Physical-Based Event Camera Simulator

Haiqian Han[1], Jiacheng Lyu[1], Jianing Li[1] (✉), Henglu Wei[1], Cheng Li[2], Yajing Wei[2], Shu Chen[2], and Xiangyang Ji[1] (✉)

[1] Tsinghua University, Haidian District, Beijing, 100084, P. R. China
[2] Beijing Xiaomi Mobile Software Co., Ltd.

Abstract. Existing event camera simulators primarily focus on the process of generating video events and often overlook the entire optical path in real-world camera systems. To address this limitation, we propose a novel Physical-based Event Camera Simulator (PECS), which is able to generate a high-fidelity realistic event stream by directly interfacing with the 3D scene. Our PECS features a lens simulation block for accurate light-to-sensor chip replication and a multispectral rendering module for precise photocurrent generation. We present two spatiotemporal event metrics to assess the similarity between simulated and actual camera events. Experimental results demonstrate that our PECS outperforms four state-of-the-art simulators by a large margin in terms of event-based signal fidelity. We integrate our PECS into the UE platform to generate extensive multi-task synthetic datasets and evaluate its effectiveness in downstream vision tasks (e.g., video reconstruction). Our open-source code is available at `https://github.com/lanpokn/PECS_trail_version`.

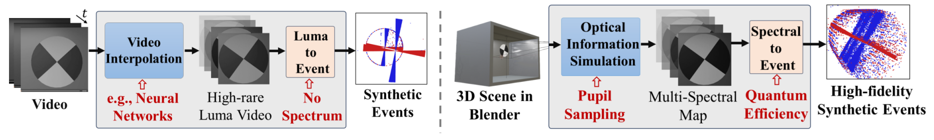
Keywords: Event camera simulation · Physics-based vision · Hyperspectral data analysis

1 Introduction

Event cameras [7,43], namely bio-inspired vision sensors, operate fundamentally differently from conventional cameras. Instead of capturing intensity images at a fixed rate, event cameras respond to brightness changes with a stream of asynchronous events. With the advantages of high temporal resolution, high dynamic range, and low power consumption [33], event cameras have found widespread use in various computer vision tasks [4,13,17,18,21,22,25,29,39,40,49].

Despite notable advancements in event-based vision, the training of deep learning-based approaches still demands a substantial amount of synthetic event data [19,53]. It is worth noting that the high cost [43] and deployment challenges [6] associated with event cameras in high-speed or low-light scenes limit the availability of real-world datasets. Thus, several event camera simulators have attempted to generate a large amount of affordable and reliable event data.

(✉) Corresponding authour: Xiangyang Ji and Jianing Li.



(a) Existing video to events simulators (b) Physical-based event camera simulator

Fig. 1: Event camera simulators. (a) The existing video to events simulators (e.g., V2E [12]). (b) Our Physical Event Camera Simulator (PECS) is designed to generate high-fidelity realistic events by directly interfacing with the 3D scene.

One category refers to optimized-based event camera simulators [5, 12, 14, 15, 20, 31, 38, 41, 57], where the objective is to design hand-crafted modules for generating event data. For example, V2E [12] converts intensity frames into event data via multiple hand-crafted modules. Note that, these simulators take three-channel videos as direct inputs, overlooking the complete optical path in real-world camera systems. It may lead to the loss of optical information, consequently seriously affecting the simulator's accuracy. While some efforts [14, 44] support offline rendering in the 3D scene, they often neglect the consideration of spectrum data in the rendering module. In fact, integrating spectrum information into the event camera simulation process has the potential to enhance the fidelity of synthetic event streams.

Another category is learning-based event camera simulators [2, 8, 10, 28, 34, 52, 54] that aim to generate event representations, improving the generalization capabilities of deep learning models directly for the target domain rather than relying on raw events. For instance, EventGAN [54] is an end-to-end neural network that directly converts images to event presentations for downstream computer vision tasks. However, these learning-based simulators, requiring retraining for diverse usage cases, may have limited generalization across different scenarios. Moreover, the generated event representations are not in the original signal domain [1], posing potential limitations for broader applications. In other words, physical-based simulators, producing spatiotemporal events, preserve the inherent characteristics of raw camera data, enhancing their versatility compared to event representations by learning-based simulators.

