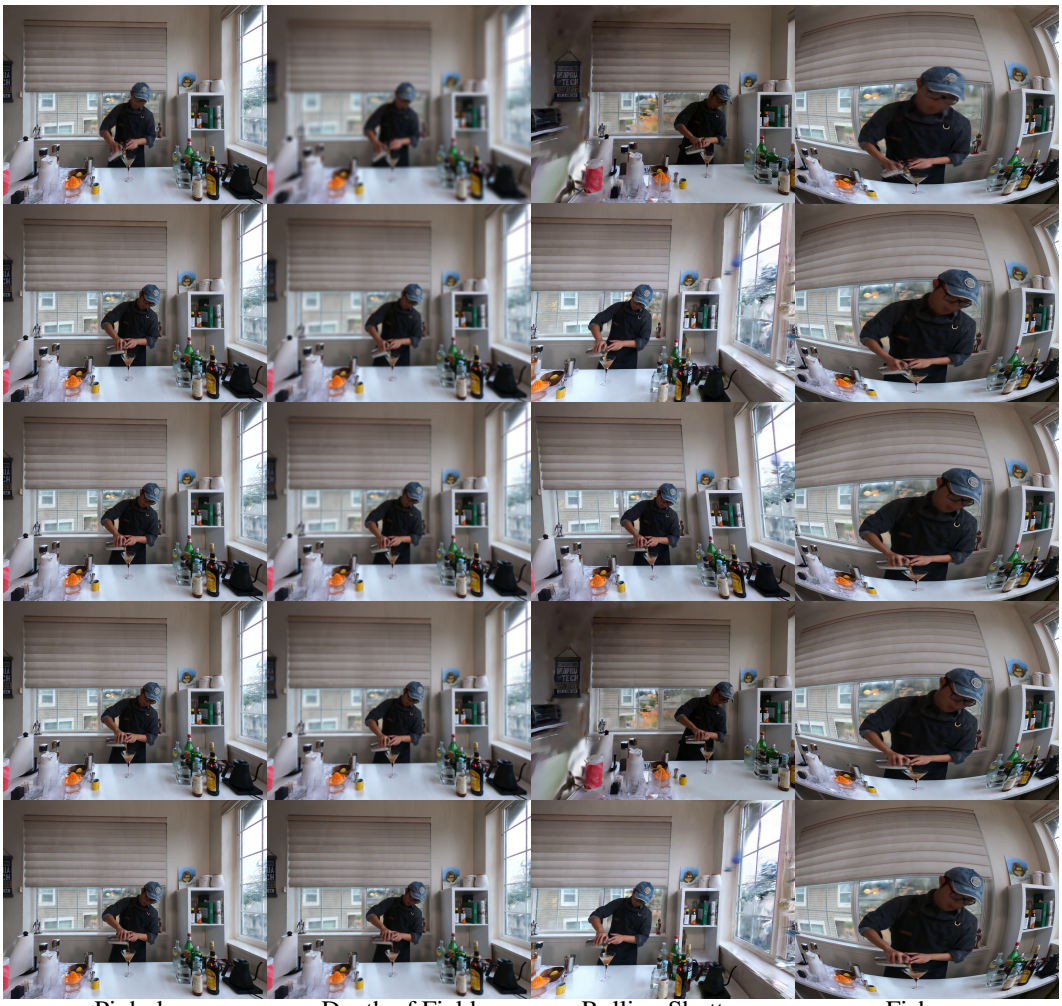

Figure 5: Qualitative results of 4D-GRT on the Neural 3D Video dataset [44].

than ours, and there are reconstruction artifacts and color temperature misalignment in the results of MSTH [72], which do not exist in our results.

Qualitative results on real-world dataset. We further validate on real-world dynamic scenes by reconstructing and rendering sequences from the Neural 3D Video (Plenoptic Video) dataset [44] with camera effects in Fig. 5; more qualitative examples are provided in the supplementary material. In addition, we present results as videos to better demonstrate temporal consistency, and also include rendering results on sequences from our own dataset in video form for visualization of scene dynamics.

5 Conclusion and Limitations

Conclusion. In this work, we identify a critical bottleneck: contemporary world models, built on pretrained video generators, fail to respect camera-effect parameters and thus produce physically inconsistent videos. To address this, we introduced **4D Gaussian Ray Tracing (4D-GRT)**, a two-stage pipeline that couples 4D Gaussian splatting with differentiable ray tracing for controllable, effect-accurate data generation. We also release a benchmark of 8 dynamic indoor scenes with synchronized multi-view videos rendered under four camera effects. Experimentally, 4D-GRT delivers substantially higher rendering speed than strong dynamic NeRF baselines across four types of camera effect while maintaining competitive or better image quality (PSNR/SSIM/LPIPS). This advancement demonstrates that our method is more suitable for generating substantial amounts of

videos with camera effects, which can better address the data insufficiency problem in understanding camera effects across world models and other visual systems. Together, 4D-GRT and our benchmark provide a strong foundation for advancing camera-aware vision.

Limitations. Our method requires well-reconstructed dynamic scenes to render high-quality videos with camera effects, necessitating sufficient multi-view video data. This dependency limits its applicability in scenarios with sparse viewpoints or monocular inputs. Future work could alleviate this limitation by integrating generative or foundation models to reconstruct dynamic scenes from monocular video, making camera effect simulation more accessible across diverse scenarios.

