A. Implementation Details

A.1. Algorithm

Our optimization and densification algorithm is shown in Algorithm 1. All modifications compared to the original Gaussian Splatting process [26] are highlighted in green.

Algorithm 1 Optimization and Densification
w, h: width and height of the training images

```
M ← Event-to-VideoGuidedPoints()                    ▷ Positions
S, C, A ← InitAttributes()              ▷ Covariances, Colors, Opacities
i ← 0                                               ▷ Iteration Count
while not converged do
    k_start, k_end, D ← GenerateTrainingView()
    I_1 ← Rasterize(M, S, C, A, k_start)
    I_2 ← Rasterize(M, S, C, A, k_end)
    L̂_1 ← Log(Remosaicing(I_1))
    L̂_2 ← Log(Remosaicing(I_2))
    ℒ ← Loss(L̂_2 − L̂_1, D)                                  ▷ Loss
    M, S, C, A ← Adam(∇ℒ)                           ▷ Backprop & Step
    if IsRefinementIteration(i) then
        for all Gaussians (μ, Σ, c, α) in (M, S, C, A) do
            if α < ε or IsTooLarge(μ, Σ) then              ▷ Pruning
                RemoveGaussian()
            end if
            if ∇_p L > τ_p then                       ▷ Densification
                if ||S|| > τ_S then            ▷ Over-reconstruction
                    SplitGaussian(μ, Σ, c, α)
                else                          ▷ Under-reconstruction
                    CloneGaussian(μ, Σ, c, α)
                end if
            end if
        end for
    end if
    i ← i + 1
end while
```

A.2. Hyper-parameters and Optimizations

Our approach adopts original 3D Gaussian Splatting as the backbone as it allows for high quality view synthesis with high-speed rendering. The Gaussian Model is initialized with spherical harmonics degree and several parameters, including xyz coordinates, features, scaling, rotation, and opacity. The model sets up essential functions for covariance, opacity, and rotation activations. The model includes functions to densify and prune Gaussians based on gradient thresholds and opacity. This ensures efficient use of computational resources by adding new Gaussians where needed and removing those that are not contributing effectively. Training utilizes the similar optimization strategies and hyper-parameter settings originally proposed for 3D Gaussian Splatting including position, feature, opacity, scaling, and rotation. The learning rate is scheduled to adjust dynamically during training. The only opacity learning rate was changed from 0.05 to 0.01 to make the training more stable. The instability seems to result from the 3D Gaussian Splatting model being supervised from multi-view points with different accumulation lengths.

A.2.1. Contrast threshold

Both Robust-e-NeRF and our method were co-optimized and trained with the symmetric contrast thresholds initialized at ${C_{+1}}/{C_{-1}} = 1.0$ (more precisely set at $C_{-1} = 0.25$) in synthetic datasets and the EDS dataset, and asymmetric contrast threshold initialized at ${C_{+1}}/{C_{-1}} = 1.458$ (set at $C_{-1} = 0.25$)[42] in the TUM-VIE dataset.

A.3. Experiment Setup

Our research and development efforts are deeply rooted in the principles of 3D Gaussian Splatting [26] methodology. In pursuit of advancing these technologies, we trained our models for more than 30k iterations (set at 40k iterations). This training was conducted on a NVIDIA GeForce RTX4090 GPU. Training time of synthetic scenes take 1-2 hours and that of real scenes take 1-3 hours at 40k iterations.

B. Additional Experimental Results

B.1. Qualitative Results in Synthetic Scenes

Fig. 6 shows the quantitative results of all methods for all seven synthetic scenes. The qualitative results are similar to the quantitative evaluation numbers as shown in Tab. 1. In the drums, lego, materials, and mic scenes, fine details seem to be well reconstructed. The chair and ficus reconstruction results appear to be similar details. In the hotdog case, it seems that the images produced by our method are not as well reconstructed compared to Robust-e-NeRF.

B.2. Qualitative Results on Different Synthetic Datasets

We evaluated our method on the same scenes used in EV-GS [70] from EventNeRF dataset [59] to compare grayscale results. We computed the mean values across 4 scenes(chair, ficus, hotdog and mic). As shown in Tab. 4, our approach outperforms EV-GS in terms of PSNR and SSIM. Furthermore, Fig. 7 shows qualitative results for 4 synthetic scenes in grayscale, demonstrating that the generated images are reconstructed effectively.

B.3. Qualitative Results in Real-World Scenes

Fig. 8 and Fig. 9 present additional quantitative results for the scenes 03_rocket_earth_dark, 07_ziggy_and_fuzz_hdr, 08_peanuts_running, 11_all_characters and 13_airplane, as

Figure 6. Generated images are shown, qualitatively comparing our work, event-based NeRF, and E2VID+3DGS in all synthetic scenes.

Table 4. Quantitative evaluation of mean values across the 4 synthetic scenes from [70].

