Supplementary Document for "Generalized Event Cameras"

Varun Sundar[†][*] vsundar4@wisc.edu
Matthew Dutson [†][*] dutson@wisc.edu
Andrei Ardelean[‡] a.ardelean@epfl.ch
Claudio Bruschini[§] Edoardo Charbon[§] {claudio.bruschini, edoardo.charbon}@epfl.ch
Mohit Gupta[†] mohitg@cs.wisc.edu

[†]University of Wisconsin-Madison [‡]NovoViz [§]École Polytechnique Fédérale de Lausanne

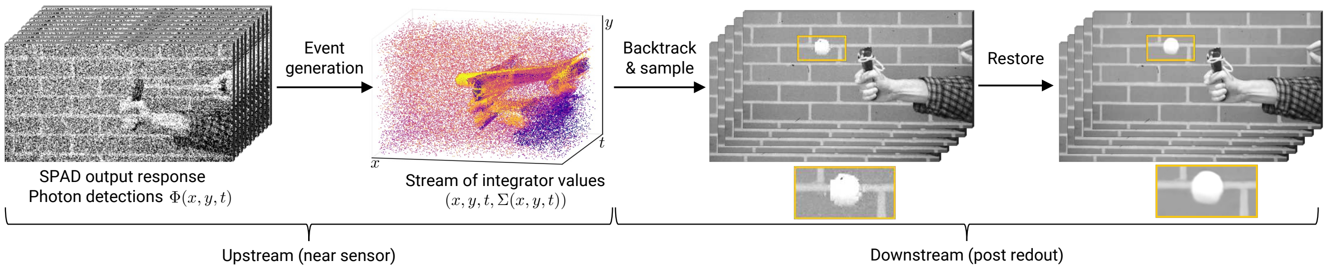
wisionlab.com/project/generalized-event-cameras/

Contents

[*]denotes equal contribution.

A. Generalized Event Algorithms

A.1. Overview



Supplementary Figure 1. **Algorithmic overview of generalized event cameras.** Our algorithms take as input the SPAD's response, $\Phi(\mathbf{x}, t)$, and output a stream of integrator values, *i.e.*, a stream of tuples $(\mathbf{x}, t, \Sigma(\mathbf{x}, t))$, where Σ represents the integrator's value. After sensor readout, we perform *backtracking*, which takes the value of the integrator as the flux estimate between the current and previous event timestamps at location $\mathbf{x}$. We then sample these flux estimates at any time t, providing a stack of backtracked frames. Notice the rich scene information present in these backtracked frames. To alleviate artifacts (shown in the insets here) arising from the pixel-wise independent emission of events, we perform video restoration to recover high-quality outputs.

Suppl. Fig. 1 shows an overview of recovering scene intensity in a bandwidth-efficient manner using generalized event cameras. This process begins from the SPAD's output response, $\Phi(\mathbf{x}\ \ t)$ at pixel location $\mathbf{x}$ and (discrete) time t, and culminates in a high-fidelity, high-temporal resolution video reconstruction. In what follows, we go over the salient steps before laying down specific algorithms (in Secs. A.2 to A.5) and detailing any algorithm-specific modifications to these steps.

Event generation. Each of our algorithms (Secs. 4 and 4.1 to 4.3) takes as input the high-speed binary frames produced by the SPAD, $\Phi(\mathbf{x}\ \ t)$, and processes it in an online manner—without any buffering of photon detections. The output of our algorithms is an asynchronous (each pixel or patch can emit events independently) spatiotemporal stream of event packets.

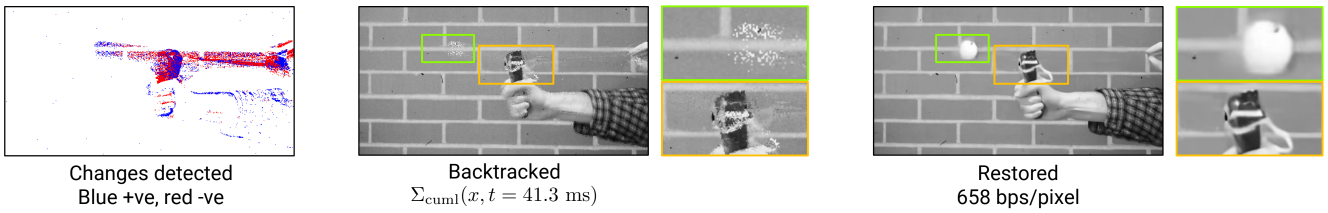
Event packets. For our methods, we assume events are sparsely encoded using a coordinate list (COO) format. In other words, we represent each event as a tuple $(\mathbf{x}\ \ t\ \ \Lambda)$, where $\mathbf{x}$ is the spatial location, t is the time, and Λ is the event payload—which is typically the integrator's value (Σ), except in the case of coded-exposure events where we transmit a set of integrator values. For the adaptive-EMA and Bayesian methods, $\Lambda = \Sigma_{\text{cuml}}$, the adaptive integrator. For the spatiotemporal chunk method, $\Lambda = \mathbf{\Sigma}_{\text{patch}}$, the patch-wise cumulative mean (a vector). For the coded method, $\Lambda = (\Sigma_{\text{long}}\ \ \{\Sigma^j_{\text{coded}}\})$, the long exposure and most recent coded exposures (note that we can exclude Σ_{long} if this is the change detector's first timestep or if we produced an event on the timestep immediately preceding this one).

Backtracking. When an event is emitted at time $t = T_1$, the integrator's value, Σ, is taken to be the flux estimate (or representation) between T_1 and the previous event emission time (T_0)—if this is the first event emission, we assume $T_0 = 0$. Thus, from the spatiotemporal event stream, we can construct a piece-wise constant representation of the incident flux, by traversing the event stream backward in time; we term this process as *backtracking*.

Sampling. After backtracking, we obtain a spatiotemporal cube of intensity estimates. We can now sample this cube discretely to obtain one or more frame-based samples. The main motivation for sampling the intensity cube, rather than processing it in its entirety, is that existing video restoration models are not capable of inference on very long video sequences (most models infer on 32–64 video frames at a time). In this work, we consider a very simple sampling strategy: temporally uniform sampling. There can be, however, more sophisticated ways to sample, potentially based on the rate of events across time [16]—we do not explore these more sophisticated variants in this paper.

Restoration. Having obtained a frame-based sampling (video representation) of our backtracked cube, we can now process it using video restoration techniques. The purpose of video restoration here is to remove artifacts arising from the independence of events between pixels (or spatial patches). For instance, neighboring pixels may fire events at different times, which tends to produce jagged artifacts around motion boundaries. We show these asynchronous artifacts (and their removal) in the insets of Suppl. Fig. 1.

A.2. Adaptive-EMA Event Camera



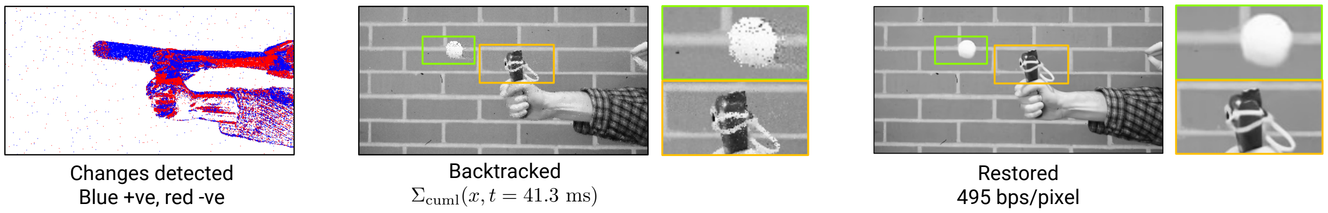
Supplementary Figure 2. **Intermediate outputs and recovered intensity from adaptive-EMA**. *(left)* We show the changes detected on the slingshot sequence, across 4000 binary frames. *(middle)* The backtracked frame bears noticeable artifacts, *(right)* however this is substantially reduced by merging information from multiple backtracked frames using a video restoration model. Improved reconstructions can be obtained using our more sophisticated methods (Secs. 4.1 to 4.3) involving more robust change detection procedures.

In Algorithm 1, we describe our simplest variant of the adaptive exposure technique, which was introduced in Sec. 3 and further specified in Sec. 4. This technique uses a fixed threshold applied to exponential moving averages (EMAs) to determine changes in scene intensity. Such a choice of change detector is motivated by early works in change-point detection that have utilized exponential moving averages [15]. We show intermediate and final outputs of adaptive-EMA on the slingshot sequence in Suppl. Fig. 2.

Algorithm 1 Adaptive-EMA Event Camera. Typical values of τ and γ are in the range 0 4–0 5, and 0 95–0 98 respectively.

```
Require: SPAD response, $\Phi(\mathbf{x}\ t)$
         Contrast threshold, $\tau$
         Exponential smoothing factor, $\gamma$
         Pixel locations, $\mathcal{X}$
         Initial time-interval, $T_0$, for computing reference moving average
         Total bit-planes, $T$
Ensure: Event-stream $E$ that contains packets of ($\mathbf{x}\ t$ cumulative mean)
 1: function GENERATEEVENTS($\Phi(\mathbf{x}\ t)$, $\tau$, $\gamma$, $T_0$)
 2:     for $\mathbf{x} \in \mathcal{X}$ do
 3:         Reference moving average, $\Sigma_{\text{ref}} \leftarrow 0$
 4:         Current moving average, $\Sigma_{\text{EMA}} \leftarrow 0$
 5:         Cumulative sum, $\Sigma_{\text{cuml}} \leftarrow 0$
 6:         Cumulative counter, $n_{\text{cuml}} \leftarrow 0$
 7:         for $t \in \{1 \quad T\}$ do
 8:             $\Sigma_{\text{EMA}} \leftarrow \gamma\Sigma_{\text{EMA}} + (1-\gamma)\Phi(\mathbf{x}\ t)$
 9:             $\Sigma_{\text{cuml}} \leftarrow \Sigma_{\text{cuml}} + \Phi(\mathbf{x}\ t)$
10:             $n_{\text{CMA}} \leftarrow n_{\text{CMA}} + 1$
11:             if $t = T_0$ then
12:                 $\Sigma_{\text{ref}}(\mathbf{x}) \leftarrow \Sigma_{\text{EMA}}(\mathbf{x}\ t)$
13:             else if $T_0 < t < T$ then
14:                 if $|\Sigma_{\text{EMA}} - \Sigma_{\text{ref}}| > \tau$ then
15:                     $E \leftarrow (\mathbf{x}\ t\ \Sigma_{\text{cuml}}\ n_{\text{cuml}})$        . change detected
16:                     $\Sigma_{\text{ref}} \leftarrow \Sigma_{\text{cuml}}$
17:                     $\Sigma_{\text{cuml}} \leftarrow 0$        . reset cumulative count
18:                     $n_{\text{cuml}} \leftarrow 0$
19:             else if $t = T$ then
20:                 $E \leftarrow (\mathbf{x}\ t\ \Sigma_{\text{cuml}}(\mathbf{x}\ t)\ n_{\text{cuml}}(\mathbf{x}\ t))$        . flush out any pending updates
21: return $E$
```

A.3. Adaptive-Bayesian Event Camera



Supplementary Figure 3. **Intermediate outputs and recovered intensity from adaptive-Bayesian**. *(left to right)* The change points detected by restarted-BOCPD are more informative that those detected by an EMA-based change detector. While there appears to be more change points here than in Suppl. Fig. 2, the changes are transmitted more parsimoniously over time—as a result, the overall readout is lower (495 bps/pixel for adaptive-Bayesian versus 658 bps/pixel for adaptive-EMA). Using an improved change detector (BOCPD) also results in backtracked images that preserve more detail, and consequently a final reconstructed image that has significantly less blur (the ping-pong ball is better recovered here).

In Algorithm 3, we describe our generalized event camera (from Sec. 4.1), which uses (a variant of) restarted Bayesian online change detection (R-BOCPD [2]) as the per-pixel change detector. As mentioned in Sec. 4.1, our main modification is the use of *extreme pruning* [21]: instead of storing one forecaster for each timestep (or per binary frame), we only retain the top-K forecasters. We found that the performance of the change detector is not significantly impacted even with substantial pruning where we retain just the top-3 forecasters; see Suppl. Fig. 4. We now describe a few variants of Algorithm 3.

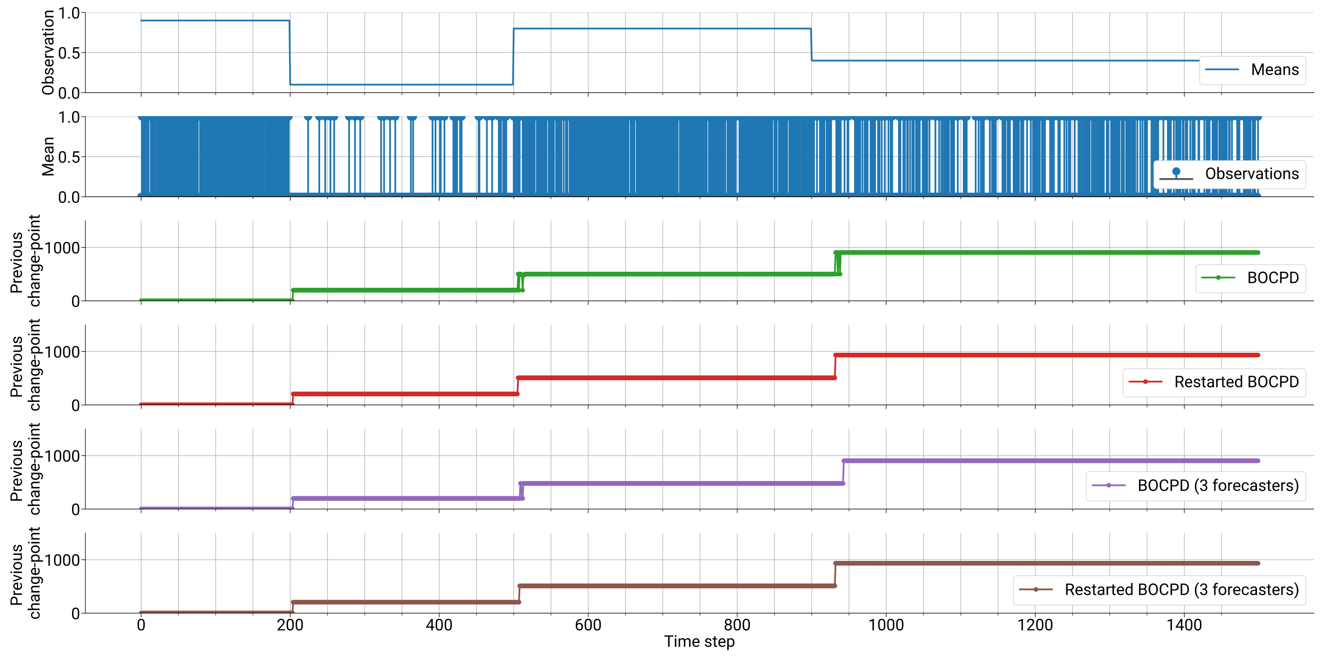
Restarts and pseudo-distribution array. We can also consider the base variant of BOCPD [1], which does not involve restarts or an array of pseudo-distribution values. In this case, we can remove the arrays in line 7 of Algorithm 3, and initialize a new forecaster as $\nu' = \gamma \sum_k \nu_k l_k$. Removing restarts allows us to skip lines 35–38 in Algorithm 3, and replace line 31 by $\text{argmax}_k\, \nu_k \neq T_0$, where T_0 is the timestamp of the previous event at the same pixel location $\mathbf{x}$. We use this simplification for our on-chip implementation for UltraPhase [3].

