LSE-NeRF: Learning Sensor Modeling Errors for Deblured Neural Radiance Fields with RGB-Event Stereo

Wei Zhi Tang[1] Daniel Rebain[1] Konstantinos G. Derpanis[2] Kwang Moo Yi[1]

[1]University of British Columbia
[2]York University

weiztang@cs.ubc.ca, drebain@cs.ubc.ca, kosta@yorku.ca, kmyi@cs.ubc.ca

Abstract

We present a method for reconstructing a clear Neural Radiance Field (NeRF) even with fast camera motions. To address blur artifacts, we leverage both (blurry) RGB images and event camera data captured in a binocular configuration. Importantly, when reconstructing our clear NeRF, we consider the camera modeling imperfections that arise from the simple pinhole camera model as learned embeddings for each camera measurement, and further learn a mapper that connects event camera measurements with RGB data. As no previous dataset exists for our binocular setting, we introduce an event camera dataset with captures from a 3D-printed stereo configuration between RGB and event cameras. Empirically, we evaluate our introduced dataset and EVIMOv2 and show that our method leads to improved reconstructions. Our code and dataset are available at https://github.com/ubc-vision/LSENeRF.

Figure 1. **Teaser –** We propose a deblur NeRF method that uses both RGB and event data. We focus on sensor modeling imperfections, which allows our method to effectively make use of both modalities. As shown, our method provides significantly sharper reconstructions compared to both when using only RGB and also other RGB/event NeRF baselines.

1. Introduction

Novel view synthesis has rapidly advanced since the recent advent of Neural Radiance Fields (NeRFs) [19]. NeRFs learn a Multi-Layer Perceptron (MLP) to represent a scene, and use volume rendering to realize an image at a given camera pose. Various extensions have been explored, including improvements to its efficiency [21], robustness to occluders and transients [29], tolerance to inexact input camera poses [14, 36], and applications to human-centric modeling [8]. More recently, 3D Gaussian splatting [9] has been proposed as an alternative representation based on rasterization instead of volume rendering. Whether using NeRFs or Gaussians splats, building a scene representation capable of rendering high-quality novel views depends on having clear and sharp images during training.

Consequently, there has been focused research on removing this constraint by augmenting existing neural rendering pipelines with a blur model, allowing for the acquisition of a deblurred reconstruction [12, 15, 36]. These works incorporate a physical model of blur formation under fast camera motions, and by doing so recover a non-blurred scene, as an inverse problem. While these methods deblur scenes to some extent, under large camera motions they still suffer from imperfect reconstructions; see Figure 1. This is inevitable as the blur removes details from the original inputs, and there is only so much that can be recovered without additional priors.

To further mitigate this issue, researchers have also sought additional data modalities, specifically event cameras that can complement blurry RGB images [3, 10, 25, 26]. Event streams, unlike typical RGB images, do not suffer from motion blur [5], hence incorporating them into NeRF pipelines can help deblur the scene. Although these

methods improve the clarity of NeRF reconstructions, they often focus on synthetic data [3, 10, 25, 26] and are limited to single-camera scenarios that have aligned RGB and event data [3, 25, 26]. The latter restricts the type of devices that can be used, and the resolution remains relatively low (640×480) [16, 25, 26]. In fact, for our target binocular setting, methods like E2NeRF [25] perform poorly, as demonstrated in Figure 1 and later in our experiments. Additionally, some methods, such as EvDeblurNeRF [3], are entirely unsuitable because their loss function requires precise alignment between the RGB and event sensors.

In this work, we aim to better utilize RGB and event data to achieve deblurred NeRFs by also focusing on improved sensor modeling. Specifically, (i) to model the sensor response differences between RGB and event data, we use a power mapping function, that is, the gamma function. (ii) to take into account the per-measure variations that may happen due to various camera hardware functions, we utilize per-time embeddings. The former, learning a gamma mapping, is performed together with the NeRF training process, in contrast to the conventional constant threshold adaptation [25, 26] and the normalization strategy [10, 27]. This is similar in spirit to EvDeblurNeRF [3], where the response functions are learned as MLPs, but as we show empirically, our solution, despite its simplicity, is superior. For the latter, while per-time embeddings is a common strategy used in training conventional NeRFs [33] in the non-blurred case, we find that learning these embeddings, and then substituting them with a *global* embedding that works well for all frames, is highly effective in mitigating blur—in fact, up to a level where it can outperform RGB/event NeRF baselines.

To validate our work, as no high-resolution binocular event-RGB dataset exists for deblurring applications that cover both indoor and outdoor environments, we introduce a new dataset with RGB images and corresponding events. Specifically, we 3D print a stereo casing that holds a GigE Blackfly S RGB camera [4] and a Prophesee EVK-3 HD event camera [24]. We then capture five outdoor scenes and five indoor scenes that exhibit fast and slow camera motion for training and testing, respectively. Unlike existing datasets that are captured with a single camera that provides both temporal and pixel *aligned* RGB and event streams, our dataset is binocular, with each camera providing high spatial resolution data of each modality. In more detail, our dataset provides 1440×1080 RGB images and event streams of resolution 1280×720, whereas existing datasets typically offer a resolution of 346×260 for both RGB and events. We believe our dataset will be helpful when experimenting with systems that have both sensor modalities, installed in different physical locations, for example on augmented reality headsets or vehicles. To facilitate research in this area, we will release our code, dataset and 3D printing schematics.

To summarize, our contributions are as follow:

- we introduce a novel method focused on sensor modeling errors for RGB and event-based deblur NeRF;
- to facilitate training and evaluation in our novel setting, we introduce a new dataset that provides high-resolution RGB and event streams in a binocular setup; and
- we significantly outperform the state of the art.

