Event-boosted Deformable 3D Gaussians for Dynamic Scene Reconstruction Supplementary Material

Wenhao Xu[1] Wenming Weng[1] Yueyi Zhang[2,✉] Ruikang Xu[1] Zhiwei Xiong[1,✉]
[1]University of Science and Technology of China [2]MiroMind
wh-xu@mail.ustc.edu.cn, yueyi.zhang@miromind.ai, zwxiong@ustc.edu.cn

This supplementary material is organized as follows:
Sec. 1 presents additional quantitative results.
Sec. 2 presents additional qualitative results.
Sec. 3 presents further dataset details.
Sec. 4 presents implementation details.

1. Additional Quantitative Results

1.1. Additional Comparisons

Table 1. Quantitative results on scene-level real-world dataset.

Method	Wielding			Jumping			Walking		
	PSNR↑	SSIM↑	FPS↑	PSNR↑	SSIM↑	FPS↑	PSNR↑	SSIM↑	FPS↑
Deformable-3DGS [15]	28.27	0.883	75	26.37	0.847	73	30.44	**0.935**	127
Event-4DGS	29.71	0.887	77	27.57	0.851	67	30.73	0.890	121
Ours	**30.56**	**0.901**	**172**	**28.37**	**0.867**	**204**	**31.89**	0.928	**303**

Comparisons on more challenging scenes. The four real-world scenes presented in the main paper are object-level, where the objects are placed on a motorized optical rotating turntable, resulting in smooth and simple camera motion. To evaluate our method on more challenging scenes, we captured three additional *scene-level* real-world scenes with complex motions, including wielding, jumping, and walking (the qualitative results are shown in Fig. 4). Quantitative results on these scenes are reported in Tab. 1. Our method still achieves the highest reconstruction quality and the fastest rendering speed, demonstrating the robustness of our approach, especially of our DSD strategy, in handling more challenging dynamic scenes.

1.2. Additional Ablation Studies

Per-scene Ablation Study. As shown in Tab. 2 and Tab. 3, we present the per-scene ablation results of our method across both synthetic and real-world scenes. The results provide compelling evidence that each proposed component contributes to the overall performance.

Notably, the optimal FPS does not show a clear pattern, as ablating components indirectly affects the number of Gaussians, leading to *acceptable fluctuations* in FPS. However, what truly stands out is that *ablating DSD significantly reduces FPS*, highlighting the crucial role of DSD in accelerating rendering. In particular, as demonstrated in Tab. 3, using GTJM leads to an average PSNR improvement of 0.70 dB, indicating that *our method effectively models the threshold variations of real event cameras*. Meanwhile, in real-world scenes, using DSD also brings an average PSNR improvement of 1.04 dB and a 2.96× speedup in rendering. In summary, *the key components of our method, GTJM and DSD, remain effective in real-world scenes*, enabling our method to achieve high-fidelity reconstruction quality and exceptional rendering speed.

Transfer our method to another baseline. In the main experiments, our method is built upon Deformable-3DGS [15]. To further demonstrate the generalizability of our method, we transfer it to another baseline, 4DGS [14]. As shown in Tab. 4, on the "Lego" scene, our key components, GTJM and DSD, still significantly improve both reconstruction quality and rendering speed, highlighting the ability of each component to *generalize across different baselines*.

Table 2. Per-scene ablation studies on synthetic dataset.

Method	Lego				Hotdog				Materials				Music box			
	PSNR↑	SSIM↑	LPIPS↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓	FPS↑
w/o GTJM	29.49	0.955	0.028	177	34.72	0.970	0.017	**261**	35.72	0.990	0.006	**243**	28.73	0.952	0.042	79
w/o Joint Optimization in GTJM	30.92	0.962	0.021	185	35.67	0.973	0.014	238	37.27	0.992	0.004	239	30.15	0.959	0.033	87
w/o DSD	29.62	0.954	0.028	58	35.69	0.972	0.014	104	37.70	0.992	0.004	76	30.29	0.959	0.032	43
w/o Buffer-based Soft Decomposition	30.27	0.959	0.025	168	35.95	0.973	0.015	216	37.31	0.991	0.004	207	30.77	**0.963**	0.030	82
Full	**31.85**	**0.967**	**0.018**	**189**	**36.15**	**0.974**	**0.013**	241	**38.02**	**0.993**	**0.003**	240	**30.78**	**0.963**	**0.029**	**92**
Method	Celestial globe				Fan				Water wheel				Man			
	PSNR↑	SSIM↑	LPIPS↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓	FPS↑
w/o GTJM	25.06	0.955	0.038	67	28.16	0.955	0.035	**169**	26.57	0.932	0.051	**113**	26.67	0.937	0.053	118
w/o Joint Optimization in GTJM	27.49	0.970	0.025	74	29.54	0.961	0.028	165	28.14	0.945	0.038	110	27.76	0.940	0.041	120
w/o DSD	27.83	0.971	0.024	37	29.73	0.959	0.029	77	27.86	0.948	0.034	31	27.54	0.935	0.040	31
w/o Buffer-based Soft Decomposition	28.16	0.973	0.022	**78**	29.65	0.959	0.029	153	28.36	0.949	**0.033**	90	27.67	0.938	0.038	106
Full	**28.83**	**0.976**	**0.020**	73	**30.18**	**0.964**	**0.025**	168	**28.47**	**0.950**	**0.033**	112	**28.21**	**0.943**	**0.037**	**129**