Direct extensions to spatial patches. While BOCPD is a per-pixel temporal change detector, there are simple ways to exploit the correlated changes observed in a patch—although these modifications should be seen as simple extensions and not a more principled approach, such as what we describe in Sec. 4.2. To reduce detection delay, we can fire events whenever a changepoint is detected in *any* pixel in a patch. While this may be preemptive for pixels where change has not yet been detected, being preemptive may not be detrimental, as a change in one pixel indicates that changes may soon be observed at other pixels in a patch.

We show intermediate and final outputs of adaptive-Bayesian on the slingshot sequence in Suppl. Fig. 3. Notice that the dynamics of the slingshot's elastic band and the propelled ping-pong ball are much better preserved in Suppl. Fig. 3 (as compared to Suppl. Fig. 2), while entailing a reduced sensor readout as well.

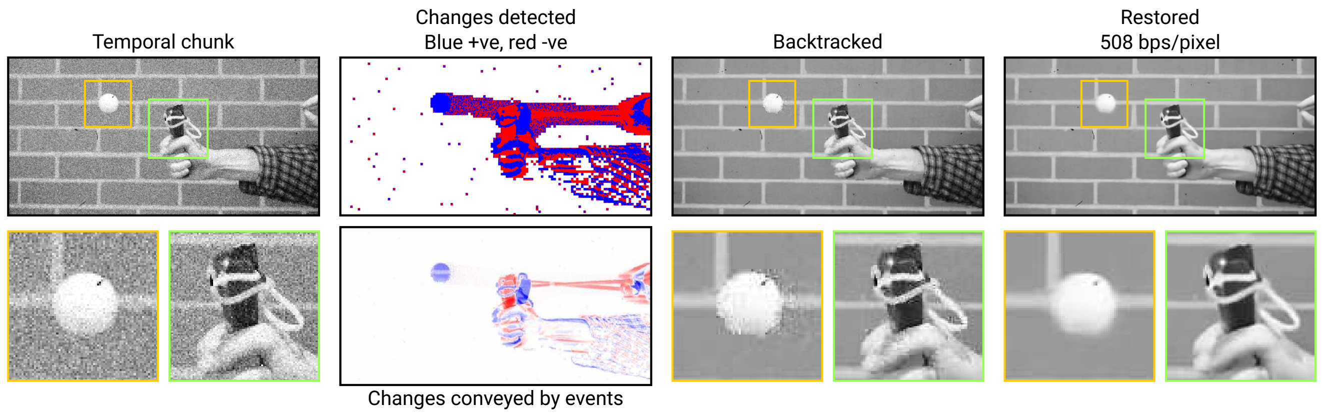
Algorithm 2 Adaptive-Bayesian Event Camera. Typical ranges for the decay factor γ are $[10^{-6}\ \ 10^{-3}]$, with larger values resulting a more frequent change detection.

```
Require: SPAD response, $\Phi(\mathbf{x}\ t)$
         Decay factor, $\gamma$
         Number of forecasters, $K$
         Pixel locations, $\mathcal{X}$
         Total bit-planes, $T$
Ensure: Event-stream $E$ that contains packets of ($\mathbf{x}\ t$ cumulative mean)
function GenerateEvents($\Phi(\mathbf{x}\ t)$, $\gamma$)
    for $\mathbf{x} \in \mathcal{X}$ do
        Forecasters, $\nu_k$ with $1 \leq k \leq K$.
            $\nu_1 \leftarrow 1$, $\nu_k \leftarrow 0$ for all $k > 1$.
        Forecaster indices, $\mathcal{I}_k$ with $1 \leq k \leq K$.
            $\mathcal{I}_1 \leftarrow 1$, $\mathcal{I}_k \leftarrow 0$ for all $k > 1$.
        Pseudo-distribution, $\tilde{l}_k$ with $1 \leq k \leq K$.
            $\tilde{l}_1 \leftarrow 1$, $\tilde{l}_k \leftarrow 0$ for all $k > 1$.
        Alphas (of a Beta prior), $\alpha_k$ with $1 \leq k \leq K$.
            $\alpha_1 \leftarrow 1$, $\alpha_k \leftarrow 0$ for all $k > 1$.
        Betas (of a Beta prior), $\beta_k$ with $1 \leq k \leq K$.
            $\beta_1 \leftarrow 1$, $\beta_k \leftarrow 0$ for all $k > 1$.
        Cumulative sum, $\Sigma_{\text{cuml}} \leftarrow 0$
        Cumulative counter, $n_{\text{cuml}} \leftarrow 0$
        for $t \in \{1 \qquad T\}$ do
            $\Sigma_{\text{cuml}} \leftarrow \Sigma_{\text{cuml}} + \Phi(\mathbf{x}\ t)$
            $n_{\text{CMA}} \leftarrow n_{\text{CMA}} + 1$
            for $k$ such that $\alpha_k\ \beta_k > 0$ do                    . compute predictive likelihoods
                Define $l_k = \alpha_k\ (\alpha_k + \beta_k)$ if $\Phi(\mathbf{x}\ t) = 1$
                    $l_k = \beta_k\ (\alpha_k + \beta_k)$ otherwise.          . update older forecasters
                $\nu_k \leftarrow (1-\gamma)\nu_k l_k$                          . update priors
                $\alpha_k \leftarrow \alpha_k + \Phi(\mathbf{x}\ t)$
                $\beta_k \leftarrow \beta_k + \Phi(\mathbf{x}\ t)$
            $\nu' = \gamma \sum_{k=1}^{K} \tilde{l}_k$                          . new forecaster
            if $\nu' > \min_k \nu_k$ then          . check if the new forecaster can replace an older one
                Denote $k_{\text{min}} = \text{argmin}_k\, \nu_k$
                $\tilde{l}_{k_{\text{min}}} \leftarrow \nu'$
                $\nu_{k_{\text{min}}} \leftarrow \nu'$
                $\mathcal{I}_{k_{\text{min}}} \leftarrow t$
                $\alpha_{k_{\text{min}}} \leftarrow 1\ \ \beta_{k_{\text{min}}} \leftarrow 1$   . initialize uniform prior
            if $\text{argmax}_k\, \nu_k \neq 1$ then                            . change detected
                $E \leftarrow (\mathbf{x}\ t\ \Sigma_{\text{cuml}}\ \ n_{\text{cuml}})$
                $\Sigma_{\text{cuml}} \leftarrow 0$                               . reset cumulative count
                $n_{\text{cuml}} \leftarrow 0$
                $\alpha_1 \leftarrow 1$, $\alpha_k \leftarrow 0$ for all $k > 1$   . reset BOCPD arrays
                $\beta_1 \leftarrow 1$, $\beta_k \leftarrow 0$ for all $k > 1$
                $\nu_1 \leftarrow 1$, $\nu_k \leftarrow 0$ for all $k > 1$
                $\mathcal{I}_1 \leftarrow 1$, $\mathcal{I}_k \leftarrow 0$ for all $k > 1$
            else if $t = T$ then
                $E \leftarrow (\mathbf{x}\ t\ \Sigma_{\text{cuml}}\ \ n_{\text{cuml}})$   . flush out any pending updates
return $E$
```



Supplementary Figure 4. **Impact of restarts and pruning on change detection**. Using a (simple 1D) synthetic example, we show the impact of the restart strategy described by Alami et al. [2] and extreme pruning (retaining just top-3 forecasters by value). *(first two rows)* We consider a piece-wise stationary time series from which we draw Bernoulli samples. *(third and fourth rows)* Incorporating restarts (and the pseudo distribution) helps reduce the detection delay. *(last two rows)* Meanwhile, we do not see a substantial impact on the detector's performance when pruning the number of forecasters (for either BOCPD or restarted BOCPD). Plotted here are the values of the previous change-point timestep, as Bernoulli observations come in.

A.4. Spatiotemporal-Chunk Event Camera



Supplementary Figure 5. **Intermediate outputs and recovered intensity from the spatiotemporal chunk method.** *(left to right)* We first accumulate photon measurements into temporal chunks of, *e.g.*, 32 binary frames. We then apply change detection on 4×4 spatial patches. Applying a restoration model to the backtracked outputs removes patch-boundary artifacts (*e.g.*, patches that recently triggered an event may appear noisier than their neighbors).

Algorithm 3 provides a formal description of the spatiotemporal-chunk event camera. Recall from Sec. 4.2 that this method generates an event whenever

$$\|\mathbf{P}(\tilde{\mathbf{\Phi}}_{\text{chunk}}(\mathbf{y}\; t) - \tilde{\mathbf{\Sigma}}_{\text{patch}}(\mathbf{y}\; t))\| \geq \tau \tag{S1}$$

Let $p \times p$ be the patch size, with $q = p^2$ the number of pixels in a patch. We use a patch size of 4×4 in all our experiments (except for Fig. 4, where we use 8×8 for illustrative purposes). $\tilde{\mathbf{\Phi}}_{\text{chunk}}(\mathbf{y}\; t) \in \mathbb{R}^q$ is the normalized mean over a temporal chunk of m binary frames; we use $m = 32$ throughout the paper. $\tilde{\mathbf{\Sigma}}_{\text{patch}}(\mathbf{y}\; t) \in \mathbb{R}^q$ is the normalized cumulative mean since the last event, excluding the current temporal chunk. $\boldsymbol{P} \in \mathbb{R}^{r \times q}$ is a feature matrix. We use $r = 16$ in our main experiments; for UltraPhase experiments, we reduce this to $r = 4$ due to on-chip memory constraints (see Sec. E.7).

We normalize via

$$\tilde{\mathbf{\Phi}}_{\text{chunk}}(\mathbf{y}\; t) = \mathbf{\Phi}_{\text{chunk}}(\mathbf{y}\; t) \oslash \mathbf{c} \tag{S2}$$

$$\tilde{\mathbf{\Sigma}}_{\text{patch}}(\mathbf{y}\; t) = \mathbf{\Sigma}_{\text{patch}}(\mathbf{y}\; t) \oslash \mathbf{c} \tag{S3}$$

where $\oslash$ indicates pointwise (Hadamard) division and $\boldsymbol{c}$ is given by

$$\boldsymbol{c}(\mathbf{y}\; t) = \sqrt{\hat{\boldsymbol{p}}(\mathbf{y}\; t)\,(1 - \hat{\boldsymbol{p}}(\mathbf{y}\; t))\frac{1}{m}\left(1 + \frac{1}{n}\right)} \tag{S4}$$

$$\hat{\boldsymbol{p}}(\mathbf{y}\; t) = \frac{n\mathbf{\Sigma}_{\text{patch}}(\mathbf{y}\; t) + \mathbf{\Phi}_{\text{chunk}}(\mathbf{y}\; t)}{n+1} \tag{S5}$$

where m is the temporal chunk size, n is the number of temporal chunks comprising $\mathbf{\Sigma}_{\text{patch}}$, and the square root is pointwise. $\boldsymbol{c}^2$ is an empirical estimate of the variance in $\mathbf{\Phi}_{\text{chunk}} - \mathbf{\Sigma}_{\text{patch}}$ under a static assumption, in which case $\mathbf{\Phi}_{\text{chunk}}$ and $\mathbf{\Sigma}_{\text{patch}}$ are binomial random variables with the same probability. We use ghost sampling when computing $\hat{\boldsymbol{p}}$ to prevent numerical instability near $\hat{p} = 0$ and $\hat{p} = 1$; specifically, we add 8 ghost Bernoulli measurements, 4 of which are successes.

Training matrix $\boldsymbol{P}$. We use the procedure outlined in Algorithm 4 to generate outputs that can be used to train the matrix $\boldsymbol{P}$ via backpropagation through time (BPTT). Algorithm 4 does not involve any branched control flow, and gives outputs identical to Algorithm 3. The event-generation decision is represented by a binary value k. On the forward pass, k is computed using a Heaviside function. On the backward pass, we approximate the gradients of the Heaviside using a surrogate gradient approach. Specifically, we replace the Heaviside gradients with those of a logistic sigmoid.

Note that Algorithm 4 also includes an autodiff-compatible backtracking step, where we fill in the values of $\boldsymbol{B}$ in time-reverse order. Unlike an indexing operation, which is not differentiable with respect to its indices, this implementation is differentiable with respect to the values of k, which delimit the piecewise-constant segments of $\boldsymbol{B}$.

We train using interpolated videos from the XVFI dataset [56]. We interpolate from the native 1 kHz frame rate to 16 kHz using RIFE [24], then linearly interpolate over time by another factor of 8 (for a total binary frame rate of 128 kHz). We use the interpolated frames as the ground truth during training. To generate SPAD frames, we treat the ground truth values (in the range $[0\ \ 1]$) as the Bernoulli photon-detection probability.

We initialize $\boldsymbol{P}$ with uniformly-distributed random values in the range $[-0\ 0125\ \ 0\ 0125]$. We train for 20 epochs, using vanilla SGD with a learning rate of $0\ 06$, reducing the learning rate by a factor of 5 after 10 epochs. Each epoch consists of 200 batches, with each batch containing 64 patch time series. Throughout training, we randomly vary the contrast threshold via uniform sampling in the range $[1\ \ 1\ 3\ 1\ \ 0\ 7]$.