2. Related Work

We first discuss previous work that focus on building a clear 3D representation from either (blurry) RGB images, event data, or a combination of both. Next, we briefly discuss RGB and event stream data used for deblur NeRF.

Deblur NeRF. Since the introduction of NeRF [19], many extensions have been developed to enhance the recovery of clear 3D neural representations from blurry images. Deblur-NeRF [15] attempts to recover a clear 3D representation by explicitly modeling the blurring process as an averaging over multiple rays that potentially caused the blur, and optimizing a clear NeRF that would have created the blurry images. More recent works [13, 36, 41] have observed that camera poses from blurry images are inherently inaccurate and include camera optimization in the training process. Other works further enforce rigid camera motion [12] or attempt to learn the 2D blur kernels with sharpness priors [11]. In our work, we show that learning a per-time embedding during NeRF training, then substituting these embeddings with a *global* scene embedding also works well for further deblurring, outperforming BADNeRF [36].

Event-based NeRF. To leverage the high temporal resolution provided by event cameras, recent works have explored using event streams, with or without RGB images, to create a NeRF. Concurrently, E-NeRF [10] and EventNeRF [28] are the first to explore the use of events and color events for neural fields, respectively. Later, methods that combine RGB and events to deblur neural fields appeared [3, 25]. The camera optimization idea of deblur NeRFs has also been extended to events and RGB NeRF [26]. Other works also look into using RGB-D data in conjunction, for both static [27] and dynamic scenes [16].

Besides NeRF, as 3D Gaussian splatting [9] has gained popularity, event-based methods have also been converted to Gaussian splats [37, 39]. These include those [37] that extend the event double integral (EDI) [22] model used in event-based NeRFs [26], or those [38] that include event sampling strategies to address the accumulation period question for frame-based event representation.

RGB and event stream data. In all of the RGB and event works discussed above, except for DE-NeRF [16], all real scenes are captured using a variant of the DAVIS 346 [34] camera—a camera that captures both RGB and event

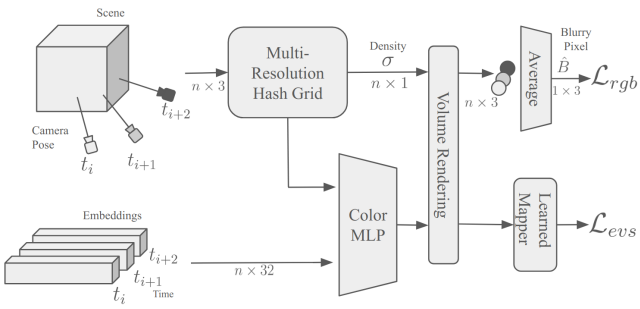

Figure 2. **Framework overview –** To render a pixel considering the camera motion blur, we pass points along the light rays of n cameras through the hash grid to obtain their density and colors. We then volume render the pixel colors for the n cameras and average them to generate the motion-blurred pixel color. We further utilize a learned mapper that maps RGB volume render to an intensity response for the event stream, which then utilizes the $n-1$ subsequent camera pairs to measure events. Note that our color Multi-Layer Perceptron (MLP) takes per-time learnable embeddings as input, to account for the sensor modeling imperfections.

streams, at a low resolution of 346×260, and furthermore, provide data where the two modalities are physically aligned. This is a strong assumption, which limits the space of device setups should one build a system that utilizes both RGB and events. Consequently, some work *require* this setup as an assumption in their loss formulations [3] that limits their applicability. In the case of DE-NeRF [16], they also utilize EVIMOv2 [2], another dataset that provides both, but *decoupled*, RGB and event streams, but this dataset is limited to indoor scenarios, has artificial texture in some scenes, and the event resolution is still lower than ours—640×480 whereas ours is 1280×720. Further, as the intent of this dataset was event-based object segmentation, optical flow, and structure-from-motion, not all scenes are useful for evaluating deblur performance. We thus collect our own high-resolution dataset, with a binocular setup that targets the deblurring task.

3. Method

Our main technical contribution are two-fold: (i) the learned per-time embedding that we later find a *global* embedding for; and (ii) the learned gamma mapper. We first discuss the forward rendering pass and then provide details about our training process. Figure 2 provides an overview of our method.

3.1. Inference

To render a scene, we use the conventional NeRF [19] paradigm with an Instant Neural Graphics Primitives (InstantNGP) [21] backbone. Specifically, given a 3D position, $\mathbf{x}$, and a direction, $\mathbf{d}$, we train a neural network that maps them into color, $\mathbf{c}$, and density, σ. Then, for a given pixel with position (x, y) with corresponding ray, $\mathbf{r}(\mathbf{t})$, in conventional NeRF, the color of this pixel, $\mathcal{C}$, is rendered by integrating along this ray between depths d_s and d_n:

$$\hat{\mathcal{C}}(\mathbf{r}) = \int_{d_s}^{d_n} \mathcal{T}(d)\sigma(\mathbf{r}(d))\mathbf{c}(\mathbf{r}(d), \mathbf{d})dd, \tag{1}$$

where $\mathcal{T}$ is the transmittance defined as

$$\mathcal{T}(d) = \exp(-\int_{d_s}^{d} \sigma(\mathbf{r}(s))ds). \tag{2}$$

While this formulation works well in the ideal case with clear training images and well modelled sensors under some chosen sensor simplification, it is insufficient when sensor modeling is critical, such as when blur is present.