Table 3. Per-scene ablation studies on real-world dataset.

Method	Excavator				Jeep				Flowers				Eagle			
	PSNR↑	SSIM↑	LPIPS↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓	FPS↑
w/o GTJM	30.30	0.918	0.083	169	29.64	0.900	0.081	79	28.00	0.910	0.081	149	30.80	0.914	0.084	178
w/o Joint Optimization in GTJM	31.14	0.923	0.073	169	30.25	0.904	0.072	**91**	28.44	0.912	0.071	140	31.14	0.916	0.076	185
w/o DSD	30.14	0.913	0.085	60	30.13	0.904	0.072	42	27.76	0.907	0.078	44	29.36	0.888	0.082	57
w/o Buffer-based Soft Decomposition	30.67	0.920	0.077	166	30.25	0.903	0.072	84	28.49	**0.915**	0.071	**158**	31.05	0.915	0.077	**193**
Full	**31.28**	**0.925**	**0.070**	**179**	**30.41**	**0.905**	**0.068**	89	**28.57**	0.913	**0.069**	149	**31.29**	**0.918**	**0.074**	192

Table 4. Our method and ablation studies based on 4DGS [14].

Methods	PSNR↑	SSIM↑	LPIPS↓	FPS↑
4DGS [14]	26.30	0.937	0.072	104
w/o GTJM	30.53	0.956	0.032	220
w/o DSD	29.95	0.952	0.052	63
Ours (based on 4DGS [14])	**31.61**	**0.961**	**0.027**	**221**

The Effect of Motion Complexity on Dynamic-static Decomposition Strategy. We measure motion complexity from two aspects: dynamic component count and motion speed. For dynamic component count, we extend the original scene "Moderate" by adding or removing dynamic components, creating two variant scenes, "Many" and "Few" (see Fig. 1). For motion speed, we adjust the animation speed in Blender [1] to create two variant scenes, "Slow" (0.5x the original speed) and "Fast" (2.0x the original speed).

As shown in Tab. 5, applying the DSD strategy enables the deformation field to focus on the dynamic regions, resulting in a significant average PSNR improvement of 0.99 dB across all scenes. More importantly, we observe that the gains from the DSD strategy increase as motion complexity grows. This highlights the potential of our method in reconstructing complex dynamic scenes.

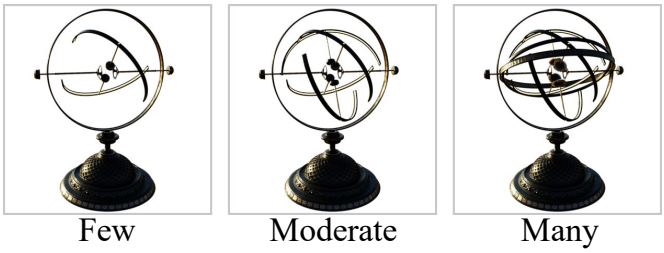

Figure 1. Scenes with varying numbers of dynamic components.

The Effect of Varying Degrees of Threshold Variations. The greater the range of threshold variations, the more intense the degree of threshold variations becomes. As shown in Fig. 2 (a), the performance of both our method and the baseline Event-4DGS declines under intense threshold variations, with Event-4DGS experiencing a more significant decline. Fig. 2

Table 5. The effect of motion complexity on dynamic-static decomposition strategy.

