







Physical-Based Event Camera Simulator

Haiqian Han¹ , Jiacheng Lyu¹, Jianing Li¹  , Henglu Wei¹ , Cheng Li²,
Yajing Wei², Shu Chen², and Xiangyang Ji¹  

¹ Tsinghua University, Haidian District, Beijing, 100084, P. R. China

² Beijing Xiaomi Mobile Software Co., Ltd.

A Summary of Event Camera Simulators

As shown in Table 1, we make a literature review of the existing event camera simulators. These simulators can be broadly classified into two categories (i.e., optimized-based and learning-based). Compared to learning-based methods, optimized-based methods have the advantage of generating 3D sparse events instead of 2D image-based representations. This approach brings them closer to the raw events generated directly from the event camera from the perspective of signal processing. Obviously, very few event camera emulators explore to interact with the input directly with the 3D scene, and most of them input three-channel video, without taking into account the rich spectral information of the real-world scenarios. In addition, none of these simulators take into account the impact of the camera lens when using videos as input. For physical-based camera simulation, we designed a realistic lens simulation block using the PBRT renderer.

Table 1: A literature review of classical event camera simulators. NL: Non-learning simulators, namely optimized-based simulators. L: Learning-based simulators.

Method	Input	Output	Ty.	Spect.	Lens	Description
PIX2NVS [2]	Frame	Events	NL	✗	✗	Using truncated chamfer loss as metrics.
ESIM [11]	3D scene	Events	NL	✗	✗	Interpolation-based simulation in UE.
V2E [4]	Frame	Events	NL	✗	✗	Using AI to interpolate the input video.
ICNS [5]	Frame	Events	NL	✗	✗	Better modeling of electronic circuits.
VOLT [7]	Frame	Events	NL	✗	✗	Modeling sensor using random process.
Mou <i>et al.</i> [8]	Frame	Events	NL	✓	✗	Integrating QE with 3-channel video.
Pantho <i>et al.</i> [10]	Frame	Event regions	L	✗	✗	Parallel processing with attention model.
EventGAN [15]	None	Event images	L	✗	✗	Generating event images using GAN.
V2CE [14]	Frame	Events	L	✗	✗	output continuous outputs using network.
Vista 2.0 [1]	Frame	Events	NL	✗	✗	Event-compatible multimodal simulation.
Tsuji <i>et al.</i> [12]	3D scene	Events	NL	✗	✓	Event-based denoising in ray tracing.
Ours	3D scene	Events	NL	✓	✓	Physics-based full-process simulation.

B Event Generation Mechanism

The event camera, a bio-inspired vision sensor, is engineered to detect changes in light intensity [3–5]. Unlike traditional image sensors that obtain the absolute level of light

^(✉) Corresponding author: Xiangyang Ji and Jianing Li.

intensity. The DVS pixel consists of a logarithmic photoreceptor that converts the input light intensity I to an output voltage V , it can be formulated as follows:

$$V = \log(I). \quad (\text{S1})$$

The logarithmic response of the photoreceptor enables event cameras to detect small changes in light intensity and reduces the impact of noise.

The output voltage V is then compared to a threshold voltage θ to determine if an event should be generated. It is described as follows:

$$|\Delta V| = |V_F(t + \Delta t) - V_c(t)| > \theta, \quad (\text{S2})$$

where $V_F(t + \Delta t)$ represents the output voltage in time $t + \Delta t$. $V_c(t)$ is the threshold voltage that be reset to $V_F(t + \Delta t)$ after event generated. Finally, an event will generated at $t = (t + \Delta t)$, and its polarity depends on the sign of ΔV .

Furthermore, the event camera incorporates a refractory period t_r , which is a period during which the pixel is not sensitive to changes in light intensity after generating an event. This refractory period prevents the event camera from generating multiple events in response to a single change of light intensity.

The aforementioned represents the fundamental principle, yet the practical implementation of the DVS model is more intricate. Notably, the photoreceptor introduces noise, rendering the relationship $V = \log(I)$ not strictly accurate. Moreover, the thresholds for On events θ_{on} and Off events θ_{off} typically differ, and these thresholds often exhibit fluctuations contingent on factors such as temperature and light intensity. In addition, inherent background noise in the DVS sensor can result in events being generated even in the absence of actual changes in light intensity.