Per-time embeddings. To encompass the sensor modeling errors, we introduce a per-time embedding. This strategy has previously been applied in other NeRF applications to model per-view illumination changes or to model transient objects [18], and is also suggested in NeRFStudio [33] to accommodate camera auto white balancing. Here, we utilize it to model characteristics of the sensors at each capture time that are not included in the typical camera model. To model how the world is perceived by the sensors and not the underlying geometry, we pass the per-time embedding only to the color branch of the neural field; see Figure 2. When rendering a frame at t, we add a D-dimensional learnable embedding, $\mathbf{E}_t \in \mathbb{R}^D$, to Eq. (1):

$$\hat{\mathcal{C}}(\mathbf{r}, t) = \int_{d_s}^{d_n} \mathcal{T}(d)\sigma(\mathbf{r}(d))\mathbf{c}(\mathbf{r}(d), \mathbf{d}, \mathbf{E}_t)dd. \tag{3}$$

These learnable embeddings give the network the freedom to model the sensor imperfections.

Rendering events. To obtain event data, we follow the definition of events. That is, an event occurs if the logarithm of the intensity of a pixel, $\log(I)$, between a time interval defined by two times, t_s and t_e, is greater than some threshold, ω, at some pixel x', y'. We thus write:

$$|\log(I_{t_e}) - \log(I_{t_s})| \geq \omega. \tag{4}$$

At instances when this condition is satisfied, in either direction, we obtain an event tuple

$$e = (x', y', t_e, p), \tag{5}$$

where $p \in \{-1, 1\}$ is the sign (polarity) of the log intensity change. As our focus is on static NeRF applications, these intensity changes typically arise from camera motion.

However, a straightforward application of Eq. (4), that is, setting $I = \mathcal{G}(\hat{\mathcal{C}})$, where $\mathcal{G}$ is the RGB to grayscale conversion function, assumes that the camera sensor response

functions are the same for the RGB and event cameras. To account for the sensor response differences, we use a gamma mapping:

$$I = \mathcal{G}(\hat{\mathcal{C}})^c, \tag{6}$$

where c is a learnable mapping parameter. In contrast, others learn the response function with MLPs (e.g., EvDeblurNeRF [3]) but here, we opt for a simple gamma mapping, the traditional mapping function used to model sensor response functions.

3.2. Training

To train our network we rely on two losses: one for the RGB data and one for the event data. Our total loss is given by

$$\mathcal{L} = \mathcal{L}_{rgb} + \lambda_{evs}\mathcal{L}_{evs}, \tag{7}$$

where $\mathcal{L}_*$ is the loss for the respective data, and λ_{evs} is the hyperparameter that balances the two losses. We next explain the individual losses.

RGB loss – $\mathcal{L}_{rgb}$. For the RGB image, we use the standard mean squared error loss, but with motion blur included. Specifically, following previous work [15, 36], we model the blur as a continuous camera pose, $\mathbf{P}(\mathbf{t})$. Denoting the exposure as τ, and the pathway a ray $\mathbf{r}$ takes as $\mathbf{r}(\mathbf{P}(\mathbf{t}))$, we express the rendered blurry pixel color as:

$$\hat{\mathcal{B}}(\mathbf{r}, t) = \int_{\tau} \hat{\mathcal{C}}(\mathbf{r}(\mathbf{P}(\mathbf{t})), t)dt. \tag{8}$$

In practice, this integral is replaced with a Monte Carlo estimate:

$$\hat{B} = \sum_{i=1}^{n} \hat{\mathcal{C}}(\mathbf{r}(\mathbf{P}(\mathbf{t_i})), t_i), \tag{9}$$

where $t_i \sim U[t - \frac{\tau}{2}, t + \frac{\tau}{2}]$, U is the uniform distribution and n is the number of samples. We then use these blurry RGB reconstructions to supervise the network's blurry approximation:

$$\mathcal{L}_{rgb} = \mathbb{E}_{\mathbf{r},t}\left[\left\|\hat{\mathcal{B}}(\mathbf{r}, t) - \mathcal{B}_{gt}\right\|^2\right], \tag{10}$$

where $\mathcal{B}_{gt}$ is the ground truth blurry RGB image.

Note here how the supervision in Eq. (10) already takes into account the motion blur, and thus allows the network to learn a sharp signal that corresponds to the blur. This, however, is inherently an under-constrained problem, as there are multiple sharp NeRFs that can render the same blur image. Thus, while deblurNeRF methods [12, 15, 36] provide a sharper reconstruction than NeRF, their reconstructions can still be blurry, as shown already in Figure 1.

Event loss – $\mathcal{L}_{evs}$. To supervise events, we use brightness increment images (BII) [7]. Specifically, for each pixel at position (x, y) at time t, we sum the intensity changes that occur within a time window Δt:

$$I_{ev}[x, y, t] = \omega_0 \sum_{e_i \in \mathcal{E}} p_i, \tag{11}$$

where considering the event tuples in Eq. (5), we have:

$$\mathcal{E} = \{e_i | x = x'_i, y = y'_i, |t_{e_i} - t| \leq \Delta t\}, \tag{12}$$

$\omega_0 = 0.2$ is the default value used previously [6, 25], and $\Delta t = 2.5$ ms is the optimal value found in prior work [17].

With the brightness increment image, $I_{ev}[x, y, t]$, we supervise our network with the conventional event loss [10] that minimizes the difference between the events that our network renders and the ground truth events. With the gamma mapped image, I, from Eq. (6), we write:

$$\mathcal{L}_{evs} = \mathbb{E}_{\mathbf{r},(t_s,t_e)}\left[\left\|\log(I(\mathbf{r}, t_e)) - \log(I(\mathbf{r}, t_s)) - I_{ev}\left[x_{\mathbf{r}}, y_{\mathbf{r}}, \frac{t_e + t_s}{2}\right]\right\|^2\right], \tag{13}$$

where t_e and t_s are the event times for each event stored in $I_{ev}[x, y, t]$, and $x_{\mathbf{r}}$ and $y_{\mathbf{r}}$ are the pixel coordinates corresponding to a ray $\mathbf{r}$.