To address the aforementioned problems, this paper proposes a novel Physical-based Event Camera Simulator (PECS), which is able to generate a highly realistic event stream by directly interfacing with the 3D scene (see Fig. 1). In fact, the goal of this work is not to optimize hand-crafted event camera simulators (e.g., ESIM or V2E). In contrast, we aim at overcoming the following challenges: (i) *How do we model a realistic lens to replicate the light-to-sensor chip process?* (ii) *How do we design a multispectral renderer to obtain high-precision photocurrents rather than three-channel videos?* Towards this end, a realistic lens simulation block using the PBRT renderer [30] is developed to accurately replicate the light-to-sensor chip process. Then, a novel multispectral rendering module is designed to generate high-precision photocurrents by quantum efficiency and Monte Carlo integration. Two asynchronous spatiotemporal event metrics (i.e., Chamfer distance and Gaussian distance) are proposed to measure the similarity between simulated and raw events. Extensive experiments show that our PECS outperforms four state-of-the-art simulators by a large margin in terms of event-based signal fidelity. Moreover, we further deploy real-time versions into the Unreal Engine (UE) platform to generate extensive multi-task synthetic datasets. Typical computer vision tasks (i.e., event-based video reconstruction) are conducted to verify the effectiveness of our PECS.

Overall, the main contributions of this work are summarized as follows:

- We propose a *novel Physical-based Event Camera Simulator* (PECS), capable of generating a high-fidelity realistic event stream by directly interfacing with the 3D scene, without relying on video-to-event conversion.
- We design a *realistic lens simulation block* using the PBRT renderer, optimizing pupil sampling to improve the quality and speed of rendering.
- We design a *novel multispectral rendering module* that generates high-precision photocurrents through quantum efficiency and Monte Carlo integration.
- We present two asynchronous spatiotemporal metrics to assess the similarity between simulated and raw events. Extensive experiments show that our PECS outperforms four state-of-the-art simulators in terms of signal fidelity.

To the best of our knowledge, this is the first attempt to explore a physical-based event camera simulator. We believe that our simulator will provide high-fidelity realistic large-scale event data for event-based vision tasks and offer proof of principle for the next generation of neuromorphic cameras.

2 Related Work

This section will review existing event camera simulators from two perspectives including optimized-based simulators and learning-based simulators.

Optimized-based Event Camera Simulators. The prior optimized-based simulators include PIX2NVS [5] and ESIM [8, 31], which convert any image or video into asynchronous events. A realistic V2E [12] is developed to model the event camera in low-light or motion blur scenarios. ICNS [14] is presented to simulate noise and estimate the latency by adding the effects of the arbiter. VOLT [20] employs the concept of stochastic processes to model the output process of event data, thereby endowing the simulated event data with good continuity and probabilistic interpretability. Nevertheless, these event camera simulators directly consider three-channel videos as the input and overlook the complete optical path in real-world camera systems. Although Mou *et al.* [24] first notice the impact of quantum efficiency on event camera simulators and implement video-to-spectrum conversion. However, the lossless conversion from three-channel data to spectral data is nearly impossible. Thus, we design a novel multispectral rendering module that generates high-precision photocurrents through quantum efficiency and Monte Carlo integration.

Learning-based Event Camera Simulators. With the rapid development of deep learning, some learning-based event camera simulators [2,8,10,28,34,52,54]

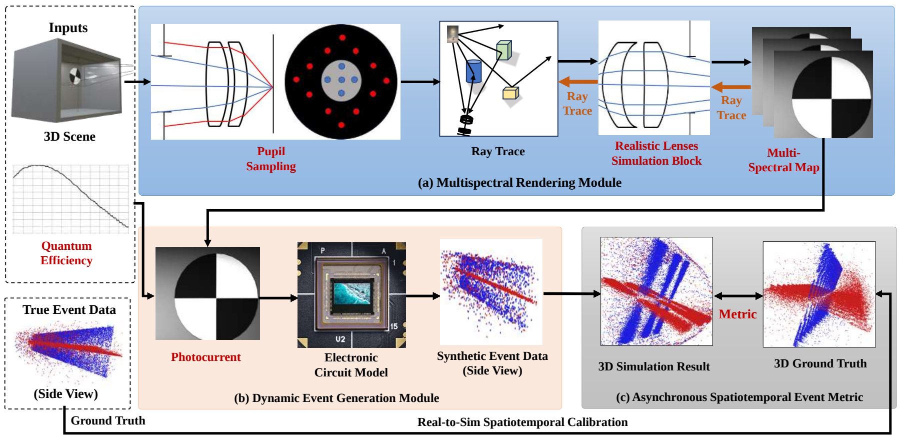

Inputs
3D Scene
Quantum Efficiency
True Event Data
(Side View)
Pupil Sampling
Ray Trace
Ray Trace
Realistic Lenses Simulation Block
Ray Trace
Multi-Spectral Map
(a) Multispectral Rendering Module
Photocurrent
Electronic Circuit Model
Synthetic Event Data (Side View)
(b) Dynamic Event Generation Module
Metric
3D Simulation Result
3D Ground Truth
(c) Asynchronous Spatiotemporal Event Metric
Ground Truth
Real-to-Sim Spatiotemporal Calibration