Method	Dynamic component count			Motion speed		
	Few	Moderate	Many	Slow	Moderate	Fast
w/o DSD	30.50	27.83	26.79	28.92	27.83	24.96
Full	**31.25**	**28.83**	**28.09**	**29.60**	**28.83**	**26.16**
△	0.75	1.00	1.30	0.68	1.00	1.20

(b) shows that the PSNR difference between our method and Event-4DGS widens as the threshold variations become more intense. For instance, at a variation range of 0.05, our method achieves a 2.66 dB PSNR improvement over Event-4DGS. When the range increases to 0.20, this improvement grows to 3.83 dB. These results demonstrate that our proposed threshold modeling method becomes increasingly beneficial as threshold variations become more intense.

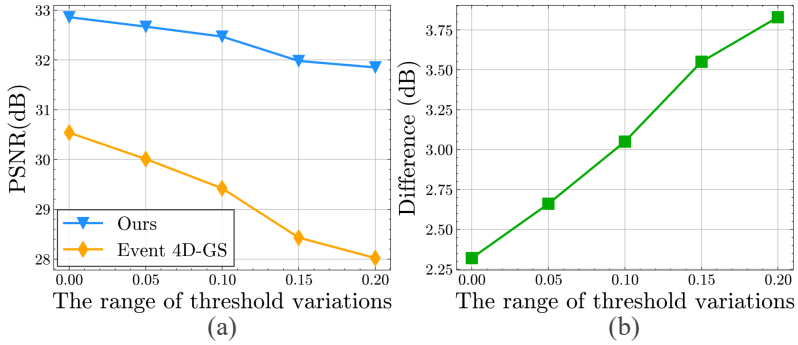

Figure 2. (a) The effect of different ranges of threshold variations on Event-4DGS and our method. (b) PSNR differences between Event-4DGS and our method across different ranges of threshold variations.

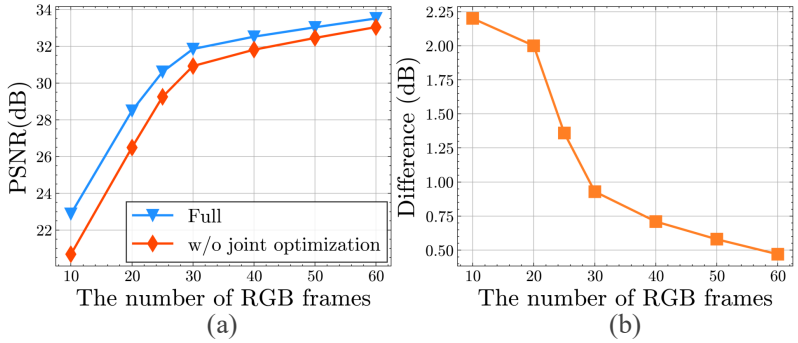

Figure 3. (a) The effect of different number of RGB frames on our method and its variant (without joint optimization in GTJM). (b) PSNR differences between our method and its variant across different numbers of RGB frames.

The Effect of Different Number of RGB Frames. In Fig. 3 (a), we demonstrate the effect of different numbers of RGB frames on our method and its variant (without joint optimization in GTJM). Denser RGB frames provide stronger geometric supervision, naturally benefiting both our method and the variant. More importantly, as shown in Fig. 3 (b), the PSNR difference between our method and the variant increases as the RGB frames become sparser. Notably, in the extremely sparse case with only 10 RGB frames, applying joint optimization in GTJM delivers a significant 2.20 dB improvement. These results support the motivation behind *GS-boosted threshold refinement* in Section 3.2 of the main paper, which leverages 3D-GS rendered intermediate frames as additional pseudo supervision to enhance the limited supervision from sparse RGB frames.

Evaluation on Longer Videos. Considering that a one-second motion may not be sufficient, we further evaluate our method

on the "Lego" scene with longer capture inputs. As shown in Tab. 6, as the number of captured inputs increases, the reconstruction quality of our method improves, demonstrating its robustness on longer videos.

Table 6. Evaluation on longer videos.

Time(s)	1.0	2.0	3.0	4.0	5.0	6.0
PSNR	31.85	33.47	34.18	34.69	35.04	35.18

1.3. Dynamic Blurry Scene Reconstruction

In dynamic scene reconstruction, severely motion-blurred training frames pose a significant challenge. Such frames exhibit inaccurate multi-view geometric correspondences, making it difficult for 3D-GS to represent the 3D scene. To address this challenge, we extend both the baselines and our approach by explicitly modeling the motion blur process.

