EventNeRF: Neural Radiance Fields from a Single Colour Event Camera

Viktor Rudnev[1, 2] Mohamed Elgharib[1] Christian Theobalt[1] Vladislav Golyanik[1]
[1]Max Planck Institute for Informatics, SIC [2]Saarland University, SIC

Abstract

Learning coordinate-based volumetric 3D scene representations such as neural radiance fields (NeRF) has been so far studied assuming RGB or RGB-D images as inputs. At the same time, it is known from the neuroscience literature that human visual system (HVS) is tailored to process asynchronous brightness changes rather than synchronous RGB images, in order to build and continuously update mental 3D representations of the surroundings for navigation and survival. Visual sensors that were inspired by HVS principles are event cameras. Thus, events are sparse and asynchronous per-pixel brightness (or colour channel) change signals. In contrast to existing works on neural 3D scene representation learning, this paper approaches the problem from a new perspective. We demonstrate that it is possible to learn NeRF suitable for novel-view synthesis in the RGB space from asynchronous event streams. Our models achieve high visual accuracy of the rendered novel views of challenging scenes in the RGB space, even though they are trained with substantially fewer data (i.e., event streams from a single event camera moving around the object) and more efficiently (due to the inherent sparsity of event streams) than the existing NeRF models trained with RGB images. We will release our datasets and the source code, see `https://4dqv.mpi-inf.mpg.de/EventNeRF/`.

1. Introduction

Regressing scene representations that are 3D-consistent by design from a set of multi-view or monocular RGB images is a challenging machine learning task. Existing approaches falling into this category, such as scene representation networks (SRNs) [42] or neural radiance fields (NeRF) [29] (and improvements thereof [22, 52, 16, 51, 34, 47, 39, 2]), enable photo-realistic novel view synthesis of challenging scenes. This research field progressed significantly in recent few years, revolutionising many computer vision and computer graphics problems through neural-based machine processing [44].

The vast body of research results in computer vision was published assuming traditional RGB cameras. At the same time, they have many limitations such as low dynamic range processing, large memory consumption and large computational demands. Biologically inspired vision sensors such as event cameras or dynamic vision sensors (DVS)—despite their growing maturity from the hardware perspective—are rarely considered. In contrast to conventional RGB cameras that record synchronous images at a pre-defined framerate, DVS record events, *i.e.,* asynchronous per-pixel brightness changes [9]. This operational principle mimics how the visual system of animals and humans (retinas) operate, *i.e.,* they record changes in the brightness of the observed regions rather than absolute brightness of the entire field of view [3]. This also means that the brains of living creatures are tailored to process such asynchronous signals (events) rather than synchronous RGB images, also when they build 3D models of the observed environments. We are intrigued and inspired by these facts.

The research question we are addressing in this work is, whether *an event stream from a DVS moving around the scene is sufficient to reconstruct a 3D representation of a static scene, e.g., NeRF.* Using a single event camera instead of an RGB camera have several merits, including its ability to handle strong motion blur that exist in RGB images and could cause several RGB-based computer vision methods to fail. Event cameras, on the other hand, do not suffer from motion blur and have already brought valuable contributions in the important and long-standing problems of hand [38] and body [49] tracking in this regards. Event cameras also have a higher dynamic range compared to RGB cameras and, thus, are able to handle dark scenes, which is also a fundamental limitation of traditional intensity-based methods. Other advantages event cameras can bring to 3D scene representation over traditional RGB cameras include lower memory and power consumption and lower computational demand. To summarise, the primary technical contributions of this paper are as follows:

1) EventNeRF: the first approach for inferring NeRF from a monocular colour event stream that enable novel view synthesis in the RGB space at test time. Our method is designed for pure event-based supervision, during which we preserve the resolution of the individual RGB event channels (while avoiding demosaicing; see Sec. 3.4).

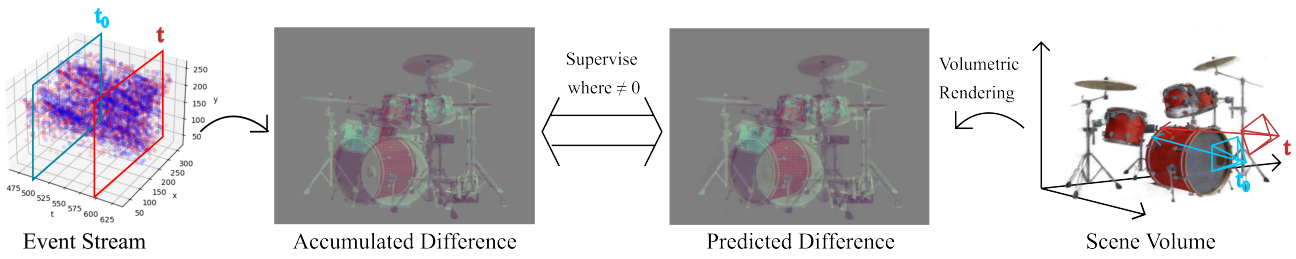


Figure 1: **Overview of the proposed method which learns a NeRF volume of a static object using just a moving event camera.** Our method connects the observed events $\{\mathbf{E}_i\}_{i=1}^{N}$ with the differences between rendered views (multiplied by Bayer colour filter) via an event-based integral (see Sec. 3.2). The projections are estimated via a fully differentiable volumetric renderer (see Sec. 3.1), thus allowing learning the NeRF volume in a self-supervised manner.

2) A ray sampling strategy that allows for data-efficient training and that is tailored to events.

3) An experimental evaluation protocol for the task of novel view synthesis from event streams in the RGB space; it establishes the benchmark for future work.

We demonstrate NeRF estimation from a single colour event camera in scenarios that would not be conceivable with an RGB camera (*e.g.,* with high speed movements, motion blur or insufficient lighting).

2. Related Work

3D Scene Representation Learning. 3D scene representation learning is a long-standing problem in machine learning (ML) [10, 7, 28, 42, 24, 29]. Existing works in this domain produce meshes [46, 11, 17] and multi-plane images (MPIs) [28, 55, 21]. Such representations can be learnt from 2D images [45, 55, 21], but the methods suffer from several limitations, including the inability to capture fine geometrical details (meshes) and being mostly bounded to 2.5D novel view synthesis (MPIs).

In contrast, the recently introduced implicit neural network-based representations learn to map the 3D spatial coordinates of a static scene to a continuous function of the local scene properties [27, 33, 42]. An alternative is voxel grid models [24, 41], where a 3D volume is discretised into 3D voxels. Neural radiance fields (NeRF) [29] models a scene through MLP as a continuous function of the scene radiance and volume density learnt from a set of 2D RGB images. At test time, NeRF accepts arbitrary 3D camera pose and viewing direction to render novel views. Due to its simplicity and high accuracy, multiple follow-up works adopted NeRF for such applications as reenactment [8, 23, 54], scene appearance and light editing [25, 4, 43], 3D shape modelling [12, 6, 5, 40] and others [50, 31, 44]. It was also shown that NeRF can operate on RGB-D inputs [1].

