

TIDES: Time-Derivative Event Simulation via Deformable Reconstruction

Christopher Thirgood*, Dipon Kumar Ghosh*, Simon Hadfield
 {c.thirgood, d.ghosh, s.hadfield}@surrey.ac.uk

University of Surrey, Guildford, Surrey, UK

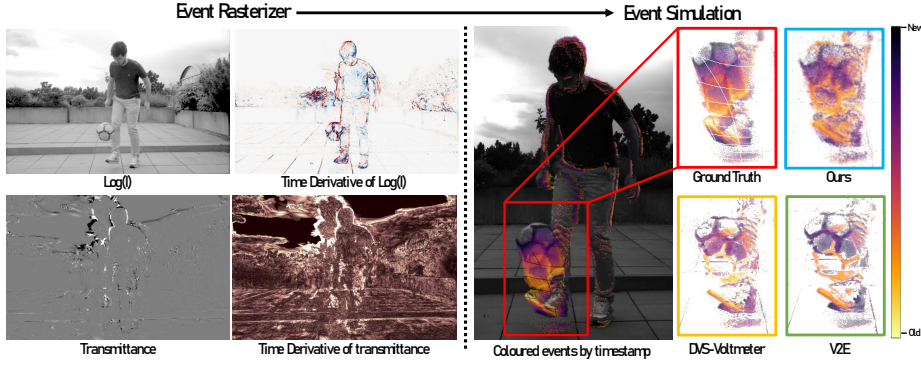


Fig. 1: TIDES at a glance: a physically grounded, end-to-end event synthesis pipeline that couples consistent scene geometry with robustness to event bursting generation

Abstract. Event cameras emit asynchronous events in response to environmental appearance changes. The scarcity of real-world event datasets makes simulation essential. However, most simulators infer event timestamps from frame sequences, forcing many threshold crossings to share a small set of discrete times; a failure mode we term timestamp batching that worsens under fast motion and occlusion.

We present TIDES, a continuous-time event simulator built on dynamic Gaussian splatting. Because TIDES operates on an explicit 3D scene representation with learnt geometry and motion, it can derive per-pixel intensity dynamics directly from the scene, rather than by differencing rendered frames. This enables accurate threshold-crossing prediction, including multiple crossings per rendering step, without temporal upsampling or frame interpolation. The same 3D scene model reveals where objects partially occlude one another; TIDES uses this to guide adaptive time stepping, concentrating computation only in regions where occlusion dynamics make simple models of brightness change unreliable.

Finally, we model finite sensor bandwidth using a tile-level arbiter whose throughput, jitter, and event drops reproduce realistic sensor artifacts.

* Equal contribution.

Across paired RGB-event benchmarks, TIDES attains state-of-the-art event-stream fidelity. We also show that events simulated by TIDES transfer more effectively to real downstream tasks than competitors’.

Keywords: Event Camera · Asynchronous Vision · Simulation

1 Introduction

Event cameras measure visual change asynchronously by emitting events when the per-pixel log-intensity change exceeds a contrast threshold [2, 6, 19]. This sensing principle provides microsecond-scale temporal resolution and high dynamic range, enabling robust perception under fast motion and challenging illumination. These capabilities make it well suited for a wide range of computer vision applications [8, 9, 17]. Since collecting and labelling large event datasets is expensive, simulation is central for training and evaluation [23, 25].

Faithful event simulation remains difficult because event timestamps are a consequence of the 3D motion and visibility at each pixel, not of differences between discretely sampled frames. The dominant failure mode in event simulation is timestamp-batching, where many threshold crossings are forced to share a small set of timestamps determined by the rendering schedule. As illustrated in Fig. 1, the issue cannot be simply resolved by interpolating sub-frame crossing points based on brightness change. This is because the real 3D world is characterised by discrete object boundaries that move continuously through space, not by smooth global appearance changes.

The timestamp-batching issue has traditionally persisted even in fully synthetic pipelines, which theoretically have access to the exact 3D motion of the scene, because existing simulators use a rendered sequence of frames as their intermediate representation. Rendering at extremely high rates can somewhat mitigate (but not fully remove) timestamp batching, but this becomes prohibitively slow and inaccurate in highly dynamic areas due to the reliance on temporal video interpolation models, which introduce artifacts. Dynamic scenes amplify these issues. Finite-difference estimates of intensity slope mix distinct visibility regimes when occlusions or disocclusions occur between samples, yielding unstable local models precisely where accurate timing matters most [11, 25].

In parallel, recent work in the field of Gaussian Splatting has provided mechanisms for real-time differentiable rendering with explicit visibility compositing. 4DGS extends these ideas to dynamic scenes via time-conditioned deformations [16, 37]. Because the scene is represented as an explicit collection of 3D primitives with known geometry and motion, the renderer provides direct access to which objects are visible, how they overlap, and how that visibility is changing. This is precisely the information that is most lacking in current frame-based event simulation methods. However, a better 3D scene model alone is not enough. The simulator must translate scene-level geometry and motion into per-pixel rates of brightness changes that are consistent with the current visibility ordering, and must allocate computation adaptively to regions where objects overlap and simple brightness models break down.

To this end, we present TIDES, a continuous-time event simulator built on dynamic 4D Gaussian splatting [37]. Rather than rendering frames and differencing them, TIDES differentiates the 3D-to-2D rendering process itself via Gaussian time dependent Jacobian vector products. Under TIDES custom Gaussian splatting rasterization method, these time-derivative rasterizations can be used to compute how fast each pixel’s brightness is changing at any instant, while correctly accounting for which objects are visible. Concretely, a forward-mode rendering pass produces log-luminance L and its instantaneous derivative $\dot{L}$ together with opacity dynamics $(\alpha, \dot{\alpha})$, all in the same compositing order as the primal image. This yields stable per-pixel threshold-crossing time prediction, including multiple crossings per step, without relying on a discretized rendering schedule.

Because the 3D scene model knows where objects overlap and how visibility is evolving, TIDES uses this information to guide adaptive time stepping, concentrating computation only where occlusion dynamics make a local linear model unreliable. Finally, TIDES models finite sensor bandwidth with a tile-level arbiter whose throughput and degradation are conditioned on the same scene-derived burst and visibility cues, reproducing burst-induced timestamp spreading and event drops. Overall, TIDES derives events from an explicit understanding of 3D scene structure and motion, rather than from frame differences.

We make the following contributions:

- A forward-mode time-derivative renderer that produces visibility-consistent $(L, \dot{L})$, enabling continuous-time threshold-crossing prediction without temporal upsampling.
- Risk-guided adaptive stepping that concentrates computation where occlusion dynamics invalidate local linearity.
- An optional tile-level readout arbiter conditioned on renderer-derived burst cues, reproducing realistic timestamp spreading and drops.
- A motion-scaled spatiotemporal fidelity metric for event streams, evaluated alongside downstream transfer benchmarks.

2 Related Work

2.1 Event simulators

Most event simulators generate events by querying intensity at discrete times and estimating threshold-crossing timestamps by interpolating changes in $\log I$. ESIM popularized this renderer-coupled design and remains a common baseline due to its simplicity and scalability [25]. However, timestamp structure is fundamentally constrained by the sampling schedule, so fast motion and visibility changes can collapse many events onto a small set of timestamps even when the comparator model is otherwise correct.