First, we replace the original RGB rendering loss Eq.(5) with the blur reconstruction loss from [16] to model the motion blur process. Specifically, we render n images $\left\{\hat{I}_i\right\}_{i=1}^{n}$ from 3D-GS along the camera trajectory during the exposure period and approximate motion blur formation via discrete averaging,

$$\hat{B} = \frac{1}{n}\sum_{i=1}^{n}\hat{I}_i. \tag{1}$$

We then minimize the photometric error between the simulated blurry image $\hat{B}$ and the ground truth blurry image B as follows,

$$\mathcal{L}_{blur} = (1-\lambda_s)\left\|\hat{B}-B\right\|_1 + \lambda_s\mathcal{L}_{D-SSIM}(\hat{B},B). \tag{2}$$

Second, for our method and Event-4DGS, we integrate the Event-based Double Integral (EDI) model [7], which derives sharp latent frames from motion-blurred inputs. The resulting sharp frames serve as $I(t)$ in Eq.(4) of the main paper, enabling the computation of the event rendering loss. Additionally, our method's GTJM and DSD components are also built upon these sharp frames, allowing them to function as intended.

Table 7. Comparison on "Blurry Lego".

Method	PSNR↑	SSIM↑	LPIPS↓
Deformable-3DGS	24.19	0.866	0.156
Event-4DGS	<u>27.41</u>	<u>0.921</u>	<u>0.054</u>
Ours	**28.98**	**0.934**	**0.047**

Table 8. Ablation studies on "Blurry Lego".

Method	PSNR↑	SSIM↑	LPIPS↓
w/o GTJM	28.25	0.927	0.050
w/o DSD	28.07	0.928	0.051
Full	**28.98**	**0.934**	**0.047**

For evaluation, we construct a dynamic blurry scene, "Blurry Lego", by averaging 32 consecutive latent frames. In Tab. 7, we report the quantitative comparison results on "Blurry Lego" corresponding to Fig.9 of the main paper. The results demonstrate that by leveraging the deblurring advantage of events, our method still achieves the best performance. Meanwhile, as shown in Tab. 8, our key components, GTJM and DSD, remain robust to motion blur.

2. Additional Qualitative Results

Additional Qualitative Comparisons. In Figs. 4 to 6, we present qualitative comparisons of baselines and our method on *scene-level real-world scenes*, *object-level real-world scenes*, and *synthetic scenes*, respectively. These visual comparisons highlight the effectiveness of our method in reconstructing high-fidelity dynamic scenes. Specifically, in the "Lego" scene of Fig. 6, only our approach successfully captures the small raised and recessed dots on the toy's surface, while other methods exhibit noticeable structural distortions and blurring.

Figure 4. Qualitative comparisons on scene-level real-world scenes.