Despite that, multiple limitations remain, such as slow training and convergence for fine details, sensitivity to the blur in the input images, and the multi-view setting assumption. No methods exist that can learn volumetric 3D scene representations such as NeRF from *event streams*, although they can address the above-mentioned limitations, as this paper shows. Moreover, HVS can produce mental 3D models from signals that are similar to events and does so effectively. Thus, we believe that an ML approach should be capable of doing it as well, and, in contrast to reviewed works, we use an *event camera* to learn NeRF for photo-realistic novel view synthesis in the RGB space. Similarly to NeRF, we use volumetric rendering; in contrast, our EventNeRF accepts events and supervises the volumetric density and the colours by the accumulated per-colour-channel differences. Next, since events are inherently sparse, we accelerate the training by supervising only the parts of the learnt neural volume that cause events in the image space.

Novel View Synthesis from Event Streams. Unique properties of event cameras (*i.e.,* low latency, ultra-low power consumption, high dynamic range and no motion blur) motivated their usage in Simultaneous Localisation and Mapping (SLAM) [48, 19, 35] and 3D reconstruction of hands [38, 30] and human bodies [49, 58]. Several event-based SLAM methods [19, 35] rely on explicit feature matching in the event space and reconstruct sparse 3D models of environments from a single event stream. As a result, novel views generated by them are sparse, colourless and mostly contain edges and details causing events. The same observation applies to the methods using stereo event cameras that further densify the 3D reconstructions [57, 56].

Several works reconstruct non-rigid objects in 3D from an event camera. The meshes tracked by EventCap [49] can be used to render novel views of the target human. However, it requires grayscale images as input and a 3D scan of the actor in advance. Thus, novel views of the actor do not contain new details observed in the input event streams (instead, the initial 3D scan is deforming). The analysis-by-synthesis method of Nevhi *et al.* [30] operates on events only and supports arbitrary non-rigid objects but still requires a 3D mesh of

the target object. EventHands [38] regresses sparse 3D hand keypoints and, thus, cannot be used for novel view synthesis of dense hand surfaces. *In stark contrast to these works, we learn—for the first time—an implicit 3D scene representation from event streams that enables photo-realistic novel view synthesis in the RGB space.*

3. Method

Our aim is to learn a neural 3D scene representation $\mathbf{C}(\mathbf{o}, \mathbf{d})$ for a static scene from colour events $\{\mathbf{E}_i\}_{i=1}^{N}$. The model $\mathbf{C}(\mathbf{o}, \mathbf{d})$ is a function a of camera position $\mathbf{o} \in \mathbb{R}^3$ and its viewing direction $\mathbf{d} \in \mathbb{R}^3, \|\mathbf{d}\| = 1$, thus allowing novel view synthesis during test in the RGB space. To learn the model, we assume the camera extrinsics $\mathbf{P}_i = [\mathbf{R_i}|\mathbf{t_i}] \in \mathbb{R}^{3\times4}$ and intrinsics $\mathbf{K}_i \in \mathbb{R}^{3\times3}$ are known for each event window. We also assume a constant colour background with a known value $\alpha \in \mathbb{R}^3$ to rectify the reconstructed image brightness and colour balance. Its known value is not a strict requirement as the equivalent brightness and colour balance corrections can be done after the rendering. Our model is learned in a self-supervised manner, by comparing the approximate difference between views computed by summing the observed events polarities $\mathbf{E}(t_0, t)$ against the difference between the predicted views $\mathbf{L}(t) - \mathbf{L}(t_0)$ (Secs. 3.3, 3.4). To this end, we use Neural Radiance Fields of Mildenhall *et al.* [29] as our 3D scene representation, where each point in 3D-space $\{\mathbf{x}_i\}_{i=1}^{M}$ is represented via a volume density σ_i and a radiance $\mathbf{c}_i$ (Sec. 3.1). We then use volumetric rendering [29] to produce 2D projections of the learned $\mathbf{C}(\mathbf{o}, \mathbf{d})$, and establish a connection with the observed events via an event-based integral (Sec. 3.2). Since the volumetric rendering is fully differentiable, our model $\mathbf{C}(\mathbf{o}, \mathbf{d})$ can be learned in a completely self-supervised manner using events only. Our model is designed for rendering of novel view in the RGB space at test time; see Fig. 1 for an overview. Lastly, we apply regularisation techniques and introduce an event-based ray sampling strategy that allows efficient learning of the model (Secs. 3.5-3.8).

3.1. Neural Radiance Fields (NeRF) and Volumetric Rendering

NeRF [29] learns a volumetric model of a static scene from multiple 2D RGB inputs captured at different camera viewpoints. It uses MLP to map each point in 3D space into a continuous field of volume density and radiance. To render a 2D image, a ray is shot from a camera location along a specific direction, and observations are projected into 2D via volumetric rendering [18, 26].

Given a ray $\mathbf{r}(q) = \mathbf{o} + q\mathbf{d}$, where $\mathbf{o} \in \mathbb{R}^3$ is the camera origin and $\mathbf{d} \in \mathbb{R}^3, \|\mathbf{d}\| = 1$ is the ray direction, we want to render its colour $\mathbf{C}(\mathbf{r}) \in \mathbb{R}^3$. Random S points $\{\mathbf{x}_i\}_{i=1}^{S}, \mathbf{x}_i \in \mathbb{R}^3$ on the ray are sampled with corresponding depths $\{t_i\}_{i=1}^{S}, t_i \in [t_n, t_f]$. Here, t_n and t_f are the near and the far camera plane depths. The points $\{\mathbf{x}_i\}_{i=1}^{S}$ are encoded through positional encoding $\gamma(x)$:

$$\gamma(x) = \{\sin(2^k\pi \cdot x), \cos(2^k\pi \cdot x)\}_{k=1}^{K}, \tag{1}$$

where $K = 12$ in our experiments. We feed the encoded representation into an MLP $f_1(\cdot)$ to obtain the point densities $\{\sigma_i\}_{i=1}^{S}$ and feature vectors $\{\mathbf{u}_i\}_{i=1}^{S}$ via $(\sigma_i, \mathbf{u}_i) = f_1(\gamma(\mathbf{x}_i))$. We then combine the result with the view direction $\mathbf{d}$ into the final colours of the ray samples $\{\mathbf{c}_i\}_{i=1}^{S}$ using MLP $f_2(\cdot)$ as $\mathbf{c}_i = f_2(\sigma_i, \mathbf{u}_i, \gamma(\theta))$. Finally, $\{\mathbf{c}_i\}_{i=1}^{S}$ are integrated into the final colour $\mathbf{C}(\mathbf{r})$ using the volumetric rendering equation:

$$\begin{aligned} &\mathbf{C}(\mathbf{r}) = \sum_{i=1}^{S} w_i \mathbf{c}_i, \text{ where } w_i = T_i(1 - \exp(-\sigma_i\kappa_i)), \\ &T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j\kappa_j\right) \text{ and } \kappa_i = q_{i+1} - q_i. \end{aligned} \tag{2}$$

We use hierarchical sampling as in the original NeRF but omit the corresponding notation.