A complementary line of work improves simulator realism by modeling sensor stages beyond an ideal comparator. v2e composes photoreceptor-style dynamics with leak, threshold mismatch, background activity, and timestamp noise to

better match device behavior [13]. V2CE [41] extends this with a learned noise model calibrated to specific hardware. Circuit-inspired models such as DVS-Voltmeter provide a more mechanistic account of event generation and show how stochastic dynamics affect temporal statistics [20]. These approaches improve noise and readout realism, but when driven by discretely sampled video or renders, timestamp batching persists because event times are still inferred from under-sampled intensity signals.

Even with realistic pixel dynamics, temporal fidelity can fail if readout constraints are ignored. ICNS and related characterization work show that shared readout and arbitration can impose structured delays and drops under bursts, motivating explicit bottleneck models rather than treating timestamps as ideal crossing times [15]. Such models, however, still rely on the upstream simulator to generate realistic continuous-time bursts as input to the arbiter.

Physically grounded analytical pipelines such as PECS aim to close the realism gap by modeling image formation and sensor behavior more explicitly, including lens simulation and multispectral rendering [11]. The trade-off is practical complexity and compute [31–33], since accurate timing can require dense temporal querying and physics-heavy rendering is difficult to scale or integrate into modern learning workflows.

Our work targets the common bottleneck across these analytical simulator families: producing faithful continuous-time timestamps under non-linear visibility, without brute-force high-rate rendering. We introduce an event rasterization module that renders instantaneous temporal derivatives and visibility-consistent cues to directly support continuous-time threshold-crossing prediction, and can optionally be paired with explicit arbitration models.

2.2 Dynamic neural rendering as a substrate for event simulation

Neural rendering offers a convenient intermediate representation for event data because it supports view-dependent image formation and can be queried at arbitrary poses and times. Most prior event-based NeRF work addresses the inverse problem, learning radiance fields from events for reconstruction and novel-view synthesis [4, 14, 18, 21, 27], with extensions to dynamic scenes via time-conditioned or deformation-based fields [1, 28]. While relevant as background, volumetric NeRF rendering is typically too slow for event-rate simulation and exposes limited visibility diagnostics near occlusions.

Gaussian splatting provides real-time differentiable rendering with explicit visibility compositing [16, 30], and 4DGS extends it to dynamic scenes via time-conditioned deformations [37]. Recent event-based splatting work similarly focuses on the inverse direction, recovering Gaussian scene representations from events (sometimes with auxiliary frames) for reconstruction and novel-view rendering [5, 10, 38–40]. In contrast, we are the first to study the forward problem; using a learned 4DGS scene as input, and simulating events by predicting continuous-time threshold crossings from renderer-native dynamics and visibility signals.

3 Method

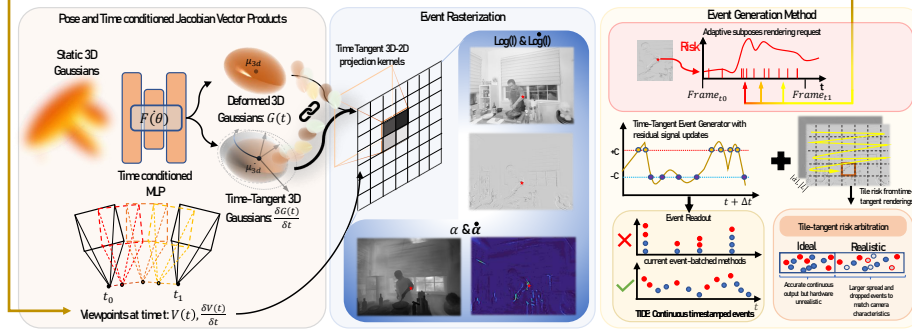


Fig. 2: System diagram for TIDES. A primal and time-derivative splatting pass outputs $(L, \dot{L}, \alpha, \dot{\alpha})$ with visibility-consistent compositing. These dynamics drive continuous-time threshold-crossing event times, avoid timestamp batching, and trigger risk-guided adaptive sub-posing under high motion and mixed visibility. An optional tile-level arbiter conditions readout spreading and drops on the same burst and visibility cues.

The central design goal of TIDES is to predict event timestamps from an understanding of the underlying 3D structure and motion of the environment, rather than from finite differences between discretely rendered frames. To this end, our event rasterizer takes a continuous-time dynamic Gaussian splatting scene (fit from RGB videos), and produces an event stream by solving per-pixel contrast threshold crossings in continuous time. Between consecutive camera timestamps, we advance time with an adaptive step Δt and query a time-derivative splatting renderer. At each query time t and pixel location $\mathbf{u}$ TIDES outputs log-luminance $L(\mathbf{u}, t)$ and its instantaneous derivative $\dot{L}(\mathbf{u}, t)$. It also outputs opacity $\alpha(\mathbf{u}, t)$ and its derivative $\dot{\alpha}(\mathbf{u}, t)$, as well as lightweight accumulation diagnostics $(b(\mathbf{u}, t), c(\mathbf{u}, t))$. Crucially, both $\dot{L}$ and $\dot{\alpha}$ are differentiated under the active front-to-back compositing order, yielding visibility-consistent instantaneous dynamics. This makes the threshold-crossing timestamp estimation a well-posed root-finding problem and provides renderer-native signals for adaptive stepping and readout modelling.

3.1 Dynamic Gaussian splatting image formation

We use EWA Gaussian splatting and front-to-back compositing as in 3DGS [16], with time-dependent parameters as in dynamic Gaussian representations [37]. At time t , each Gaussian k induces a screen-space elliptical kernel with projected mean $\mathbf{u}_k(t)$ and precision (inverse covariance) $\mathbf{Q}_k(t)$. For rendered pixel $\mathbf{u}$, the unnormalized contribution from Gaussian k is

$$g_k(\mathbf{u}, t) = \exp\left(-\frac{1}{2}(\mathbf{u} - \mathbf{u}_k(t))^T \mathbf{Q}_k(t)(\mathbf{u} - \mathbf{u}_k(t))\right). \quad (1)$$

We define the per-splat opacity contribution

$$\tau_k(\mathbf{u}, t) = \text{clamp}(\sigma_k(t) g_k(\mathbf{u}, t), 0, \tau_{\max}), \quad (2)$$

where $\sigma_k(t)$ is the Gaussian’s time dependent opacity and $\tau_{\max} < 1$ is a fixed saturation constant for numerical stability. To find the Gaussians’ final visibility weights w_k , their transmittance is composited in front-to-back order, with transmittance $T_0(\mathbf{u}, t) = 1$,

$$w_k(\mathbf{u}, t) = T_{k-1}(\mathbf{u}, t) \tau_k(\mathbf{u}, t), \quad T_k(\mathbf{u}, t) = T_{k-1}(\mathbf{u}, t)(1 - \tau_k(\mathbf{u}, t)). \quad (3)$$

Thus, the rendered RGB ($\mathbf{I}$) and opacity (α) are combined across Gaussians as

$$\mathbf{I}(\mathbf{u}, t) = \sum_k w_k(\mathbf{u}, t) \mathbf{c}_k(t) + T_K(\mathbf{u}, t) \mathbf{I}_{bg}(t), \quad \alpha(\mathbf{u}, t) = 1 - T_K(\mathbf{u}, t). \quad (4)$$

Where $\mathbf{c}_k(t)$ is the time-dependent color of Gaussian k and $\mathbf{I}_{bg}$ is the time-dependent background color. We convert to luminance $Y(\mathbf{u}, t) = \mathbf{w}^\top \mathbf{I}(\mathbf{u}, t)$ using channel-weightings $\mathbf{w}$ and define the log signal that defines event-generation

$$L(\mathbf{u}, t) = \log(Y(\mathbf{u}, t) + \epsilon). \quad (5)$$

This compositing chain defines the visibility ordering; TIDES evaluates time dynamics under the same ordering to obtain visibility-consistent event timings. We provide the full projection Jacobians and screen-space covariance expressions used to compute $\mathbf{u}_k(t)$ and $\mathbf{Q}_k(t)$ in the supplementary material.