Finding the *global* embedding. Finally, once the network is trained, we note that to render a novel view, we do not have the per-time embedding for a new frame. We thus find a *global* embedding, $\mathbf{E}_* \in \mathbb{R}^D$, that works well for all frames. To achieve this, we freeze the entire network except the global embedding weights and then retrain for a few thousand steps to determine the global embedding for evaluation. In doing so, we utilize only the RGB loss, $\mathcal{L}_{rgb}$, to find the global embedding, as we empirically found that using the event loss, $\mathcal{L}_{evs}$, did not work well—we ablate this in Section 5.4.

Camera optimization. Camera poses recovered from blurry images are often inaccurate. To mitigate this, we allow *both* the RGB and the event camera poses to be optimized. Since we do not have camera pose for BIIs in between RGB frames, we initialize these poses as linear interpolations and optimize them. We parameterize camera poses with exponential maps as in BADNeRF [36]. We then optimize the camera poses directly with gradients from the respective losses. In more detail:

- **RGB cameras:** As the RGB loss utilizes blurry images, it is important to consider 'intermediate' cameras for which we do not have ground-truth images for. As in BADNeRF [36], we assume a linear trajectory for the RGB camera using the exponential map parameterization. We then optimize the poses of the cameras that we have ground-truth images for directly with gradients from the RGB loss.

Figure 3. **Sample RGB frames from each scene in our dataset –** Our dataset consists of **(top row)** five outdoor scenes and **(bottom row)** five indoor scenes. The substantial image blur is caused by rapid camera movements.

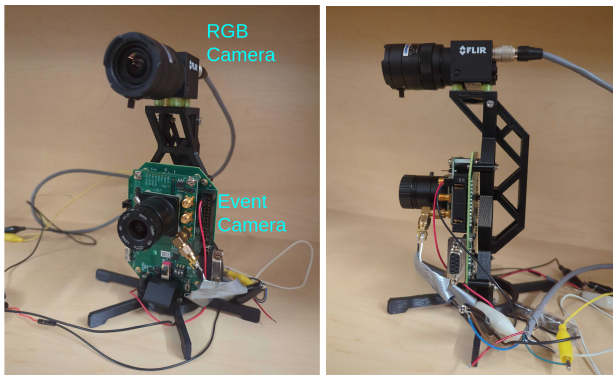

Figure 4. **Capture rig –** We 3D print a stereo casing that holds a GigE Blackfly S camera and a Prophesee EVK-3 HD camera.

- **Event cameras:** For the event camera positions we discretize the trajectory into individual cameras for each brightness increment image. We initially set the event camera pose as linear interpolations of the RGB camera pose and optimize them according to the event loss.

As the initial reconstructions are unstable, we only enable camera optimization after the first 10K steps of training.

4. Data Collection

To validate our method, we collect our own dataset containing five indoor and five outdoor forward facing scenes.

Capture rig. While various other datasets exist, they are either synthetic, or contain data collected from variants of the DAVIS 346 camera that captures aligned RGB and events. The assumption of aligned streams is a strong one, limited to certain devices, one that our method relaxes. Furthermore, typically, these devices operate on a lower spatial resolution to support both modalities. Hence, we opt for a more versatile capture setup, where, as shown in Figure 4, we 3D print a stereo casing and use a GigE Blackfly S [4] camera to capture RGB images and a Prophesee EVK-3 HD camera [24] to capture events. Each device provides high-resolution data, with the RGB camera capturing 1440×1080 images and the event camera capturing 1280×720 events. The two devices are temporarily hardware-synced by triggers sent from the RGB camera.

Capture protocol. We capture our dataset by hand-holding the stereo rig in various locations. Each scene is approximately 20 seconds long, with the first 13 seconds being blurry with fast swinging motions and the last seven seconds exhibiting slow stable camera movement to provide clear frames for evaluation.

Calibration. To calibrate the two cameras (i.e., recover the intrinsics and relative extrinsics), we use the standard checkerboard calibration pattern and slowly move our stereo rig to create events around the edges of the checkerboard.

We use the model from E2Calib [20] to map event streams to event images and then stereo calibration from OpenCV [1] to recover the intrinsics and extrinsics.

Camera pose. To recover the camera poses, we use COLMAP [30–32]. To allow COLMAP to work well with blurry images, we loosen its constraints on initialization and match filtering. Specifically, we reduce the initialization constraint to have 28 inliers with a minimum of five degrees. We also reduce the observation filter constraint by setting the reprojection error to 12 pixels and reducing the triangulation filter's minimum angle to 0.08 radian.

Furthermore, as the extrinsics between the two cameras for our calibrated stereo rig are provided in their own scene scale (specifically metric scale), we need to also recover the scene scale. To do this, we manually triangulate three points from the RGB images and annotate the corresponding keypoints in the event brightness increment images (BII).