3.2. Event-based Single Integral

We denote the absolute instantaneous intensity image at time t as $\mathbf{B}(t) \in \mathbb{R}^{W\times H}$ and its logarithmic image as $\mathbf{L}(t) = \frac{\log \mathbf{B}(t)}{g}$, where g a gamma correction value fixed to 2.2 in all experiments. Gamma correction is required to obtain linear colour from $\mathbf{B}(t)$ as it is intended for viewing on the computer screen. It is encoded with sRGB gamma curve, and Gamma 2.2 is its recommended smooth approximation [15]. Each event is denoted as a tuple (t, x, y, p), where t is the event timestamp, (x, y) are the 2D coordinates and $p \in \{-1, +1\}$ is the polarity. It is assumed to cause a $p\Delta$ change in the logarithmic absolute value $L_{x,y}(t)$ as compared to the value $L_{x,y}(t')$ at the previous timestamp t'. Hence, $L_{x,y}(t) - L_{x,y}(t') = p\Delta$, where Δ is a fixed and polarity-wise symmetric event threshold. We model the change in $L_{x,y}(t)$ based on the events as follows:

$$L_{x,y}(t) - L_{x,y}(t_0) = \int_{t_0}^{t} \Delta e_{xy}(\tau) d\tau. \tag{3}$$

Here, $e_{xy}(\tau) = p\delta_\tau^t$ for all event tuples (t, x, y, p), where δ is the impulse function with unit integral. By substituting $e_{xy}(\tau)$ into the integral, the following sum is obtained:

$$L_{x,y}(t) - L_{x,y}(t_0) = \sum_{(\tau,p):t_0<\tau\leq t} p\Delta \stackrel{\text{def}}{=\!=} E_{x,y}(t_0, t). \tag{4}$$

The right side of Eq. (4) is a constant computed from the event stream by accumulating per-pixel event polarities. The left part is the difference between the logarithmic absolute intensity values of the views rendered from $\mathbf{C}(\mathbf{o}, \mathbf{d})$ at t_0 an t.

Thus, Eq. (4) establishes a link between the observed events $\{\mathbf{E}_i\}_{i=1}^N$ and the intensity images. We propose keeping the right side constant and model $\mathbf{L}$ with the NeRF representation described in Sec. 3.1. We note that the described model of Eq. (4) is similar to Event-based Double Integral (EDI) model proposed in Pan *et al.* [32]. However, with NeRF—as opposed to conventional RGB cameras—we can render instantaneous moments, and, hence, there is no motion blur caused by the shutter of such cameras. Therefore, we reduce Double EDI to Single EDI in the model of Pan *et al.*

3.3. Self-Supervision through NeRF-based Volumetric Rendering

We substitute the left hand side of Eq. (4) by the NeRF-based volumetric rendering Eq. (2) described in Sec. 3.1. The differentiable nature of these renderings allows learning $\mathbf{C}$ of the observed static scene in a self-supervised manner from a single event stream. We define our reconstruction loss function as the mean square error $\mathbf{MSE}(\cdot)$ of the left side of (4) versus its right side:

$$\mathcal{L}_{\text{recon},xy}(t_0, t) = \mathbf{MSE}(\hat{\mathbf{L}}_{xy}(t) - \hat{\mathbf{L}}_{xy}(t_0), E_{xy}(t_0, t)), \tag{5}$$

where $\hat{\mathbf{L}}_{xy}(t) = \frac{1}{g}\log\hat{\mathbf{B}}_{xy}(t)$, with g is a constant gamma correction as discussed earlier. As per Eq. (2), $\hat{\mathbf{B}}_{xy}(t) = \mathbf{C}(\mathbf{r}_{xy}(t))$, where $\mathbf{r}_{xy}(t)$ is the ray corresponding to the pixel at location (x, y). Note that the derivations concern so far the grayscale model.

The right-hand argument of Eq. (5) is a scalar in case of a grayscale model. In this case, the $\mathbf{MSE}$ notation should be interpreted in the following way: The same grayscale value of the right argument is supervising *all three colour channels* of the left argument. This results in learning a grayscale $\hat{\mathbf{L}}(t)$. The supervision for the case of learning from colour events will be described next.

3.4. Using Colour Events

The difference between colour and grayscale event cameras is the presence of a colour ("Bayer") filter in front of the sensor [36]. Hence, each pixel perceives one colour channel at the time. It is known and constant for each pixel and usually it is arranged in 2×2 blocks.

In traditional camera imaging, all the missing colour channel values are recovered in a process called *debayering* [20]. For each pixel, its neighbouring values are used to fill in the missing colour channels. With the event camera, events fire per pixel asynchronously and, hence, it is not possible to use traditional debayering. Another possible solution is to downsample all 2×2 pixel blocks into a single pixel, at the cost of losing $3/4$ of spatial resolution. We next describe an approach to learn fully coloured model with neither maintaining full frame reconstructions nor losing spatial resolution.

We denote the Bayer filter as $\mathbf{F} \in \mathbb{R}^{W\times H\times 3}$, where $W\times H$ is the spatial resolution. Here, for each pixel (x, y), $\mathbf{F}_{x,y,z} = 1$, only if z is the colour channel of the examined pixel, and 0 elsewhere. In the DAVIS 346C event camera which we use in our experiments, $\mathbf{F}$ consists of the following tiled 2×2 pattern: $[[[1,0,0],[0,1,0]],[[0,1,0],[0,0,1]]]$ (RGGB colour filter).

To model the colour events, we point-wise pre-multiply the colour filter mask $\mathbf{F}$ with both the rendered (left) and the event (right) arguments of the $\mathbf{MSE}$ in Eq. (5):

$$\mathcal{L}_{\text{recon}}(t_0, t) = \mathbf{MSE}(\mathbf{F}\odot\hat{\mathbf{L}}(t) - \mathbf{F}\odot\hat{\mathbf{L}}(t_0), \mathbf{F}\odot E(t_0, t)), \tag{6}$$

where "$\odot$" denotes pixel-wise multiplication.

In pixels, where only red events happen, the multiplication by F removes all other channels than red. This does not mean that a particular point in 3D space will only ever be supervised with by the red channel and have no green or blue channel information. Instead, when the same 3D point is observed from different views, it will be seen by green and blue pixels as well and, hence, have all the needed information to reconstruct the full RGB colour at the full resolution. Hence we can recover all three RGB channels without demosaicing or loss of spatial resolution of the event windows.

For learning the correct white balance of the scene, we make use of the following observation. Recall that we assume $\alpha \in \mathbb{R}^3$ is known in advance. The object's silhouette generates events that capture the difference between the background and foreground colour. This propagates the background colour information into the foreground object, thus recovering its colour as well and allowing learning both the correct brightness offset and the colour balance between the channels.

3.5. Distortion Regularisation

We adopt the distortion regulariser from MipNeRF 360 [2] to handle potential floating cloudy artefacts in the learned model. The basic idea is that we compute and minimise the term that measures how localised the sampled weights $\{w_i\}_{i=1}^N$ of ray $\mathbf{r}$ from Eq. (2) are:

$$\begin{aligned}\mathcal{L}_{\text{dist}}(\mathbf{r}) &= \mathcal{L}_{\text{dist}}(\{w_i\}_{i=1}^N, \{\kappa_i\}_{i=1}^N) = \\ &= \sum_{i,j} w_i w_j \left|\frac{\kappa_i + \kappa_{i+1}}{2} - \frac{\kappa_j + \kappa_{j+1}}{2}\right| + \\ &+ \frac{1}{3}\sum_i w_i^2(\kappa_{i+1} - \kappa_i),\end{aligned} \tag{7}$$

with κ_i defined in Eq. (2). When there are floating artefacts in front of the surface, the term becomes higher as the weights are distributed not only near the object surface, but also in front of it.