3.2 Forward-mode time-derivative Gaussian splatting

Accurate crossing-time prediction requires an instantaneous log-luminance slope that is consistent with renderer visibility. To be clear, the volumetric rendering process described above is differentiable. However, the derivatives of the rendering process itself are not what drives event-generation. Rather, we must first extract the derivatives of the scene parameters with respect to time, and then propagate them through the derivatives of our rendering process. This allows us to understand analytically and explicitly, how the final rendered appearance will change with respect to time ($\dot{L}$), instead of approximating it by differencing two rendering outputs.

More formally, a learned deformation model maps time to Gaussian parameters

$$\Theta(t) = (\boldsymbol{\mu}(t), \mathbf{s}_{\text{raw}}(t), \mathbf{q}_{\text{raw}}(t), \sigma_{\text{raw}}(t), \mathbf{a}(t)), \quad \Theta : \mathbb{R} \rightarrow \mathbb{R}^{P \times d}, \quad (6)$$

where P is the number of Gaussians. This model is differentiable, so we recover the time derivatives $\dot{\Theta}(t) \triangleq \frac{d}{dt} \Theta(t)$ via a Jacobian-vector product (JVP) along the time axis, yielding per-Gaussian $(\dot{\boldsymbol{\mu}}_k, \dot{\mathbf{s}}_{\text{raw},k}, \dot{\mathbf{q}}_{\text{raw},k}, \dot{\sigma}_{\text{raw},k}, \dot{\mathbf{a}}_k)$. This gives exact time derivatives in one evaluation, avoiding the inaccuracy of finite difference

approximation, while remaining lightweight compared to reverse-mode differentiation through the rasterizer.

We compute this Jacobian–vector product in forward-mode automatic differentiation through the deformation network, not by finite differences in time. Because the renderer applies element-wise activations to the raw deformation outputs, we propagate the time derivatives through these activations analytically via the chain rule, for example

$$\mathbf{s}_k = \exp(\mathbf{s}_{\text{raw},k}), \quad \dot{\mathbf{s}}_k = \mathbf{s}_k \odot \dot{\mathbf{s}}_{\text{raw},k}, \quad (7)$$

$$\sigma_k = \text{sigmoid}(\sigma_{\text{raw},k}), \quad \dot{\sigma}_k = \sigma_k(1 - \sigma_k) \dot{\sigma}_{\text{raw},k}. \quad (8)$$

Quaternions are normalized before rasterization, and we propagate $\dot{\mathbf{q}}_k$ consistently through this normalization.

If the camera is nonstatic, we treat the camera parameters as time-dependent renderer inputs. When ground truth camera time derivatives $\dot{\mathbf{V}}(t), \dot{\mathbf{P}}(t), \dot{\mathbf{c}}(t)$ are not available, they are estimated from neighboring camera timestamps. Note that this does not reintroduce finite differences for contrast slopes, since $\dot{L}$ is still computed analytically by differentiating the resulting renderer traversal itself.

The time-tangent renderer computes per-Gaussian screen-space primitives $(\mathbf{u}_k, \dot{\mathbf{u}}_k, \mathbf{Q}_k, \dot{\mathbf{Q}}_k)$ and appearance $(\mathbf{c}_k, \dot{\mathbf{c}}_k)$, then composites them in the same front-to-back order as the primal pass. This ensures that temporal derivatives are consistent with visibility decisions such as early termination, clamping, and occlusion ordering. In particular, $\dot{\mathbf{c}}_k$ includes derivatives through the view direction $\mathbf{v}_k(t)$ and the spherical-harmonic basis, and therefore captures motion-induced view-dependent appearance changes.

We now derive $\dot{L}$ by differentiating each stage of the compositing chain in Sec. 3.1, starting from per-Gaussian screen-space contributions and propagating through front-to-back blending. Using Eq. 1, let

$$p_k(\mathbf{u}, t) = -\frac{1}{2} \mathbf{d}_k^\top \mathbf{Q}_k \mathbf{d}_k, \quad \mathbf{d}_k = \mathbf{u} - \mathbf{u}_k(t). \quad (9)$$

Since $\mathbf{u}$ is fixed, $\dot{\mathbf{d}}_k = -\dot{\mathbf{u}}_k$ and

$$\dot{p}_k = -\frac{1}{2} \left(\mathbf{d}_k^\top \dot{\mathbf{Q}}_k \mathbf{d}_k + 2 \dot{\mathbf{d}}_k^\top \mathbf{Q}_k \mathbf{d}_k \right), \quad \dot{g}_k = g_k \dot{p}_k. \quad (10)$$

Using Eq. 2, define $\tau_k^{\text{raw}} = \sigma_k g_k$ and

$$\dot{\tau}_k^{\text{raw}} = \dot{\sigma}_k g_k + \sigma_k \dot{g}_k, \quad \dot{\tau}_k = \begin{cases} 0 & \text{if } \tau_k \text{ is saturated at } \tau_{\text{max}}, \\ \dot{\tau}_k^{\text{raw}} & \text{otherwise.} \end{cases} \quad (11)$$

With per-Gaussian derivatives in hand, we differentiate the front-to-back compositing. Differentiating Eq. 3 yields

$$w_k = T_{k-1} \tau_k, \quad \dot{w}_k = \dot{T}_{k-1} \tau_k + T_{k-1} \dot{\tau}_k, \quad (12)$$

$$T_k = T_{k-1}(1 - \tau_k), \quad \dot{T}_k = \dot{T}_{k-1}(1 - \tau_k) - T_{k-1} \dot{\tau}_k. \quad (13)$$

The final opacity derivative follows from $\alpha = 1 - T_K$ as $\dot{\alpha} = -\dot{T}_K$. Finally, differentiating Eq. 4 yields the time derivative of the rendered image

$$\dot{\mathbf{I}} = \sum_k (\dot{w}_k \mathbf{c}_k + w_k \dot{\mathbf{c}}_k) + \dot{T}_K \mathbf{I}_{bg} + T_K \dot{\mathbf{I}}_{bg}. \quad (14)$$

With luminance $Y = \mathbf{w}^\top \mathbf{I}$ and $\dot{Y} = \mathbf{w}^\top \dot{\mathbf{I}}$, we obtain

$$\dot{L}(\mathbf{u}, t) = \frac{\dot{Y}(\mathbf{u}, t)}{Y(\mathbf{u}, t) + \epsilon}. \quad (15)$$

3.3 Comparator and sensor model

We next extend $(L, \dot{L})$ from the time-derivative renderer to define a per-pixel driving signal $S(\mathbf{u}, t) = \mathcal{F}(L(\mathbf{u}, t))$, where $\mathcal{F}$ is an optional sensor-front-end operator that models pixel-level analog processing. By default $\mathcal{F}$ is identity, so $S = L$ and $\dot{S} = \dot{L}$ directly. When enabled, $\mathcal{F}$ composes standard sensor-modelling blocks: optical Point-Spread-Function blur (applied to both L and $\dot{L}$), a first-order photoreceptor low-pass, a center-surround spatial filter, and additive leak or noise terms. This sensor front-end is orthogonal to our continuous-time timing formulation.