Fig. 2: The pipeline of the proposed Physical-based Event Camera Simulator (PECS). It starts with pupil sampling, precalculating the rays expected to pass through the lens. Ray tracing calculates light refraction and reflection within the lens and scene, producing multispectral maps. Incorporating quantum effects, these maps generate accurate photocurrents. Our PECS then leverages logarithmic differences to generate events, enabling the measurement of similarity between simulated and raw camera events in signal fidelity.

are developed to generate event representations in the target domain. For instance, Zhu *et al.* [54] use the modified GAN model to generate synchronous event representations without raw events. Gu *et al.* [10] propose a domain-adaptive event simulator that can generate both event data and the camera's trajectory. Pantho *et al.* [28] use a neural network to generate event data represented by regions, which is convenient for use in moving region detection. While these simulators can enhance overall accuracy to a certain degree, the intrinsic challenge lies in the limited interpretability of neural networks. Consequently, these simulators, requiring re-training for diverse usage cases, may have limited generalization across different scenarios. In other words, simulators that rely on learning-based approaches are expected to exhibit lower robustness compared with optimized-based event camera simulators.

3 Methods

3.1 Framework Overview

This work aims at designing a novel Physical-based Event Camera Simulator, termed PECS, which is able to generate a highly realistic event stream by directly interfacing with the 3D scene. As illustrated in Fig. 2, our PECS mainly consists of two modules: a multispectral rendering module and a dynamic event generation module. In the multispectral rendering module, we employ pupil sampling as a pre-processing step to obtain a set of rays originating from the direction in which the lens may emit. Then, ray tracing is employed to compute the refraction and reflection of light within both the lens and the scene, allowing us to acquire multispectral maps. Meanwhile, multispectral maps with quantum effects generate high-precision photocurrent. Particularly, we design a realistic lens simulation block using the PBRT renderer, which aims at optimizing pupil sampling to improve the quality and speed of rendering. In the dynamic event generation module, we leverage the principle of logarithmic differences in the event camera to generate events. This module tackles the constraints of current simulation systems by incorporating spectrum data and accounting for quantum efficiency. This enhancement contributes to improved accuracy in estimating photocurrent, thereby creating a more realistic depiction of the physical processes involved in the conversion of light signals into photocurrent. Finally, two event metrics (i.e., Chamfer distance and Gaussian distance) are proposed to measure the distance between simulated and raw events in signal fidelity.

3.2 Realistic Camera Lenses Simulation

Realistic camera lens simulation in computer graphics replicates actual camera lens behavior [16] by employing ray tracing to accurately simulate light paths through multiple lens elements. This approach takes into account the intricacies of the lens system, including refraction at interfaces, and improves image quality by estimating incident radiance along arbitrary rays.

Our realistic camera lens simulation block models a lens system with various elements, tracing rays through them with multispectral information. It considers interactions until rays either exit the optical system or are absorbed by the aperture stop or lens housing. To reduce computational waste, our PECS performs pupil sampling before conducting the actual simulation. In this study, the pupil's definition is a little different from in optics [36]. It refers to all the directions from which rays emit from the lenses at any specific point on the sensor film, not just the center point. Let the sensor plane coordinates be denoted as (u, v). The pupil is a set of feasible directions represented by points (x, y) on the direction plane. Each point (x, y) in the pupil corresponds to a unique direction. Thus, the mathematical definition of the pupil can be expressed as:

$$Pupil(u, v) = \{(x, y) \mid Ray(\{u, v\}, \{x, y\})), \tag{1}$$

where $Ray(\{u, v\}, \{x, y\})$ denotes the ray originating from (u, v) and pointing towards (x, y) passing through the lenses.

Pupil sampling is the algorithm that identifies the correct pupil for each point on the sensor film. In the case of lenses commonly used in event cameras (e.g., Prophesee Gen4), which are often centrally symmetric with circular apertures, representing the pupil shape as an ellipse simplifies the pupil sampling process. Therefore, the pupil can be approximated as follows:

$$Pupil(u, v) = \{(x, y) \mid (x, y) \in Elli(p, a, b)\}, \tag{2}$$

where $Elli(p, a, b)$ is an ellipse centered at point p on the direction plane with semi-major axis length a and semi-minor axis length b.