We then use Powell's method [23] to minimize the reprojection error of the triangulated points to the corresponding keypoints in the brightness increment images, which gives

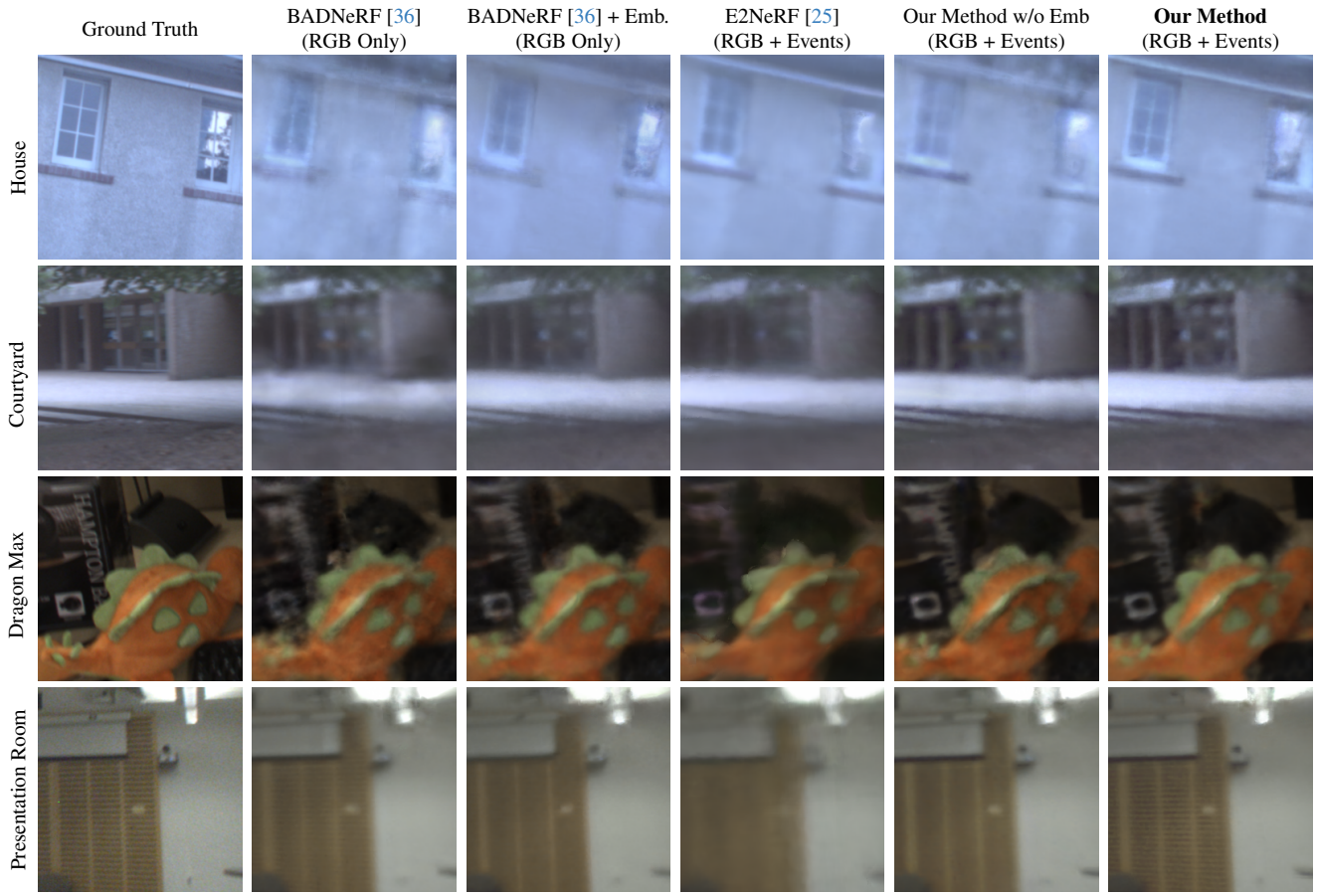

Figure 5. **Qualitative examples –** We show qualitative examples of zoomed-in reconstruction cutouts. Our method provides the sharpest reconstructions. Interestingly, BADNeRF [36], combined with our embedding strategy, also provides clear results, being on par or slightly better than even when event data is used. The best results, however, are obtained with our method where, RGB images are used together with event data.

us the scene scale. Finally, we obtain the event camera poses by applying the relative extrinsics to the RGB camera poses, with the scene scale.

5. Results

5.1. Experimental setup

Dataset. In addition to our dataset, we further utilize the EVIMOv2 [2] dataset for evaluation. EVIMOv2 [2] is originally designed for object segmentation, optical flow and Structure-from-Motion (SFM) tasks. As it is not a dataset designed for deblurring tasks, we manually examine all images in the sequences chosen for our experiments and manually select clear images to be held out for testing. We choose three sequences from the dataset that have a sufficient number of clear images: depth_var_1_lr_000000, scene7_00_000001, scene8_01_000000. Because the dataset is not designed to test debluring, we note that the selected clear images may still contain slight blur.[1]

Baselines. We compare our method with the following:

- **BADNeRF [36]:** We reimplement their method in our framework but without the exposure time optimization, as in our data we have the exact exposure time. We use this baseline to compare against the case when we do not use the event data. See the supplemental for method details.
- **E2NeRF [25]:** We use the official code to demonstrate the performance of a recent event-based NeRF method.
- **Ablations:** We further compare our method against ablations, such as without the embedding, without the mapper, and without the camera optimization to demonstrate the effectiveness of each component.

Metrics. We use the following standard image quality evaluation metrics: Peak Signal-to-Noise Ratio (PSNR), Struc-

[1] To ensure the reproducibility of our results, we will release the data processing code for the selected scenes.