We adopt a residual-state comparator. Each pixel maintains a reference $S_{\text{ref}}(\mathbf{u})$, initialized to $S(\mathbf{u}, t_0)$ and updated at each firing. An event is triggered when the signal departure from this reference reaches a contrast threshold: $S(\mathbf{u}, t) - S_{\text{ref}}(\mathbf{u}) = +C_{\mathbf{u}}^+$ for a positive event, or $= -C_{\mathbf{u}}^-$ for a negative one. In the simplest case both polarities share a single global threshold $C_{\mathbf{u}}^+ = C_{\mathbf{u}}^- = C$. To model realistic sensor variation we optionally allow asymmetric thresholds ($C^+ \neq C^-$) and per-pixel jitter by drawing $C_{\mathbf{u}}^+, C_{\mathbf{u}}^-$ independently around nominal values; once sampled, thresholds remain fixed for the simulation. We optionally also model comparator leak as a slow reference drift $S_{\text{ref}} \leftarrow S_{\text{ref}} - \lambda_{\text{leak}} \Delta t$.

Within a simulation window $(t_0, t_0 + \Delta t]$ we approximate S locally. The first-order (linear) model is

$$S(\mathbf{u}, t_0 + \delta) \approx S_0(\mathbf{u}) + \dot{S}_0(\mathbf{u}) \delta, \quad \delta \in (0, \Delta t], \quad (16)$$

where $S_0 = S(\cdot, t_0)$ and $\dot{S}_0 = \dot{S}(\cdot, t_0)$. Because $\dot{S}_0$ is obtained by analytic differentiation of the rendering pipeline, it inherits per-pixel visibility ordering and reflects occlusion and disocclusion dynamics without finite-difference approximation. When a second render $S_1 = S(\cdot, t_0 + \Delta t)$ is available, we optionally fit a quadratic local model using S_0 , $\dot{S}_0$, and S_1 , improving timing under signal curvature.

Given the local model, we solve for crossing times. Under the linear model with $\dot{S}_0(\mathbf{u}) \neq 0$, the candidates are

$$\delta_+(\mathbf{u}) = \frac{S_{\text{ref}}(\mathbf{u}) + C_{\mathbf{u}}^+ - S_0(\mathbf{u})}{\dot{S}_0(\mathbf{u})}, \quad \delta_-(\mathbf{u}) = \frac{S_{\text{ref}}(\mathbf{u}) - C_{\mathbf{u}}^- - S_0(\mathbf{u})}{\dot{S}_0(\mathbf{u})}. \quad (17)$$

Under the quadratic model, the candidates are the smallest positive roots of the corresponding quadratic in $(0, \Delta t]$. We consider δ_+ only when $\dot{S}_0 > 0$ and δ_- only when $\dot{S}_0 < 0$, and accept a candidate only if $\delta \in (0, \Delta t]$. The earliest valid crossing δ^* yields the event timestamp $t^* = t_0 + \delta^*$. We append $(t^*, \mathbf{u}, +1)$ to the output stream $\mathcal{E}$ and set $S_{\text{ref}} \leftarrow S_{\text{ref}} + C_{\mathbf{u}}^+$ if the positive threshold was reached, or append $(t^*, \mathbf{u}, -1)$ and set $S_{\text{ref}} \leftarrow S_{\text{ref}} - C_{\mathbf{u}}^-$ otherwise. The procedure repeats on the remaining interval $(t^*, t_0 + \Delta t]$, up to N_{max} events per pixel per step.

3.4 Adaptive rendering

Crossing-time prediction is least reliable near disocclusions and mixed visibility. The compositing loop exposes three diagnostic signals at negligible extra cost. The transmittance rate $|\dot{\alpha}(\mathbf{u}, t)|$ (rapid visibility change), the contributor count $c(\mathbf{u}, t)$ (depth competition), and the coverage proxy $b(\mathbf{u}, t)$ (accumulated blending mass). We combine them into a per-pixel risk score

$$r(\mathbf{u}, t) = \text{clip}(w_b (1 - \text{norm}(b)) + w_c \text{norm}(c) + w_\alpha \text{norm}(|\dot{\alpha}|)), \quad (18)$$

and use r to shrink the step size via $\Delta t \leftarrow \Delta t \kappa(r)$ with $\kappa(r) \in (0, 1]$, and to suppress event emission when $b < b_{\text{min}}$.

3.5 Derivative-conditioned finite-bandwidth readout

Real sensors serialize events through a finite-bandwidth readout, so localized bursts cause queuing delays, jitter, and drops. We optionally model this with a tile-scanning arbiter: tiles maintain FIFO queues and a deterministic scanner releases at most K_m events from tile m per tick, with drop probability $p_{\text{drop},m}$ and additive timestamp jitter $\sigma_{t,m}$. We condition these on a tile-level activity score $\psi_m \in [0, 1]$ aggregating the event-rate proxy $|\dot{L}|/C$, the transmittance rate $|\dot{\alpha}|$, and the contributor count c :

$$K_m = \lfloor K_0 (1 - \beta_K \psi_m) \rfloor, \quad p_{\text{drop},m} = p_0 + \eta \psi_m, \quad \sigma_{t,m} = \sigma_0 + \gamma_\sigma \psi_m, \quad (19)$$

where K_0, p_0, σ_0 are steady-state baselines and $\beta_K, \eta, \gamma_\sigma$ control the degradation rate. Under high activity, throughput falls while drops and jitter grow, reproducing realistic readout saturation.

4 Experiments

4.1 Experimental setting

We evaluate simulators along three axes: (i) event-stream fidelity against real events, (ii) event timestamp ratio metrics to analyze event bursts and (iii) synthetic-to-real transfer when training on simulated events and testing on real data. The results presented here are the average across all sequences in each dataset, while the supplementary material provides per-sequence results.

We compare TIDES against ESIM [25], V2E [13], DVS-Voltmeter [20], and ICNS/IEBCS [24]. We do not include PECS [11] as a baseline because its released pipeline targets a different scene-based setup than our shared-renderer protocol; evaluating PECS within our shared 4DGS-renderer protocol would require disabling its native physical image-formation pipeline, yielding a comparison that is methodologically misaligned and potentially unfair to PECS. Furthermore, PECS uses its own benchmarking protocol isolating other methods against itself.