By uniformly sampling points on the beam direction plane, the exit pupil for each sampled point on the sensor film is efficiently computed. For other points, the exit pupil is determined through linear interpolation between the ellipse's center and its major and minor axes. Despite potential errors in the elliptical model for complex lenses, this method still notably decreases the time of ray tracing within the lens, thereby speeding up the lens system simulation.

3.3 Multispectral Rendering Module

The multispectral rendering module transforms incoming light signals into photocurrent, adjusting for display convenience [12, 14, 31]. An event camera involves detecting photons at various frequencies, initiating photocurrent through the photoelectric effect. After filtering and logarithmic conversion, the currents are converted into photovoltages. The module concentrates on the photon-to-photocurrent conversion, excluding subsequent electronic circuit model steps.

Most simulation systems estimate photocurrent using the L component of LUV values, but actual photon receivers don't convert different frequency lights into the same number of electrons, as revealed by the Quantum Efficiency (QE) curve. The definition of absolute quantum efficiency [35] can be described as:

$$\mathrm{QE}_{absolute}(\lambda) = \frac{N_e}{N_v} = \frac{R_\lambda}{\lambda} \times \frac{hc}{e}, \tag{3}$$

where N_e is the number of electrons, N_v is the number of photons, R_λ is the current per unit of incident light power (A/W), λ is the wavelength in nm, h is the Planck constant, c is the speed of light in vacuum, and e is the elementary charge. The quantum efficiency is generally given as a relative value ranging from 0 to 1. Its definition can be formulated as follows:

$$\mathrm{QE}_{relative}(\lambda) = \frac{QE_{\text{absolute}}(\lambda)}{\max(QE_{\text{absolute}}(\lambda))}. \tag{4}$$

Our PECS overcomes limitations by providing multi-spectral data for precise photocurrent estimation. Leveraging known spectrum information and quantum efficiency, PECS directly computes photocurrent as follows:

$$i_p = C \cdot \int \lambda \cdot QE(\lambda) \cdot L(\lambda) \cdot d\lambda, \tag{5}$$

where i_p is the photocurrent after Photoelectric conversion, which is a key parameter in subsequent electronic circuits. C is a constant that depends on the pixel area and units of physical quantities, and $L(\lambda)$ is the input spectrum. If L is represented by a finite set of discrete channels, the following simple numerical integration method can be used to achieve the aforementioned integration:

$$i_p \approx C \cdot \sum_{k=1}^{N} \lambda_k \cdot QE(\lambda_k) \cdot L(\lambda_k) \cdot (\lambda_{k+1} - \lambda_k), \tag{6}$$

where N is the number of known channels of the spectrum. If $N = 3$, the information carried by the spectrum degrades into a common RGB image.

The spectrum representation can be more complex in certain applications, such as using neural networks for implicit continuous representations. Simple data integration may lead to insufficient sampling in regions with larger function values, causing significant errors. To address this, Monte Carlo integration and importance sampling are commonly employed as:

$$i_p = \int_a^b f(x)dx = \lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}, \tag{7}$$

where N is the number of samples. $p(x)$ is probability density function, which generated samples x_i. Applying Eq. 7 to Eq. 5, it can be re-written as:

$$i_p \approx I_N = C \cdot \frac{1}{N}\sum_{k=1}^{N} \frac{\lambda_k \cdot QE(\lambda_k) \cdot L(\lambda_k)}{p(\lambda_k)}, \tag{8}$$

where I_N is an approximation. Choosing a well-behaved $p(\lambda)$ is crucial to minimize N while maintaining accuracy. In probability theory, it is equal to make the variance small. Assuming that different samples are independent and identically distributed, we can obtain the variance of I_N as:

$$\mathrm{Var}(I_N) = \frac{1}{N^2}\sum_{k=1}^{N} \mathrm{Var}\left(\frac{\lambda_k \cdot QE(\lambda_k) \cdot L(\lambda_k)}{p(\lambda_k)}\right) = \frac{\sigma_N^2}{N^2}, \tag{9}$$

In the optimal condition, $\sigma_N^2 = 0$, which means $p(\lambda)$ is totally equal to the original function except for scale. Thus, we can model $p(\lambda)$ as a piecewise constant function and estimate it by pre-calculating the original function as:

$$p(\lambda) = B \cdot \lambda'_j \cdot QE(\lambda'_j) \cdot L(\lambda'_j), j = \mathrm{argmin}_m |\lambda - \lambda'_m|, \tag{10}$$

where m is the channel number. For each m, $L(\lambda'_m)$ is pre-calculated through the implicit representation of the spectrum. B is a constant that ensures the integral of $p(\lambda)$ equals 1. Then, the photocurrent can be obtained as follows:

$$i_p \approx C \cdot \frac{1}{N}\sum_{k=1}^{N} \frac{\lambda_k \cdot QE(\lambda_k) \cdot L(\lambda_k)}{B \cdot \lambda'_{j_k} \cdot QE(\lambda'_{j_k}) \cdot L(\lambda'_{j_k})}, j_k = \mathrm{argmin}_m |\lambda_k - \lambda'_m|. \tag{11}$$