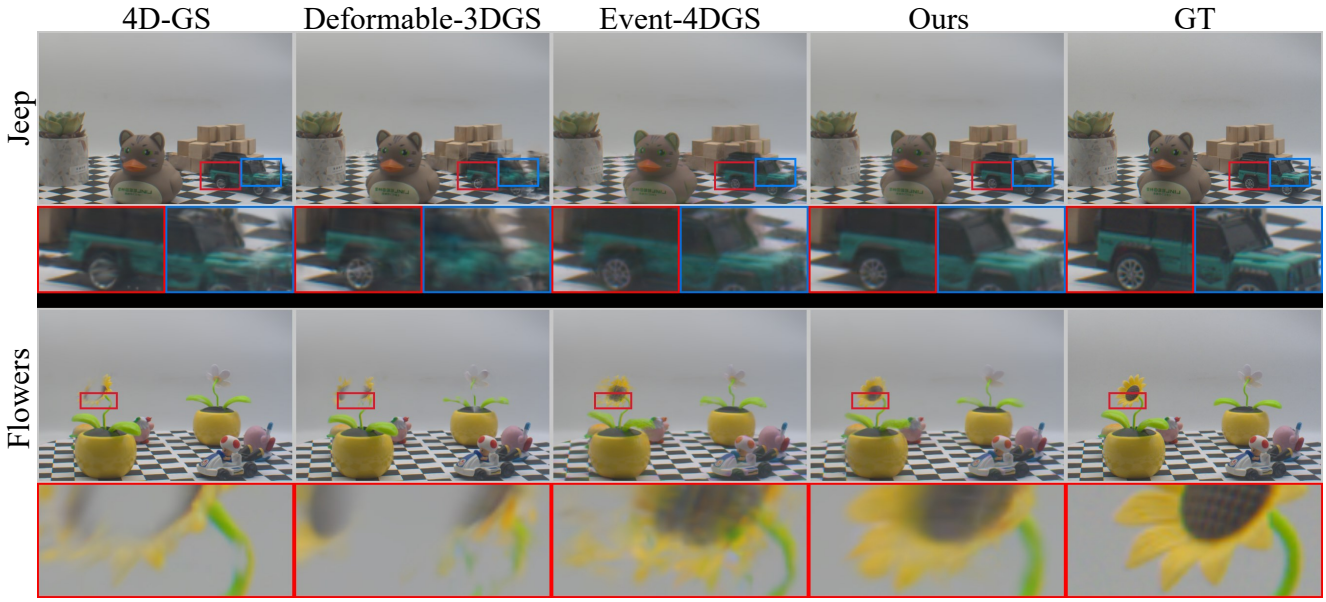

Figure 5. Qualitative comparisons on object-level real-world scenes.

Figure 6. Qualitative comparisons on synthetic scenes.

Qualitative Results of Ablation Study. Additionally, as shown in Fig. 7, we provide qualitative results of ablation study on synthetic dataset. Notably, in the subplots with red borders, the absence of GTJM or joint optimization results in color artifacts (e.g., the yellow-green haze in the "Lego" scene and the purple haze in the "Celestial Globe" scene). In the subplots with blue borders, the absence of DSD and buffer-based soft decomposition leads to deficient and distorted dynamic structures.

Figure 7. Qualitative results of ablation study on synthetic dataset.