We report a fixed-budget setting with a $10\times$ query schedule for all methods and additionally ablate our adaptive rendering setting for TIDES as a separate row of results. Our adaptive pose-sampler varied greatly in the frequency of sub-pose requests for each scene. Baseline methods struggled to operate with non-linear timestamp intervals at inference time, and performed universally worse when combined with our adaptive rendering.

4.2 Datasets and shared renderer

We evaluate on four real datasets spanning ego-motion and dynamic-scene regimes: EDS and DSEC capture ego-motion, while HS-ERGB and BS-ERGB contain dynamic objects [7, 12, 34, 35]. When 6-DoF trajectories are provided, we use the official poses; otherwise, we estimate camera motion and sparse geometry from RGB using COLMAP for static scenes and Easi3R for dynamic scenes [3, 36]. For each sequence we fit a dynamic Gaussian splatting scene from which we can render log-luminance $L(\mathbf{u}, t)$ at any timestamp, providing a shared continuous-time rendering source for all simulators. Frame-driven baselines are given renders at regular intervals, while step-based methods query only at their chosen times. This isolates differences in time-stamping, thresholding, readout modeling, and occlusion handling from interpolation artifacts in the inputs.

4.3 Metrics

We report three complementary metric families that target distinct simulator failure modes: (i) inter-event timing statistics, (ii) spatiotemporal alignment to real events, and (iii) batching/burst distortions.

Event generation can be characterized by the distribution of inter-spike intervals (ISI) τ at each pixel. Following DVS-Voltmeter-style analysis, we fit an inverse-Gaussian (IG) model to observed ISIs and report the negative log-likelihood (IG-NLL) of real events under the simulator’s ISI distribution. Lower IG-NLL indicates that the simulator reproduces the fine-grained timing of events rather than only their spatial support as a batch.

To measure whether simulated events occur at the correct location and time, we compare polarity-matched real and simulated event sets in a motion-scaled spatiotemporal embedding. Each event (x, y, t) is mapped to $\mathbf{e} = (x, y, vt)$, where v (pixels/s) converts time errors into an expected spatial displacement so that a temporal offset Δt is penalized as $v\Delta t$. We compute a bidirectional nearest-neighbour (Chamfer-style) distance between the embedded point sets over fixed

Table 1: Event fidelity per dataset. Best, second-best, and third-best per column are highlighted in red, orange, and yellow, respectively (excluding TIDES (adaptive) from color ranking).

Simulator	EDS		HS-ERGB		BS-ERGB		DSEC	
	IG-NLL ↓	Chamfer ↓	IG-NLL ↓	Chamfer ↓	IG-NLL ↓	Chamfer ↓	IG-NLL ↓	Chamfer ↓
ESIM	0.004748	0.046932	0.006763	0.021335	0.007669	0.039233	0.096768	0.150414
V2E	0.004058	0.044124	0.005823	0.016149	0.006092	0.035944	0.099792	0.153248
ICNS/IEBCS	0.004575	0.046710	0.008341	0.028272	0.010608	0.045028	0.092576	0.142781
DVS-Voltmeter	0.006050	0.050914	0.006388	0.014871	0.005163	0.033895	0.080542	0.133779
TIDES (10x)	0.003840	0.043245	0.004984	0.014892	0.004392	0.033002	0.068641	0.121962
TIDES (adaptive)	0.003733	0.042622	0.002343	0.013772	0.004251	0.032260	0.068439	0.120611

temporal windows, with voxel-grid density control to reduce sensitivity to repeated near-duplicate events. Lower values indicate better spatiotemporal agreement without requiring explicit correspondences.

To diagnose timestamp artifacts that are visually evident as “layered” event bands, we report: (i) Same-ts, the fraction of events sharing identical timestamps within a window (higher implies stronger batching); (ii) ISI spike ratio, the fraction of ISIs near zero or concentrated at discrete clock quanta (indicating over-quantized timestamps); and (iii) Fano-factor error, which measures whether the simulator matches real burst dispersion by comparing the Fano factor $F = \text{Var}(N)/\mathbb{E}[N]$ of event counts in fixed spatiotemporal bins:

$$\mathcal{E}_{\text{Fano}} = \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} \left| \log \frac{F_b^{\text{pred}}}{F_b^{\text{gt}}} \right|. \quad (20)$$

Lower Same-ts, lower ISI spike ratio, and lower $\mathcal{E}_{\text{Fano}}$ indicate reduced batching and more realistic burst statistics. Full definitions and implementation details (binning, windowing, and normalization) are provided in the supplementary material.

4.4 Downstream tasks

To measure downstream task transfer, we train a range of models on simulated events and evaluate on real data. The specific tasks and models we use are: (1) E2VID [26] for reconstruction, (2) TimeLens-XL [22] for video frame interpolation (VFI), (3) Depth Any Event Stream [42] for depth estimation, and (4) ESS on DSEC [29] for segmentation. For VFI, we use the image reconstructed from the ground-truth event stream as reference and report PSNR and LPIPS. Additional motion-task results (flow and odometry) are reported in the supplementary tables.

4.5 State-of-the-art comparison

Table 1 and Table 2 report event-stream fidelity and timestamp-burst diagnostics, while Table 3 reports downstream transfer.