3.6. Final Loss Function

Combining all regularisation and integrating over the windows, we obtain the following loss function:

$$\mathcal{L} = \frac{1}{N_{\text{windows}}} \sum_{i=1}^{N_{\text{windows}}} (\mathcal{L}_{\text{recon}}(t_0, t) + \lambda \mathcal{L}_{\text{dist}}(t_0, t)),$$
$$t = \frac{i}{N_{\text{windows}}}, t_0 \sim U[t - L_{\max}, t), \tag{8}$$

where $\lambda = 10^{-4}$ both for the real and synthetic dataset experiments.

3.7. Ray Direction Jitter

A frequently-used camera trajectory is a circle around the object, so that the camera always keeps roughly the same altitude and distance to the object. This makes it easy for the model to overfit to the input views. Following NeRF-OSR [37], we use ray direction jitter for better generalisation to unseen views, *i.e.,* instead of shooting the rays exactly through the pixel centre, we randomly offset the ray direction within the pixel. This policy is fast and easy to implement.

3.8. Event-based Ray Sampling Strategy

Our method can be used with individual events or event windows; we use event windows for efficiency and for simplicity. Thus, we accumulate events from the start to the end of the window. Now we describe how we select the starts and the ends of the windows. First, we split the whole event stream into intervals $t_i = i/N_{\text{windows}}$, where $i \in \{1, \ldots, N_{\text{windows}}\}$; t_i are the window ends. We empirically noticed that using constant short windows results in poor propagation of high-level lighting; using constant long windows often results in poor local details. Hence, we sample the window length randomly as $t - t_0 \sim U(0, L_{\max}]$. Equivalently, if the window end t is fixed, the window start is sampled as $t_0 \sim U[t - L_{\max}, t)$. This allows the model to learn both global and local colour information. For real data experiments, we use $L_{\max} = 0.1$. For synthetic data experiments, we use $L_{\max} = 0.05$.

In every event window, N_e pixels are sampled from the positions where $E(t_0, t) \neq 0$, *i.e.*, at least one event happens and is not cancelled by others. However, if sampling is limited only by non-zero rays (*positive sampling*), information about pixels with no events will be ignored and would degrade results. Hence, we also apply *negative sampling*. For this, we sample βN_e rays from every other pixel in the frame, where $\beta = 0.1$. This makes our method more robust to noisy events (which we filter in advance) and leads to better reconstruction of the uniformly coloured surfaces as shown in the ablation studies (Sec. 4.4). It also helps in reconstructing thin structures which is detailed next.

In total, our sampling strategy uses only $O((1+\beta)N_e) = O(N_e)$ rays per epoch. This is much more efficient than the ray sampling strategy used in traditional NeRF for processing RGB images, where in each epoch, every pixel is rendered from every camera viewpoint, thus requiring $O(W \times H \times N_{\text{views}})$ rays, where $W \times H$ is the resolution and N_{views} the number of camera viewpoints. This observation proposes to discard the traditional notion of training view count [44] in our work and to instead use new data-efficiency metrics based on the number of events. It also suggests that redundant views do not help model learning as much as when they were diversely sampled.

3.9. Implementation Details

Our code is based on NeRF++ [52]. We disable the background network as our test scenes can completely fit inside the unit sphere, which is rendered only using the foreground network. We replace ReLU with $\tanh$ activation function as it resulted in more consistent training results. We train the models for $5 \cdot 10^5$ iterations on two NVIDIA Quadro RTX 8000 GPUs with the optimiser and hyper-parameters unchanged from the original NeRF++ paper. This takes about six hours to complete. Rendering a full 346×260 image (*i.e.,* the resolution of DAVIS 346C) then takes only about 1.7 seconds using the same hardware setup.

4. Experiments

We evaluate our technique EventNeRF on synthetic and real sequences. We show visual results and numerical results in the case of synthetic data. We evaluate the various design choices of our method through an ablative study and compare them against a baseline method. Here, we compare against an implementation that first converts events to RGB through Rebecq *et al.* [36], followed by a regular NeRF regression from the generated images. We also include video results in the supplement.

Examined Sequences. We have examined seven synthetic and ten real sequences. For synthetic sequences, we take the source models from Mildenhall *et al.* [29]. For each scene, we render a one-second-long 360° rotation of camera around the object at 1000 fps as RGB images, resulting in 1000 views. From these images, we simulate the events stream using the model from [38]. The corresponding camera intrinsics and extrinsics are used directly in our method. For the real data experiments, we record ten objects with our DAVIS 346C colour event camera in front of a uniform white background. We have noticed that it is hard to make a stable and calibrated setup in real life where the event camera rotates around an object. In the absence of the background events, rotating the object while keeping the camera still will result in the same event stream as when the camera rotates around the static object. Hence we placed the objects on a direct drive vinyl turntable, resulting in stable and consistent object rotation at the speed of 45 RPM. In this setting, it is also essential to provide lighting that remains constant

Figure 2: Reconstruction by our method EventNeRF on synthetic ("Materials", "Ficus") and real ("Goatling", "Sewing") sequences. Our approach reconstructs specularities ("Materials"), thin structure ("Ficus", "Sewing") and even the glasses frame on the "Goatling" sequence.

	E2VID+NeRF			Our EventNeRF		
Scene	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Drums	19.71	0.85	0.22	**27.43**	**0.91**	**0.07**
Lego	20.17	0.82	0.24	**25.84**	**0.89**	**0.13**
Chair	24.12	0.92	0.12	**30.62**	**0.94**	**0.05**
Ficus	24.97	0.92	0.10	**31.94**	**0.94**	**0.05**
Mic	23.08	0.94	0.09	**31.78**	**0.96**	**0.03**
Hotdog	24.38	0.93	0.12	**30.26**	**0.94**	**0.04**
Materials	22.01	0.92	0.13	**24.10**	**0.94**	**0.07**
Average	22.64	0.90	0.15	**28.85**	**0.93**	**0.06**

Table 1: Comparing our method against E2VID+NeRF. Our method consistently produces better results.

regardless of the current object's rotation angle. To satisfy this, we mounted a single ring light right above the object. A detailed photo of the setup is included in the supplementary material (Fig. 7).

Metrics To account for the varying event generation thresholds, colour balance, and exposure, we do the following procedure for all our and baseline results. In our model, the event threshold only affects the overall image contrast in logarithmic space. Exposure and colour balance are modelled as an offset in logarithmic space. Hence for every sequence of predicted images $\mathbf{I}_k{}_{k=1}^{N_{\text{images}}}$ and the corresponding ground-truth images $\mathbf{G}_k{}_{k=1}^{N_{\text{images}}}$, we fit a single linear colour transform $f(\mathbf{I})$, which we apply to the predictions in the logarithmic space:

$$f(\mathbf{I}) = \exp(\mathbf{a} \odot \log \mathbf{I} + \mathbf{b}),$$
$$(\mathbf{a}, \mathbf{b}) = \arg\min_{\mathbf{a},\mathbf{b} \in \mathbb{R}^3} \sum_{k=1}^{N_{\text{images}}} \|\mathbf{a} \odot \log \mathbf{I}_k + \mathbf{b} - \log \mathbf{G}_k\|^2. \tag{9}$$

Then, for the transformed images $f(\mathbf{I}_k)_{k=1}^{N_{\text{images}}}$, we compute and report PSNR, SSIM, and LPIPS [53] based on AlexNet.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
Fixed 50 ms win.	27.32	0.90	0.09
W/o neg. smpl.	26.48	0.87	0.16
W/o dist. reg	27.33	**0.91**	**0.07**
Full EventNeRF	**27.43**	**0.91**	**0.07**

Table 2: Ablation studies computed on the *Drums*.