Note that, Monte Carlo integration in Eq. 11 and simple numerical integration in Eq. 6 each have their own advantages and disadvantages. When the spectral data is represented continuously, Monte Carlo integration produces significantly smaller errors with a sharp spectral distribution curve compared to simple integration. However, if the spectrum is only represented by discrete channels, the lack of original information renders Monte Carlo integration somewhat redundant, and simple numerical integration can be used instead.

Note that, our PECS estimates photocurrent by considering spectral data and quantum efficiency, offering more accurate simulations of light signal conversion. Consequently, the accuracy is significantly enhanced compared to the LUV method. Meanwhile, the quantum efficiency of different cameras often varies and can generally be provided by the corresponding sensor manufacturers.

3.4 Asyncrhonous Spatiotemporal Event Metric

To measure the similarity between simulated and raw camera events, we present two asynchronous spatiotemporal event metrics (i.e., Chamfer distance and Gaussian distance), which is similar to the metric part in iterative closest point tasks [3, 37]. The Chamfer distance metric C_D can be described as:

$$C_D(R,Q) = \frac{1}{|R|}\sum_{r\in R}\min_{q\in Q}\ d(r,q) + \frac{1}{|Q|}\sum_{q\in Q}\min_{r\in R}\ d(r,q), \tag{12}$$

where the lower C_D its value, the closer the two events are. R and Q are two point clouds, r or q is a single point in R or Q. Each point can be expressed as a four-dimensional tuple $\{x, y, p, t\}$ representing the spatial coordinates, polarity, and temporal information. The distance in the metric can be depicted as:

$$d(r,q) = ||r - q||_2, \tag{13}$$

By utilizing KDTREE [9] for acceleration, the time complexity is only $O(n\log(n))$.

Similarly, the Gaussian distance can be formulated as:

$$G_D(R,Q) = \frac{1}{|R|}\sum_{r\in R} g(\min_{q\in Q}\ d(r,q)) + \frac{1}{|Q|}\sum_{q\in Q} g(\min_{r\in R}\ d(r,q)), \tag{14}$$

where the function g makes $G_D(R,Q)$ still become lower when the two events get closer while reducing the impact of outliers, and it can be described as:

$$g(x) = 1 - \exp\left(-\frac{||x||_2^2}{\sigma}\right), \tag{15}$$

where σ is a hyperparameter used to adjust the sensitivity range of the original metric. In this study, we set 0.4 as the default.

In fact, these two metrics describe the attributes of asynchronous events from different dimensions and can be necessary to assess the similarity between raw and simulated data. The Gaussian distance exhibits relative stability against outliers compared to the Chamfer distance, making it more robust to noise. Nevertheless, its measurement results have a narrower range of variation, making it less distinctive in simultaneous evaluations of multiple simulators.

Note that, the units of x, y, p, and t in the original data are different, and the range of values for t often far exceeds that of the other components, resulting in Eq. 13 and Eq. 14 almost do not consider information beyond t. To address this issue, it is necessary to normalize the original measurements first to resolve this scaling problem as follows:

$$L_\beta[i] = (\frac{L'_\beta[i] - \min L'_\beta}{\max L'_\beta - \min L'_\beta + \epsilon}) * \alpha, \tag{16}$$

where L'_β is the original measurements, β can be x, y, p and t element. L_β is the corresponding processed data. i is the i'th event of the raw data, which means $R[i] = \{L_x[i], L_y[i], L_p[i], L_t[i]\}$. ϵ is a small number that is to avoid dividing zero, and α is a hyperparameter. In this work, we choose $\alpha = 100$ for space x, y and polarity p, and choose $\alpha = 1000$ for time t.