Acknowledgement

This research is supported by National Science and Technology Council, Taiwan (R.O.C), under the grant number of NSTC-114-2221-E-001-016, NSTC-113-2634-F-002-008, NSTC-112-2222-E-A49-004-MY2, NSTC-113-2628-E-A49-023-, and Academia Sinica under the grant number of AS-CDA-110-M09 and AS-IAIA-114-M10.

References

[1] 3dhaupt. Indoor pot plant 2. `https://free3d.com/3d-model/indoor-pot-plant-77983.html`, 2019. Accessed: 2025-08-14, Licensed for Personal Use Only.

[2] Philip J. Ball, Jakob Bauer, Frank Belletti, Bethanie Brownfield, Ariel Ephrat, Shlomi Fruchter, Agrim Gupta, Kristian Holsheimer, Aleksander Holynski, Jiri Hron, Christos Kaplanis, Marjorie Limont, Matt McGill, Yanko Oliveira, Jack Parker-Holder, Frank Perbet, Guy Scully, Jeremy Shar, Stephen Spencer, Omer Tov, Ruben Villegas, Emma Wang, Jessica Yung, Cip Baetu, Jordi Berbel, David Bridson, Jake Bruce, Gavin Buttimore, Sarah Chakera, Bilva Chandra, Paul Collins, Alex Cullum, Bogdan Damoc, Vibha Dasagi, Maxime Gazeau, Charles Gbadamosi, Woohyun Han, Ed Hirst, Ashyana Kachra, Lucie Kerley, Kristian Kjems, Eva Knoepfel, Vika Koriakin, Jessica Lo, Cong Lu, Zeb Mehring, Alex Moufarek, Henna Nandwani, Valeria Oliveira, Fabio Pardo, Jane Park, Andrew Pierson, Ben Poole, Helen Ran, Tim Salimans, Manuel Sanchez, Igor Saprykin, Amy Shen, Sailesh Sidhwani, Duncan Smith, Joe Stanton, Hamish Tomlinson, Dimple Vijaykumar, Luyu Wang, Piers Wingfield, Nat Wong, Keyang Xu, Christopher Yew, Nick Young, Vadim Zubov, Douglas Eck, Dumitru Erhan, Koray Kavukcuoglu, Demis Hassabis, Zoubin Gharamani, Raia Hadsell, Aäron van den Oord, Inbar Mosseri, Adrian Bolton, Satinder Singh, and Tim Rocktäschel. Genie 3: A new frontier for world models. 2025. URL `https://deepmind.google/discover/blog/genie-3-a-new-frontier-for-world-models`.

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

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

[5] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[6] Blender Documentation Team. *The Blender 4.5 Manual*, 2025. URL `https://docs.blender.org/manual/en/latest/`. Licensed under CC-BY-SA v4.0.

[7] Blender Online Community. Blender – a 3d modelling and rendering package. `http://www.blender.org`, 2025. Version 4.5, Accessed: 2025-08-14.

[8] BlenderKit Community. Blenderkit – free 3d assets for blender. `https://www.blenderkit.com`, 2025. Accessed: 2025-08-14.

[9] BlendSwap Community. Blendswap: Free blender 3d models. `https://www.blendswap.com`. Accessed: 2025-08-20.

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

[11] Chen-Wei Chang, Cheng-De Fan, Chia-Che Chang, Yi-Chen Lo, Yu-Chee Tseng, Jiun-Long Huang, and Yu-Lun Liu. Gcc: Generative color constancy via diffusing a color checker. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10868–10878, 2025.

[12] Bo-Yu Chen, Wei-Chen Chiu, and Yu-Lun Liu. Improving robustness for joint optimization of camera pose and decomposed low-rank tensorial radiance fields. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 38, pages 990–1000, 2024.

[13] Ernie Chu, Shuo-Yen Lin, and Jun-Cheng Chen. Video controlnet: Towards temporally consistent synthetic-to-real video translation using conditional image diffusion models. *arXiv preprint arXiv:2305.19193*, 2023.

[14] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. *SIGGRAPH Comput. Graph.*, 18(3):137–145, January 1984. ISSN 0097-8930. doi: 10.1145/964965.808590. URL `https://doi.org/10.1145/964965.808590`.

[15] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(9):10850–10869, 2023.

[16] dailyfree3d. Lowpoly cat rigged run animation. `https://free3d.com/3d-model/lowpoly-cat-rigged-run-animation-756268.html`, 2019. Accessed: 2025-08-14, Licensed for Personal Use Only.

[17] Cheng-De Fan, Chen-Wei Chang, Yi-Ruei Liu, Jie-Ying Lee, Jiun-Long Huang, Yu-Chee Tseng, and Yu-Lun Liu. Spectromotion: Dynamic 3d reconstruction of specular scenes. *arXiv*, 2024.

[18] I-Sheng Fang and Jun-Cheng Chen. Camerabench: Benchmarking visual reasoning in mllms via photography, 2025. URL `https://arxiv.org/abs/2504.10090`.

[19] I-Sheng Fang, Yue-Hua Han, and Jun-Cheng Chen. Camera settings as tokens: Modeling photography on latent diffusion models. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024.

[20] 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*, 2022.

[21] 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.

[22] Free3D. Free3d: Download free 3d models. `https://www.free3d.com`. Accessed: 2025-08-20.