3. More Details on Dataset

3.1. More Details on Synthetic Dataset

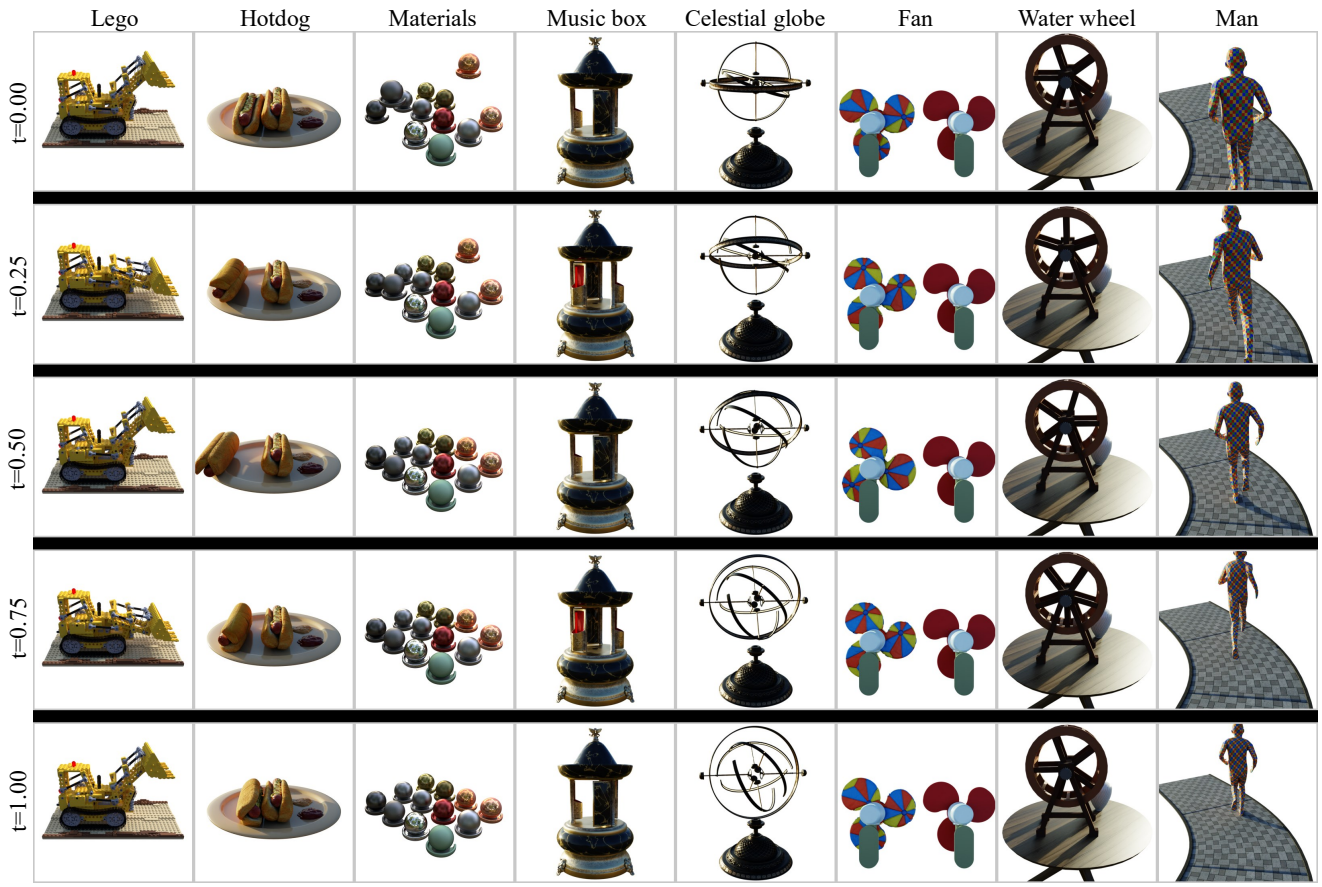

Figure 8. Visualization of our dataset, captured at different time steps under a fixed viewpoint. **Please see the supplementary video for details.**

Scene Motion Visualization. Fig. 8 presents all synthetic scenes in our dataset at different time steps under a fixed viewpoint. To better visualize the dynamic content, we recommend watching the supplementary video. Specifically, we developed new animations based on the static scenes provided by NeRF [6] to create the dynamic scenes "Lego", "Hotdog", and "Materials". The "Music box" and "Celestial globe" are brand-new single-object dynamic scenes. The "Fan", "Water wheel", and "Man" are combination scenes, consisting of independent dynamic and static objects. Notably, the scene "Fan" (a rotating electric fan) represents a typical *high-speed* 4D scene, while "Man" is another particularly challenging scene due to the *large object displacements* and complex texture variations it involves.

Simulating Threshold Variations. By default, the simulated thresholds C vary between 0.1 and 0.3, encompassing the approximate nominal thresholds of commonly used event cameras, such as the Prophesee Gen 4 and DAVIS 346. To simulate threshold variations over polarity, the positive thresholds C_p and negative thresholds C_n are constrained by the relation: $C_n = C_p \times n$, where $n \in \mathcal{N}(\mu, \sigma^2)$ with $\mu = 1.0$ and $\sigma = 0.1$. To simulate threshold variations over time, the threshold parameters in the event camera simulator ESIM [11] are updated every 50 frames. Threshold variations over space are simulated by introducing noisy events. Specifically, Gaussian noise with $\mu = 0$ and $\sigma = 1.0$ is generated, converted to integer values to form discrete noisy events. These noisy events are introduced at a specified fraction, controlled by a masking process to regulate the noise amount.

Please note that *the distribution of simulated threshold variations is unknown to our method.* The threshold modeling in Eq.8 of the main paper, relies solely on the event generation model [2, 5], meaning it holds true under any threshold distributions. In other words, Eq.8 is fundamentally data-driven and capable of modeling arbitrary threshold variations. This

capability allows our method to remain effective even when encountering the complex threshold distributions in real-world. This is also the reason why we do not follow [3] to oversimplify the modeling of threshold variations using a Gaussian distribution.

3.2. Hybrid Camera System

Following prior work [2, 8–10, 16], we adopt a coaxial camera system instead of a stereo setup. In a stereo system, the optical centers of the event and frame cameras are misaligned, which makes precise alignment challenging even with rigorous calibration. Such misalignment could degrade the performance of both the event rendering loss in Eq.(4) of the main paper and the EDI in Sec. 1.3 of this supplementary material. Additionally, the real-world scenes captured in prior work [2, 8–10, 16] suffer from low-resolution and low-quality events. Therefore, we build a hybrid camera system rather than using DAVIS346, in order to capture higher-quality datasets.

As mentioned in the main paper, our hybrid camera system consists of a frame camera (Basler acA1440-220uc with a resolution of 1440 $\times$ 1080 and an 8mm lens), an event camera (Prophesee Gen4 with a resolution of 1280 $\times$ 720 and an 8mm lens), a beam splitter (Thorlabs CCM1-BS013) and a microcontroller (STM32F103C8). Following [13], we performed geometric calibration and temporal synchronization between the event camera and the frame camera.

Geometric Calibration. We first display a blinking chessboard pattern (6 $\times$ 9) on a 27-inch monitor positioned 1 meter in front of the hybrid camera system. The frame camera captures frames at 30 FPS, while the event camera accumulates a one-second event stream to generate an event image. After obtaining both the frames and event images, we apply OpenCV's corner detection algorithm to extract the chessboard corners. The detected corners are visualized in Fig. 9. To ensure accuracy, any incorrectly detected corners are manually removed.