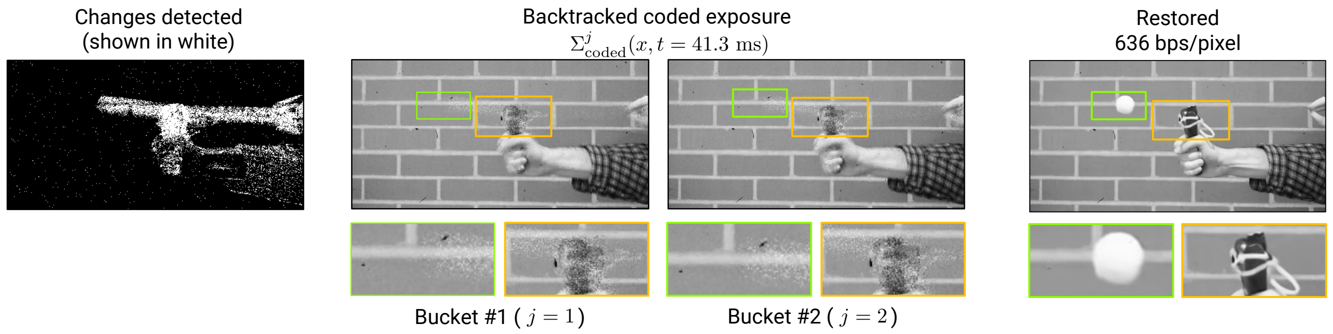
Algorithm 3 Spatiotemporal-Chunk Event Camera. We use $m = 32$. Typical values of τ are $[0\ 6\ \ 1\ 4]$; however, this depends on $\boldsymbol{P}$, as there is a redundant degree of freedom between τ and $\boldsymbol{P}$

```
Require: Patch-wise SPAD response, $\mathbf{\Phi}(\mathbf{y}\ \ t)$
         Contrast threshold, $\tau$
         Feature matrix, $\mathbf{P}$
         Temporal chunk size, $m$
         Patch locations, $\mathcal{Y}$
         Total bit-planes, $T$
Ensure: Event-stream $E$ that contains packets of ($\mathbf{y}\ \ t$ patch-wise cumulative mean)
 1: function GenerateEvents($\mathbf{\Phi}(\mathbf{y}\ \ t)$, $\tau$, $\mathbf{P}$)
 2:     for $\mathbf{y} \in \mathcal{Y}$ do
 3:         Chunk-wise mean $\mathbf{\Phi}_{\text{chunk}} \leftarrow \mathbf{0}$
 4:         Cumulative patch-wise mean $\mathbf{\Sigma}_{\text{patch}} \leftarrow \mathbf{0}$
 5:         Cumulative counter $n \leftarrow 1$
 6:         for $t \in \{1\ \ 2\ \ \ \ \ T\}$ do
 7:             $\mathbf{\Phi}_{\text{chunk}} \leftarrow \mathbf{\Phi}_{\text{chunk}} + \mathbf{\Phi}(\mathbf{y}\ \ t)\ \ m$      . accumulate into chunk-wise mean
 8:             if $t \in \{2m\ \ 3m\ \ \ \ \ \}$ then      . check for changes
 9:                 $\hat{\boldsymbol{p}} \leftarrow (n\mathbf{\Sigma}_{\text{patch}} + \mathbf{\Phi}_{\text{chunk}})\ \ (n+1)$      . empirical probability
10:                 $\boldsymbol{c} \leftarrow \sqrt{\hat{\boldsymbol{p}}(1-\hat{\boldsymbol{p}})(1+1\ \ n)\ \ m}$      . normalization
11:                 if $\|\mathbf{P}((\mathbf{\Phi}_{\text{chunk}} - \mathbf{\Sigma}_{\text{patch}}) \oslash \boldsymbol{c})\|_2 \geq \tau$ then.      . change detected
12:                     $E \leftarrow (\mathbf{y}\ \ t\ \ \mathbf{\Sigma}_{\text{patch}})$
13:                     $\mathbf{\Sigma}_{\text{patch}} \leftarrow \mathbf{0}$
14:                     $n_{\text{cumul}} \leftarrow 0$
15:             if $t \in \{m\ \ 2m\ \ 3m\ \ \ \ \ \}$ then
16:                 $\mathbf{\Sigma}_{\text{patch}} \leftarrow (n\mathbf{\Sigma}_{\text{patch}} + \mathbf{\Phi}_{\text{chunk}})\ \ (n+1)$      . update cumulative mean
17:                 $n \leftarrow n+1$
18:                 $\mathbf{\Phi}_{\text{chunk}} \leftarrow \mathbf{0}$      . reset chunk-wise mean
19: return $E$
```

Algorithm 4 Autodiff-Compatible Event Generation and Backtracking for Spatiotemporal-Chunk Event Camera. We treat Surrogate-Heaviside as a Heaviside function on the forward pass, and a logistic sigmoid on the backward pass.

```
Require: Patch-wise SPAD response, $\mathbf{\Phi}(\mathbf{y}\ \ t)$
         Contrast threshold, $\tau$
         Feature matrix, $\mathbf{P}$
         Temporal chunk size, $m$
         Patch locations, $\mathcal{Y}$
         Total bit-planes, $T$
         Assume that $T \equiv 0 \pmod{m}$
Ensure: Patch-wise $\boldsymbol{B}(\mathbf{y}\ \ t)$ that contains backtracked integrator values
 1: function GENERATEEVENTSANDSAMPLE($\mathbf{\Phi}(\mathbf{y}\ \ t)$, $\tau$, $\mathbf{P}$)
 2:     for $\mathbf{y} \in \mathcal{Y}$ do
 3:         for $i \in \{1\ \ 2\qquad T\ \ m\}$ do                . compute mean of each temporal chunk
 4:             Chunk-wise mean for $i^{\text{th}}$ temporal chunk $\mathbf{\Phi}^{(i)}_{\text{chunk}} \leftarrow \mathbf{0}$
 5:             for $1 \leq j \leq m$ do
 6:                 $\mathbf{\Phi}^{(i)}_{\text{chunk}} \leftarrow \mathbf{\Phi}^{(i)}_{\text{chunk}} + \mathbf{\Phi}(\mathbf{y}\ \ m(i-1)+j)\ \ m$     . add bitplane to chunk-wise mean
 7:         Cumulative patch-wise mean $\mathbf{\Sigma}^{(1)}_{\text{patch}} \leftarrow \mathbf{\Phi}^{(1)}_{\text{chunk}}$
 8:         Cumulative counter $n \leftarrow 1$
 9:         for $i \in \{2\ \ 3\qquad T\ \ m\}$ do                . generate events starting at the second chunk
10:             $\hat{\boldsymbol{p}} \leftarrow \left(n\mathbf{\Sigma}^{(i-1)}_{\text{patch}} + \mathbf{\Phi}^{(i)}_{\text{chunk}}\right)\ \ (n+1)$     . empirical probability
11:             $\boldsymbol{c} \leftarrow \sqrt{\hat{\boldsymbol{p}}(1-\hat{\boldsymbol{p}})(1+1\ \ n)\ \ m)}$     . normalization
12:             $k^{(i)} \leftarrow$ SurrogateHeaviside $\left(\|\mathbf{P}((\mathbf{\Phi}^{(i)}_{\text{chunk}} - \mathbf{\Sigma}^{(i-1)}_{\text{patch}}) \oslash \boldsymbol{c})\|_2 - \tau\right)$     . binary value indicating change detection
13:             $n \leftarrow n\left(1-k^{(i)}\right)+1$
14:             $\mathbf{\Sigma}^{(i)}_{\text{patch}} \leftarrow \mathbf{\Sigma}^{(i-1)}_{\text{patch}}(n-1)\ \ n + \mathbf{\Phi}^{(i)}_{\text{chunk}}\ \ n$     . update cumulative mean
15:         $\boldsymbol{B}(\mathbf{y}\ \ T\ \ m) \leftarrow \mathbf{\Sigma}^{(T/m)}_{\text{patch}}$
16:         for $i \in \{T\ \ m-1\qquad 2\ \ 1\}$ do                . iterate backward
17:             $\boldsymbol{B}(\mathbf{y}\ \ i) \leftarrow \left(1-k^{(i)}\right)\boldsymbol{B}(\mathbf{y}\ \ i+1) + k^{(i)}\mathbf{\Sigma}^{(i)}_{\text{patch}}$     . use next value of $\boldsymbol{B}$ if there is no event
18: return $\boldsymbol{B}$
```

A.5. Coded-Exposure Event Camera



Supplementary Figure 6. **Intermediate outputs and recovered intensity from (two-bucket) coded-exposure events**. *(left)* We show the changes detected using the confidence-interval test between two coded measurements that were computed across a temporal chunk of 1000 binary frames. *(middle)* The coded-exposure measurements, Σ^j_{coded}, differ predominantly in dynamic regions, while being statistically similar in static regions—this fact forms the basis of the change detector that we design for coded-exposure events. *(right)* Reconstructions obtained using our video restoration model on a stack of pre-processed (pseudo-inverse step) and backtracked coded exposures.

Mask pattern (coding) details. We choose $J \geq 2$ random binary mask sequences of length N each, *i.e.*, $\boldsymbol{C}^j \in \{0 \ 1\}^N$. Each of these J sequences is non-overlapping, which is

$$\boldsymbol{C}^j(n)\boldsymbol{C}^k(n) = 0 \, \forall \, j \neq k \quad \forall \, 1 \leq n \leq N \tag{S6}$$

The motivation behind such a choice is to ensure that the pseudo-inverse step (Moore-Penrose inverse), which is applied to the coded measurements as a processing step, can be computed efficiently. With these constraints, the pseudo-inverse step involves multiplication (of the coded measurements) by a diagonal matrix, which can be carried out efficiently.

One way to construct such a mask sequence is by choosing $j^* \sim \text{Uniform}(1 \ J)$ at each sequence location $n \in 1 \ 2 \quad N$, and then setting $\boldsymbol{C}^{j^*}(n) = 1$ and $\boldsymbol{C}^j(n) = 0$ for all other $j \neq j^*$. In other words, we pick a random "bucket" at each subframe index n.

For instance, when $J = 2$, this amounts to picking to mask sequences, $\boldsymbol{C}^1 \ \boldsymbol{C}^2 \in \{0 \ 1\}^N$, such that

$$\boldsymbol{C}^1(n) = 1 - \boldsymbol{C}^2(n) \, \forall \, 1 \leq n \leq N \tag{S7}$$

This choice corresponds to the "coded two-bucket" camera [22]. Thus, these random binary masks can be seen as a generalization of computing a single coded measurement to computing J coded measurements [17].

Finally, we remark that we do not consider the case of $J = 1$ here, since we implement coded exposures *computationally* on single-photon sensors—hence, the "complementary" coded exposure (or the two-bucket measurement) is always readily available. In other words, there is no drastic compute or memory overhead of a (computational) two-bucket coded exposure over a single coded exposure. This would not be the case if an *optical* setup (*e.g.*, using digital micromirror devices, DMDs) were used to implement coded exposures—where using a two-bucket measurement would likely entail a beam splitter and a second DMD.

Pseudo-inverse step. After backtracking, we can sample a J bucket coded-exposure measurement at any time t. Of course, if the pixel is static, we only get 1 static measurement at the pixel location—so we repeat static measurements J times. Before applying our video restoration module, we perform a pseudo-inverse step that is derived from the linear forward model of J-bucket coded exposures and is similar to the pseudo-inverse pre-processing step adopted for single-bucket compressive captures [4, 24]. This pre-processing step involves computing

$$Y(\mathbf{x} \ n) = \sum_{j=1}^{J} \frac{1}{\sum_{m=1}^{N} \boldsymbol{C}^j(\mathbf{x} \ m)} \Sigma^j(\mathbf{x} \ t)\boldsymbol{C}^j(\mathbf{x} \ n) \tag{S8}$$

giving us N subframes from the J coded measurements.

To summarize, for coded-exposure events, we follow these steps:

$$\text{Backtrack and sample} \rightarrow \text{Repeat static regions by } J\times \rightarrow \text{Pseudo-inverse pre-processing} \rightarrow \text{Video restoration} \tag{S9}$$

We show intermediate and final outputs of (two-bucket) coded-exposure events on the slingshot sequence in Suppl. Fig. 6.

Algorithm 5 Coded-Exposure Event Camera. We typically choose J to be 2, 4 or 8 and N to (correspondingly) be 8, 16 or 32. Further, we set γ in the range 1 8–3 5, with smaller values resulting in more sensitive (frequent) change detection.

```
Require: SPAD response, $\Phi(\mathbf{x}\ t)$
         Temporal extent of integrator, $T_{\text{code}}$
         Number of buckets, $J$
         Subframes per code, $N$
         Wilson's score significance, $\gamma$
         Pixel locations, $\mathcal{X}$
         Total bit-planes, $T$
         Assume that $T_{\text{code}} \equiv 0 \pmod N$, $T \equiv 0 \pmod{T_{\text{code}}}$
Ensure: Event-stream $E$ that contains packets of $(\mathbf{x}\ t\ J$ coded exposures)
 1: function GENERATEEVENTS($\Phi(\mathbf{x}\ t)$, $\gamma$)
 2:     for $\mathbf{x} \in \mathcal{X}$ do
 3:         $J$ binary mask sequences $\boldsymbol{C}^j \in \{0\ 1\}^N$ such that:
 4:             $\sum_{j=1}^{J} \boldsymbol{C}^j(n) = 1$ for all $1 \le n \le N$        . codes sum to one at each subframe index
 5:             $\boldsymbol{C}^j(n)\boldsymbol{C}^k(n) = 0$ for all $j \ne k$ and for all $1 \le n \le N$        . codes are mutually orthogonal
 6:         Cumulative sum, $\Sigma_{\text{long}} \leftarrow 0$
 7:         Cumulative counter, $n_{\text{long}} \leftarrow 0$
 8:         for $t_{\text{code}} \in \{1 \quad T\ T_{\text{code}}\}$ do
 9:             Coded exposure $\Sigma^j_{\text{coded}} \leftarrow 0$ for all $1 \le j \le J$
10:             for $n \in \{1 \quad N\}$ do
11:                 Denote $t_{\text{start}} = T_{\text{code}}(t_{\text{code}} + n\ N)$
12:                 Compute $\Phi_{\text{avg}}$: the average of $\Phi(\mathbf{x}\ t)$ in the interval $[t_{\text{start}}\ t_{\text{start}} + T_{\text{code}}\ N)$
13:                 $\Sigma^j_{\text{coded}} \leftarrow \Sigma^j_{\text{coded}} + \boldsymbol{C}^j(n)\Phi_{\text{avg}}$, for all $1 \le j \le J$        . multiplex photon detections
14:             if $\Sigma^j_{\text{coded}} \in$ conf$(T_{\text{code}}\ J \sum_s \Phi(\mathbf{x}\ s)\ T_{\text{code}})$ for all $j$ then        . binomial proportional test
15:                 $\Sigma_{\text{long}} \leftarrow \sum_j \Sigma^j_{\text{coded}}$        . static region
16:                 $n_{\text{long}} \leftarrow n_{\text{long}} + 1$
17:                 if $t_{\text{code}} = T\ T_{\text{code}}$ then
18:                     $E \leftarrow (\mathbf{x}\ t_{\text{code}}\ \Sigma_{\text{long}}\ n_{\text{long}})$        . flush out any pending updates
19:             else
20:                 $E \leftarrow (\mathbf{x}\ t_{\text{code}}\ \{\Sigma^j_{\text{coded}}\}_{1 \le j \le J}\ \Sigma_{\text{long}}\ n_{\text{long}})$
21:                 $\Sigma_{\text{long}} \leftarrow 0$        . reset long exposure
22:                 $n_{\text{long}} \leftarrow 0$
23: return $E$
```

B. Extended Discussion

B.1. More Sophisticated Integrators

In this work, we propose two integrators: adaptive exposures and coded exposures. There remains an extensive, unexplored space of alternate integrators. Both adaptive and coded exposures are linear projections of a pixel's photon detections over time. We could consider more general linear mappings, such as continuous-valued temporal codes. Further, we are not restricted to projections over time; it may also be advantageous to consider spatial projections, *e.g.*, frequency-domain transforms. With these alternate projections, we might be able to reduce bitrates while maintaining reconstruction quality.