Metric	PSNR ↑		SSIM ↑	
	Ours	EV-GS	Ours	EV-GS
Mean	**29.48**	26.6	**0.959**	0.925

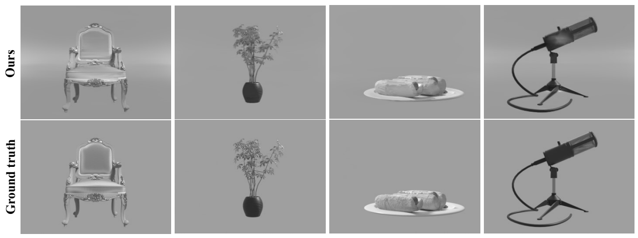

Figure 7. Generated images qualitatively comparing our method with ground truth across 4 synthetic scenes.

well as for the mocap-1d-trans, mocap-desk2 scenes, respectively. Our method demonstrates the ability to reconstruct fine detail in both real-world data. However, there is still room for improvement in the quality of reconstruction for some real-world scenes, particulary concerning floating point clouds and the back wall.

Figure 8. For each scene in the EDS dataset, we show generated images from two viewpoints alongside the ground truth image, comparing our work with event-based NeRF and E2VID+3DGS.

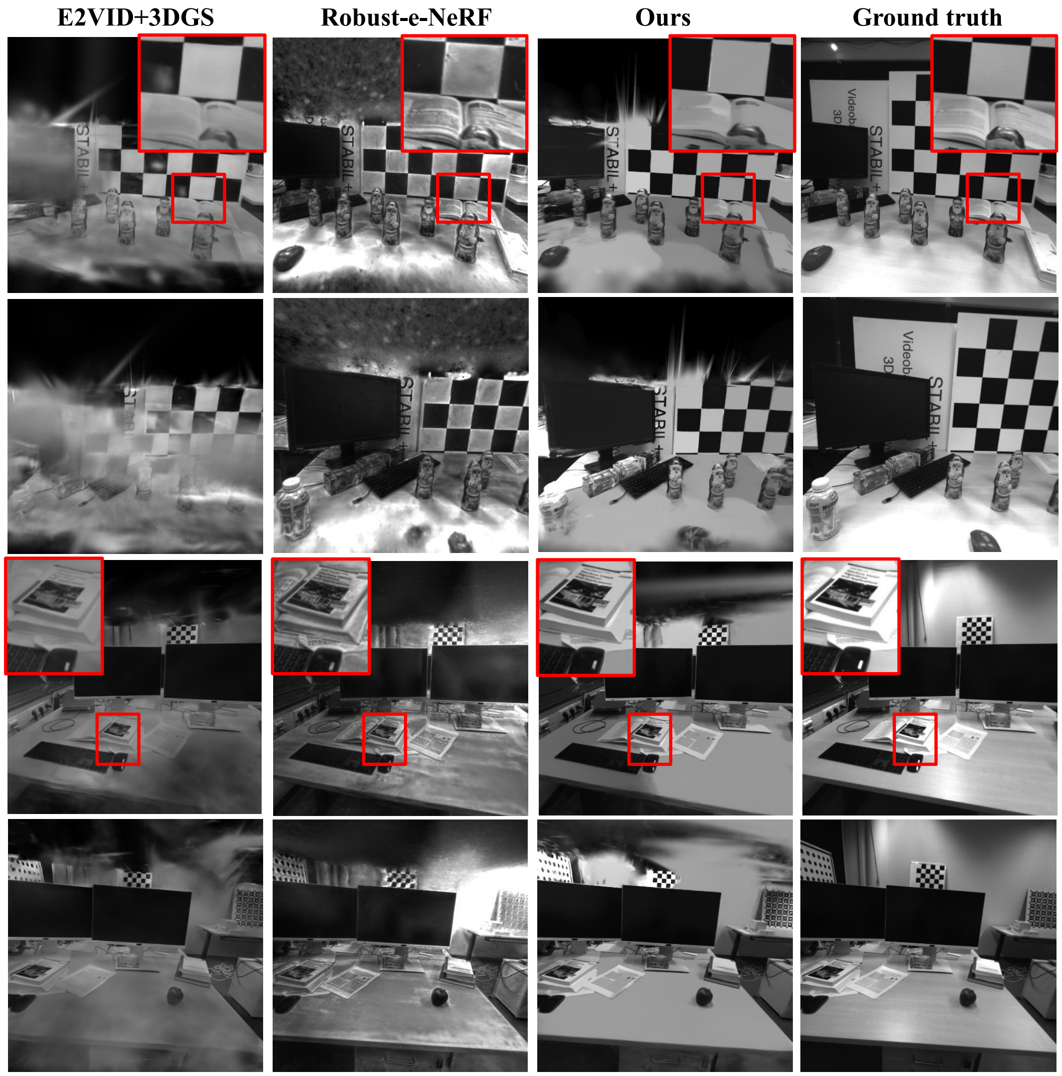

Figure 9. For each scene in TUM-VIE dataset, we show generated images from two viewpoints alongside the ground truth image, comparing our work with event-based NeRF and E2VID+3DGS.