	Courtyard			Bag			House			Engineer Building			Bicycle			Average		
	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓
BADNeRF [36]	0.712	21.618	0.475	0.735	26.450	0.380	0.734	22.670	0.457	0.806	24.169	0.372	0.759	25.775	0.398	0.749	24.137	0.416
BADNeRF [36] + Emb.	0.716	*22.359*	*0.451*	*0.754*	*26.895*	**0.357**	0.751	*24.227*	*0.420*	*0.823*	*25.446*	*0.339*	0.760	*26.363*	0.398	*0.761*	*25.058*	*0.393*
E2NeRF [25]	0.691	21.904	0.592	0.696	26.172	0.508	0.757	23.754	0.501	0.785	24.37	0.451	0.726	24.637	0.574	0.731	24.168	0.525
Our method w/o Emb.	*0.719*	22.314	0.472	0.749	27.118	0.374	*0.758*	23.855	0.442	0.819	25.179	0.346	*0.760*	26.278	*0.390*	0.761	24.949	0.405
Our method	**0.723**	**22.660**	**0.442**	**0.760**	**27.575**	*0.367*	**0.778**	**25.047**	**0.416**	**0.827**	**25.461**	**0.324**	**0.778**	**26.620**	**0.388**	**0.772**	**25.473**	**0.388**
	Grad Lounge			Presentation Room			Teddy Grass			Dragon Max			Lab			Average		
	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓
BADNeRF [36]	0.737	24.918	0.517	0.692	24.706	0.503	0.673	23.485	0.572	0.826	25.330	0.373	0.816	22.389	0.356	0.749	24.166	0.464
BADNeRF [36] + Emb.	**0.775**	26.261	*0.440*	0.692	*25.623*	*0.475*	*0.699*	*25.120*	**0.552**	0.850	*26.195*	0.328	*0.841*	22.970	0.319	*0.771*	*25.250*	*0.423*
E2NeRF [25]	0.707	24.927	0.564	0.634	23.24	0.613	0.61	23.363	0.681	0.777	23.921	0.427	0.746	21.667	0.487	0.695	23.424	0.554
Our method w/o Emb.	0.760	*26.506*	0.471	*0.699*	25.338	0.510	0.692	25.128	0.601	*0.850*	26.123	*0.318*	0.841	*22.986*	*0.310*	0.768	25.216	0.442
Our method	*0.773*	**26.589**	**0.434**	**0.610**	**25.845**	**0.465**	**0.701**	**25.618**	*0.570*	**0.856**	**26.349**	**0.307**	**0.855**	**23.144**	**0.288**	**0.777**	**25.509**	**0.413**

Table 1. **Quantitative results for our dataset –** We report the quantitative metrics for our method and the baselines on the test views. We report results both for **(top rows)** outdoor scenes and **(bottom rows)** indoor scenes. We mark **best** and *second-best* results in each column. Our method performs best in terms of all metrics. Interestingly, BADNeRF with our embedding strategy performs second-best, although it is trained only on RGB images. This demonstrates using per-time embeddings to measure sensor modeling imperfections helps significantly. In Sec. 5.4, we further show how one obtains the global embedding for inference is also critical.

tural Similarity Index (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS) [40]. As in prior work [14], since we perform camera optimization, we perform camera pose optimization for all methods to align novel views before calculating metrics for fair comparison.

Implementation details. We implement our method with Nerfstudio [33]. We use the instant NGP [21] backbone with the default configurations and train all methods for 200k iterations to ensure convergence. We empirically set λ_{evs}=1 for the event loss. After training, to find the global embeddings, we perform 3k iterations of optimization. When computing the metrics, we optimize the camera pose for 6k iterations. We choose $n = 4$ in Eq. (9) to make our RGB and event loss have a similar batch size of 597 and 588, respectively, which is the largest batch size that fits on our NVIDIA 3090 GPU with 24GB VRAM. Following Nerfstudio [33], we choose D=32 as our per-time embedding dimension; we ablate our choice in the supplement.

5.2. Results on our dataset

We show example qualitative results in Fig. 5 and a quantitative summary in Tab. 1. As shown in Tab. 1, our method with the embeddings and the learned mapper performs best. It is interesting to note that our embedding strategy, combined with BADNeRF, already outperforms the state of the art, up to a degree where it outperforms methods that use events, although it uses only RGB data. This highlights the importance of considering camera sensor modeling errors. Still, our method of using events together with the per-time embeddings and the learned gamma mapping performs best.

Besides the quantitative results, the differences between the reconstructions are more pronounced in Fig. 5. With our method, sharper reconstructions are obtained.

5.3. Results on the EVIMOv2 [2] dataset

We also evaluate our method on EVIMOv2 [2]. We report our results in Tab. 2[2]. It is interesting to note that while our method performs best, once our per-time embedding strategy is used, RGB-only reconstruction with BADNeRF [36] works almost as well as using events without per-time embeddings. This further demonstrates the importance of incorporating these sensor modeling imperfections.

5.4. Ablation study

We now ablate our design choices. For all our ablations, we use one indoor scene ('Dragon Max') and one outdoor scene ('Courtyard') from our dataset.

Obtaining a good *global* embedding is critical. We first examine our learned per-time embeddings, which we already demonstrated its effectiveness in Sec. 5.2 and Sec. 5.3, but this time, focusing on how one obtains the embeddings. In Tab. 3 we report the performance of BADNeRF [36] with and without learned per-time embeddings, including our strategy of finding a global embedding that minimizes the RGB loss, and the typical default strategy of training with embeddings, which is to use a zero embedding. As shown, our strategy significantly improves reconstruction quality, while the zero embedding offers only a minor improvement in SSIM and LPIPS, with negligible impact to PSNR. Combining the event loss with the RGB loss is detrimental, likely because event data is sparse, whereas

[2]E2NeRF [25] is excluded as the official implementation failed to reconstruct a clear scene despite our best efforts. We suspect this could be because E2NeRF [25] is MLP-based using positional encodings [35], which requires careful scaling of the camera coordinates and the scene content at the center. We tried various scaling and centering approaches, but none produced reasonable results despite our best efforts.