4 Experiments

4.1 Experimental Setting

Real-to-Sim Scenario Construction. Spatiotemporal synchronization is essential for quantitatively evaluating, we record the real-world event data using a Prophesee Gen4 camera in a professional optical laboratory. We capture six sequences including rotating disk and translating checkerboards with light changes and various motion speeds. For example, T_0.6_H means translating checkerboards, 0.6 meter-per-seconds, and high light. R_360_L means rotating disk, 360 rpm, and low light. To ensure consistency between the simulated and real-world scenes, we utilize the blender for high-fidelity 3D modeling. We export the scenes to PBRT and simulate a naturally calibrated camera using real lenses [26,42]. This approach enables us to acquire temporally and spatially aligned rendering results for use in subsequent sensor modules. More details of spatiotemporal alignment can be found in the supplementary material. In other methods, as regular RGB cameras struggle to capture clear images in these scenes, we use Blender's rendered RGB video as input. By manually calibrating pinhole camera parameters, we ensure temporally and spatially consistent clear images for simulation inputs, enabling the proper functioning of these simulators.
Implementation Details. All experiments are conducted on a GeForce RTX 3050 Ti and 11th Gen Intel(R) Core(TM) i7-11800H CPU. We compare our PECS with four open-source event camera simulators (i.e., ESIM [31], VOLT [20], V2E [12], and ICNS [14]). In our PECS, the hyperparameters in Eq. 11 and Eq. 6 are set to $C = 100$ and $N = 31$ to make an accuracy-speed trade-off. Regarding the ESIM method, we extract its core code from its original ties to Unreal Engine (UE) and replace UE output with video output. For the V2E method, given the clarity and compactness of Blender-generated images, we disable the network interpolation module to prevent extremely slow simulation speed. In the ICNS method, all parameters have been adjusted to achieve optimal results. In the VOLT method, we select a set of parameters for the best performance. We present two event metrics (i.e., Chamfer distance and Gaussian distance) to quantitatively evaluate the similarity between raw events and synthetic data. Besides, we select three representative sequences of the HS-ERGB dataset [45] in the event-based video reconstruction task.

Table 1: Comparison with state-of-the-art simulators on our six recording sequences. Note that, our Physical Event Camera Simulator (PECS) shows superior performance on both metrics across a diverse range of movement speeds and lighting conditions.

Scene	Gaussian distance ↓					Chamfer distance ↓				
	ESIM	VOLT	V2E	ICNS	PECS	ESIM	VOLT	V2E	ICNS	PECS
Rotate 360 High	1.995	1.923	1.671	1.558	0.960	45.512	22.086	6.356	3.816	2.010
Rotate 360 Low	1.965	1.931	1.831	1.681	1.631	35.082	25.391	7.729	4.131	3.708
Rotate 60 High	1.995	1.911	1.816	1.574	1.504	39.425	20.814	9.297	4.237	3.459
Trans 1mps High	1.963	1.655	1.827	1.806	1.667	62.206	10.573	15.450	9.453	3.975
Trans 06mps High	1.967	1.723	1.839	1.837	1.798	63.939	10.198	18.154	9.588	4.169
Trans 1mps Low	1.987	1.814	1.700	1.663	1.710	68.115	10.952	14.071	9.182	4.467
Average	1.979	1.826	1.781	1.687	**1.545**	52.380	16.669	11.843	6.735	**3.631**

4.2 Effective Test

Quantitative Evaluation. To quantitatively evaluate the effectiveness of our PECS, we conduct a comparison with four open-source event camera simulators (i.e., ESIM [31], VOLT [20], V2E [12], and ICNS [14]) on six recording sequences (see Table 1). Note that, our PECS outperforms four state-of-the-art simulators in both two event metrics. More precisely, in terms of the Gaussian distance, our PECS reduces the average by 0.434, 0.281, 0.236, and 0.142 when compared to ESIM, VOLT, V2E, and ICNS, respectively. Meanwhile, in the Chamfer distance, our simulator exhibits an average decrease of 48.749, 13.038, 8.212, and 3.104 compared to ESIM, VOLT, V2E, and ICNS, respectively.
Visualization Evaluation. Some representative visualization results on two motion scenarios (i.e., translating checkerboards and rotation disks) recorded in a professional optical laboratory are illustrated in Fig. 3 and Fig. 4. Obviously, our PECS achieves the best performance against four state-of-the-art simulators including ESIM, VOLT, V2E, and ICNS. We can find that V2E and ESIM simulators are too primitive, resulting in concentrated data distribution in 3D space and a lack of continuity. The event data of ESIM is entirely concentrated at the timestamp of the input frame, lacking continuity, which is the main reason for its poor performance. VOLT performs well in terms of the continuity of event output, but its noise model suffers from significant distortion. Additionally, VOLT simply converts three-channel video to grayscale as photocurrent, without considering quantum efficiency and spectrum. Although the ICNS has an advanced sensor model and supports scene input, the lack of algorithms in the section leads to less accurate event data generation.