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

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

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

[26] Yu Gao, Haoyuan Guo, Tuyen Hoang, Weilin Huang, Lu Jiang, Fangyuan Kong, Huixia Li, Jiashi Li, Liang Li, Xiaojie Li, Xunsong Li, Yifu Li, Shanchuan Lin, Zhijie Lin, Jiawei Liu, Shu Liu, Xiaonan Nie, Zhiwu Qing, Yuxi Ren, Li Sun, Zhi Tian, Rui Wang, Sen Wang, Guoqiang Wei, Guohong Wu, Jie Wu, Ruiqi Xia, Fei Xiao, Xuefeng Xiao, Jiangqiao Yan, Ceyuan Yang, Jianchao Yang, Runkai Yang, Tao Yang, Yihang Yang, Zilyu Ye, Xuejiao Zeng, Yan Zeng, Heng Zhang, Yang Zhao, Xiaozheng Zheng, Peihao Zhu, Jiaxin Zou, and Feilong Zuo. Seedance 1.0: Exploring the boundaries of video generation models, 2025. URL `https://arxiv.org/abs/2506.09113`.

[27] Yunpeng Gong, Yongjie Hou, Chuangliang Zhang, and Min Jiang. Beyond augmentation: Empowering model robustness under extreme capture environments. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2024.

[28] Google DeepMind. Veo 3 model card. `https://storage.googleapis.com/deepmind-media/Model-Cards/Veo-3-Model-Card.pdf`, 2025. Accessed: 2025-08-19.

[29] Karoline H. Plenoblendernerf: A blender add-on for neural radiance fields. https://github.com/KarolineH/PlenoBlenderNeRF, 2025. Accessed: 2025-08-14.

[30] Haoran He, Yang Zhang, Liang Lin, Zhongwen Xu, and Ling Pan. Pre-trained video generative models as world simulators, 2025. URL https://arxiv.org/abs/2502.07825.

[31] Heinzelnisse. Lego 856 bulldozer. https://www.blendswap.com/blend/11490, 2014. Accessed: 2025-08-14, Licensed under CC-BY-NC 3.0.

[32] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

[33] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

[34] Hao-Yu Hou, Chia-Chi Hsu, Yu-Chen Huang, Mu-Yi Shen, Wei-Fang Sun, Cheng Sun, Chia-Che Chang, Yu-Lun Liu, and Chun-Yi Lee. 3d gaussian splatting with grouped uncertainty for unconstrained images. In *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2025. doi: 10.1109/ICASSP49660.2025.10887619.

[35] Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. Gaia-1: A generative world model for autonomous driving, 2023. URL https://arxiv.org/abs/2309.17080.

[36] Xiaotao Hu, Wei Yin, Mingkai Jia, Junyuan Deng, Xiaoyang Guo, Qian Zhang, Xiaoxiao Long, and Ping Tan. Drivingworld: Constructing world model for autonomous driving via video gpt, 2024. URL https://arxiv.org/abs/2412.19505.

[37] 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.

[38] Quan Huynh-Thu and Mohammed Ghanbari. Scope of validity of PSNR in image/video quality assessment. *Electronics letters*, 44(13):800–801, 2008.

[39] Bo-Hsu Ke, You-Zhe Xie, Yu-Lun Liu, and Wei-Chen Chiu. Stealthattack: Robust 3d gaussian splatting poisoning via density-guided illusions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[40] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

[41] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, Kathrina Wu, Qin Lin, Junkun Yuan, Yanxin Long, Aladdin Wang, Andong Wang, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Hongmei Wang, Jacob Song, Jiawang Bai, Jianbing Wu, Jinbao Xue, Joey Wang, Kai Wang, Mengyang Liu, Pengyu Li, Shuai Li, Weiyan Wang, Wenqing Yu, Xinchi Deng, Yang Li, Yi Chen, Yutao Cui, Yuanbo Peng, Zhentao Yu, Zhiyu He, Zhiyong Xu, Zixiang Zhou, Zunnan Xu, Yangyu Tao, Qinglin Lu, Songtao Liu, Dax Zhou, Hongfa Wang, Yong Yang, Di Wang, Yuhong Liu, Jie Jiang, and Caesar Zhong. Hunyuanvideo: A systematic framework for large video generative models, 2025. URL https://arxiv.org/abs/2412.03603.

[42] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

[43] Jungho Lee, Suhwan Cho, Taeoh Kim, Ho-Deok Jang, Minhyeok Lee, Geonho Cha, Dongyoon Wee, Dogyoon Lee, and Sangyoun Lee. Cocogaussian: Leveraging circle of confusion for gaussian splatting from defocused images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16101–16110, 2025.

[44] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. Neural 3d video synthesis from multi-view video, 2022. URL https://arxiv.org/abs/2103.02597.

[45] Yiqing Liang, Numair Khan, Zhengqin Li, Thu Nguyen-Phuoc, Douglas Lanman, James Tompkin, and Lei Xiao. Gaufre: Gaussian deformation fields for real-time dynamic novel view synthesis. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2025.

[46] Zimu Liao, Siyan Chen, Rong Fu, Yi Wang, Zhongling Su, Hao Luo, Li Ma, Linning Xu, Bo Dai, Hengjie Li, Zhilin Pei, and Xingcheng Zhang. Fisheye-gs: Lightweight and extensible gaussian splatting module for fisheye cameras, 2024. URL https://arxiv.org/abs/2409.04751.

[47] Chin-Yang Lin, Chung-Ho Wu, Chang-Han Yeh, Shih-Han Yen, Cheng Sun, and Yu-Lun Liu. Frugalnerf: Fast convergence for extreme few-shot novel view synthesis without learned priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11227–11238, 2025.

[48] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21136–21145, 2024.

[49] 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 (CVPR)*, 2023.

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

[51] 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.