We have so far assumed that our objective is intensity reconstruction. However, there may be situations where we know that the final goal is a specific inference task. In such a scenario, we could design integrators that only encode information relevant to the task at hand; this may be significantly more bandwidth-efficient than transmitting a generic intensity encoding. This integrator could take the form of a learned module, *e.g.*, a neural network, that operates near-sensor and computes a compressed, task-specific scene representation. We could train this module end-to-end with the downstream layers that perform final inference. The resulting system would involve a neural network that spans multiple compute devices, with an event-based communication layer in the middle. This setup somewhat resembles spiking neural networks and other event-based networks, although the goal with such methods is generally reduced computation costs (arising from sparse layer inputs) and not reduced bandwidth along a data-transfer interface.

B.2. Entropy Coding and Quantization

Generalized event cameras encode intensity levels, in the range $[0 \;\; 1]$, representing the photon-detection rate. In our experiments, we apply uniform quantization to the transmitted values; this keeps our comparisons straightforward and fair.

However, in a practical deployment, it may be more bandwidth-efficient to encode changes rather than values and apply entropy coding to the changes. When transmitting changes, the first event encodes a value in $[0 \;\; 1]$, and subsequent events encode a difference in $[-1 \;\; +1]$. This approach is functionally equivalent to transmitting levels or values, assuming the event camera correctly tracks quantization effects (to avoid drift). For natural scenes, the distribution of changes is non-uniform. The shape of this distribution depends in part on the change-detection algorithm; in general, we would expect changes near zero to be unlikely, as these would not trigger an event. We can apply entropy coding (*e.g.*, Huffman coding) to exploit the nonuniformity in the distribution and achieve some additional compression. Entropy coding could give substantial bandwidth savings, albeit at the cost of some increased near-sensor computation.

In addition to entropy coding, we could apply non-uniform quantization, either to values or changes. For example, we could non-uniformly quantize values in $[0 \;\; 1]$ based on perceptual considerations (*e.g.*, human sensitivity to intensity differences). With changes, we could exclude portions of the range $[-1 \;\; +1]$ based on the characteristics of the change detector; for example, with a fixed contrast threshold τ, there is no need to represent changes in $(-\tau \;\; \tau)$.

B.3. Spatial Compression

One advantage of patch-wise events (*e.g.*, as described in Sec. 4.2) is that they permit some spatial compression. For example, we can apply JPEG block compression to the payload of an 8×8 patch-wise event. Such a change would bring our techniques more in line with existing video-compression algorithms, which compress in both space and time. The compression ratio we observe would depend on the sensor resolution; with higher resolution, we would expect image patches to be more uniform, and thus more easily compressible. Likewise, patches with more noise (*e.g.*, due to a short adaptive integration window) would be more difficult to compress. Under ideal conditions, patch-wise compression might give us an additional ~$5\times$ reduction in bitrate. We leave this idea as a topic for future work.

B.4. Alternate Sparse Formats

As described in Sec. A.1, we assume a COO event format $(\mathbf{x} \;\; t \;\; \Lambda)$ when evaluating the bandwidth requirements of our methods. We could improve the sparse format to reduce bandwidth costs further. One such improvement would be to use an implicit time encoding. Specifically, on each binary frame, we would transmit a header $(t \;\; n)$ indicating the current timestamp and the number of events on this frame. We would then send n event packets $(\mathbf{x} \;\; \Lambda)$. If $n \gg 1$, this approach would virtually eliminate the overhead associated with transmitting values of t. Note, however, that this approach assumes we communicate events in time order; *i.e.*, if $t_2 > t_1$, all events at time t_1 are sent before any events at time t_2.

To reduce the overhead associated with $\mathbf{x}$, we could employ sparse matrix formats such as compressed sparse row (CSR) or compressed sparse column (CSC). These formats compress one of the two spatial coordinates (the row and column indices, respectively). The savings relative to COO would depend on the density of events on each frame, with denser events leading to more compression with CSR or CSC.

For pixel-wise methods (*i.e.*, adaptive-EMA, Bayesian, and coded), another potential optimization is to adaptively group events into patch-wise packets. Assume we are using the COO format $(\mathbf{x}\ t\ \Lambda)$. If a spatial patch contains many events, it may be more efficient to transmit them in a single packet, as this amortizes the overheads of $\mathbf{x}$ and t. To implement this optimization, we would add an indicator bit b to each event packet. If $b = 0$, then the packet should be treated as a pixel-wise event; if $b = 1$, it is a patch-wise event, meaning Λ encodes integrator information for an entire patch. Within a patch-wise event we would use dense format for Λ, marking the pixels that triggered an event with a one-bit mask. Each patch would adaptively determine whether to encode its events pixel-wise or patch-wise. This decision would be based on the number of events, with the optimal threshold depending on the number of bits required for $\mathbf{x}$, t, and Λ, as well as the patch size.

B.5. Latency of the Adaptive Integrator

One limitation of the adaptive integrator $\Sigma_{\text{cumul}}(\mathbf{x}\ T_1)$ is that it creates some latency in the intensity estimates. The estimate for the duration $[T_0\ T_1]$—which the adaptive integrator computes as the mean of $\Phi(\mathbf{x}\ t)$ over $[T_0\ T_1]$—is not known until T_1. Thus, there is a latency of $T_1 - t$ to obtain an estimate for $t \in [T_0\ T_1]$. The expected delay depends on the frequency of events, with lower delay in more dynamic regions. The latency of $\Sigma_{\text{cumul}}(\mathbf{x}\ T_1)$ is not an issue for offline reconstruction and inference (which we assume throughout this paper). However, it may be of concern for certain real-time applications.

As a potential solution, we could introduce a second type of event, which we call an *eager* event, in contrast to the *change* events we have considered thus far. Assume again an event is generated at time T_1. In addition to transmitting $\Sigma_{\text{cumul}}(\mathbf{x}\ T_1)$ at that moment, we could send an eager event encoding a noisy estimate of $\Phi(\mathbf{x}\ T_1)$. We could then send periodic eager events encoding refinements to the value of the adaptive integrator. Assuming constant flux after T_1, the adaptive integrator converges to the true flux value as time passes; transmitting refinements would allow downstream components to leverage this improved estimate in the absence of a subsequent change event. We could consider various schedules for sending refinements—*e.g.*, at exponentially increasing intervals, or at fixed intervals until some maximum time has passed. Note, however, that to keep the event camera's bandwidth coupled with (proportional to) the scene dynamics, there would need to be an upper bound to the number of eager refinement events, *i.e.*, with each change event triggering at most N refinements.

The solution described above would involve some additional bandwidth costs. We can thus imagine a generalized event camera that operates in two modes: "online mode" and "offline mode," with eager events only being used in the online mode, where low latency is necessary.

C. Restoration Model Details

C.1. Model Architecture

While any video restoration model could be used, we choose the densely-connected residual network proposed in Wang et al. [20] (EfficientSCI) for our video restoration architecture—which was successful at restoring backtracked outputs of all of our generalized event cameras (adpative-EMA, adaptive-Bayesian, spatiotemporal chunk, and coded-exposure events). We also experimented with the spatial-temporal shift-based model of Li et al. [10] (ShiftNet): while this architecture was successful at restoring three of our four proposed events, it did not succeed at restoring backtracked coded exposures. We attribute this to the fact that EfficientSCI was designed with video compressive sensing in mind, whereas ShiftNet is targeted for more general video-restoration tasks.

The memory cost of EfficientSCI scales with the number of input frames. Thus, we are constrained by the device memory in the number of frames we can reconstruct. For example, with 24 GB of GPU memory and a resolution of 512×256, we can reconstruct about 96 video frames—which can cover a temporal extent of 3000–9000 binary frames (30–90 ms). Increasing the temporal extent requires sampling the backtracked cube at an increased temporal stride, which can lead to blurring and lower-quality results. One solution to this problem might be to employ a recurrent architecture with a fixed-size memory. However, forward-mode recurrent inference must be causal; *i.e.*, the predicted frame at time t may only consider backtracked samples from times before t. An efficient recurrent model may enable reconstructing videos at frame-rates faster than what we show in this work (~3000 FPS)—indeed, our methods that operate the granularity of individual binary frames (*e.g.*, Sec. 4.1) can, in principle, provide reconstructions at the frame-rate of SPAD photon-detections, *i.e.*, 96 8 kHz.

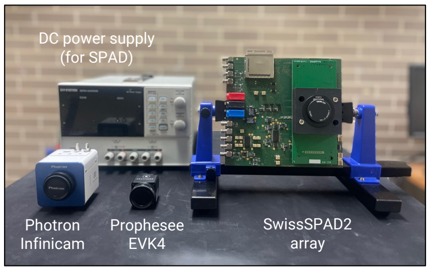
C.2. Dataset

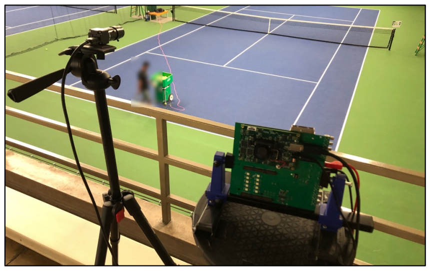
We use 4403 high-speed videos from the training split of the XVFI dataset [56] to train our restoration models. We temporally interpolate these 1000 FPS to 16000 FPS using RIFE [24] and treat each video frame, normalized between 0   1 as the photon-detection probability. We then draw 6 binary frames per video frame (as a per-pixel Bernoulli random variable, based on the photon-detection probabilities)—thereby giving us binary-valued responses at 96 kHz. We remark that our dataset generation approach here (unlike in Secs. 5.3 and E.6) is not physically accurate, since we directly treat the video's values as photon-detection probabilities instead of average photon-arrival rates. We adopt this approach for its simplicity and note that the difference (between detection probabilities and Poisson rates) manifests as a tone-mapping operation by the SPAD's response function ($f(x) = 1 - e^{-x}$). We did not see any generalization issues when applying our models trained this synthetic dataset to real SPAD data—in both ambient and low-light scenarios.

C.3. Training Parameters

All restoration models were trained until convergence (40–60 epochs) using the Adam optimizer [9] and the mean squared error (MSE) loss objective. We used an initial learning rate of 10^{-5}, which was decayed as per a cosine-annealed scheduler [11] to a minimum value of 10^{-8}. When training the spatiotemporal chunk method, we randomly choose τ on each training iteration from a uniform distribution covering the range $[0\ 76\quad 1\ 43]$. We also clip the gradient norm to 1 0 to resolve instability during training.

D. Imaging Setup





Supplementary Figure 7. **Camera setup for experimental acquisition**. *(left)* Our setup comprises of the SwissSPAD2 array [19], Prophesee EVK4 event camera and the Photron Infinicam high-speed camera. *(right)* An example setup, which we used for capturing the tennis scenes shown in Figs. 1 and 7. We blur out the player for anonymity.

Suppl. Fig. 7 shows the cameras we used for demonstrating the capabilities of generalized events: our algorithms process the outputs of the SwissSPAD2 array. We compare the performance of generalized events against the imaging capabilities of a commercial event camera (Prophesee EVK4) and a high-speed camera that is capable of real-time streaming (Photron Infinicam).

D.1. Imager Specifications

SwissSPAD2 Array. The sensor has a resolution of 512×512 pixels: however, we operate the device in its "half-array mode", meaning that only one subarray of resolution 512×256 pixels is used. Each pixel has a pixel pitch of 16 4 μm and a fill factor of less than 10%. The fill factor can be improved by the inclusion of microlens arrays which this prototype lacks.

The SPAD arrays features a certain number (~5%) of "hot pixels": pixels whose dark count rate (DCR) is abnormally high, and as a result, almost always return 1s. We calibrate a mask of hot pixels by using sensor responses that were taken in the dark. We find that hot pixels do not impact the generation of generalized events—thus, we inpaint hot pixels (either using the Telea algorithm [18] or nearest-neighbor replacement) using the calibrated mask after backtracking is performed, *i.e.*, post-sensor readout.

Prophesee EVK4. This commercially available event camera has a sensor resolution of 1280×720 pixels. Each pixel has a pixel pitch of 4 86 μm and a fill factor of $> 77\%$.

Photron Infinicam. This high-speed camera provides a USB-C data interface, which is enabled by performing compression on the fly. The camera SDK does not provide access to parameters that control this online compression—as a result, we find that when imaging scenes where the camera is per-frame SNR limited, such as the indoor scene of Fig. 5, read-noise induced artifacts translate to severe compression artifacts. The camera has a resolution of 1246×1024 pixels.

D.2. Scene Acquisition Details

We list the (focal lengths of C-mount) lenses used to capture each of the scenes shown in our main paper, along with the acquisition time of each scene (in terms of the number of binary frames captured by the SPAD at 96 8 kHz).