	depth_var_1_lr_000000			scene7_00_000001			scene8_01_000000			Average		
	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓
BADNeRF [36]	0.819	23.358	0.252	0.786	20.726	0.396	0.754	20.144	0.402	0.786	21.410	0.350
BADNeRF [36] + Emb.	0.852	*24.912*	0.217	*0.825*	22.126	*0.339*	0.804	21.368	0.322	*0.827*	22.802	0.292
Our method w/o Emb.	*0.856*	24.486	*0.212*	0.819	*22.439*	0.341	*0.805*	*21.619*	*0.321*	*0.827*	*22.848*	*0.291*
Our method	**0.880**	**26.152**	**0.177**	**0.855**	**24.247**	**0.288**	**0.839**	**22.927**	**0.257**	**0.858**	**24.442**	**0.241**

Table 2. **Quantitative results for the EVIMOv2 [2] dataset –** We report the quantitative metrics for our method and the baselines on the test views. Our method performs best.

Method	SSIM↑	PSNR↑	LPIPS↓
BADNeRF [36]	0.769	23.474	0.424
BADNeRF [36] + Zero Emb.	0.776	23.440	0.403
BADNeRF [36] + Our Emb.	*0.783*	24.277	0.389
Ours $\mathcal{L}_{rgb} + \mathcal{L}_{evs}$	*0.783*	*24.390*	*0.387*
Ours $\mathcal{L}_{rgb}$	**0.790**	**24.504**	**0.374**

Table 3. **Ablation: per-time embedding –** We report the quantitative metrics with various strategies of using per-time embeddings. Using our strategy to obtain the *global* embedding significantly improves performance while simply training with embeddings and then using a zero embedding, which is the default strategy for NeRFStudio [33] only shows a minor improvement. Furthermore, training only with the RGB loss, Eq. (10), to find the embeddings provides better results than using also the event loss.

Method	SSIM↑	PSNR↑	LPIPS↓
Ours w/o events	0.783	24.277	0.389
Ours + Linear mapping	0.743	22.995	0.503
Ours + Normalized linear mapping	*0.783*	24.302	0.400
Ours + MLP mapping [3]	*0.783*	*24.350*	*0.386*
Ours + MLP mapping (both) [3]	0.777	24.257	0.411
Ours + Gamma mapping	**0.790**	**24.504**	**0.374**

Table 4. **Ablation: mapper –** We report quantitative metrics for our gamma mapping (Eq. (6)), MLP mapping [3], and simply no mapping (linear) with and without event normalization [10]. Using an MLP mapper on either events or both events and RGB helps, but it does not perform as well as our simple gamma mapper.

finding an embedding that accurately represents the scene requires considering all parts of the scene. We further remind the reader that in Tables 1 and 2 our *global* embeddings provide a significant boost in performance.

Gamma mapping outperforms MLP mapping. We report various mapping options for linking between RGB and event data in Tab. 4. Our learned gamma mapping provides a simple yet effective mapping strategy. Using a simple linear mapping, that is, setting a constant default threshold for events as $w = 0.2$ performs significantly worse. Other mapping strategies, such as using an MLP [3] or normalizing the events [10] do not perform as well as ours.

Method	SSIM↑	PSNR↑	LPIPS↓
BADNeRF [36] w/o cam. opt.	0.702	20.781	0.527
BADNeRF [36]	*0.769*	*23.474*	*0.424*
Our method w/o cam. opt.	0.755	23.227	0.452
Our method	**0.790**	**24.504**	**0.374**

Table 5. **Ablation: camera optimization –** We report quantitative metrics with and without camera optimization. Camera optimization is critical to achieving the best performance.

Camera optimization is important for blurry scenes. We further look into the importance of camera optimization in Tab. 5. As pointed out in previous work [36], camera optimization is especially important when it comes to deblur settings. For our data, this also holds. Camera optimization during training significantly improves rendering quality.

6. Conclusion and Future Work

We introduced a NeRF framework that utilizes both RGB and event data to reconstruct a deblurred scene from data captured with fast camera motion. Distinct from prior work, our cameras are separate, decoupled sensors rather than aligned. Central to our approach is learning sensor imperfections through a data-driven manner using per-time embeddings, combined with a gamma mapping between RGB and event data. To evaluate our method in this unique stereo capture setting, we introduced a new dataset, consisting of five indoor and outdoor captures each. Empirically, we showed that our method outperforms the state of the art.

Limitations. Our data collection process currently requires manual human annotation. While this involves selecting three or more points to properly scale the scene to the calibrated extrinsics, it remains a manual effort. Additionally, our work relies on NeRF, while much of the field is transitioning to the 3D Gaussian splatting [9] framework. Nonetheless, we believe our findings are *backbone agnostic* and will also benefit 3D Gaussian splatting-based methods.

References

[1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 5

[2] Levi Burner, Anton Mitrokhin, Cornelia Fermüller, and Yiannis Aloimonos. Evimo2: An event camera dataset for motion segmentation, optical flow, structure from motion, and visual inertial odometry in indoor scenes with monocular or stereo algorithms, 2022. 3, 6, 7, 8, 1

[3] Marco Cannici and Davide Scaramuzza. Mitigating motion blur in neural radiance fields with events and frames. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9286–9296, 2024. 1, 2, 3, 4, 8

[4] FLIR. Blackfly s gige. https://www.flir.ca/products/blackfly-s-gige/?vertical=machine+vision&segment=iis, 2024. Accessed: 2024-08-12. 2, 5

[5] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J. Davison, Jörg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based vision: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(1):154–180, 2022. 1

[6] Daniel Gehrig, Mathias Gehrig, Javier Hidalgo-Carrió, and Davide Scaramuzza. Video to events: Recycling video datasets for event cameras. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2020. 4