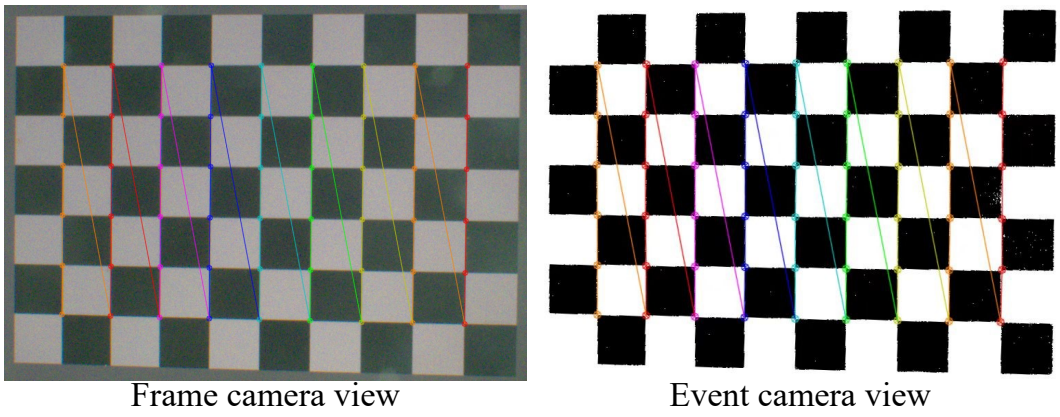

Figure 9. The detected corners for homograpy computation from the frames and event images.

Our goal is to geometrically map pixels from the frame camera view to the corresponding pixels in the event camera view within the overlapping field of view. To achieve this, we employ a homography matrix, estimated using the detected corner correspondences between the two camera views. Formally, this transformation is expressed as,

$$p_i^E = H \cdot p_i^F, \tag{3}$$

where H denotes a 3 $\times$ 3 transformation matrix, $p_i^E = [x_i^E, y_i^E, 1]^T$ and $p_i^F = [x_i^F, y_i^F, 1]^T$ are the homogeneous coordinates in the event images and frames. Finally, we warp the frame using the computed matrix H to spatially align the two camera views.

Temporal Synchronization. In our hybrid camera system, two cameras are temporally synchronized by external triggers. A microcontroller acts as the master device, generating a square wave synchronization signal, while the frame and event cameras act as slave devices. On each falling edge of the square wave, the frame camera captures a frame, while the event camera records an external event (logging the synchronization timestamp). The FPS of the frame camera is determined by the frequency of the square wave generated by the microcontroller.

4. More Details on Implementation

The Algorithm. Our method builds upon the differentiable rasterizer of 3D-GS [4] and is implemented in PyTorch, running on an NVIDIA RTX 3090 GPU. Given sparse RGB frames, we initialize from random point clouds rather than structure-from-motion (SfM) [12] points. Hyperparameters are provided in the supplementary materials.

We summarize the detailed training procedure in Algorithm 1. Our method takes sparse RGB frames I and the corresponding event streams $\mathcal{E}$ as input.

Stage 1: RGB-assisted Initial Estimation. We first leverage the intensity information from the RGB frames to estimate the initial threshold C. We initialize C as a constant and optimize it using $\mathcal{L}_{\text{thres}}$ for 7k iterations.

Stage 2: Dynamic-static Decomposition. Next, we decompose the scene into dynamic and static components. Warm-up Gaussians GS_{init} are initialized using randomly generated points and optimized for 3k iterations using $\mathcal{L}_{\text{rgb}}$. Through the dynamic-static decomposition pipeline in Section 3.3, we decompose GS_{init} into static Gaussians GS_{s} and dynamic Gaussians GS_{d}.

Stage 3: Deformation Warm-up. We then warm up the deformation field using only RGB frames to accelerate overall training speed. We randomly initialize the deformation field parameters θ and jointly optimize GS_{s}, GS_{d}, and θ using $\mathcal{L}_{\text{rgb}}$.

Stage 4: Joint Threshold and GS Optimization. Finally, we introduce event supervision and perform joint threshold and GS optimization. The total loss $\mathcal{L}_{\text{total}}$ is used to jointly optimize GS_{s}, GS_{d}, θ, and C.