- Slingshot sequence in Fig. 1: 25 mm lens, 4000 binary frames.
- Tennis sequence in Fig. 1: 50 mm lens, 3000 binary frames.
- Vertical wheel in Fig. 1: 12 mm lens, 3000 binary frames.
- Nighttime traffic sequence in Fig. 1: 25 mm lens, 3000 binary frames.
- Dartboard sequence in Fig. 1: 16 mm lens, 3000 binary frames.
- Jack-in-the-box sequence in Fig. 2: 16 mm lens, 4000 binary frames.
- Rotating hole-saw bit sequence in Fig. 3: 35 mm lens, 6000 binary frames.
- Casino roulette sequence in Fig. 4: 25 mm lens, 4000 binary frames.
- Stress ball sequence in Fig. 5: 25 mm lens, 4096 binary frames.
- Nighttime traffic sequence in Fig. 6: 25 mm lens, 8000 binary frames.
- Tennis sequence in Fig. 7: 100 mm lens, 8192 binary frames.

E. Extended Experiments

In this supplementary note, we provide extended versions of our experiments in the main paper. We include detailed descriptions of our constructed baselines and make fine-grained comparisons across our proposed methods.

E.1. Baseline Details

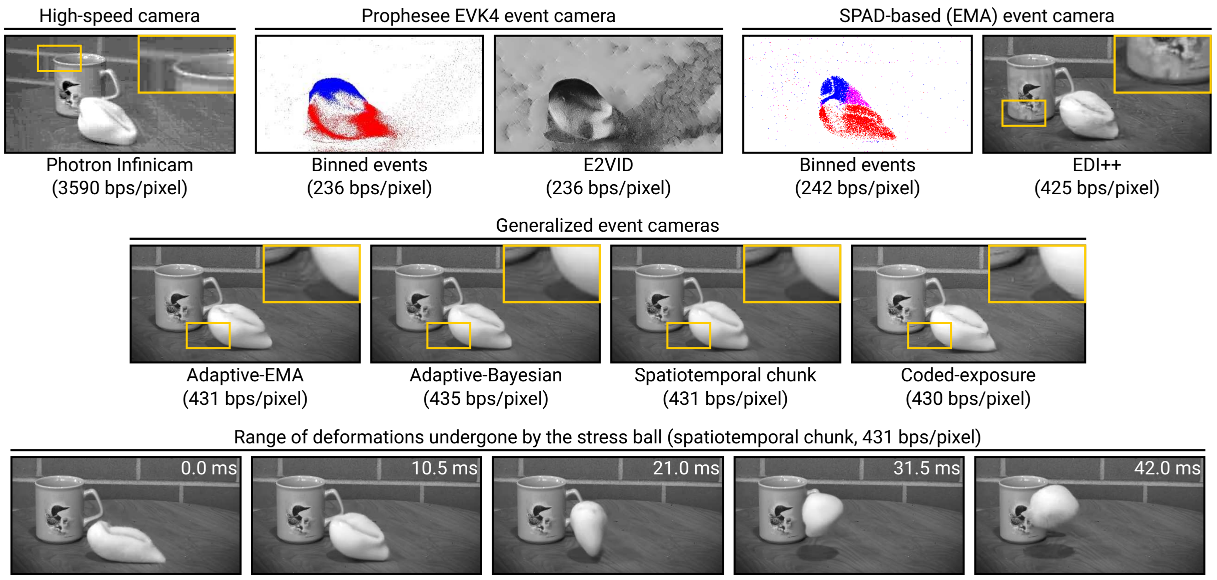
EDI++ construction. EDI [14] is a hybrid event-plus-frame technique that can produce a series of sharp images from an event stream and a long exposure spanning the same time duration. There are two practical difficulties in implementing this method: (1) precise spatial- and temporal-alignment is needed between the frames and events, (2) differences in imaging modalities between conventional CMOS cameras and DVS event sensors. There is also a third (DVS specific) difficulty, which Pan et al. [14] overcome by solving an optimization problem (either across one pair of event streams and frames or across multiple such inputs): the contrast threshold in conventional event cameras is not necessarily known and can have inherent randomness [5, 7]. In contrast, when implementing EDI using SPAD-events and frames, we have none of these difficulties: frames and events are perfectly aligned (by construction), the same imaging modality (SPAD photon detection) is used to obtain both events and frames, and finally, the forward (or event generation) model is precisely controlled (no optimization problem has to be solved). Thus, SPAD-based EDI can be thought of as an ideal version of EDI. We further refine EDI outputs using a trained restoration model (with the same architecture used for video restoration of our generalized events), and call this idealized, refined version of EDI "EDI++".

8-bucket compressive sensing. We use the multi-bucket video compressive sensing method described in Sundar et al. [17], with 8 buckets and spanning 32 subframes—where the sum of 64 binary frames comprises a single subframe. For restoring these coded 8-bucket capture, we use EfficientSCI [20], while retaining the same densely-connected residual architecture that constitutes the video restoration model for the proposed generalized events.

Burst denoising baseline. We perform burst denoising on a stack of 32 short exposures—where the sum of 64 binary frames constitutes one short exposure—using the align-and-merge technique of Hasinoff et al. [6]. After the merging step, we additionally perform BM4D denoising [13] (with $\sigma = 0\ 02$, after binomial variance stabilization [23]). Further, we find that BM4D applied directly to the stack of short exposures (with binomial variance stabilization) results in poor performance (~28 dB PSNR on the rate-distortion plot of Fig. 8).

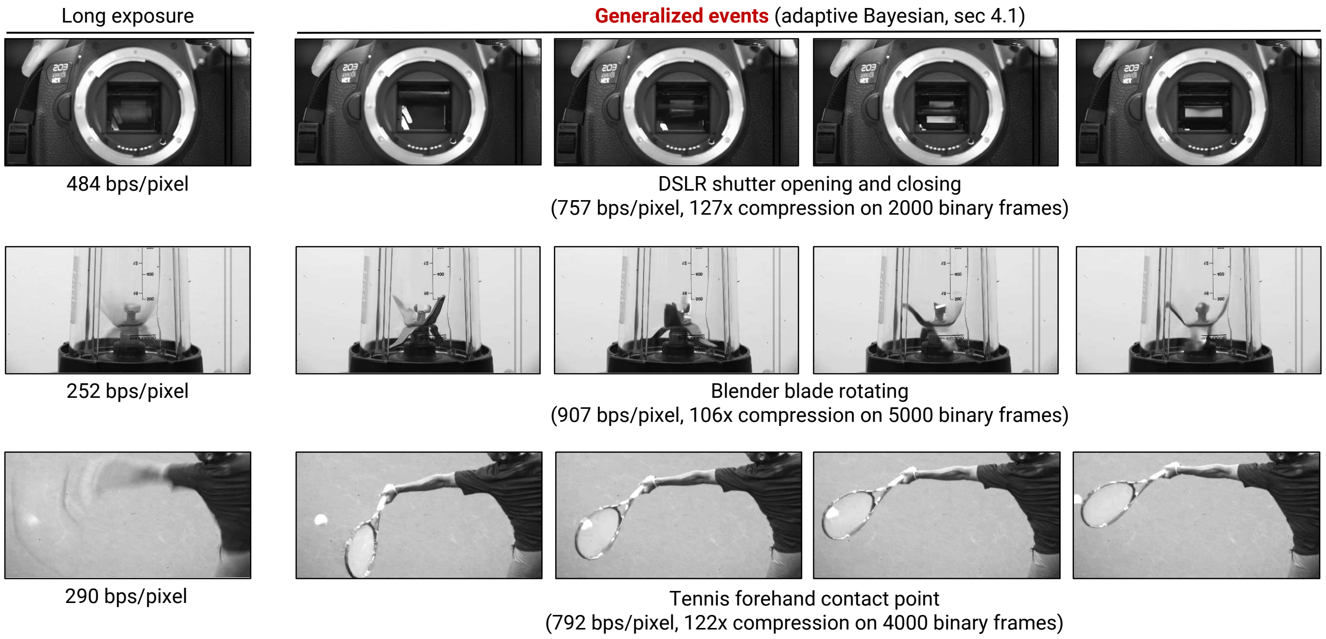
E.2. High-Speed Videography

Suppl. Fig. 8 contains extended results for high-speed videography on the stress ball sequence from Fig. 5. We show that all of our generalized event cameras give comparable results for this scene.



Supplementary Figure 8. **Extended high-speed videography results.** In addition to the results shown in Fig. 5, we show outputs obtained using our other event cameras, *viz.* adaptive-EMA from Sec. 4, adaptive-Bayesian from Sec. 4.1, and coded-exposure events from Sec. 4.3.

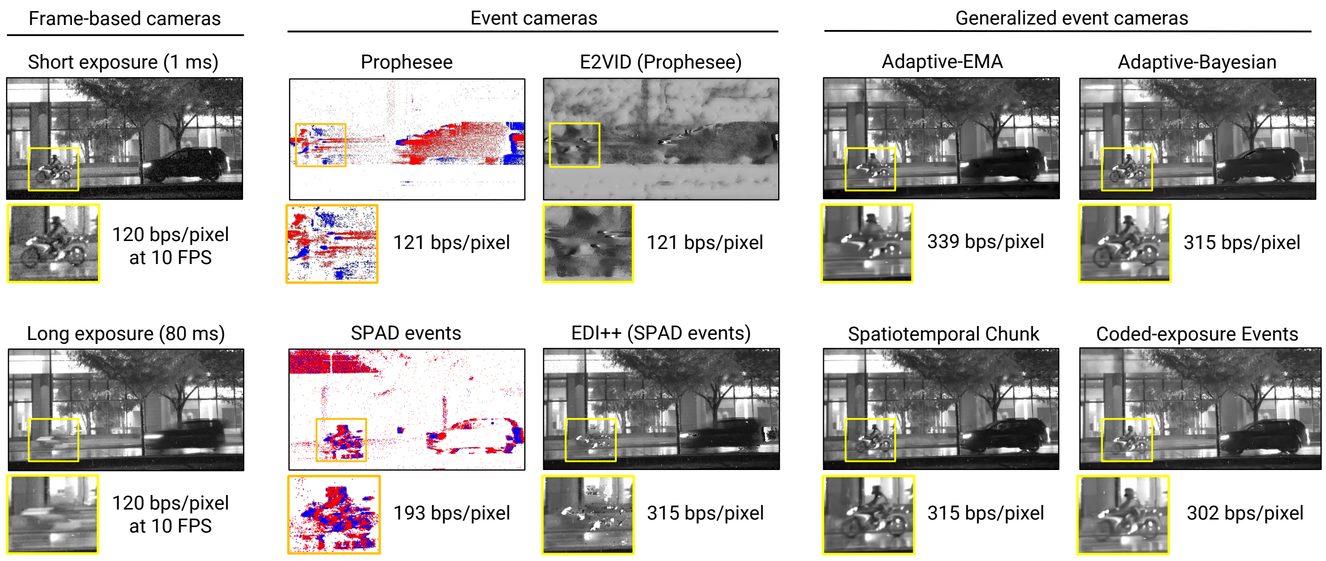
We include additional results on three other sequences—a DSLR shutter in motion, the rotating blade of a blender, and a tennis forehand shot—in Suppl. Fig. 9.


Long exposure
Generalized events (adaptive Bayesian, sec 4.1)
484 bps/pixel
DSLR shutter opening and closing
(757 bps/pixel, 127x compression on 2000 binary frames)
252 bps/pixel
Blender blade rotating
(907 bps/pixel, 106x compression on 5000 binary frames)
290 bps/pixel
Tennis forehand contact point
(792 bps/pixel, 122x compression on 4000 binary frames)

Supplementary Figure 9. **Additional high-speed videography results.** *(left column)* We show a long exposure (average over all binary frames in the sequence) to depict the extent of motion in each sequence, *(right)* and show 4 video frames (uniformly separated in time across the capture duration) using our proposed method from Sec. 4.1. All videos were reconstructed at a frame rate of 3025 FPS.

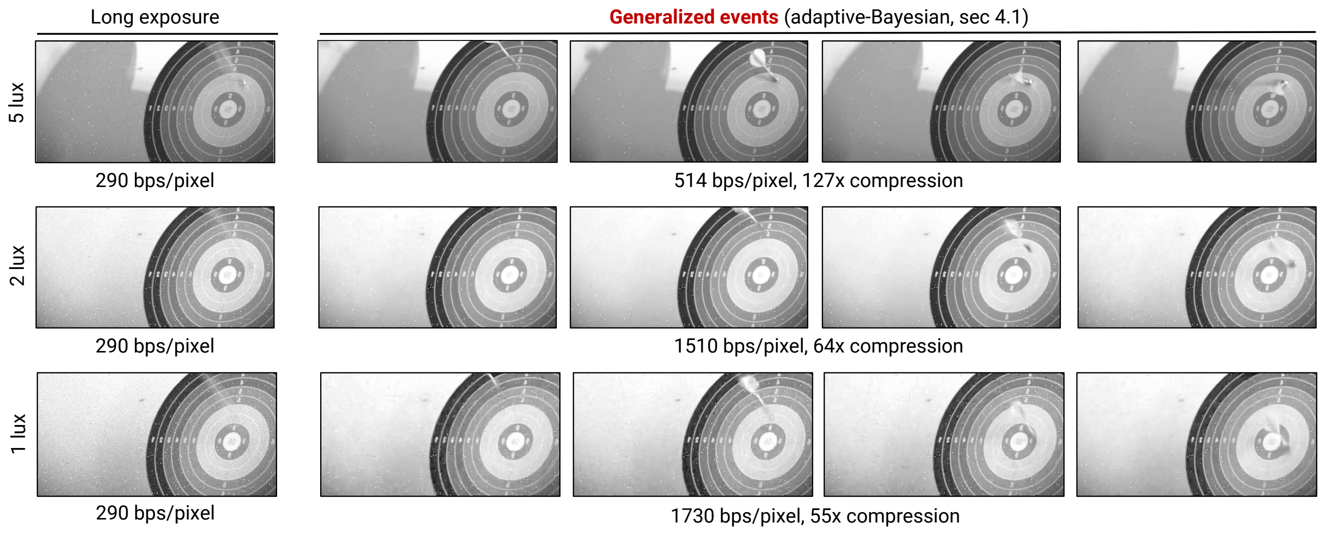
E.3. Low-light Scenes

Suppl. Fig. 10 shows extended low-light results for the nighttime traffic scene in Fig. 6. The Bayesian, spatiotemporal chunk, and coded methods give better quality than adaptive-EMA, which fails to detect low-contrast changes (*e.g.*, on the motorcycle wheel) due to its simplistic fixed-threshold change detector.



Supplementary Figure 10. **Extended low-light results.** In addition to the results shown in Fig. 6, we show outputs obtained using our other event cameras, *viz.* adaptive-EMA from Sec. 4, spatiotemporal chunk from Sec. 4.2, and coded-exposure events from Sec. 4.3.