[52] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3d gaussian ray tracing: Fast tracing of particle scenes. *ACM Transactions on Graphics and SIGGRAPH Asia*, 2024.

[53] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3d gaussian ray tracing: Fast tracing of particle scenes, 2024. URL https://arxiv.org/abs/2407.07090.

[54] 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.

[55] Sergey I Nikolenko et al. *Synthetic data for deep learning*, volume 174. Springer, 2021.

[56] Muyao Niu, Tong Chen, Yifan Zhan, Zhuoxiao Li, Xiang Ji, and Yinqiang Zheng. Rs-nerf: Neural radiance fields from rolling shutter images, 2024. URL https://arxiv.org/abs/2407.10267.

[57] OpenAI. Sora system card. https://openai.com/index/sora-system-card/, 2024. Accessed: 2025-08-19.

[58] 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 (ICCV)*, pages 5865–5874, 2021.

[59] 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.

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

[61] Maxime Raafat. BlenderNeRF, August 2024. URL https://github.com/maximeraafat/BlenderNeRF.

[62] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022.

[63] RonKo. Box animation template. https://www.blendswap.com/blend/30950, 2023. Accessed: 2025-08-14, Licensed under CC-BY 4.0.

[64] Olivier Saurer, Kevin Koser, Jean-Yves Bouguet, and Marc Pollefeys. Rolling shutter stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 465–472, 2013.

[65] Otto Seiskari, Jerry Ylilammi, Valtteri Kaatrasalo, Pekka Rantalankila, Matias Turkulainen, Juho Kannala, and Arno Solin. Gaussian splatting on the move: Blur and rolling shutter compensation for natural camera motion, 2024.

[66] Liao Shen, Tianqi Liu, Huiqiang Sun, Jiaqi Li, Zhiguo Cao, Wei Li, and Chen Change Loy. Dof-gaussian: Controllable depth-of-field for 3d gaussian splatting, 2025. URL https://arxiv.org/abs/2503.00746.

[67] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 29(5):2732–2742, 2023. doi: 10.1109/TVCG.2023.3247082.

[68] Chih-Hai Su, Chih-Yao Hu, Shr-Ruei Tsai, Jie-Ying Lee, Chin-Yang Lin, and Yu-Lun Liu. Boostmvsnerfs: Boosting mvs-based nerfs to generalizable view synthesis in large-scale scenes. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–12, 2024.

[69] 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 (CVPR)*, 2022.

[70] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 969–977, 2018.

[71] Andrey Voynov, Amir Hertz, Moab Arar, Shlomi Fruchter, and Daniel Cohen-Or. Curved diffusion: A generative model with optical geometry control. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 149–164. Springer, 2024.

[72] Feng Wang, Zilong Chen, Guokang Wang, Yafei Song, and Huaping Liu. Masked space-time hash encoding for efficient dynamic scene reconstruction. *Advances in neural information processing systems*, 36:70497–70510, 2023.

[73] Shunxin Wang, Raymond Veldhuis, Christoph Brune, and Nicola Strisciuglio. A survey on the robustness of computer vision models against common corruptions. *arXiv preprint arXiv:2305.06024*, 2023.

[74] Xi Wang, Robin Courant, Marc Christie, and Vicky Kalogeiton. Akira: Augmentation kit on rays for optical video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2609–2619, 2025.

[75] Yujie Wang, Praneeth Chakravarthula, and Baoquan Chen. Dof-gs:adjustable depth-of-field 3d gaussian splatting for post-capture refocusing, defocus rendering and blur removal, 2025. URL https://arxiv.org/abs/2405.17351.

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

[77] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20310–20320, June 2024.

[78] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20310–20320, 2024.

[79] Qi Wu, Janick Martinez Esturo, Ashkan Mirzaei, Nicolas Moenne-Loccoz, and Zan Gojcic. 3dgut: Enabling distorted cameras and secondary rays in gaussian splatting. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[80] Zijin Wu, Xingyi Li, Juewen Peng, Hao Lu, Zhiguo Cao, and Weicai Zhong. Dof-nerf: Depth-of-field meets neural radiance fields. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 1718–1729. ACM, October 2022. doi: 10.1145/3503161.3548088. URL http://dx.doi.org/10.1145/3503161.3548088.

[81] Zhen Xing, Qijun Feng, Haoran Chen, Qi Dai, Han Hu, Hang Xu, Zuxuan Wu, and Yu-Gang Jiang. A survey on video diffusion models. *ACM Computing Surveys*, 57(2):1–42, 2024.

[82] 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 (CVPR)*, pages 8285–8295, 2023.

[83] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting, 2024. URL `https://arxiv.org/abs/2310.10642`.

[84] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20331–20341, 2024.

[85] Po-Hung Yeh, Kuang-Huei Lee, and Jun-Cheng Chen. Training-free diffusion model alignment with sampling demons. *arXiv preprint arXiv:2410.05760*, 2024.

[86] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA)*, 38(6), 2019.

[87] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4578–4587, 2021.

[88] Yu-Ting Zhan, Cheng-Yuan Ho, Hebi Yang, Yi-Hsin Chen, Jui Chiu Chiang, Yu-Lun Liu, and Wen-Hsiao Peng. Cat-3dgs: A context-adaptive triplane approach to rate-distortion-optimized 3dgs compression. *arXiv preprint arXiv:2503.00357*, 2025.

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

[90] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018.

[91] Zhihang Zhong, Yinqiang Zheng, and Imari Sato. Towards rolling shutter correction and deblurring in dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9219–9228, 2021.