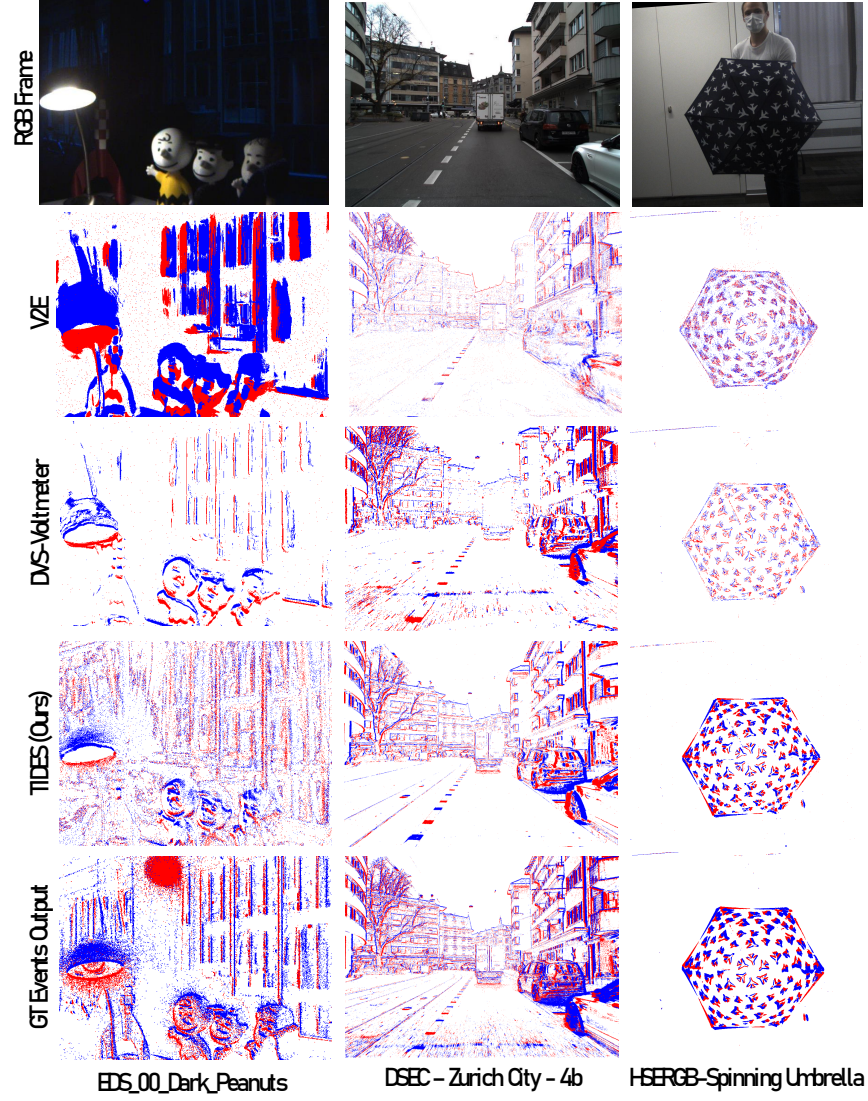


Fig. 3: Illustrations of the event frames created from the ground truth, TIDES and popular event simulation methods V2E and DVS-Voltmeter.

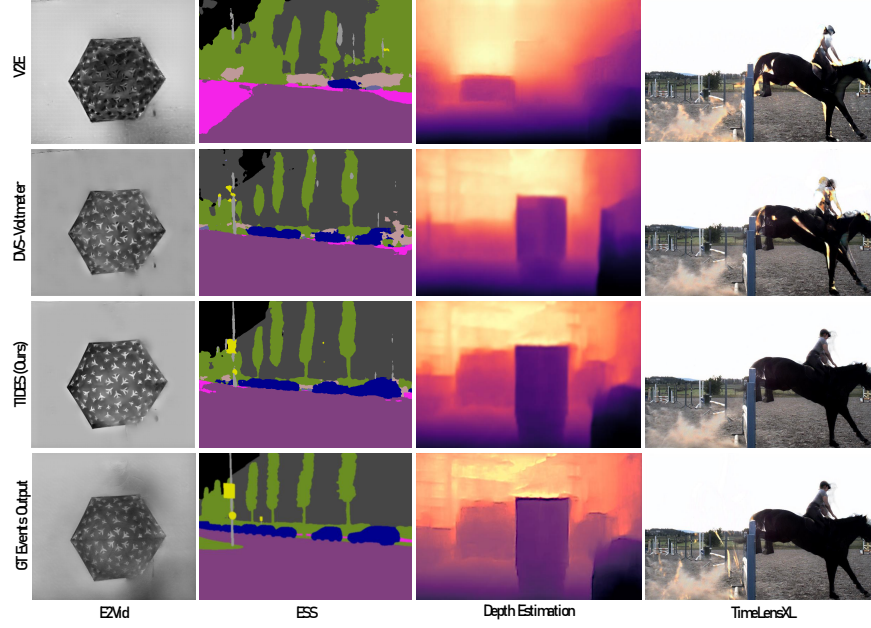


Fig. 4: Visualizations of the downstream models using individual event generations including the ground truth event.

Table 2: Batching diagnostics across datasets (average)

Simulator	EDS			HS-ERGB			BS-ERGB			DSEC		
	Same-ts ↓	Fano ↓	ISI spike ↓	Same-ts ↓	Fano ↓	ISI spike ↓	Same-ts ↓	Fano ↓	ISI spike ↓	Same-ts ↓	Fano ↓	ISI spike ↓
ESIM	0.281985	0.506379	0.096938	0.166213	3.305385	0.064902	0.279757	1.689136	0.111152	0.165757	3.296069	0.050237
V2E	0.076272	4.284378	0.047712	0.126701	6.409716	0.081733	0.155642	6.473020	0.089496	0.226925	6.495386	0.141100
ICNS/IEBCS	0.046909	2.417609	0.034290	0.074536	4.975071	0.049699	0.102743	5.063936	0.065669	0.114483	4.957434	0.073942
DVS-Volmeter	0.102028	0.313285	0.061608	0.076948	2.698272	0.051943	0.112803	1.436576	0.069397	0.060686	3.155488	0.039239
TIDES (10x)	0.040882	0.228440	0.052497	0.050215	2.069280	0.038454	0.085554	0.816516	0.041768	0.057469	2.332386	0.024181
TIDES (adaptive)	0.025234	0.338062	0.055599	0.051124	2.017263	0.046340	0.080981	0.608790	0.033215	0.056436	2.199856	0.021470

TIDES achieves the best fidelity across all four datasets under both a fixed $10\times$ query budget and our adaptive budget (Table 1). Gains are largest under fast motion and occlusions, where frame-driven simulators alias visibility changes and finite-difference slope estimates collapse events onto a few timestamps. In contrast, TIDES renders instantaneous, visibility-consistent $\dot{L}$ in the same compositing order as the primal image, enabling continuous-time threshold crossings (including multi-crossing per pixel) without choosing an arbitrary differencing interval.

Burst diagnostics in Table 2 show that improvements come from corrected timestamp structure, not event-rate inflation like v2e typically highlights from its extreme batching. TIDES consistently reduces Same-ts batching and burst distortion (Fano, ISI spikes). Adaptive stepping further improves IG-NLL and Chamfer by refining time only in mixed-visibility regions indicated by $(\alpha, \dot{\alpha}, b, c)$, with the largest gains on HS/BS-ERGB where pose sampling is sparse. Among

baselines, V2E and ICNS/IEBCS improve realism via sensor/readout blocks, and DVS-Voltmeter models stochastic timing, but they remain limited by discretely sampled driving signals; TIDES removes this bottleneck by coupling continuous-time rendering with visibility-consistent derivatives and risk-aware time control. These results are exacerbated by the visual event frames seen in Fig. 3.

TIDES produces smoother temporal dispersion along motion boundaries and fewer frame-aligned timestamp bands than frame-driven baselines, matching the reduced batching statistics in Table 2. These improvements translate to downstream transfer, models trained on TIDES events achieve the best overall performance across reconstruction, frame interpolation, depth, and segmentation (Table 3), with qualitative examples shown in Fig. 4.

Table 3: Quantitative downstream transfer (train on simulated, test on real).

Simulator	E2VID (HS-ERGB)		VFI (BS-ERGB)		EDA (DSEC)		ESS (DSEC)
	PSNR $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	LPIPS $\downarrow$	RMSE $\downarrow$	RMSE log $\downarrow$	mIoU $\uparrow$
ESIM	12.2321	0.3283	28.5777	0.0580	12.6627	0.3163	<i>17.2170</i>
V2E	16.3587	0.2469	28.1585	0.0616	12.6108	0.3125	7.5678
ICNS/IEBCS	15.2137	<i>0.2097</i>	<i>31.0376</i>	<i>0.0422</i>	19.6507	0.8791	16.3643
DVS-Voltmeter	<i>17.7328</i>	0.2254	28.6220	0.0514	<i>10.9048</i>	<i>0.2865</i>	16.9245
TIDES (ours)	18.1587	0.1997	34.3054	0.0335	10.8160	0.2795	17.6898

4.6 Ablations

We ablate each module while keeping the rendering source, sensor parameters, and evaluation protocol fixed. Table 4 shows that visibility-consistent time-tangent derivatives are the largest contributor as replacing the time-tangent slope with finite differences increases IG-NLL and Chamfer substantially. Adaptive control is the next most important factor, confirming that refining time in mixed visibility improves timing beyond a fixed query schedule. Disabling multi-crossing or mixed-visibility masking also degrades fidelity, while removing the readout model causes a smaller but consistent drop.