4.1. Synthetic Sequences

As described earlier, we examine seven synthetic sequences from Mildenhall *et al.* [29]: drums, caterpillar, ficus, hotdog, materials, microphone, and chair. They cover a wide variety of effects, including thin structures (drums, caterpillar, ficus, microphone), view-dependent effects (drums, caterpillar, materials), large uniformly coloured surfaces (chair). Fig. 2 shows sample visual results from two synthetic sequences (left). Our method learns view-dependent

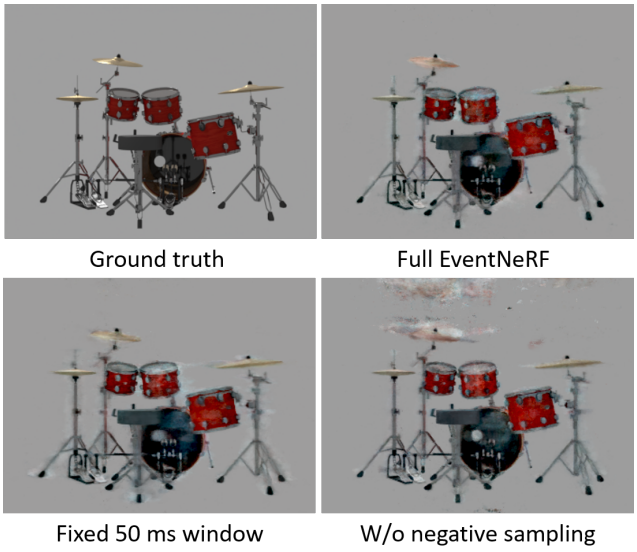


Figure 3: Importance of our various design choices (see Sec. 4.4). The full model produces the best results.

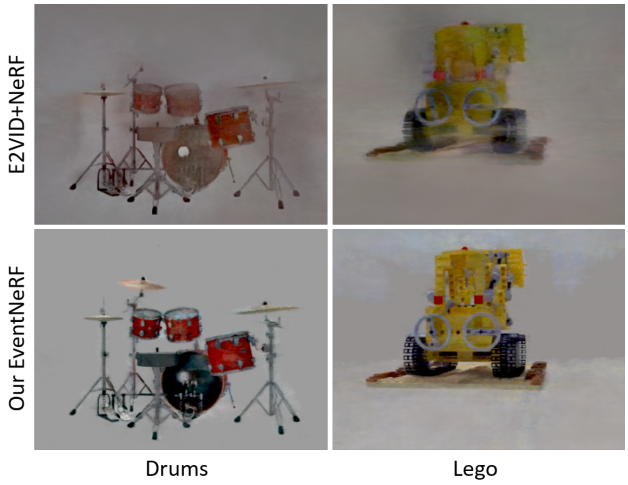


Figure 4: EventNeRF better resembles the ground truth than the baseline E2VID+NeRF.

effects (Materials) and thin structures (Ficus). Fig. 4 shows our reconstruction on two more synthetic sequences. Here, we capture textured regions well (Lego). Corresponding numerical results are reported in Tab. 1. Please refer to our video for the visualisations.

4.2. Real Sequences

We examine ten scenes that we recorded on a turntable: goatling, dragon, chicken, sewing machine, game controller, lego cube, beer bottle, fake plant, multimeter, microphone. They show the performance of our method on scenes with thin structures (goatling, sewing machine, fake plant, microphone), fine-print coloured text (game controller, multimeter, beer bottle), view-dependent effects (goatling, sewing machine, lego cube, beer bottle), dark details (goatling, game controller). As there is no ground-truth RGB data, we can only evaluate results visually, as shown in Fig. 2. Note how can we can reconstruct a one-pixel-wide sewing needle in the "Sewing" sequence, and we can also reconstruct the eyeglasses on the "Goatling" scene; see Fig. 5 for depth map visualisations and the supplementary material and the video for more sequences. In Fig. 5, the colour scheme assigns per-pixel distances in meters from a virtual camera to the object in a novel view. See Figs. 9 and 10 in the supplement for more qualitative visualisations.

4.3. Baseline Comparison

We compare against the approach of first recovering traditional RGB frames from events using E2VID [36] and then learning regular NeRF with them as the training data. Tab. 1 reports numerical evaluations for this experiment. Our method clearly outperforms the E2VID+NeRF approach in all metrics and on all sequences. Fig. 4 shows visual samples of this experiment on the Drums and Lego sequences. Our method resembles the ground truth better than the baseline.

Note that such approaches do not account for the sparsity and asynchronousity of the event stream. Hence, they need much more memory and disk space to store all the reconstructed views. Moreover, there is a limit to how short the E2VID window can be made, and hence to the number of reconstructed views. In contrast, EventNeRF respects the asynchronous nature of event streams and reconstructs the neural representation directly from the event stream. Moreover, it can use an arbitrary number of windows, which allows it to reconstruct the scene even from 3% of the data, as we show in Sec. 4.5. That makes it significantly more scalable than first reconstructing the frames and using traditional NeRF [29].

4.4. Ablation Study

We ablate the choice of regularisers and analyse window size randomisation as an alternative to a fixed long window. The results are shown both in Tab. 2 and in Fig. 3 on the "Drums" sequence. The most significant degradation comes from disabling negative sampling, as this leads to the highly visible artefacts in the background and in from of the object. Next, using fixed 50 ms window results in the loss of fine detail in the reconstruction. And on this scene, disabling distortion regulariser leads to a minor loss in the reported metrics.

4.5. Data Efficiency

By varying the event generation threshold in the simulated event streams, we can vary the total number of events. Hence,

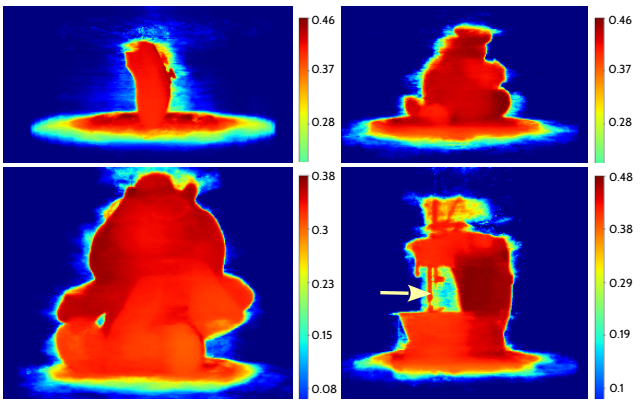


Figure 5: Examples of depth maps at arbitrary novel views obtained in our experiments with EventNeRF with real data (row-wise from left to right: "Controller", "Chicken", "Goatling" and "Sewing"). Note the fine details such as controller buttons, glasses frame and the sewing needle highlighted by an arrow. The distance units are meters.