Hyperparameter Settings. The learning rate for 3D-GS [4] is initialized at 1.6×10^{-4} and decays to 1.6×10^{-5} by the end of training. The deformation field, modeled as an MLP network [15], starts with the same learning rate of 1.6×10^{-4}, which similarly decreases to 1.6×10^{-5}. The learnable threshold uses a learning rate of 2.4×10^{-4} during the RGB-assisted initial estimation phase, and 1.6×10^{-4} during the joint optimization phase, decaying to 1.6×10^{-5}. The time bin for the ECM is set to 64, enabling the capture of fine-grained threshold variations over time. The weighting factors of the loss functions, λ_{s}, λ_{event}, and λ_{thres}, are set to 0.2, 5.0, and 5.0, respectively.

Algorithm 1 Training for Event-boosted Deformable 3D Gaussians

Require: Sparse RGB frames I, event streams $\mathcal{E}$

Stage 1: RGB-assisted Initial Estimation

Initialize threshold C to a constant.

Optimize C for 7k iterations as

$$C^* = \underset{C}{\text{argmin}} \mathcal{L}_{\text{thres}}(C, I, \mathcal{E}).$$

Stage 2: Dynamic-static Decomposition

Randomly initialize static Gaussians GS_{init}.

Optimize GS_{init} for 3k iterations as

$$GS^*_{init} = \underset{GS_{init}}{\text{argmin}} \mathcal{L}_{rgb}(GS_{init}, I).$$

Decompose GS_{init} into static Gaussians GS_s and dynamic Gaussians GS_d.

Stage 3: Deformation Warm-up

Randomly initialize deformation field parameters θ.

Joint optimize GS_s, GS_d, and θ for 5k iterations as

$$GS^*_s, GS^*_d, \theta^* = \underset{GS_s, GS_d, \theta}{\text{argmin}} \mathcal{L}_{rgb}(GS_s, GS_d, \theta, I).$$

Stage 4: Joint Threshold and GS Optimization

Joint optimize GS_s, GS_d, θ, and C for 42k iterations as

$$GS^*_s, GS^*_d, \theta^*, C^* = \underset{GS_s, GS_d, \theta, C}{\text{argmin}} \mathcal{L}_{total}(GS_s, GS_d, \theta, C, I, \mathcal{E}).$$

References

[1] Blender. A 3d modelling and rendering package. https://www.blender.org/, 2018. 2

[2] Marco Cannici and Davide Scaramuzza. Mitigating motion blur in neural radiance fields with events and frames. In *CVPR*, 2024. 8, 9

[3] Yuhuang Hu, Shih-Chii Liu, and Tobi Delbruck. v2e: From video frames to realistic dvs events. In *CVPR workshop*, 2021. 9

[4] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):139–1, 2023. 10

[5] Simon Klenk, Lukas Koestler, Davide Scaramuzza, and Daniel Cremers. E-nerf: Neural radiance fields from a moving event camera. *IEEE Robotics and Automation Letters*, 8(3):1587–1594, 2023. 8

[6] 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. 8

[7] Liyuan Pan, Cedric Scheerlinck, Xin Yu, Richard Hartley, Miaomiao Liu, and Yuchao Dai. Bringing a blurry frame alive at high frame-rate with an event camera. In *CVPR*, 2019. 4

[8] Yunshan Qi, Lin Zhu, Yu Zhang, and Jia Li. E2nerf: Event enhanced neural radiance fields from blurry images. In *ICCV*, 2023. 9

[9] Yunshan Qi, Jia Li, Yifan Zhao, Yu Zhang, and Lin Zhu. E^3 nerf: Efficient event-enhanced neural radiance fields from blurry images. *arXiv preprint arXiv:2408.01840*, 2024.

[10] Yunshan Qi, Lin Zhu, Yifan Zhao, Nan Bao, and Jia Li. Deblurring neural radiance fields with event-driven bundle adjustment. In *MM*, 2024. 9

[11] Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza. Esim: an open event camera simulator. In *CoRL*, 2018. 8

[12] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 10

[13] Jin Wang, Wenming Weng, Yueyi Zhang, and Zhiwei Xiong. Unsupervised video deraining with an event camera. In *ICCV*, 2023. 9

[14] 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 *CVPR*, 2024. 1, 2

[15] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *CVPR*, 2024. 1, 10

[16] Wangbo Yu, Chaoran Feng, Jiye Tang, Xu Jia, Li Yuan, and Yonghong Tian. Evagaussians: Event stream assisted gaussian splatting from blurry images. *arXiv preprint arXiv:2405.20224*, 2024. 4, 9