Supplementary Material Overview

This supplementary document is structured as follows:

A. **Dataset Construction**

B. **Video Generation Experiment**

C. **Implementation Detail**

D. **Additional Qualitative Results**

A Dataset Construction

In practice, capturing the same dynamic scene with different camera effects is infeasible; re-takes change motion timing, illumination, and other conditions. Public datasets therefore lack multi-view, same-scene recordings under multiple effects.

To fill this gap, we construct a synthetic multi-view dataset in Blender 4.5 [7]. We create 8 dynamic indoor scenes across 4 environments (basketball court, warehouse, living room, bathroom), each with 50 viewpoints and 50 frames at 512×512, rendered under four effects (normal, fisheye, rolling shutter, depth of field).

Section A.1 details the Blender [7] pipeline, tools, and plugins used for multi-view dynamic rendering, and Section A.2 describes the scenes and their material/physics characteristics.

A.1 Scene Construction using Blender

To establish a dataset suitable for evaluating model performance under different camera effects, we selected **Blender 4.5** [7] as our primary scene construction and rendering tool. Blender provides highly realistic physics simulation capabilities, enabling accurate modeling of object motion, collisions, bouncing, and other dynamic behaviors. Additionally, it ensures completely consistent scene configurations across multiple renders, guaranteeing perfect content alignment under different camera effects while avoiding the limitations of real-world filming where identical dynamics cannot be reproduced.

For scene sourcing, we selected 4 high-quality indoor scene models through the **BlenderKit** [8] asset library. In addition, certain dynamic objects, such as the basketball model used in the basketball court scene, were also obtained from BlenderKit. In the scenes of living room and bathroom, the dynamic objects were obtained from other online asset platforms: the LEGO bulldozer model [31] and the Box animation model [63] from **Blend Swap** [9], the running cat model [16] and a pot plant model [1] from **free3D.com** [22]. Within each indoor environment, we designed 2 different types of dynamic objects and animations, such as bouncing spheres, jumping cubes, or other wind-influenced motion behaviors. In total, we constructed **8 distinct dynamic scenes**.

We chose 8 scenes primarily because they encompass diverse material properties and physical interaction patterns, including reflective floor surfaces and rigid body collision dynamics. This diversity adequately simulates various common lighting and motion scenarios found in the real world, providing comprehensive coverage for subsequent model generalization testing. Simultaneously, maintaining a reasonable number of scenes ensures that each scene's dynamic design and camera configuration can undergo thorough manual adjustment and verification, preventing quality degradation due to excessive scene quantities.

For camera configuration, we established **50 different camera viewpoints** for each scene, with each viewpoint rendered separately under four camera effects: pinhole, fisheye, rolling shutter, and depth of field. We selected 50 viewpoints to maintain manageable data volumes while covering sufficient observation angles, ensuring comprehensive capture of dynamic object shape variations and lighting responses. This design also provides adequate observational diversity for our 4D Gaussian ray tracing model, facilitating learning of complete 3D-4D scene structure during training.

Our 50 camera viewpoint placement, corresponding camera pose data, and initial scene point cloud generation were all accomplished through the **PlenoBlenderNeRF** [29] Blender plugin that is originally forked from **BlenderNeRF** [61]. This plugin automatically distributes multiple cameras

uniformly throughout the scene and generates corresponding initial point clouds from Blender mesh vertex data before rendering, ensuring complete consistency between rendering results under different camera effects and geometric foundations, facilitating subsequent quality analysis and comparison.

For data output, we generated **50-frame videos** at **512×512** resolution for each camera under all four camera effects in each scene, ensuring complete capture of dynamic object motion trajectories, lighting changes, and camera effect characteristics for subsequent analysis.

A.2 Scene Descriptions and Characteristics

Our 8 constructed dynamic scenes derive from 4 different environment types: **basketball court**, **warehouse**, **living room**, and **bathroom**. Each scene incorporates distinct physical interaction patterns and surface properties in modeling and material settings, ensuring dataset coverage of diverse lighting responses and motion behaviors. The following sections describe each scene's dynamic design and attributes.

Basketball Court:

1. **Rolling Basketball:** We simulate external forces acting on the basketball surface using Blender's **wind force**, generating rolling motion on the ground. During rolling, the ball experiences both friction and inertia effects, creating gradually decelerating dynamic effects that closely approximate real physical behavior.
2. **Falling Basketballs:** The scene contains three basketballs positioned at height, falling under gravity to the ground surface. After elastic collisions with the floor, the basketballs interact through further collisions, causing directional changes and eventual rolling in different directions. This design simultaneously simulates multiple physical phenomena including multi-body collisions, bouncing, and rolling, effectively testing model capabilities in handling multi-target dynamics and interactions during rendering and geometric reconstruction.

Warehouse:

1. **Bouncing Spheres:** Three rigid spheres positioned at height fall under gravity to the ground surface, undergoing elastic collisions followed by mutual interactions. The collision process alters sphere motion directions and velocities, causing final rolling in different directions and creating complex multi-body interaction and scattering dynamics.
2. **Bouncing Cubes:** Four cubes fall from height to the ground surface, experiencing elastic collisions and mutual interactions. During collisions, cube motion trajectories and rotation angles continuously change, simulating non-spherical rigid body motion characteristics under multiple collision scenarios.