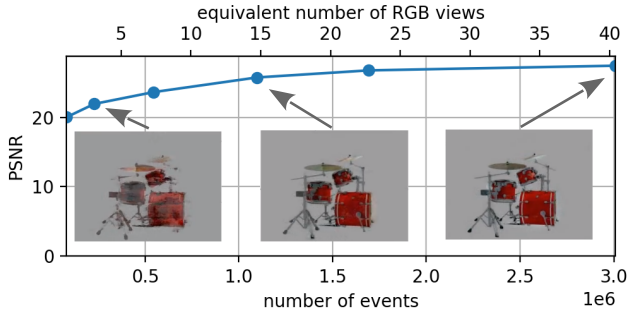


Figure 6: PSNR performance of our EventNeRF on "Drums" with varying event numbers (multiplied by 10^6 on the x-axis). The images underneath the function graph show exemplary novel views obtained with the 7%, 36% and 100% of the events, respectively.

we can compare how our method performs with different event numbers.

In the Fig. 6, we plot PSNR of the reconstructed "Drums" scene as a function of the number of events and the number of equivalent RGB frames (that would occupy the same space). The generated novel views with 36% and 100% of the available events are barely distinguishable and with 7% of the data, the scene is still well recognisable. Note how we can still reconstruct the scene with the amount of data that is less than **one** equivalent RGB frame at 20 PSNR. For reference, NeRF's performance degrades significantly with less than 100 views [16].

5. Discussion

All in all, our hypothesis—that a sparse event input is sufficient to reconstruct a dense volumetric 3D scene representation capable of novel view synthesis in the RGB space—is verified. The experiments show that directly supervising a 3D volume results in higher-quality novel views than first reconstructing RGB images from events for the supervision. Next, we observe that EventNeRF handles thin structured well and converges faster than NeRF. Moreover, we can reconstruct a 3D-consistent model with only 3% of the initial events, whereas NeRF cannot do so with fewer than 40 input views. Finally, our ablative study confirms that all event-specific design choices are crucial to achieving high visual accuracy. We foresee that EventNeRF can find applications and extensions not only in computer vision and graphics but also in robotics, especially in systems where event cameras are preferential compared to RGB cameras.

Limitations. Despite high quantitative scores, several artefacts can be observed in our novel views (*e.g.,* halos and tails near the object silhouettes); most of them are due to the imperfection of event cameras (such as a small resolution and non-deterministic threshold in the real case). As per construction, the RGB colours in the rendered novel views can be distorted as they depend on the assumption on the background colour (simplified boundary conditions). In the case of only a few events in uniformly textured areas, we observe colour propagation (gradient) artefacts in poorly textured areas due to scarcity of the corresponding input events (*cf.* the synthetic chair in Fig. 9 of the supplement).

Social Impact. Event cameras could soon find their way to consumer devices such as mobile phones. Even though EventNeRF can generate photo-realistic novel views in the RGB space that can be maliciously claimed real, it overfits to the input static scene and cannot automatically modify it in a controlled way.

6. Conclusion

We introduced the first method to reconstruct a 3D model of a static scene from event streams that enables photo-realistic RGB view synthesis. Thanks to the proposed combination of event supervision, volumetric rendering and several event-specific regularisers, EventNeRF outperforms the compared baselines in terms of the rendered image quality, thin structure handling, and data storage requirements. Thus, this paper extends the spectrum of practical event-based techniques with a 3D representation learning approach, and we hope to see follow-up works in this new field, *i.e.,* neural rendering from event streams.

References

[1] Benjamin Attal, Eliot Laidlaw, Aaron Gokaslan, Changil Kim, Christian Richardt, James Tompkin, and Matthew O'Toole. Törf: Time-of-flight radiance fields for dynamic scene view synthesis. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021.

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022.

[3] Kwabena Boahen. Neuromorphic microchips. *Scientific American*, 292:56–63, 2005.

[4] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P.A. Lensch. Nerd: Neural reflectance decomposition from image collections. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[5] Eric Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Computer Vision and Pattern Recognition (CVPR)*, 2021.

[6] Enric Corona, Tomas Hodan, Minh Vo, Francesc Moreno-Noguer, Chris Sweeney, Richard Newcombe, and Lingni Ma. Lisa: Learning implicit shape and appearance of hands. In *Computer Vision and Pattern Recognition (CVPR)*, 2022.

[7] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH*, 1996.

[8] Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *Computer Vision and Pattern Recognition (CVPR)*, pages 8649–8658, 2021.

[9] Guillermo Gallego, Tobi Delbruck, 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 Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 44(01):154–180, 2022.

[10] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *SIGGRAPH*, page 43–54, 1996.

[11] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. 37(6):257:1–257:15, 2018.

[12] Yang Hong, Bo Peng, Haiyao Xiao, Ligang Liu, and Juyong Zhang. Headnerf: A real-time nerf-based parametric head model. In *Computer Vision and Pattern Recognition (CVPR)*, 2022.

[13] iniVation. DV software. `https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/dvsnoisefilter/dvsnoisefilter.cpp`.

[14] iniVation. User Guide - Biasing Dynamic Sensors. `https://gitlab.com/inivation/inivation-docs/blob/master/Advanced%20configurations/User_guide_-_Biasing.md`.

[15] International Electrotechnical Commission. International Standard IEC 61966-2-1:1999: Amendment 1 - Multimedia systems and equipment – Colour measurement and management – Part 2-1: Colour management – Default RGB colour space – sRGB. Standard, 2003.

[16] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *International Conference on Computer Vision (ICCV)*, pages 5885–5894, 2021.

[17] Navami Kairanda, Edith Tretschk, Mohamed Elgharib, Christian Theobalt, and Vladislav Golyanik. ϕ-SfT: Shape-from-template with a physics-based deformation model. In *Computer Vision and Pattern Recognition (CVPR)*, 2022.

[18] James T. Kajiya and Brian P Von Herzen. Ray tracing volume densities. page 165–174. Association for Computing Machinery, 1984.

[19] Hanme Kim, Stefan Leutenegger, and Andrew J. Davison. Real-time 3d reconstruction and 6-dof tracking with an event camera. In *European Conference on Computer Vision (ECCV)*, 2016.

[20] Xin Li, Bahadir Gunturk, and Lei Zhang. Image demosaicing: A systematic survey. In *Visual Communications and Image Processing*, volume 6822, pages 489–503, 2008.

[21] Zhengqi Li, Wenqi Xian, Abe Davis, and Noah Snavely. Crowdsampling the plenoptic function. In *European Conference on Computer Vision*, pages 178–196. Springer, 2020.

[22] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[23] Lingjie Liu, Marc Habermann, Viktor Rudnev, Kripasindhu Sarkar, Jiatao Gu, and Christian Theobalt. Neural actor: Neural free-view synthesis of human actors with pose control. *ACM Trans. Graph. (ACM SIGGRAPH Asia)*, 2021.

[24] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 38(4):65:1–65:14, July 2019.

[25] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021.

[26] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 1(2):99–108, 1995.

[27] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[28] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Trans. Graph.*, 38(4), 2019.

[29] 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. In *European Conference on Computer Vision (ECCV)*, 2020.