C Ray Tracing

The core method for converting a modeled 3D scene into light intensities is ray tracing, and the ray tracing method can be mathematically represented rigorously and elegantly using path integral formulation [13]. To simplify the presentation, this work omits the details of volume rendering. However, the path integral formulation can be easily generalized to be compatible with volume rendering. The light intensity at a point on the sensor can be represented as:

$$I = \sum_{N=1}^{\infty} \int_{\Omega_N} f(\mathbf{x}) d\mu_N(\mathbf{x}), \quad (\text{S3})$$

where I is the intensity received by the sensor film. Ω_N is the path space with N units long path \mathbf{x} and can be defined by $N+1$ points. The first and last two points must be located on the film and light source respectively, while the middle point must be on the surface of the scene object. $f(\mathbf{x})$ can be understood as the contribution value of a specified path to intensity as:

$$f(\bar{x}) = \prod_{n=0}^N f_v(x_{n-1} \rightarrow x_n \rightarrow x_{n+1}) \times \left(\prod_{n=1}^N G(x_{n-1} \leftrightarrow x_n) \right), \quad (\text{S4})$$

where the geometric term $G(\cdot)$ is an analytic function related to mutual visibility, $f_v(\cdot)$ is determined by the optical properties between the points on the path, and its specific

definition is as follows:

$$f_v(x_{n-1} \rightarrow x_n \rightarrow x_{n+1}) = \begin{cases} f_s(x_{n-1} \rightarrow x_n \rightarrow x_{n+1}), & \text{if } 0 < n < N \\ L_e(x_0 \rightarrow x_1), & \text{if } n = 0 \\ W_e(x_{N-1} \rightarrow x_N), & \text{if } n = N \end{cases}, \quad (\text{S5})$$

where the $f_s(x_{n-1} \rightarrow x_n \rightarrow x_{n+1})$ is related to BSDF, and represents the energy ratio of a ray with a specified incident direction reflected in a specified exit direction. $L_e(x_0 \rightarrow x_1)$ is determined by the optical properties of the light source. W_e is determined by the nature of the film on the sensor, describing the different sensitivities of different points on the film to light rays. Finally, the rigorous description of the integrating factor in Eq. S3 can be written as follows:

$$d\mu_N(\bar{x}) = \prod_{n=0}^N dA(x_n), \quad (\text{S6})$$

where $dA(x_n)$ represents the infinitesimal area element at point x_n . We have completed the fundamental mathematical description of the standard path-tracing algorithm.

D Real to Sim: Scene Construction

To achieve high-fidelity results in the path tracing algorithm, it is essential to maintain precise geometric and optical information throughout the rendering process. This involves creating highly realistic 3D virtual scenes based on real-world scenarios, a practice commonly known as scene construction. It is noteworthy that scene construction constitutes a comprehensive and mature industry, supported by numerous professionals and sophisticated software designed for streamlined workflows. In this study, we will provide a concise introduction to scene construction tailored for small-scale research purposes. The process of scene construction can be divided into two main phases.

Alignment of Geometric Information. The primary goal of this procedure is to capture all the geometric details within a given scene. For instance, in documenting the translation of a chessboard within a darkroom, it is essential to initially record the relative positions of key elements in the scene, including the camera center, light sources, chessboard, darkroom, and other opaque objects. Subsequently, the geometry of each object must be recorded. A commonly employed technique is mesh representation, wherein the surface of any object is expressed as a composition of triangular primitives. Simple objects, such as a chessboard or a darkroom, can be described as combinations of cuboids. However, for more intricate objects, the mesh representation may require continuous refinement to better approximate their true shapes. Advanced methods, such as inverse rendering [9] for obtaining mesh representation from images or reconstructing object mesh representations from point clouds obtained [6] from radar, are available to facilitate this process.

In the PECS system with real lens simulation, camera calibration becomes more straightforward compared to using a pinhole camera model. Accurate geometric positioning of the camera eliminates the need for additional calibration, highlighting a key advantage in realistic camera lens simulation.