Living Room:

1. **LEGO Bulldozer Model:** We utilize a LEGO bulldozer model as the primary dynamic object in the living room scene. Through animation design, we simulate mechanical arm swinging motions to present realistic mechanical operation dynamics and joint motion trajectories while preserving material details and lighting variations in rendering.
2. **Running Cat Model:** We also employ a cat model with added temporal displacement and running animations, creating continuous motion postures within the scene. The animation encompasses both overall positional changes and continuous limb movements with posture variations, simulating realistic animal running physics and rhythmic characteristics.

Bathroom:

1. **Rotating Pot Plant Model:** We utilize a detailed pot plant model as an additional dynamic element in the bathroom scene. Using Blender's key frame animation system, we create a smooth rotational motion where the pot plant rotates 180 degrees around the z-axis throughout the animation sequence. This rotation provides a controlled, predictable motion pattern that allows for systematic evaluation of how different camera effects handle circular motion and changing viewpoints of complex organic geometry.

Figure 6: Fisheye video evaluation. Sora most closely follows the specified lens parameters, while the others often fail to produce a true fisheye effect. Each cell shows the 60th frame of the generated video.

2. **Box animation Model:** The second dynamic element in the bathroom scene is a self-opening box whose lid gradually reveals its interior. The motion involves coordinated hinge movement, producing geometric changes, occlusion effects, and shifting light as inner surfaces are exposed—ideal for testing camera effects on mechanical motion and surface transitions.

B Video Generation Experiment

To evaluate the capabilities of state-of-the-art video generation models in synthesizing videos with realistic camera effects, we conduct comprehensive experiments using four leading commercial video generation models: OpenAI Sora [57], Google Veo 3 [28], Tencent HunyuanVideo [41], and ByteDance Seedance 1.0 Mini [26]. Our evaluation focuses on three fundamental camera distortion effects: fisheye distortion, rolling shutter artifacts, and depth of field effects. These effects represent critical aspects of realistic video synthesis that require accurate understanding of camera optics and sensor behavior.

B.1 Video Generation Configuration

To ensure fair comparison while respecting each model's individual design constraints and optimal operating conditions, we utilize the default output specifications for each video generation model. This approach allows us to evaluate the models under their native configurations rather than forcing uniform parameters that may disadvantage certain architectures.

B.2 Experimental Design and Prompting Strategy

We design a systematic evaluation protocol using three distinct prompts for each camera effect to assess both implicit understanding and explicit parameter-driven generation capabilities. This multi-prompt approach enables us to evaluate model performance across different levels of technical specification complexity.

B.2.1 Fisheye Distortion Generation

To evaluate fisheye rendering, we use three prompts of increasing specificity. Figure 6 shows the video results; we use the 60th frame from each video as the representative image.

Basic prompt. We first probe implicit fisheye understanding with a natural-language description:

> *"A fisheye-lens video of a person skateboarding in a city square."*

Detailed camera specification (extreme wide-angle). We test explicit parameter control using an extreme wide-angle setup:

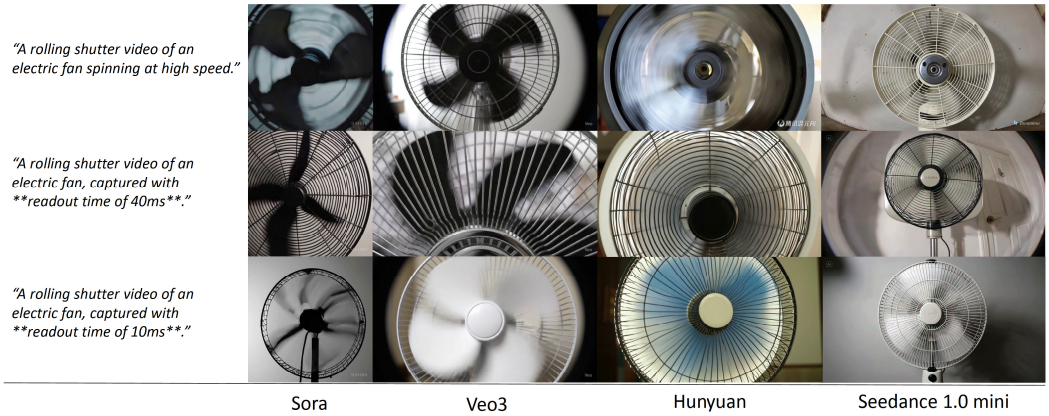

Figure 7: Rolling-shutter video evaluations. Only Veo 3 exhibits the expected rolling-shutter skew in the fan blades; the other models produce uneven blade areas without the characteristic skew.

> *"A video of a person skateboarding in a city square, captured using an **8mm fisheye lens** with a **180° field of view**."*

Detailed camera specification (moderate wide-angle). We also assess sensitivity with a less extreme configuration:

> *"A video of a person skateboarding in a city square, captured using a **15mm fisheye lens** with a **150° field of view**."*

Findings. Across the three prompts, **Sora** is the only model that consistently produces recognizable fisheye imagery and approximately respects the specified parameters. The other models (**Veo 3**, **HunyuanVideo**, and **Seedance 1.0 Mini**) frequently fail to render a convincing fisheye effect or to follow the camera settings. In particular, **HunyuanVideo** under the *basic* prompt shows no fisheye distortion, and **Seedance 1.0 Mini** under the *moderate wide-angle* prompt yields only a generic wide-angle view, losing the characteristic fisheye curvature (see Fig. 6).

B.2.2 Rolling-Shutter Effect Generation

For rolling-shutter evaluation, we use three prompts that vary only the strength of the effect. Figure 7 shows the results from four models (3 prompts × 4 models = 12 images); for visualization, we display the 60th frame from each generated video.