[30] Jalees Nehvi, Vladislav Golyanik, Franziska Mueller, Hans-Peter Seidel, Mohamed Elgharib, and Christian Theobalt. Differentiable event stream simulator for non-rigid 3d tracking. In *CVPR Workshop on Event-based Vision*, 2021.

[31] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Computer Vision and Pattern Recognition (CVPR)*, 2021.

[32] 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 *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[33] Jeong Joon Park, Peter R. Florence, Julian Straub, Richard A. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[34] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. Derf: Decomposed radiance fields. In *Computer Vision and Pattern Recognition (CVPR)*, pages 14148–14156, 2021.

[35] Henri Rebecq, Guillermo Gallego, Elias Mueggler, and Davide Scaramuzza. Emvs: Event-based multi-view stereo–3d reconstruction with an event camera in real-time. *Int. J. Comput. Vision (IJCV)*, 126(12):1394–1414, 2018.

[36] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. High speed and high dynamic range video with an event camera. *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, 2019.

[37] Viktor Rudnev, Mohamed Elgharib, William Smith, Lingjie Liu, Vladislav Golyanik, and Christian Theobalt. Neural radiance fields for outdoor scene relighting. *arXiv*, 2021.

[38] Viktor Rudnev, Vladislav Golyanik, Jiayi Wang, Hans-Peter Seidel, Franziska Mueller, Mohamed Elgharib, and Christian Theobalt. Eventhands: Real-time neural 3d hand pose estimation from an event stream. In *International Conference on Computer Vision (ICCV)*, 2021.

[39] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2022.

[40] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[41] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[42] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[43] Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR*, 2021.

[44] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edith Tretschk, Wang Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, Tomas Simon, Christian Theobalt, Matthias Nießner, Jonathan T. Barron, Gordon Wetzstein, Michael Zollhöfer, and Vladislav Golyanik. Advances in Neural Rendering. *Computer Graphics Forum (EG STAR)*, 2022.

[45] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Trans. Graph.*, 38(4), 2019.

[46] Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2Face: Real-time Face Capture and Reenactment of RGB Videos. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2016.

[47] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *International Conference on Computer Vision (ICCV)*, 2021.

[48] Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios. *IEEE Robotics and Automation Letters*, 3(2):994–1001, 2018.

[49] Lan Xu, Weipeng Xu, Vladislav Golyanik, Marc Habermann, Lu Fang, and Christian Theobalt. Eventcap: Monocular 3d capture of high-speed human motions using an event camera. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[50] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *International Conference on Computer Vision (ICCV)*, 2021.

[51] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *Computer Vision and Pattern Recognition (CVPR)*, 2021.

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

[53] 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.

[54] Yufeng Zheng, Victoria Fernández Abrevaya, Marcel C. Bühler, Xu Chen, Michael J. Black, and Otmar Hilliges. I M Avatar: Implicit morphable head avatars from videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2022.

[55] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018.

[56] Yi Zhou, Guillermo Gallego, Henri Rebecq, Laurent Kneip, Hongdong Li, and Davide Scaramuzza. Semi-dense 3d reconstruction with a stereo event camera. In *European Conference on Computer Vision (ECCV)*, pages 235–251, 2018.

[57] Yi Zhou, Guillermo Gallego, and Shaojie Shen. Event-based stereo visual odometry. *IEEE Transactions on Robotics*, 37(5):1433–1450, 2021.

[58] Shihao Zou, Chuan Guo, Xinxin Zuo, Sen Wang, Hu Xiaoqin, Shoushun Chen, Minglun Gong, and Li Cheng. Eventhpe: Event-based 3d human pose and shape estimation. In *International Conference on Computer Vision (ICCV)*, 2021.

Supplementary Material

A. Real Data Capture

This supplemental document provides more details on the experiments and technical aspects for the proposed technique. We show photos of our experimental setup, and describe our event camera settings, event threshold calibration and density clipping. We also show more results on synthetic and real data.

A.1. Setup

Fig. 7 shows photos of the setup we used to record real data.

Figure 7: Real data recording setup. The object is placed on a 45 RPM direct-drive vinyl turntable and lit by a ring light mounted directly above it. The scene is recorded with a DAVIS 346C colour event camera (right bottom).

A.2. Event Capture

In the event camera settings, we set the event generation threshold to produce an event in about 33% of luminance change. This low threshold also brings lots of noise events, mostly of positive polarity [14]. We add a "dvsnoisefilter" node in the DV software with background activity filtering enabled. According to the source code [13], the background activity filter works in the following way. For every event, they count the number of its neighbouring pixels that had an event in the last 2000μs (supporting pixels). Then they count the numbers of supporters for the supporting pixels themselves. If they have at least one supporting pixel that is itself supported, they let the original event through.

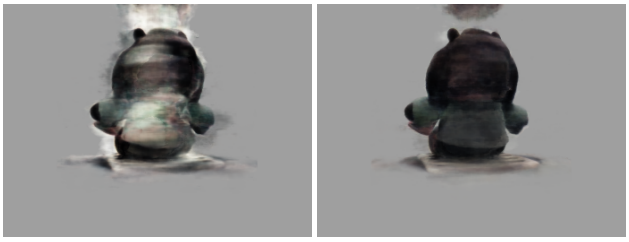


Figure 8: Effect of using a single threshold (top) versus calibrated separate positive and negative event thresholds (bottom) on the "Goatling" scene.

A.3. Threshold Calibration

While the noise filter removes most of the background noise, it still keeps some of the noise in the foreground. This means, if we directly accumulate the events via Eq. (4) of the main paper, this difference will always be biased towards the positive side. Ignoring this effect results in significant degradation in the reconstruction quality, as shown on Fig. 8.

Hence, we account for this effect by calibrating separate positive Δ_{+1} and negative Δ_{-1} thresholds.

In our setup, the object rotates at constant speed with revolution period P. Hence the view $L(t+P)$ is the same as the view $L(t)$ for any t. Therefore we can substitute $L(t) = L(t+P)$ into Eq. (4) of the main paper:

$$\begin{aligned} 0 &= L_{x,y}(t+P) - L_{x,y}(t) = \\ &= E_{x,y}(t, t+P) = \sum_{(\tau,p):t<\tau\leq t+P} p\Delta_p. \end{aligned} \tag{10}$$

We use that observation to formulate an optimisation objective for the thresholds, given fixed average threshold Δ:

$$\begin{aligned} \Delta_{+1}, \Delta_{-1} = \arg\min_{\Delta_{+1},\Delta_{-1}} \sum_{x,y} \left\| \sum_{(\tau,p):t<\tau\leq t+P} p\Delta_p \right\|^2, \\ \text{s.t. } \frac{1}{2}(\Delta_{+1} + \Delta_{-1}) = \Delta. \end{aligned} \tag{11}$$

We optimise the thresholds per-scene using this equation with $t = 0$. The results vary in range $\Delta_{+1} - \Delta_{-1} \approx [-0.9\Delta, -0.7\Delta]$. For real scenes, the average threshold Δ is selected manually so that the reconstructed scene is not over-/under-exposed. For synthetic scenes, we use symmetric thresholds $\Delta_{+1} = \Delta_{-1} = \Delta = 0.5$.