We provide an additional result on an indoor sequence shot in the dark (see Suppl. Fig. 11), which features lower (and controllable) light-levels—1, 2 and 5 lux measured using a light meter on the sensor side, as opposed to 7 lux in Suppl. Fig. 10. We note that the low-light performance shown here could be improved upon with the inclusion of microlens arrays in the SPAD prototype, which could increase its fill factor (and in turn photon detection efficiency) from 10% to $> 40\%$.


Long exposure
Generalized events (adaptive-Bayesian, sec 4.1)
5 lux
290 bps/pixel
514 bps/pixel, 127x compression
2 lux
290 bps/pixel
1510 bps/pixel, 64x compression
1 lux
290 bps/pixel
1730 bps/pixel, 55x compression

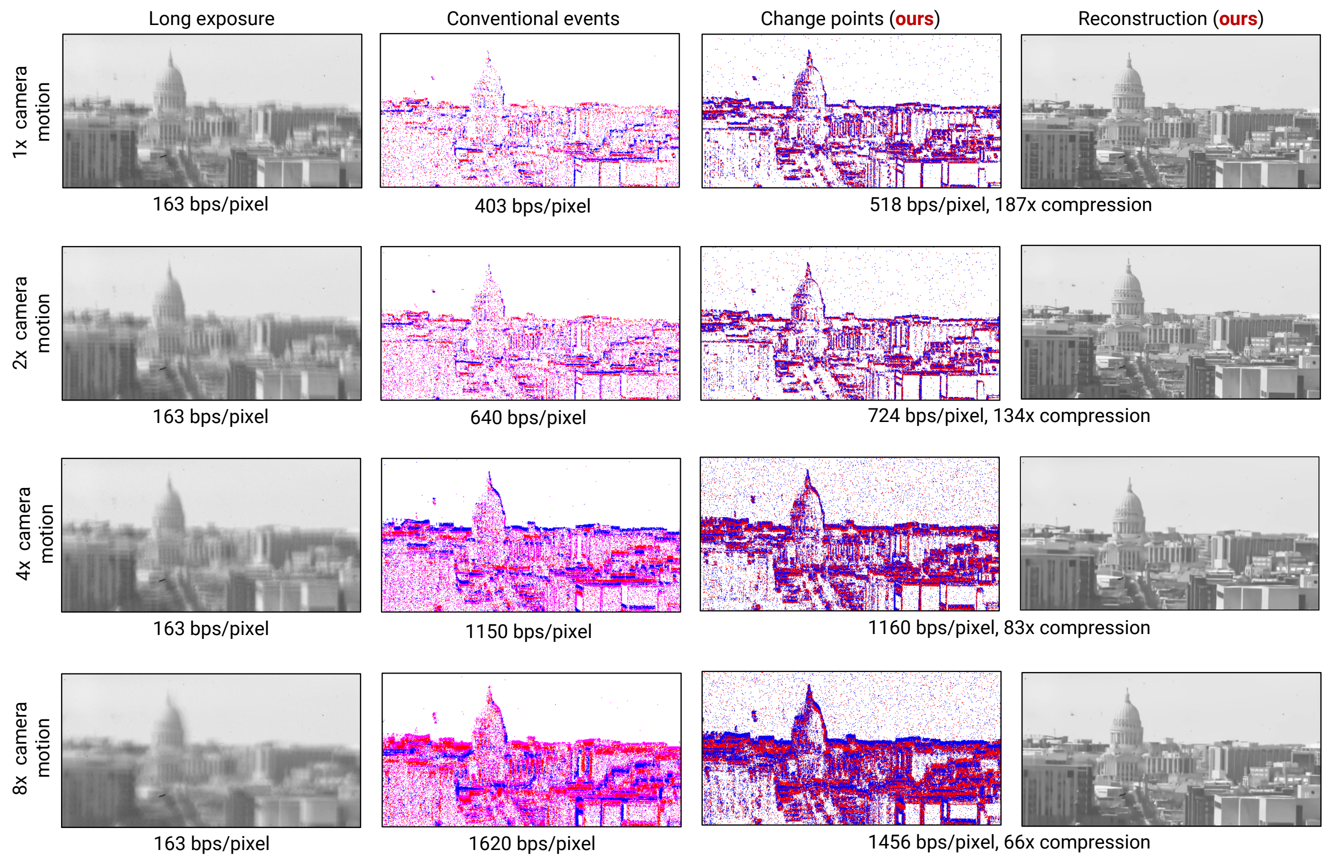
Supplementary Figure 11. **Additional low-light results.** We throw darts in the dark, at light levels of 5, 2 and 1 lux (measured on the sensor side, *top to bottom rows*). All three sequences span a duration of 41 ms (4000 binary frames). Our restoration models are not trained on low-light sequences, despite this, we see reasonable low-light performance (*e.g.*, at 5 lux). At 1 lux and 2 lux, we see (somewhat graceful) degradation in our reconstructions. Lower the light level, harder it is to reliably distinguish scene motion from noise—which results in lower compression rates (and image quality) at lower lux values. Please zoom in to see details.

E.4. Scenes with Camera Motion

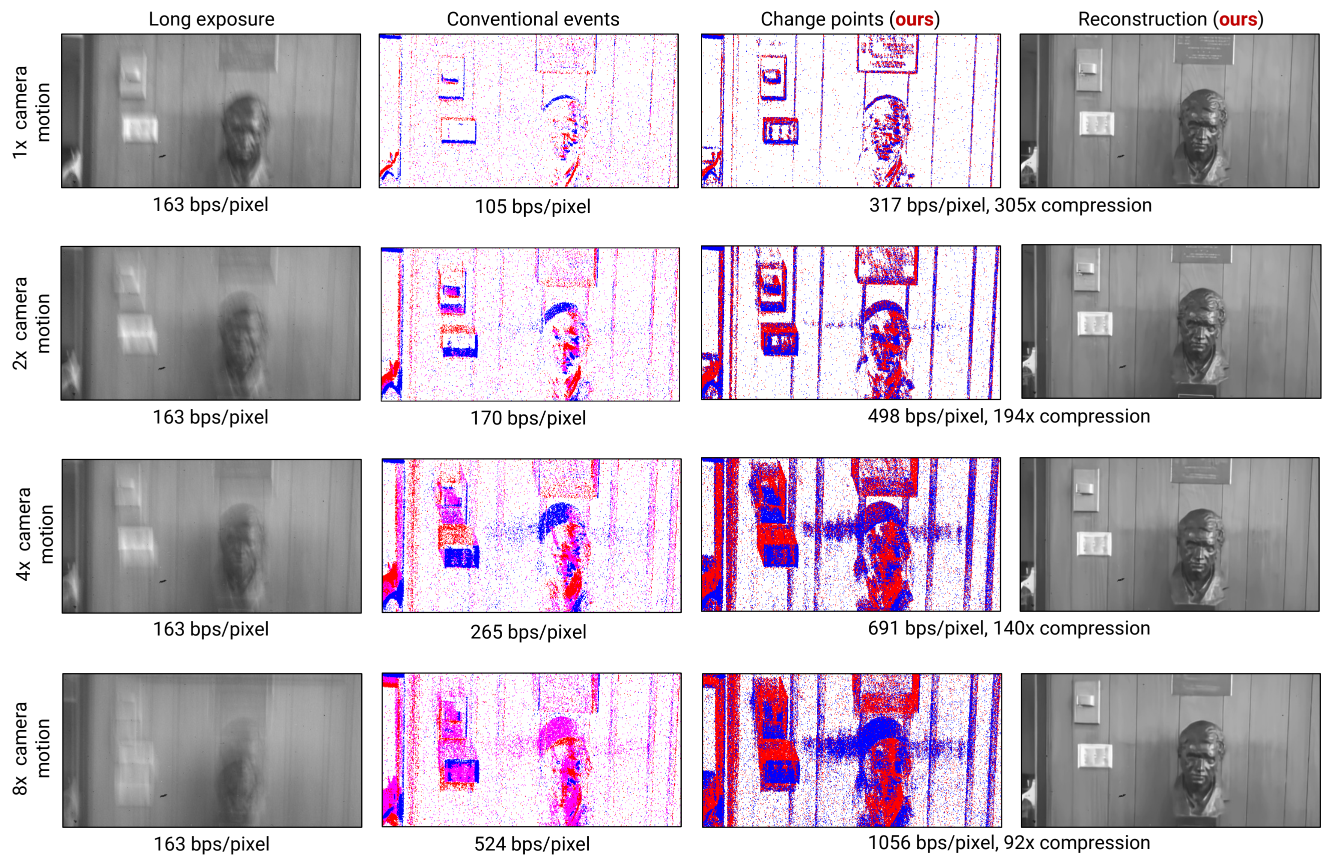
In this subsection, we provide a qualitative analysis of the impact of ego-motion on the event-generation rate and the output-image quality. We consider three scenes with camera motion: the “building” sequence that features a significant amount of spatial structure and image detail (downtown buildings); the “Ramanujan bust” sequence, which is an indoor scene with a moderate amount of texture (from the bust and metal plaque); and a nighttime driving sequence where the SPAD was placed on the car’s dashboard.

We show the change points and reconstructions obtained using one of our proposed generalized events (adaptive-Bayesian, Sec. 4.1) for these sequences in Suppl. Figs. 12 to 14. To bring out the effect of camera motion, we speed up the SPAD’s output response (by skipping binary frames) by factors of $2\times$, $4\times$ and $8\times$. Thus, a $4\times$ sped up sequence sees $4\times$ as much camera motion. For additional context, we also show the extent of motion using a long exposure and the response of a (SPAD-based) event camera across the same duration.

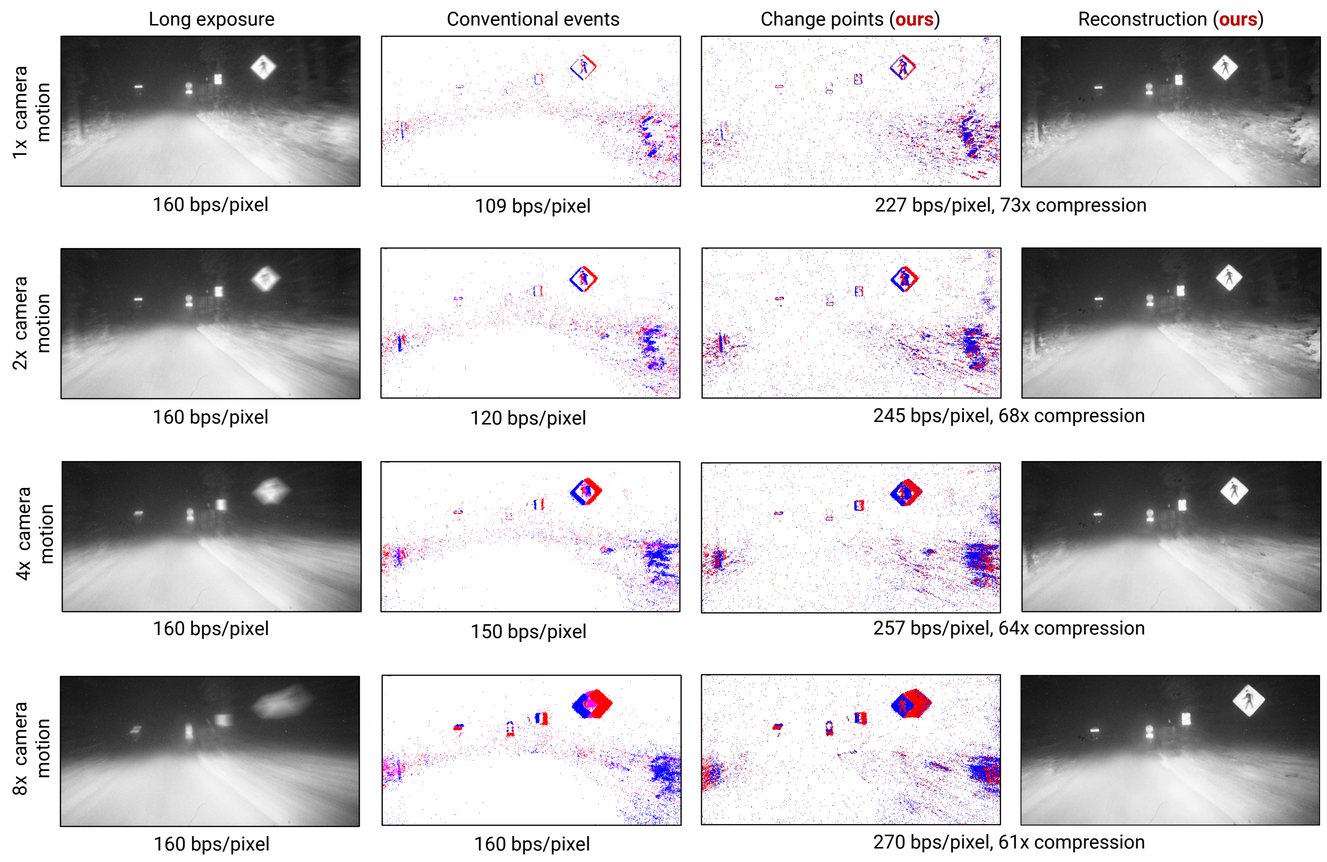
Across all sequences, we observe that even with an exaggerated amount of camera motion (*e.g.*, $4\times$ and $8\times$ sped-up sequences), we still see a significant amount of compression (reported with respect to raw photon readout) and a modest sensor readout (reported in bps/pixel).



Supplementary Figure 12. **Ego-motion results on the “building” sequence**. We run our generalized event technique on 8000 binary frames in each row. In the second to fourth rows, we speed up photon-detections by processing only every n-th binary frame ($n = 2, 4, 8$)—this exaggerates ego-motion. *(left to right columns)* We depict the extent of motion using a long exposure across the same duration. We also show the changes detected by a (SPAD-based) event camera [17]. With increasing ego-motion, our techniques output more change points (as expected), but the reconstructions remain relatively sharp (there is some amount of blur induced), even with $8\times$ more motion.



Supplementary Figure 13. **Ego-motion results on the “Ramanujan bust” sequence**. The sequence comprises 8000 binary frames, each acquired at a frame rate of 96.8 kHz. Column and row descriptions are identical to Suppl. Fig. 12. This sequence has comparatively less texture than the building sequence, consequently, the blur in our output reconstructions (at $8\times$ the camera motion) is far less perceptible.