Restoration of Optical Information. The goal is to reconstruct light source details and object surface materials from geometric information. Reconstructing light source information is relatively straightforward, given that professional light sources usually

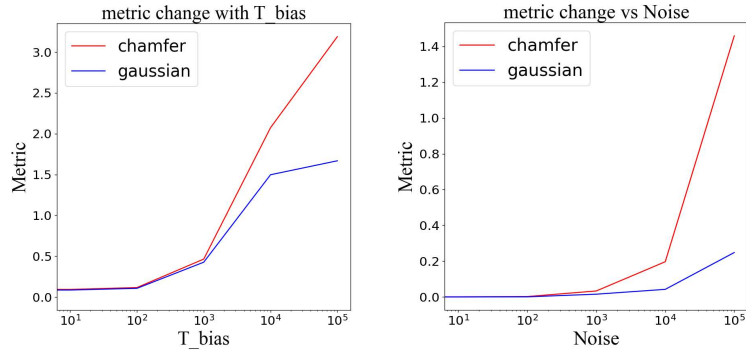


Fig. S1: The spatiotemporal event metrics vary with two typical distortion operations (i.e., adding bias and adding noise).

offer attributes like color temperature. Recording illuminance at key positions in the scene is often enough for accurate recreation during the modeling process.

Object surface materials are pivotal parameters determining information like the Bidirectional Scattering Distribution Function (BSDF) in path tracing. The prevalent method for material modeling is the Physically Based Rendering (PBR) system. PBR, a shading model in computer graphics, realistically simulates lighting and materials. Incorporating properties such as albedo, roughness, metallicness, and normal maps, PBR calculates how light interacts with surfaces. Through textures and equations, PBR materials replicate reflections, refractions, and scattering across various materials like metals, plastics, and fabrics. Widely embraced in graphics pipelines and game engines, PBR ensures high-quality, physically accurate visuals, offering artists a standardized workflow for crafting realistic materials in virtual environments. In essence, recording real-world material properties enables the optical information to be effectively reconstructed using the PBR model.

E Additional Results

Effective Test on Spatiotemporal Event Metrics. To verify the effectiveness of spatiotemporal event metrics, we conduct two additional experiments in Fig. S1. The initial experiment is the T_bias operation, encompassing the following steps: Initially, a segment of real event data P is selected. Subsequently, distortion is applied to the timestamps of this data (stretching the t-axis and capturing the same duration as the origin), resulting in the distorted data P' . Following this, metrics are employed to calculate the differences between P and P' . A larger bias (bias greater than 1) indicates a greater disparity between P and P' . Similarly, the noise operation involves the addition of noisy points to the real event data P to obtain P' . The greater the quantity of noisy points added, the more noticeable the distinction between P' and P .

The experimental results highlight the robust evaluation capabilities of both metrics. When applied to entirely identical event data, both two metrics yield a result of 0, indicating no discernible difference between the two datasets. With an increase in noise or distortion, the measurement results exhibit a clear positive correlation. Notably, a key difference emerges: the Gaussian distance shows lower sensitivity to increases in

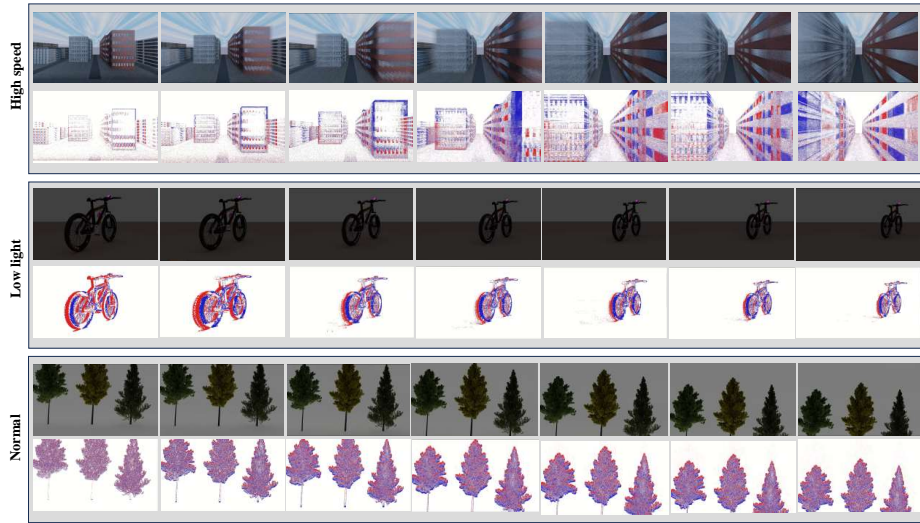


Fig. S2: Representative visualization results using our PECS on three scenarios. The first row refers to high-speed motion blur scenarios, the second row is low light scenarios, and the last row represents normal scenarios.

noise. In instances involving both distortion and noise, when the chamfer distance reaches 1, the Gaussian distance in the distortion case exceeds 0.5, while in the noise case, it remains below 0.2.