In the data efficiency experiments, we vary the threshold from $\Delta = 0.5$ up to $\Delta = 3.0$ for the lowest amount of data. The latter threshold is equivalent to $\exp(3.0) = 20$ times change in luminance per single generated event.

B. Density Clipping

For the real scenes, we know that the object always lies inside the cylinder defined by the turntable plate. Hence,

to filter the noise and artefacts in the unobserved areas, we force the density to zero everywhere outside of this cylinder:

$$\sigma(x, y, z) = 0, \text{ if } x^2+y^2 > r_{\max}^2 \text{ or } z > z_{\max} \text{ or } z < z_{\min}. \tag{12}$$

The cylinder parameters $z_{\min}$, $z_{\max}$ and $r_{\max}$ are tuned manually to fit the recorded experimental setup. In our case, $z_{\min} = -0.35$, $z_{\max} = 0.15$ and $r_{\max} = 0.25$.

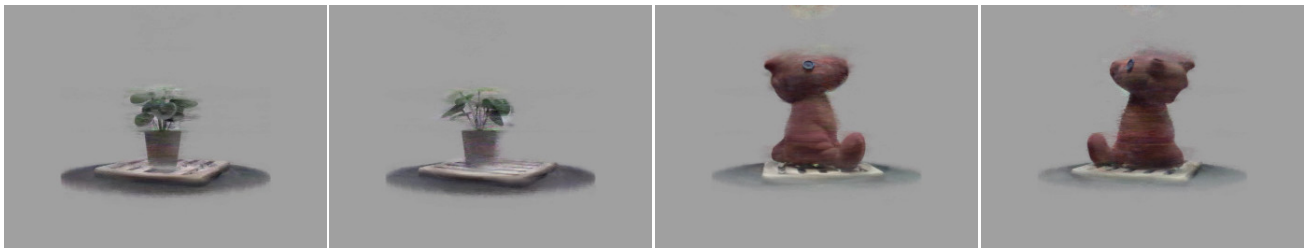
C. Synthetic Data Results

We provide more synthetic data results and comparisons in Fig. 9 and in the supplementary video. E2VID+NeRF struggles with background reproduction and separation, resulting in less clear images and background artefacts. In addition, the colour and detail reproduction are also a concern for E2VID+NeRF. Our method do not suffer from such problems. It produces photorealistic results, capturing specularities (Lego, Materials, Microphone), thin structures (Drums, Ficus, Lego, Microphone), and textured regions (Hotdog, Chair).

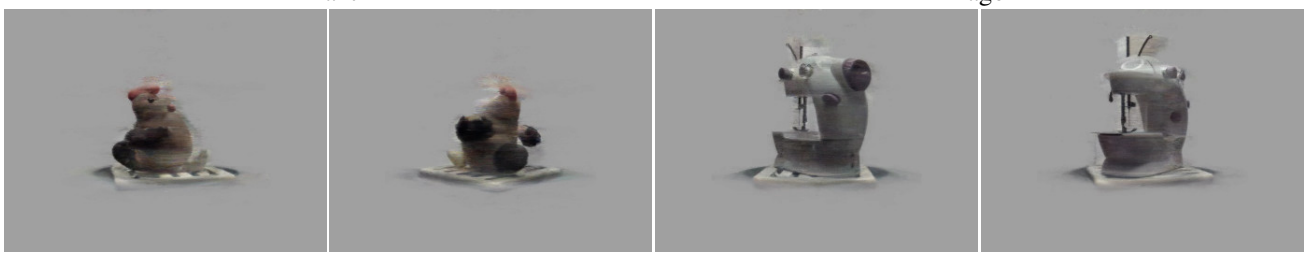
D. Real Data Results

We provide more real data results in Fig. 10 and in the supplementary video. In the "Plant" scene, we can reconstruct every stem and thin leaf. In the "Sewing" scene, we recover even a one-pixel-wide needle of the machine (best viewed with zoom). In the "Microphone" scene, we can reconstruct such fine details as a microphone grid. In the "Controller" scene, we preserve its details in the dark regions despite having low contrast. Similarly, in the "Goatling" scene, we can reconstruct both the details in the dark and bright highlights on the glasses. In the "Lego Cube" scene, EventNeRF recovers sharp colour details of the bricks. "Multimeter", "Lego Cube" and "Sewing" show how we recover view-dependent effects. In the "Bottle" scene, we can even read the text on the reconstructed label. However, there are highly visible trail artefacts behind the bottle. We speculate that it is because our event camera model might not be sophisticated enough to model intricacies of the real camera which lead to this problem.

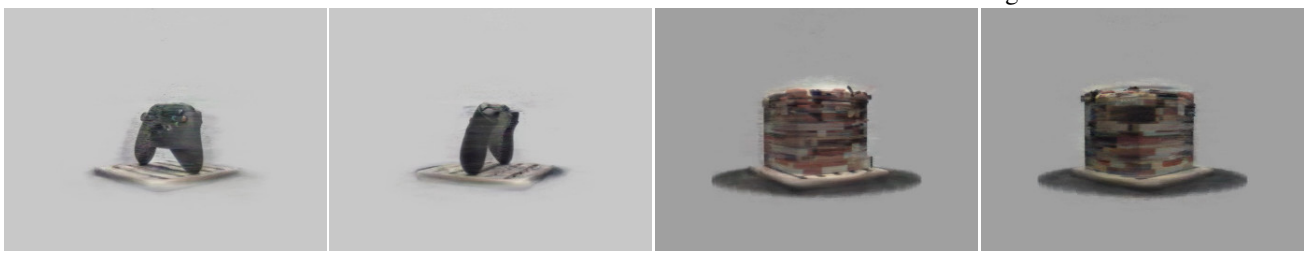
Figure 9: Additional results and E2VID+NeRF comparisons on all of the used synthetic scenes (from top: Hotdog, Chair, Drums, Ficus, Lego, Materials and Microphone).



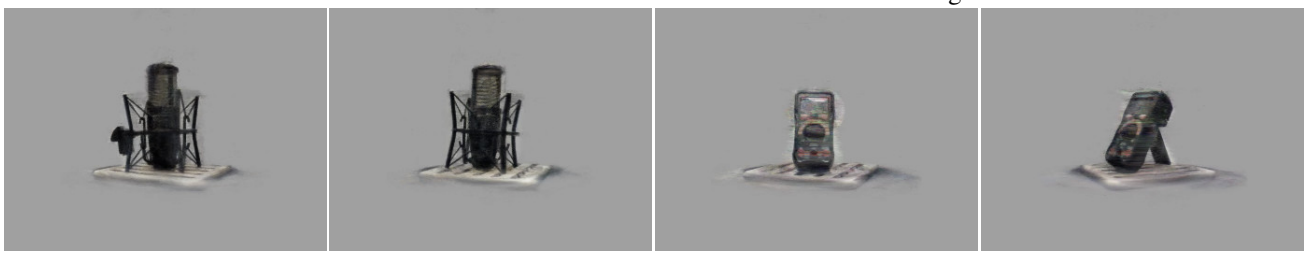
Plant Dragon



Chicken Sewing



Controller Lego Cube



Microphone Multimeter

Bottle Goatling

Figure 10: Additional EventNeRF reconstructions of the real scenes. We show two views per scene.