5 Conclusion

We presented TIDES, a continuous-time event simulator built on dynamic Gaussian splatting that predicts per-pixel threshold crossings without discretizing time. TIDES renders log-luminance and its instantaneous derivative in the same visibility ordering, making threshold-crossing prediction well-posed without the timestamp batching artifacts prevalent in rendering-based simulators. Across four paired RGB-event benchmarks, TIDES achieves state-of-the-art fidelity and demonstrates stronger synthetic-to-real transfer on downstream event-vision tasks than existing frame-based simulators. The quality of events simulated

Table 4: Ablations isolating TIDES components on EDS scenes.

Variant	IG-NLL ↓	Chamfer ↓
Full TIDES	0.00373	0.04262
Finite-diff slope ($\dot{L}$ by FD)	0.00521	0.04801
No multi-crossing (1 event max / step)	0.00439	0.04487
Fixed step size (no controller)	0.00486	0.04672
No risk-gated refinement (no $(\alpha, \dot{\alpha}, b, c)$ shrink)	0.00458	0.04593
No mixed-visibility masking (emit even when $b < b_{\min}$)	0.00422	0.04431
No $\dot{\alpha}$ cue	0.00461	0.04711
No readout model (ideal timestamps)	0.00388	0.04294
Compute-matched TIDES (adaptivity off, fixed #queries)	0.00405	0.04371

through TIDES is ultimately bounded by the fidelity of the underlying 4DGS scene reconstruction. However, our technique is agnostic to any particular 4DGS technique. Future efforts to improve the 4DGS rendering substrate will naturally translate to TIDES ongoing improvement.

Acknowledgement

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC), part of UK Research and Innovation (UKRI), under grant reference UKRI560.

References

1. Bhattacharya, A., Madaan, R., Cladera, F., Vemprala, S., Bonatti, R., Daniilidis, K., Kapoor, A., Kumar, V., Matni, N., Gupta, J.K.: Evdnerf: Reconstructing event data with dynamic neural radiance fields. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 5846–5855 (Jan 2024)
2. Brandli, C., Berner, R., Yang, M., Liu, S.C., Delbruck, T.: A 240×180 130 db 3 μ s latency global shutter spatiotemporal vision sensor **49**(10), 2333–2341 (2014). <https://doi.org/10.1109/JSSC.2014.2342715>
3. Chen, Y., Alliez, P., Aubry, M., Barrau, A., Cabon, Y., Revaud, J.: Easi3r: Estimating disentangled motion from dust3r. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2025). <https://doi.org/10.48550/arXiv.2503.24391>, <https://www.cvlibs.net/publications/Chen2025ICCV.pdf>
4. Feng, C., Yu, W., Cheng, X., Tang, Z., Zhang, J., Yuan, L., Tian, Y.: Ae-nerf: Augmenting event-based neural radiance fields for non-ideal conditions and larger scene. arXiv preprint arXiv:2501.02807 (2025)
5. Feng, C., Yu, W., Cheng, X., Tang, Z., Zhang, J., Yuan, L., Tian, Y.: E-4dgs: High-fidelity dynamic reconstruction from the multi-view event cameras. In: Proceedings of the 33rd ACM International Conference on Multimedia (ACM MM). pp. 7356–7365 (2025). <https://doi.org/10.1145/3746027.3754777>

6. Gallego, G., Delbrück, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A.J., Conradt, J., Daniilidis, K., Scaramuzza, D.: Event-based vision: A survey **44**(1), 154–180 (2022). <https://doi.org/10.1109/TPAMI.2020.3008413>
7. Gehrig, M., Aarents, W., Gehrig, D., Scaramuzza, D.: Dsec: A stereo event camera dataset for driving scenarios. *IEEE Robotics and Automation Letters* (7 2021). <https://doi.org/10.1109/LRA.2021.3068942>, <https://magehrig.github.io/publication/gehrig-2021-ral-dsec/>
8. Ghosh, D.K., Jung, Y.J.: Two-stage cross-fusion network for stereo event-based depth estimation. *Expert Systems with Applications* **241**, 122743 (2024). <https://doi.org/https://doi.org/10.1016/j.eswa.2023.122743>, <https://www.sciencedirect.com/science/article/pii/S0957417423032451>
9. Ghosh, D.K., Jung, Y.J.: Depth cue fusion for event-based stereo depth estimation. *Information Fusion* **117**, 102891 (2025). <https://doi.org/https://doi.org/10.1016/j.inffus.2024.102891>, <https://www.sciencedirect.com/science/article/pii/S1566253524006699>
10. Han, H., Li, J., Wei, H., Ji, X.: Event-3dgs: Event-based 3d reconstruction using 3d gaussian splatting. In: *Advances in Neural Information Processing Systems*. vol. 37 (2024). <https://doi.org/10.52202/079017-4069>, https://proceedings.neurips.cc/paper_files/paper/2024/hash/e73ad1f690542144ce354637bb913c35-Abstract-Conference.html
11. Han, H., Lyu, J., Li, J., Wei, H., Li, C., Wei, Y., Chen, S., Ji, X.: Physical-based event camera simulator. In: *Computer Vision – ECCV 2024*. pp. 19–35 (2024). https://doi.org/10.1007/978-3-031-72995-9_2, https://www.ecva.net/papers/eccv_2024/papers_ECCV/papers/06110.pdf
12. Hidalgo-Carrió, J., Gallego, G., Scaramuzza, D.: Event-aided direct sparse odometry. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022). <https://doi.org/10.48550/arXiv.2204.07640>, https://openaccess.thecvf.com/content/CVPR2022/html/Hidalgo-Carrio_Event-Aided_Direct_Sparse_Odometry_CVPR_2022_paper.html
13. Hu, Y., Liu, S.C., Delbruck, T.: v2e: From video frames to realistic dvs events. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. pp. 1312–1319 (2021). <https://doi.org/10.1109/CVPRW53098.2021.00152>, https://openaccess.thecvf.com/content/CVPR2021W/EventVision/html/Hu_v2e_From_Video_Frames_to_Realistic_DVS_Events_CVPRW_2021_paper.html
14. Hwang, I., Kim, J., Kim, Y.M.: Ev-nerf: Event based neural radiance field. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. pp. 837–847 (Jan 2023)
15. Joubert, D., Marcireau, A., Ralph, N., Jolley, A., van Schaik, A., Cohen, G.: Event camera simulator improvements via characterized parameters **15**, 702765 (2021). <https://doi.org/10.3389/fnins.2021.702765>
16. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering (2023). <https://doi.org/10.1145/3592433>
17. Kim, J., Ghosh, D.K., Jung, Y.J.: Event-based video deblurring based on image and event feature fusion. *Expert Systems with Applications* **223**, 119917 (2023). <https://doi.org/https://doi.org/10.1016/j.eswa.2023.119917>, <https://www.sciencedirect.com/science/article/pii/S0957417423004189>
18. Klenk, S., Koestler, L., Scaramuzza, D., Cremers, D.: E-nerf: Neural radiance fields from a moving event camera. *IEEE Robotics and Automation Letters* **8**(3), 1587–1594 (2023). <https://doi.org/10.1109/LRA.2023.3240646>