Basic prompt. We probe implicit rolling-shutter understanding with a motion-centric description:

> *"A rolling-shutter video of an electric fan spinning at high speed."*

Detailed camera specification (pronounced effect). We enforce stronger distortion by specifying a longer readout:

> *"A rolling-shutter video of an electric fan, captured with a **readout time of 40 ms**."*

Detailed camera specification (subtle effect). We reduce the distortion by shortening the readout:

> *"A rolling-shutter video of an electric fan, captured with a **readout time of 10 ms**."*

Findings. Across the three prompts, only **Veo 3** reliably exhibits the characteristic rolling-shutter skew in the fan blades. The other models (**Sora**, **HunyuanVideo**, **Seedance 1.0 Mini**) produce blades with uneven apparent areas and lack the expected geometric skew, indicating poor adherence to the rolling-shutter effect (see Fig. 7).

Figure 8: Depth-of-field video evaluation. All models produce pronounced blur regardless of the specified aperture ($f/2.0$ vs. $f/8.0$), revealing weak compliance with DoF parameters and poor control of depth-of-field effects.

B.2.3 Depth-of-Field Effect Generation

For depth-of-field evaluation, we use three prompts that test models' understanding of blur and focus control. Figure 8 shows the results from the four models; for visualization, we display the 60th frame from each generated video.

Basic prompt. We first probe implicit depth-of-field understanding with a natural description:

> *"A video of a chess piece on a board with very shallow depth of field."*

Detailed camera specification (shallow focus). We test explicit parameter control for shallow depth of field:

> *"A video of a chess piece on a board, shot with a **35 mm focal length** and **aperture f/2.0**."*

Detailed camera specification (moderate focus). We evaluate sensitivity under a smaller aperture:

> *"A video of a chess piece on a board, shot with a **35 mm focal length** and **aperture f/8.0**."*

Findings. Across all three prompts—including the explicit *f/2.0* (shallow) and *f/8.0* (moderate) settings—every model renders pronounced background blur in the 60th-frame images, regardless of the specified aperture. This indicates that the models do not respect aperture parameters or depth relationships, and therefore perform poorly on depth-of-field video generation (see Fig. 3).

In summary, across fisheye, rolling-shutter, and depth-of-field evaluations, current video generation models exhibit limited camera awareness. Only Sora reliably produces recognizable fisheye imagery, and only Veo 3 consistently displays the expected rolling-shutter skew; depth-of-field outputs remain strongly blurred regardless of the specified aperture (e.g., $f/2.0$ vs. $f/8.0$). These results indicate poor compliance with camera parameters and insufficient physical fidelity, underscoring the need for camera-aware training data and modeling approaches.

C Implementation Detail

C.1 Gaussian Deformation Network Architecture

We adopt the 4DGS [78] formulation of the Gaussian deformation network to predict the deformation of 3D Gaussians at a specific time step. Specifically, the network consists of a spatio-temporal structure encoder and a multi-head Gaussian deformation decoder.

The spatio-temporal encoder is constructed using six multi-resolution plane modules

$$R_l(i,j) \in \mathbb{R}^{h \times lN_i \times lN_j},$$

and an MLP φ, where l is the upsampling scale, h is the dimension of the voxel feature, and N_i and N_j are the base resolutions of the plane. Given the mean value (x, y, z) of a 3D Gaussian

$$G = \{(x, y, z), r, s, \sigma, \mathcal{C}\},$$

we first compute the voxel feature

$$f_h = \bigcup_l \prod \mathbf{interp}(R_l(i,j)) \in \mathbb{R}^{h \times l}, \tag{4}$$

$$(i,j) \in \{(x,y),(x,z),(y,z),(x,t),(y,t),(z,t)\}, \tag{5}$$

where $\mathbf{interp}$ denotes bilinear interpolation. The feature f_h is then merged by φ to produce

$$f = \varphi(f_h).$$

Next, the multi-head Gaussian deformation decoder predicts the deformation of 3D Gaussians. The decoder consists of separate MLPs $\{\varphi_x, \varphi_r, \varphi_s\}$, which predict the deformation residuals of each Gaussian attribute:

$$(\Delta x, \Delta y, \Delta z) = \varphi_x(f), \quad \Delta r = \varphi_r(f), \quad \Delta s = \varphi_s(f).$$

These residuals are then applied to deform a 3D Gaussian G into

$$G_t = \{(x + \Delta x, y + \Delta y, z + \Delta z),\ r + \Delta r,\ s + \Delta s,\ \sigma,\ \mathcal{C}\}.$$

C.2 Training details

The hyperparameters in our experiments are mainly based on those reported in 4DGS [78]. In our observation, we find the setting following 4DGS [78] will not perform well in scenes with rapidly moving objects. Therefore, we increase the learning rate, the capacity of the deformation field, the basic resolution of the plane modules and the training iterations. Specifically, we adopt the following setting for scenes with objects undergoing vigorous motion. The width of deformation MLP is set to 128. The basic resolution of the plane modules is set to 128, which is upsampled by 2, 4 and 8. The densification threshold is set as 2.5×10^{-5} and increases to 1×10^{-4}. The training iteration in the coarse stage is set to 5×10^4 and in the fine stage it is set to 8×10^4.

We adopt a two-stage optimization strategy similar to 4DGS [78]. In the first stage, we initialize the positions and colors of the 3D Gaussians using point clouds generated from Blender's synthetic scenes. We then train the 3D Gaussian scene—without the deformation network—using only the first frame. This step reconstructs the static components of the scene and provides a strong initialization for dynamic objects. In the second stage, we introduce the deformation network and jointly optimize both the 3D Gaussians and the network across all time frames.