Visualization Results on Simulation Data. To further demonstrate the versatility of our PECS, we conduct additional experiments in a variety of complex scenarios. In particular, our PECS has the capability to accept any 3D scene as input, and given the extensive availability of high-fidelity 3D scenes, it possesses the capacity to generate large-scale datasets. Fig. S2 shows the simulation capabilities of our PECS in three scenes. The first row is a high-speed motion blur scenario, simulating the event output of a camera moving quickly through a city. The second row is a low-light scene, simulating the event output of a camera shooting a bicycle in extremely low light. The third row is normal light and speed, simulating the event output of a camera shooting trees with complex details. It can be seen that our PECS exhibits strong versatility by generating high-precision event data across diverse scenes.

References

1. Amini, A., Wang, T.H., Gilitschenski, I., Schwarting, W., Liu, Z., Han, S., Karaman, S., Rus, D.: Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles. In: ICRA. pp. 2419–2426 (2022)
2. Bi, Y., Andreopoulos, Y.: Pix2nvs: Parameterized conversion of pixel-domain video frames to neuromorphic vision streams. In: ICIP. pp. 1990–1994 (2017)
3. Graça, R., McReynolds, B., Delbruck, T.: Shining light on the dvs pixel: A tutorial and discussion about biasing and optimization. In: CVPR. pp. 4044–4052 (2023)
4. Hu, Y., Liu, S.C., Delbruck, T.: v2e: From video frames to realistic dvs events. In: CVPRW. pp. 1312–1321 (2021)

5. Joubert, D., Marcireau, A., Ralph, N., Jolley, A., van Schaik, A., Cohen, G.: Event camera simulator improvements via characterized parameters. *Frontiers in Neuroscience* **15**, 702765 (2021)
6. Ladicky, L., Saurer, O., Jeong, S., Maninchedda, F., Pollefeys, M.: From point clouds to mesh using regression. In: *Proceedings of the IEEE International Conference on Computer Vision*. pp. 3893–3902 (2017)
7. Lin, S., Ma, Y., Guo, Z., Wen, B.: Dvs-voltmeter: Stochastic process-based event simulator for dynamic vision sensors. In: *ECCV*. pp. 578–593 (2022)
8. Mou, X., Feng, K., Yi, A., Wang, S., Chen, H., Hu, X., Guo, M., Chen, S., Suess, A.: Accurate event simulation using high-speed videos. *Electronic Imaging* **34**(7), 242–1 (2022)
9. Nicolet, B., Jacobson, A., Jakob, W.: Large steps in inverse rendering of geometry. *ACM Transactions on Graphics (TOG)* **40**(6), 1–13 (2021)
10. Pantho, M.J.H., Mbongue, J.M., Bhowmik, P., Bobda, C.: Event camera simulator design for modeling attention-based inference architectures. *Journal of Real-Time Image Processing* **19**(2), 363–374 (2022)
11. Rebecq, H., Gehrig, D., Scaramuzza, D.: Esim: An open event camera simulator. In: *CoRL*. pp. 969–982 (2018)
12. Tsuji, Y., Yatagawa, T., Kubo, H., Morishima, S.: Event-based camera simulation using monte carlo path tracing with adaptive denoising. *arXiv* (2023)
13. Zhang, C.: Path-space differentiable rendering. University of California, Irvine (2022)
14. Zhang, Z., Cui, S., Chai, K., Yu, H., Dasgupta, S., Mahbub, U., Rahman, T.: V2ce: Video to continuous events simulator. *arXiv* (2023)
15. Zhu, A.Z., Wang, Z., Khant, K., Daniilidis, K.: Eventgan: Leveraging large scale image datasets for event cameras. In: *ICCP*. pp. 1–11 (2021)