4.3 Ablation Study

Influence of the Sampling Number N**.** To analyze the sampling number N of numerical integration in our PECS, we set the multispectral rendering module with various values of N, and set C as 100. As described in Table 2, we find that the Chamfer distance and Caussian distance decrease with the increase of N. More precisely, Comparing N=31 and N=6, the Chamfer distance and Gaussian distance decrease by 6.58 and 0.619, respectively. Overall, this results show that

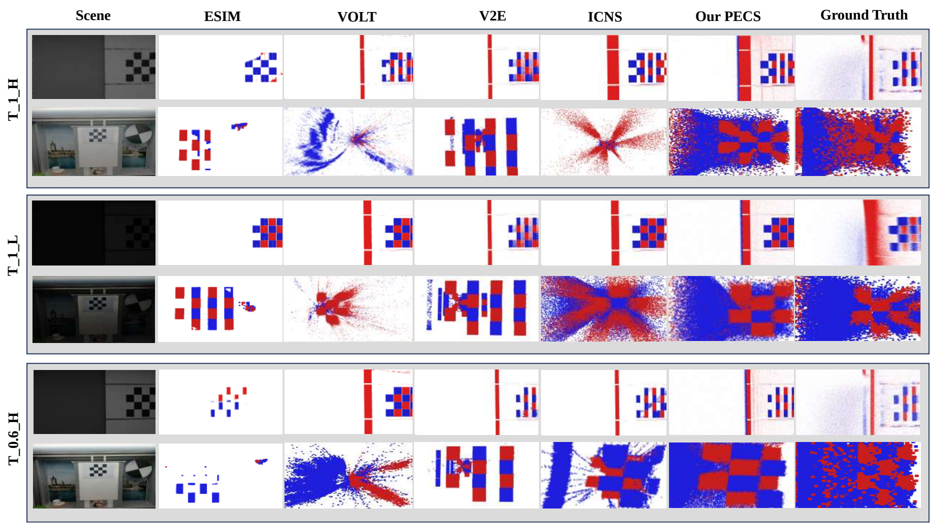


Fig. 3: Representative examples of **translating checkerboards** with various motion speed and light conditions. In each scenario, the first row is a 2D visualization schematic of the event data, and the second row is a 3D visualization schematic of the event data.

Table 2: The influence of the sampling number N of Monte-Carlo integration in the multispectral rendering module. It's worth noting that our PECS exhibits improved performance with an increase in N.

N	1	3	4	6	7	11	16	31
Chamfer distance $\downarrow$	13.96	8.826	16.28	8.588	8.466	2.869	2.476	**2.008**
Gaussian distance $\downarrow$	1.597	1.624	1.569	1.583	1.582	1.176	1.091	**0.964**

the larger the N, the closer the final simulation data is to the real data. Of course, an increase in N also entails a rise in time complexity. Thus, there is often a consideration of the trade-off between accuracy and computational speed.

Influence of the Hyperparameter C. To analyze the effect of the hyperparameter C of numerical integration on the final performance, we set the multispectral rendering module with various values of C, and set N as 31. From Table 3, we can find that there is no simple correlation between the hyperparameter C and the final performance. The distances of the two metrics gradually decrease with an increase in C until they start to increase at nearly 100. Essentially speaking, the hyperparameter C is introduced to align the standard unit system with the unit system used in the actual code. Theoretical analysis indicates the existence of an optimal value, a notion supported by ablation experiment results. The optimal value is observed to be around 100, and the distances of the two metrics decrease as the value of C deviates from this optimal point.