[7] Daniel Gehrig, Henri Rebecq, Guillermo Gallego, and Davide Scaramuzza. EKLT: Asynchronous Photometric Feature Tracking Using Events and Frames. *Int. J. Comput. Vis.*, 128(3):601–618, 2020. 4

[8] Wei Jiang, Kwang Moo Yi, Golnoosh Samei, Oncel Tuzel, and Anurag Ranjan. NeuMan: Neural human radiance field from a single video. In *European Conference on Computer Vision*, pages 402–418. Springer, 2022. 1

[9] 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. 1, 2, 8

[10] Simon Klenk, Lukas Koestler, Davide Scaramuzza, and Daniel Cremers. E-NeRF: Neural radiance fields from a moving event camera. *IEEE Robotics and Automation Letters*, 2023. 1, 2, 4, 8

[11] Byeonghyeon Lee, Howoong Lee, Usman Ali, and Eunbyung Park. Sharp-NeRF: Grid-based fast deblurring neural radiance fields using sharpness prior. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3709–3718, 2024. 2

[12] Dogyoon Lee, Minhyeok Lee, Chajin Shin, and Sangyoun Lee. DP-NeRF: Deblurred neural radiance field with physical scene priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12386–12396, 2023. 1, 2, 4

[13] Dongwoo Lee, Jeongtaek Oh, Jaesung Rim, Sunghyun Cho, and Kyoung Mu Lee. ExBluRF: Efficient radiance fields for extreme motion blurred images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17639–17648, 2023. 2

[14] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. BARF: Bundle-adjusting neural radiance fields. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5741–5751, 2021. 1, 7

[15] Li Ma, Xiaoyu Li, Jing Liao, Qi Zhang, Xuan Wang, Jue Wang, and Pedro V Sander. Deblur-NeRF: Neural radiance fields from blurry images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12861–12870, 2022. 1, 2, 4

[16] Qi Ma, Danda Pani Paudel, Ajad Chhatkuli, and Luc Van Gool. Deformable neural radiance fields using RGB and event cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3590–3600, 2023. 2, 3

[17] Ana I Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso García, and Davide Scaramuzza. Event-based Vision Meets Deep Learning on Steering Prediction for Self-Driving Cars. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 4

[18] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the wild: Neural radiance fields for unconstrained photo collections. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7210–7219, 2021. 3

[19] 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. 1, 2, 3

[20] Manasi Muglikar, Mathias Gehrig, Daniel Gehrig, and Davide Scaramuzza. How to calibrate your event camera. In *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*, 2021. 5

[21] 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. 1, 3, 7

[22] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6820–6829, 2019. 2

[23] Michael JD Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, 1964. 5

[24] Prophesee. Event-based evk 3. https://www.prophesee.ai/event-based-evk-3/, 2024. Accessed: 2024-08-12. 2, 5

[25] Yunshan Qi, Lin Zhu, Yu Zhang, and Jia Li. E^2NeRF: Event enhanced neural radiance fields from blurry images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13254–13264, 2023. 1, 2, 4, 6, 7

[26] Yunshan Qi, Lin Zhu, Yifan Zhao, Nan Bao, and Jia Li. Deblurring neural radiance fields with event-driven bundle adjustment. *arXiv preprint arXiv:2406.14360*, 2024. 1, 2

[27] Delin Qu, Chi Yan, Dong Wang, Jie Yin, Qizhi Chen, Dan Xu, Yiting Zhang, Bin Zhao, and Xuelong Li. Implicit event-

rgbd neural slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19584–19594, 2024. 2

[28] Viktor Rudnev, Mohamed Elgharib, Christian Theobalt, and Vladislav Golyanik. EventNeRF: Neural radiance fields from a single colour event camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4992–5002, 2023. 2

[29] Sara Sabour, Suhani Vora, Daniel Duckworth, Ivan Krasin, David J Fleet, and Andrea Tagliasacchi. RobustNeRF: Ignoring distractors with robust losses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20626–20636, 2023. 1

[30] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[31] Johannes Lutz Schönberger, True Price, Torsten Sattler, Jan-Michael Frahm, and Marc Pollefeys. A vote-and-verify strategy for fast spatial verification in image retrieval. In *Asian Conference on Computer Vision (ACCV)*, 2016.

[32] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 5

[33] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023. 2, 3, 7, 8

[34] Gemma Taverni, Diederik Paul Moeys, Chenghan Li, Celso Cavaco, Vasyl Motsnyi, David San Segundo Bello, and Tobi Delbruck. Front and back illuminated dynamic and active pixel vision sensors comparison. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 65(5):677–681, 2018. 2

[35] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 7

[36] Peng Wang, Lingzhe Zhao, Ruijie Ma, and Peidong Liu. BAD-NeRF: Bundle adjusted deblur neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4170–4179, 2023. 1, 2, 4, 6, 7, 8, 3

[37] Yuchen Weng, Zhengwen Shen, Ruofan Chen, Qi Wang, and Jun Wang. EaDeblur-GS: Event assisted 3D deblur reconstruction with Gaussian splatting. *arXiv preprint arXiv:2407.13520*, 2024. 2

[38] Tianyi Xiong, Jiayi Wu, Botao He, Cornelia Fermuller, Yiannis Aloimonos, Heng Huang, and Christopher A Metzler. Event3DGS: Event-based 3D Gaussian splatting for fast egomotion. *arXiv preprint arXiv:2406.02972*, 2024. 2

[39] 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. 2

[40] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 7

[41] Lingzhe Zhao, Peng Wang, and Peidong Liu. BAD-Gaussians: Bundle adjusted deblur Gaussian splatting. *arXiv preprint arXiv:2403.11831*, 2024. 2