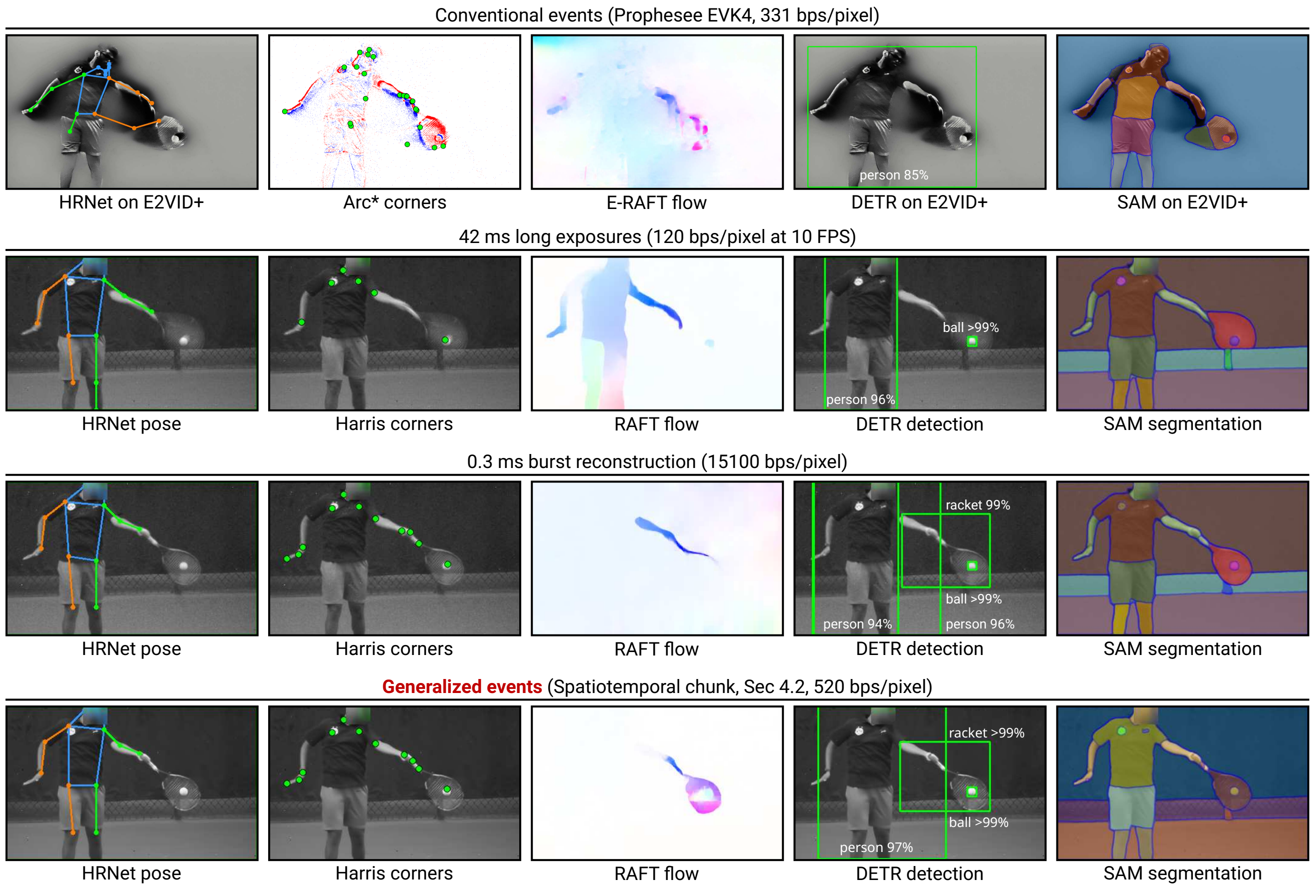
Supplementary Figure 14. **Ego-motion results on the nighttime driving sequence**. Column and row descriptions are identical to Suppl. Fig. 12. The SPAD was operated at a lower speed in this sequence (16.6 kHz instead of 96.8 kHz)—we report compression factors with respect to photon-detection readout at 16.6 kHz. While this sequence also has lesser texture than Suppl. Fig. 12, the low-light conditions here make it more challenging. When ego-motion is exaggerated by $4\times$ (or higher, last two rows), we observe that details of the bushes are blurred out. However, the pedestrian crossing sign, which is has better contrast, is still recovered.

E.5. Plug-and-Play Event Inference

Expanded results. Suppl. Fig. 15 shows an expanded set of plug-and-play inference results. This figure includes results for a 42 ms long exposure (4096 binary frames, second row) and a burst reconstruction method [12], run on consecutive 0 3 ms short exposures (32 binary frames). The long exposure fails to capture fast-moving objects; there is significant blur in the arm, racket, and ball, causing inference to fail in these regions. The burst reconstruction gives good-quality inference in both static and dynamic regions; however, it requires a high readout bandwidth (about 30$\times$ higher than our method).

Experiment details. We manually trim the Prophesee outputs to a temporal extent corresponding to the SPAD capture (consisting of 8192 binary frames). We run a Prophesee-provided E2VID model on these events to reconstruct a video. For our method and the burst reconstruction in Suppl. Fig. 15, we run pose detection, corner detection, object detection, and segmentation on the reconstructed frame corresponding to the 2224th binary frame. For the long exposure results in Suppl. Fig. 15, we run these methods on the mean over binary frames 0–4095. See below for optical flow extents. Below we provide additional details for some of the experiments in Fig. 7 and Suppl. Fig. 15.

- **HRNet pose:** We use the HRNet-W48 version of the model.
- **Harris corners:** We run a standard Harris corner detector with $\sigma = 4$.
- **RAFT flow:** For our method and the burst reconstruction in Suppl. Fig. 15, we run RAFT between the reconstructions corresponding to the 2224th and 2864th binary frames. For the long exposure results in Suppl. Fig. 15, we consider the interval between the 0–4095 and 4096–8191 exposures. We use the RAFT-Large version of the model.
- **DETR detection:** In most cases, we use a confidence threshold of 90%. We lower the threshold to 80% for the E2VID reconstruction given the lack of high-confidence predictions. We use the ResNet-50 version of the model.
- **Arc* corners:** We run the algorithm on the entire event sequence. We temporally trim the predicted corners to a 0 8 ms window around the target instant. In the figure, we show events within an 8 ms window to provide visual context.
- **E-RAFT flow:** We prepossess events into a voxel grid with 30 bins (divided into 2 sub-durations), with a time span corresponding to that used in the RAFT experiments.



Supplementary Figure 15. **Expanded plug-and-play results.** In addition to the results shown in Fig. 7, we show results for long exposures (consisting of 4096 binary frames) and burst reconstructions [12].

E.6. Rate-Distortion Evaluation

Dataset details. We source 15 videos from YouTube that were captured by a Phantom Flex 4K camera at 1000 FPS. Suppl. Fig. 16 shows thumbnails depicting the scene content in each video, as well as a long exposure over 42 ms that shows the extent of motion. We download these videos at a resolution of 854×480 pixels and further downsize (by $1\ 6\times$) and vertically crop them to the SPAD's resolution of 512×256 pixels.

We did not utilize the XVFI dataset [56], which we used for training our video restoration models, for evaluating rate-distortion tradeoffs to prevent the possibility of data leakage. Further, we find that these YouTube-sourced videos have a more extreme range of motion than XVFI videos.

Simulating SPAD responses. To simulate SPAD photon detections from high-speed videos, we adopt the following steps:

1. We interpolate videos at 1000 FPS by $16\times$ using RIFE [24] to 16000 FPS.
2. We treat the videos as sRGB-encoded frames, convert them to linear RGB images, and then grayscale.
3. Next, we determine the average count of incident photo-electrons as

$$N(\mathbf{x}\ t) = \alpha \mathcal{I}(\mathbf{x}\ t) + d \tag{S10}$$

where we $\mathcal{I}(\mathbf{x}\ t)$ represents a video frame, d is number of spurious detections ($7\ 74 \times 10^{-4}$ counts per binary frame, using values reported in Ulku et al. [19]). We choose α such that the average value of $\alpha\mathcal{I}(\mathbf{x}\ t)$ over each pixel location $\mathbf{x}$ and time t is 1—we find that our ambient captures with the SwissSPAD2 have an average photon per pixel per binary frame (PPP) of 1.

4. Finally, we draw binary frames with the probability of a 1 given by

$$\Pr\{\Phi(\mathbf{x}\ t) = 1\} = 1 - e^{-N(\mathbf{x},t)} \tag{S11}$$

From each frame at 16000 FPS, we draw 6 binary frames, thereby simulating photon detections at 96000 Hz.

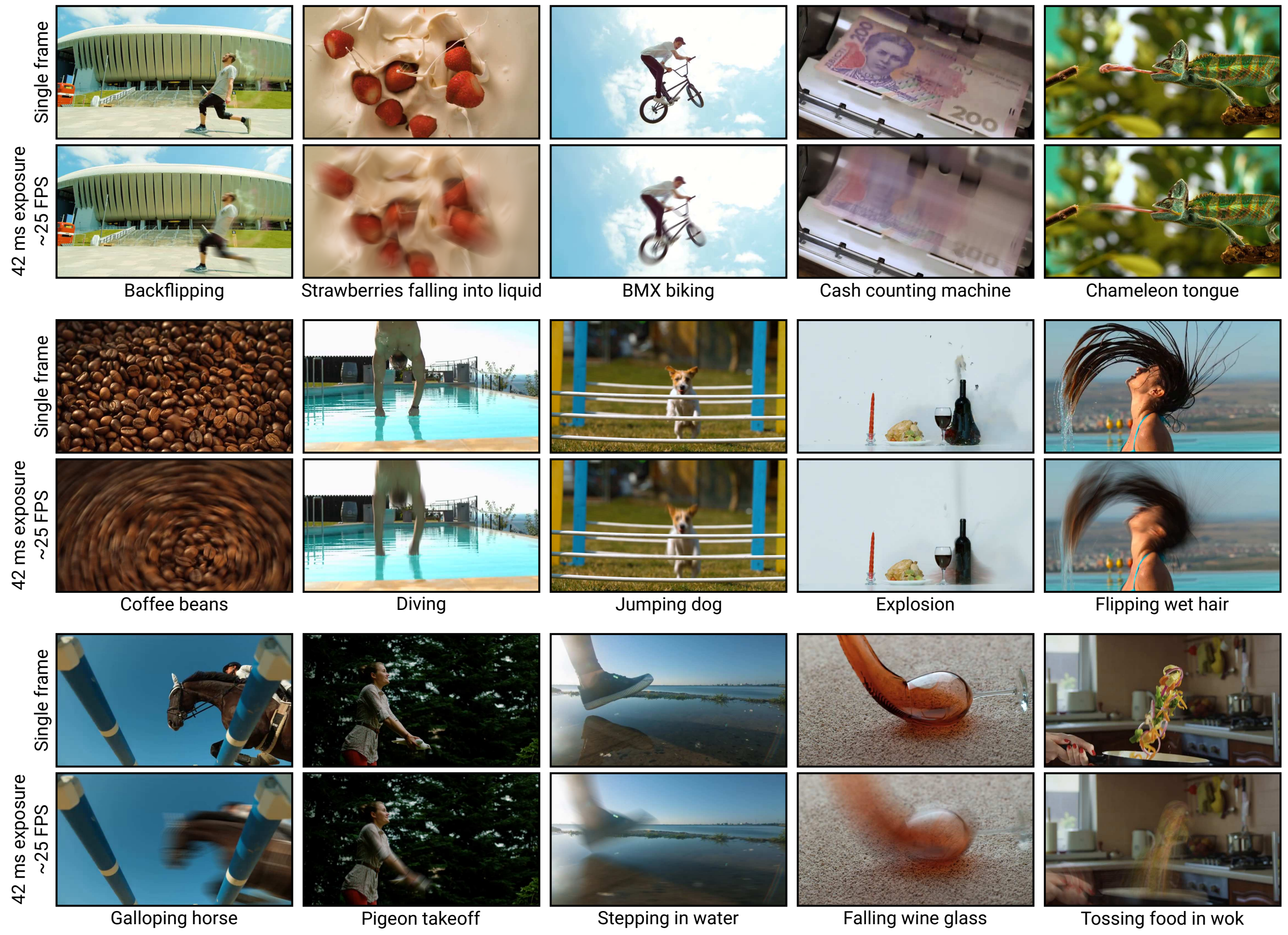
Computing metrics. We evaluate perceptual distortion using PSNR (computed from the average mean squared error across the entire video [8]), SSIM (computed per-frame and averaged) and MS-SSIM (computed per-frame and averaged) metrics. Both metrics are converted with respect to linear values. Specifically, each of our generalized event cameras and baselines (EDI++, burst denoising, coded 8-bucket) recovers the time-varying estimate of $1 - e^{-N(\mathbf{x},t)}$, *i.e.*, the probability of photon detection. Let us denote this by $\hat{p}(\mathbf{x}\ t)$. We can obtain linear estimates by computing

$$\frac{1}{\alpha}\left(\log\left(\frac{1}{1-\hat{p}(\mathbf{x}\ t)}\right) - d\right) \tag{S12}$$

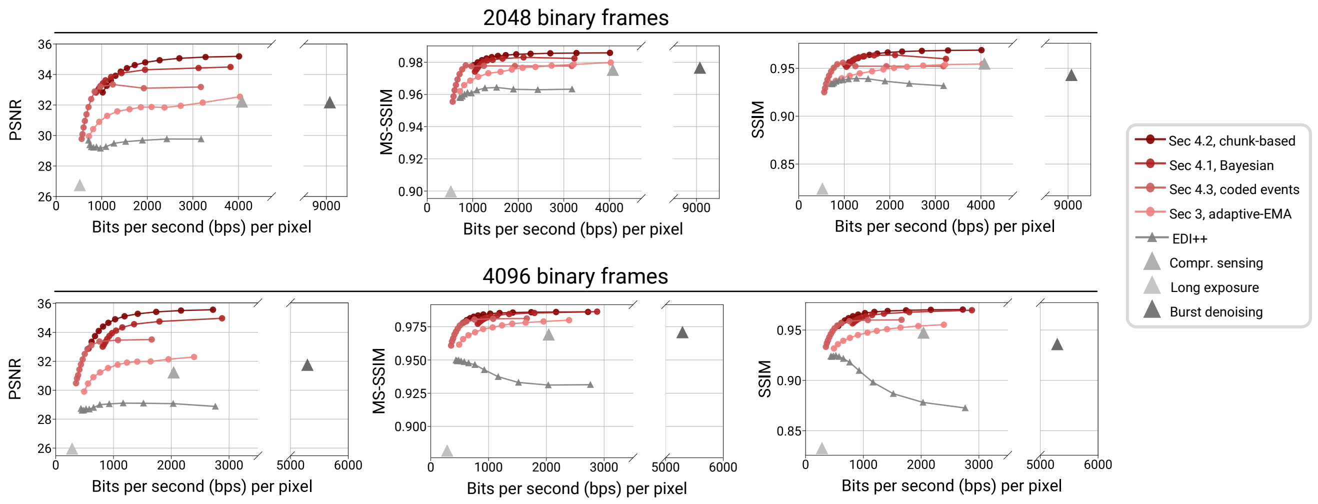
Baseline parameter sweeps. We implement EDI++ with SPAD events, with an exponential decay of $0\ 95$, and sweep the threshold between $0\ 3$–$0\ 54$. The other baselines (burst denoising, long exposure, compressive sensing) are frame-based, and do not feature a tunable parameter that controls their readout rate.