In 3DGS [40], densification is guided by the magnitude of the 2D positional gradients of Gaussians. In contrast, since our rendering relies on ray tracing rather than rasterization, we follow the approach of [52] and use the magnitude of the 3D gradient of the Gaussian means to decide when to densify a Gaussian. Additionally, as in [52], we scale the gradient magnitude by the distance of the deformed Gaussian from the camera, encouraging densification in the background.

D Additional Qualitative Result

We present a qualitative comparison of rolling-shutter rendering with different chunk sizes. As shown in 9, smaller chunk sizes produce more accurate renderings, while larger chunk sizes lead to noticeable blocking artifacts.

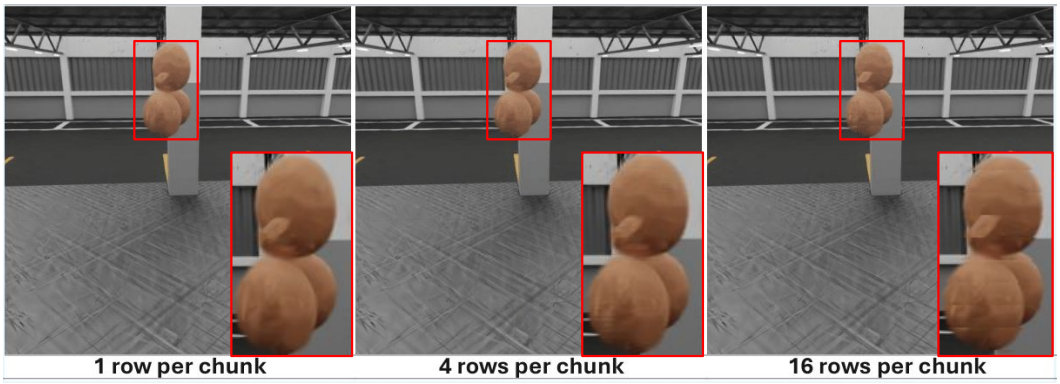

Figure 9: Qualitative comparison of rolling shutter effect with different number of rows per chunk.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly summarize the main contributions and scope of the work, and these claims are consistent with the experimental results presented in the paper (see Abstract, Section 1, and Section 4).

   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: The paper explicitly discusses limitations in the Conclusion and Limitations section, highlighting the requirement of well-reconstructed dynamic scenes with sufficient multi-view data, and the reduced applicability in sparse or monocular settings (see Section 5).

Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [NA]

   Justification: The paper does not contain formal theoretical results. Instead, our claims are empirically supported through experiments where a video generation model produces videos from prompts with camera-effect parameters, and the quality of these generated videos is evaluated (see Figure 2 for examples).

   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: The paper provides sufficient details to reproduce the main experimental results, including dataset construction, baseline configurations, training setup, and evaluation metrics (Section 4). Additional technical details and scene descriptions are provided in the supplementary material.

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: We provide full code and instructions, along with sample data in the supplementary material. The sample data allows reviewers to run the pipeline and verify functionality, but it is not sufficient to fully reproduce the main experiments (which require data from 50 cameras). The complete dataset will be released upon acceptance to enable full reproducibility of all results.

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: The paper specifies the datasets (Blender-generated multi-view dynamic scenes with four camera effects), baselines (HexPlane and MSTH dynamic NeRF models), training setup (all methods trained on the same input videos until convergence), and evaluation metrics (PSNR, SSIM, LPIPS). Additional implementation details are provided in the supplementary material.

   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 report mean performance values across all frames and scenes using PSNR, SSIM, LPIPS, rendering FPS, training time, and Storage (Section 4). Limited to computational cost, we did not run multiple random seeds or report error bars, but the evaluation covers diverse dynamic scenes to provide robustness.

   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 report the compute resources used for our experiments, including GPU type (NVIDIA RTX 4090), memory, and training time (see Section 4 and supplementary material). This information is sufficient for reproducing the experiments under similar hardware conditions.

   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: The research conforms to the NeurIPS Code of Ethics. Our study uses synthetic datasets constructed in Blender and does not involve human subjects or personally identifiable information (see Section 4).

   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: The paper discusses both potential positive and negative societal impacts: on the positive side, it highlights applications in areas such as video editing and visualization, where controllable and physically accurate camera effect simulation can be beneficial; on the negative side, it acknowledges the risk of misuse, for example in generating misleading or deceptive visual content.

    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: Our released dataset is fully synthetic, constructed in Blender without human subjects or scraped web content, and thus does not pose significant risks requiring special safeguards.

    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: All external assets used in constructing our dataset, including Blender, BlenderKit, PlenoBlenderNeRF, BlendSwap, and Free3D, are properly cited with attribution to their creators. We respect the corresponding licenses and terms of use, as noted in the references and supplementary material.

    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: [Yes]

    Justification: We introduce a new benchmark dataset consisting of rendered multi-view videos with multiple camera effects. The dataset is fully synthetic, does not involve human subjects, and is documented with construction details and scene descriptions.

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

    Answer: [NA]

    Justification: Our study does not involve crowdsourcing or human subjects; all datasets are fully synthetic and generated using Blender.

    Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [NA]

    Justification: Our work does not involve human subjects or crowdsourcing; thus, IRB approval is not applicable.

    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.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core methodology of this work does not involve large language models; they are only referenced in analysis for performance comparison, not as part of the proposed method.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.