19. Lichtsteiner, P., Posch, C., Delbruck, T.: A 128×128 120 db 15 μ s latency asynchronous temporal contrast vision sensor **43**(2), 566–576 (2008). <https://doi.org/10.1109/JSSC.2007.914337>
20. Lin, S., Ma, Y., Guo, Z., Wen, B.: Dvs-voltmeter: Stochastic process-based event simulator for dynamic vision sensors. In: Computer Vision – ECCV 2022. pp. 578–593 (2022). https://doi.org/10.1007/978-3-031-20071-7_34
21. Low, W.F., Lee, F.C., Lim, L.P., Khor, K.H., Ng, H.Z., Cheong, L.F.: Robust e-nerf: Nerf from sparse & noisy events under non-uniform motion. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (Oct 2023)
22. Ma, Y., Guo, S., Chen, Y., Xue, T., Gu, J.: Timelens-xl: Real-time event-based video frame interpolation with large motion. In: European Conference on Computer Vision (ECCV) (2024)
23. Mueggler, E., Gallego, G., Rebecq, H., Scaramuzza, D.: The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam (2017)
24. neuromorphicsystems: IEBCS: ICNS event based camera simulator (2023), <https://github.com/neuromorphicsystems/IEBCS>
25. Rebecq, H., Gehrig, D., Scaramuzza, D.: Esim: An open event camera simulator. In: Proceedings of the Conference on Robot Learning (CoRL). pp. 969–982 (2018), <https://proceedings.mlr.press/v87/rebecq18a.html>
26. Rebecq, H., Ranftl, R., Koltun, V., Scaramuzza, D.: High speed and high dynamic range video with an event camera (2019)
27. Rudnev, V., Elgharib, M., Theobalt, C., Golyanik, V.: Eventnerf: Neural radiance fields from a single colour event camera. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4992–5002 (Jun 2023)
28. Rudnev, V., Fox, G., Elgharib, M., Theobalt, C., Golyanik, V.: Dynamic eventnerf: Reconstructing general dynamic scenes from multi-view rgb and event streams. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. pp. 4905–4915 (Jun 2025)
29. Sun*, Z., Messikommer*, N., Gehrig, D., Scaramuzza, D.: Ess: Learning event-based semantic segmentation from still images. European Conference on Computer Vision. (ECCV) (2022)
30. Thirgood, C., Mendez, O., Ling, E., Storey, J., Hadfield, S.: HyperGS: Hyperspectral 3d gaussian splatting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5970–5979. <https://doi.org/10.1109/CVPR52734.2025.00560>, https://openaccess.thecvf.com/content/CVPR2025/html/Thirgood_HyperGS_Hyperspectral_3D_Gaussian_Splatting_CVPR_2025_paper.html
31. Thirgood, C., Mendez Maldonado, O., Ling, E.C., Storey, J., Hadfield, S.: FeatureSLAM: Feature-enriched 3d gaussian splatting SLAM in real time, <https://arxiv.org/abs/2601.05738>
32. Thirgood, C., Mendez Maldonado, O., Ling, E.C., Storey, J., Hadfield, S.: HYDRA: HYbrid knowledge distillation and spectral reconstruction algorithm for high-channel hyperspectral camera applications, <https://arxiv.org/abs/2510.16664>
33. Thirgood, C.T., Mendez Maldonado, O.A., Ling, C., Storey, J., Hadfield, S.J.: RaSpecLoc: RAmAn SPECTroscopy-dependent robot LOCalisation. In: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 5296–5303. IEEE. <https://doi.org/10.1109/IROS55552.2023.10342198>

34. Tulyakov, S., Boicchio, A., Gehrig, D., Georgoulis, S., Li, Y., Scaramuzza, D.: Time lens++: Event-based frame interpolation with parametric non-linear flow and multi-scale fusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2022), https://openaccess.thecvf.com/content/CVPR2022/html/Tulyakov_Time_Lens_Event-Based_Frame_Interpolation_With_Parametric_Non-Linear_Flow_and_CVPR_2022_paper.html
35. Tulyakov, S., Gehrig, D., Georgoulis, S., Erbach, J., Gehrig, M., Li, Y., Scaramuzza, D.: Time lens: Event-based video frame interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021), https://openaccess.thecvf.com/content/CVPR2021/html/Tulyakov_Time_Lens_Event-Based_Video_Frame_Interpolation_CVPR_2021_paper.html
36. Wang, S., Leroy, V., Cabon, Y., Chidlovskii, B., Revaud, J.: Dust3r: Geometric 3d vision made easy. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2024), <https://github.com/naver/dust3r>
37. Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., Wang, X.: 4d gaussian splatting for real-time dynamic scene rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 20310–20320 (2024). <https://doi.org/10.1109/CVPR52733.2024.01920>, https://openaccess.thecvf.com/content/CVPR2024/papers/Wu_4D_Gaussian_Splatting_for_Real-Time_Dynamic_Scene_Rendering_CVPR_2024_paper.pdf
38. Xiong, T., Wu, J., He, B., Fermüller, C., Aloimonos, Y., Huang, H., Metzler, C.A.: Event3dgs: Event-based 3d gaussian splatting for high-speed robot egomotion. In: Conference on Robot Learning, 6-9 November 2024, Munich, Germany. Proceedings of Machine Learning Research, vol. 270, pp. 4100–4118. PMLR (2024), <https://proceedings.mlr.press/v270/xiong25b.html>
39. Yura, T., Mirzaei, A., Gilitschenski, I.: Eventsplat: 3d gaussian splatting from moving event cameras for real-time rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 26876–26886 (Jun 2025)
40. Zahid, S., Rudnev, V., Ilg, E., Golyanik, V.: E-3dgs: Event-based novel view rendering of large-scale scenes using 3d gaussian splatting. In: International Conference on 3D Vision (3DV). pp. 926–934 (2025). <https://doi.org/10.1109/3DV66043.2025.00090>
41. Zhang, Z., Cui, S., Chai, K., Yu, H., Dasgupta, S., Mahbub, U., Rahman, T.: V2CE: Video to continuous events simulator. In: IEEE International Conference on Robotics and Automation (ICRA 2024), Yokohama, Japan, May 13–17, 2024. pp. 12455–12461. IEEE (2024). <https://doi.org/10.1109/ICRA57147.2024.10609864>, <https://doi.org/10.1109/ICRA57147.2024.10609864>
42. Zhu, J., Pan, T., Cao, Z., Liu, Y., Kwok, J.T., Xiong, H.: Depth any event stream: Enhancing event-based monocular depth estimation via dense-to-sparse distillation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 5146–5155 (October 2025)