Generalized-event parameter sweeps. For adaptive-EMA, we set the exponential decay (of the EMA) to $0\ 95$, and sweep the threshold between $0\ 3$–$0\ 54$. For adaptive-Bayesian (Sec. 4.1), we retain the top-3 forecasters and vary the sensitivity γ of BOCPD uniformly (on a logarithmic scale) between 10^{-7} and $10^{-2.2}$. For the spatiotemporal chunk method (Sec. 4.2), we use a patch size of 4×4 pixels, average 32 binary frames per temporal chunk, and vary the threshold (τ) of the change detector uniformly between $0\ 6$ to $1\ 28$. For coded-exposure events, we use a chunk size of 1024 binary frames, with 4 coded buckets (measurements per chunk) that multiplex 16 subframes each—thus, each subframe consists of $1024\ \ 16 = 64$ binary frames. We vary the confidence level of Wilson's score ("α" parameter) uniformly between $1\ 2$ to $6\ 8$.

Extended results. In addition to the PSNR-based rate-distortion plot shown in Fig. 8, we include evaluations based on SSIM and MS-SSIM metrics in Suppl. Fig. 17 *(top)*. Across metrics, we see that generalized event cameras provide a pronounced difference in performance over the considered baselines (shown in shades of gray). *(bottom)* We include another evaluation on 4096 binary frames (instead of 2048). We observe that the readout rate, measured as bps/pixel, is lower across this extended duration—since the fixed readout costs of static (and less dynamic) regions is amortized over a longer duration, unlike frame-based cameras that involve fixed readout for all regions of an image. Finally, we remark that the performance gap between EDI++ and our techniques is further widened across this longer duration.



Supplementary Figure 16. **Thumbnails for YouTube-sourced videos.** Thumbnails from our YouTube-sourced evaluation dataset. We show the central frame from each clip, along with a 42 ms long exposure to illustrate the extent of motion.



Supplementary Figure 17. **Extended rate-distortion evaluation.** *(top)* Rate-distortion evaluations based on PSNR, SSIM and MS-SSIM metrics across 2048 binary frames. *(bottom)* When considering a longer temporal extent, *i.e.*, 4096 binary frames, we see that the readout rate at which a significant PSNR (or other metrics) drop-off is noticed becomes smaller—in other words, we obtain more compression of raw photon detections.

E.7. UltraPhase Experiments

System description. UltraPhase consists of 3×6 cores, each of which processes data from a 4×4 patch of SPAD pixels. The chip operates at a frequency of 0 42 GHz, implying a maximum of 4341 instructions per binary frame at 96 8 kHz. We additionally refer readers to Ardelean [3] for a detailed description of the chip architecture.

Implementing event cameras on UltraPhase. We implement our event camera designs in UltraPhase assembly code. Text files containing this code are included with this supplement (please see `assembly_adaptive_ema.txt`, `assembly_adaptive_bocpd.txt`, `assembly_spatiotemporal_chunk.txt` and `assembly_coded_exposure.txt`).

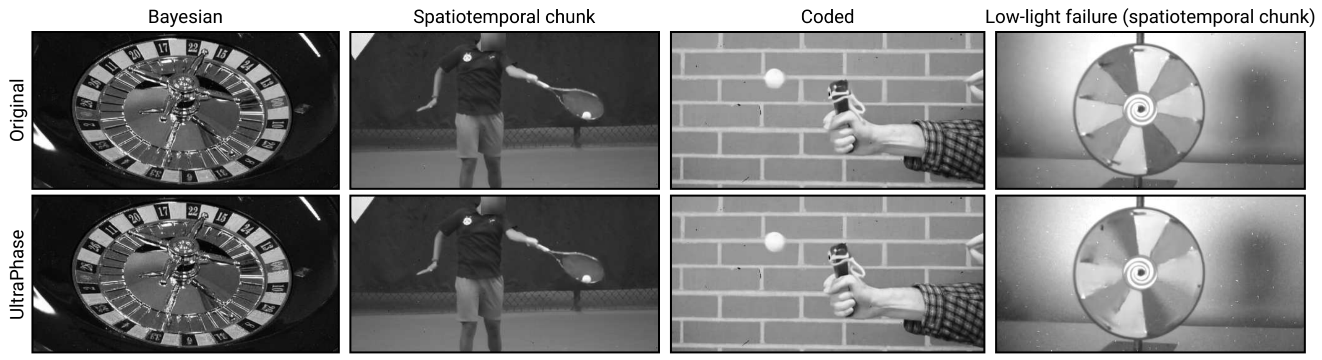
At this point, UltraPhase does not have native hardware for computing divisions—so we modify our methods to work avoid division operations. For the Bayesian method, we skip restarts and avoid division when computing the likelihood by multiplying arguments with their least common multiple—this is possible because the min and argmax operations carried out in BOCPD are invariant to the scale (of forecaster values). For the spatiotemporal chunk method, we skip the normalization step and use a 4×16, 8-bit quantized feature matrix $\boldsymbol{P}$, in contrast to the 16×16 matrix we use in the rest of our experiments. For the coded-exposure method, we consider (2-bucket, 8-subframe) masks; additionally, we replace Wilson's score by a fixed confidence interval (essentially amounting to a fixed threshold operation). In Suppl. Fig. 18 we show comparisons between the modified and original methods; we see that, for most scenes, the above modifications do not significantly reduce the quality of the results.

Clock cycle measurements. Due to circumstances beyond our control, we were unable to run our methods on a physical testbed system. We evaluate the runtime characteristics of our methods by assembling them for UltraPhase and measuring the number of compute cycles required to execute them. Given the deterministic nature of the digital hardware, the compute and memory requirements we measure in this evaluation are identical to those we would measure in physical hardware.

We confirm that all methods operate within the memory budget of the chip. In Tab. 1, we show the measured clock cycles (the average per binary frame) for each method. In the case of branching, we assume the more computationally expensive branch is taken. Therefore, all compute values are an upper bound.

Readout estimation. The chip readout depends on the dynamics of the scene. To estimate the readout, we run our methods on a 12×24 crop from the tennis sequence in Fig. 7, over 2500 binary frames. We show frames from this crop in Suppl. Fig. 19. We scale the measured readout values to units of kilobytes per second; see Tab. 1 for results.

Power estimation. We estimate two components of power consumption: compute power and chip readout power. We base our analysis on [17]. We assume the chip consumes 3 5 picojoules per clock cycle spent executing an instruction, and that the chip expends 54 nanowatts per kilobit of readout. Tab. 1 shows our estimated compute and readout power.



Supplementary Figure 18. **The effect of modifications for UltraPhase.** Some minor modifications are required to make our methods compatible with UltraPhase. As we observe in the first three columns, these modifications do not usually have any noticeable impact on the quality of the results; the modified methods *(bottom)* give results that closely match the original methods *(top)*. However, we do observe differences in low light, as seen in the rightmost column, which shows a scene captured at 0.3 lux. In low light, a lack of noise-aware thresholding (*e.g.*, the Wilson's bound for the coded method or the normalization step in the spatiotemporal chunk method) leads to less reliable change detections.

Supplementary Figure 19. **Sequence used for UltraPhase experiments.** This sequence covers 2500 binary frames (25.8 ms). The cropped portion captures the edge of a swinging racket as it makes contact with a tennis ball.

Method	Compute cycles	Bandwidth estimate (kB/s)	Compute power (W)	Readout power (W)	Total power (W)
Raw photons	12	3600	$5\ 43 \times 10^{-9}$	$1\ 57 \times 10^{-3}$	$1\ 57 \times 10^{-3}$
Adaptive-EMA	2367	59 0	$2\ 11 \times 10^{-4}$	$2\ 57 \times 10^{-5}$	$2\ 37 \times 10^{-4}$
Bayesian	3095	57 6	$3\ 62 \times 10^{-4}$	$2\ 51 \times 10^{-5}$	$3\ 87 \times 10^{-4}$
Spatiotemporal chunk	380	53 3	$5\ 45 \times 10^{-6}$	$2\ 32 \times 10^{-5}$	$2\ 87 \times 10^{-5}$
Coded	177	33 1	$1\ 18 \times 10^{-6}$	$1\ 44 \times 10^{-5}$	$1\ 56 \times 10^{-5}$

Table 1. **UltraPhase results.** We measure the number of compute cycles (per binary frame) required to implement each of our methods, and estimate the readout bandwidth. Based on these values, we estimate the power required for on-chip computation and readout. All of our methods fit within the chip's computational budget of 4341 instructions per binary frame and give two orders of magnitude reduction in bandwidth compared to reading out raw photon detections. Due to these bandwidth reductions, our methods are also much more power efficient than raw photon readout.

Supplementary References

[1] Ryan Prescott Adams and David JC MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007. 4

[2] Réda Alami, Odalric Maillard, and Raphael Féraud. Restarted bayesian online change-point detector achieves optimal detection delay. In *International conference on machine learning*, pages 211–221. PMLR, 2020. 4, 6

[3] Andrei Ardelean. *Computational Imaging SPAD Cameras*. PhD thesis, École polytechnique fédérale de Lausanne, 2023. 4, 26

[4] Xin Yuan Chengshuai Yang, Shiyu Zhang. Ensemble learning priors driven deep unfolding for scalable video snapshot compressive imaging. In *ECCV*, 2022. 10

[5] Rui Graça, Brian McReynolds, and Tobi Delbruck. Shining light on the DVS pixel: A tutorial and discussion about biasing and optimization. In *CVPRW*, pages 4045–4053, 2023. 16

[6] Samuel W. Hasinoff, Dillon Sharlet, Ryan Geiss, Andrew Adams, Jonathan T. Barron, Florian Kainz, Jiawen Chen, and Marc Levoy. Burst photography for high dynamic range and low-light imaging on mobile cameras. *ACM TOG*, 35(6):1–12, 2016. 16

[7] Yuhuang Hu, Shih-Chii Liu, and Tobi Delbruck. v2e: From video frames to realistic DVS events. In *CVPRW*, pages 1312–1321, 2021. 16

[8] Onur Keleş, M Akιn Yιlmaz, A Murat Tekalp, Cansu Korkmaz, and Zafer Doğan. On the computation of psnr for a set of images or video. In *2021 Picture Coding Symposium (PCS)*, pages 1–5. IEEE, 2021. 24

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 14

[10] Dasong Li, Xiaoyu Shi, Yi Zhang, Ka Chun Cheung, Simon See, Xiaogang Wang, Hongwei Qin, and Hongsheng Li. A simple baseline for video restoration with grouped spatial-temporal shift. In *CVPR*, pages 9822–9832, 2023. 14

[11] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2016. 14

[12] Sizhuo Ma, Shantanu Gupta, Arin C. Ulku, Claudio Bruschini, Edoardo Charbon, and Mohit Gupta. Quanta burst photography. *ACM TOG*, 39(4):1–16, 2020. 22, 23

[13] Matteo Maggioni, Vladimir Katkovnik, Karen Egiazarian, and Alessandro Foi. Nonlocal transform-domain filter for volumetric data denoising and reconstruction. *IEEE TIP*, 22(1):119–133, 2012. 16

[14] Liyuan Pan, Richard Hartley, Cedric Scheerlinck, Miaomiao Liu, Xin Yu, and Yuchao Dai. High frame rate video reconstruction based on an event camera. *IEEE TPAMI*, 44(5):2519–2533, 2022. 16

[15] SW Roberts. Control chart tests based on geometric moving averages. *Technometrics*, 42(1):97–101, 2000. 3

[16] Trevor Seets, Atul Ingle, Martin Laurenzis, and Andreas Velten. Motion adaptive deblurring with single-photon cameras. In *WACV*, pages 1945–1954, 2021. 2

[17] Varun Sundar, Andrei Ardelean, Tristan Swedish, Claudio Bruschini, Edoardo Charbon, and Mohit Gupta. Sodacam: Software-defined cameras via single-photon imaging. In *ICCV*, pages 8165–8176, 2023. 10, 16, 19, 26

[18] Alexandru Telea. An image inpainting technique based on the fast marching method. *Journal of graphics tools*, 9(1):23–34, 2004. 15

[19] Arin Can Ulku, Claudio Bruschini, Ivan Michel Antolovic, Yung Kuo, Rinat Ankri, Shimon Weiss, Xavier Michalet, and Edoardo Charbon. A 512×512 SPAD Image Sensor With Integrated Gating for Widefield FLIM. *IEEE Journal of Selected Topics in Quantum Electronics*, 25(1): 1–12, 2019. 15, 24

[20] Lishun Wang, Miao Cao, and Xin Yuan. Efficientsci: Densely connected network with space-time factorization for large-scale video snapshot compressive imaging. In *CVPR*, pages 18477–18486, 2023. 14, 16

[21] Zhaohui Wang, Xiao Lin, Abhinav Mishra, and Ram Sriharsha. Online changepoint detection on a budget. In *2021 International Conference on Data Mining Workshops (ICDMW)*, pages 414–420. IEEE, 2021. 4

[22] Mian Wei, Navid Sarhangnejad, Zhengfan Xia, Nikita Gusev, Nikola Katic, Roman Genov, and Kiriakos N. Kutulakos. Coded two-bucket cameras for computer vision. In *ECCV*, 2018. 10

[23] Guan Yu. Variance stabilizing transformations of poisson, binomial and negative binomial distributions. *Statistics & Probability Letters*, 79 (14):1621–1629, 2009. 16

[24] Xin Yuan, Yang Liu, Jinli Suo, Frédo Durand, and Qionghai Dai. Plug-and-play algorithms for video snapshot compressive imaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7093–7111, 2022. 10