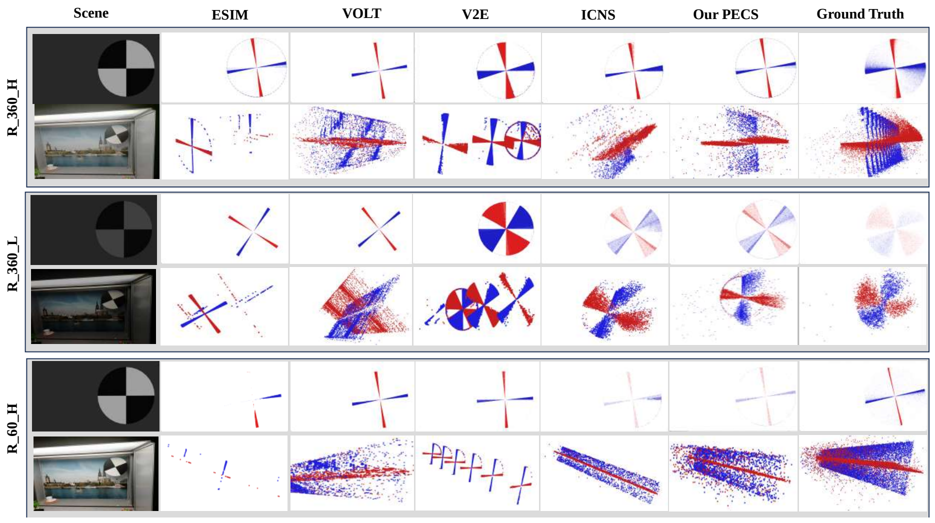


Fig. 4: Representative visualization results of **rotation disks** with various motion speed and light conditions. In each scenario, the first row is a 2D visualization schematic of the event data, and the second row is a 3D visualization schematic of the event data.

Table 3: The influence of the hyperparameter C of Monte-Carlo integration in the multispectral rendering module. The distances gradually decrease with an increase in C until they start to increase at nearly 100.

C	10	30	60	80	100	120	140	160
Chamfer distance ↓	6.493	7.445	2.508	5.066	**2.008**	2.011	2.264	2.396
Gaussian distance ↓	1.342	1.488	1.106	1.408	**0.964**	0.986	1.050	1.076

5 Downstream Applications

In this section, we integrate our PECS with the Unreal Engine (UE) platform to generate extensive event-based vision datasets that support multiple tasks. We then proceed to validate the simulator's performance on the event-based video reconstruction task. In contrast to existing simulators, the representative algorithm trained on our synthetic events performs favorably on real event data.
Extensive Multi-task Synthetic Datasets. To seamlessly link our PECS with downstream computer vision tasks, we incorporate it into the widely used UE platform, a popular tool for virtual environments. Our PECS, excluding the lens simulation module, is deployed on the UE platform to achieve real-time generation of multi-modal datasets (e.g., RGB frames and events). To facilitate various visual tasks, we also develop an automatic label annotation tool on the UE platform, providing bounding boxes for object detection and scene distance values for depth estimation. As depicted in Fig. 5, we present the synthetic data of a stereo hybrid camera on a drone platform, which mainly includes

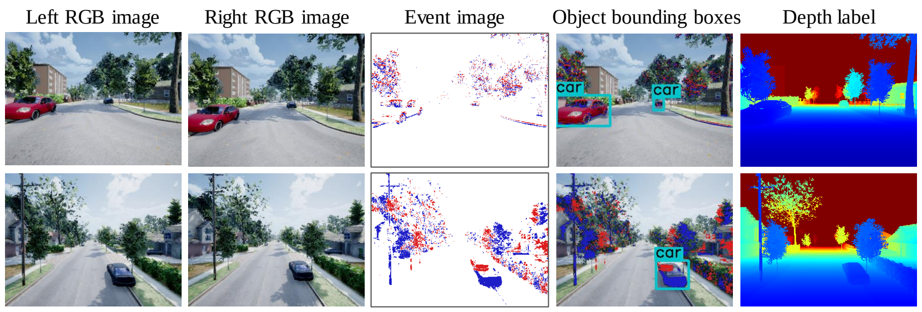


Fig. 5: Representative examples of multi-task synthetic datasets. We integrate our PECS into the UE platform to generate extensive synthetic datasets supporting multiple tasks, such as video reconstruction, object detection, and depth estimation.

Table 4: Performance comparison on event-based video reconstruction. The E2VID network [32] trains on synthetic data from each event camera simulator and tests on three representative sequences of the real-world HS-ERGB dataset [45].

Sequences	PSNR↑			SSIM↑			LPIPS↓		
	ESIM	V2E	PECS	ESIM	V2E	PECS	ESIM	V2E	PECS
spinning_umbrella	8.756	8.437	**9.236**	0.472	0.471	**0.480**	0.451	0.449	**0.430**
fountain_bellevue2	6.139	5.867	**6.564**	0.213	0.211	**0.218**	0.675	0.670	**0.662**
bridge_lake_03	11.770	12.686	**13.243**	0.541	0.560	**0.563**	0.332	0.274	**0.250**
Mean	8.889	8.997	**9.678**	0.409	0.414	**0.420**	0.486	0.464	**0.447**

RGB images, event streams, object detection labels, and scene depth values. Notably, our PECS can generate comprehensive synthetic datasets using both monocular and stereo cameras, supporting a variety of vision tasks, including video reconstruction, object detection, monocular depth estimation, binocular depth estimation, and more. Besides, we also adjust the speed of motion cameras and scene light intensity, generating diverse datasets with high-speed motion and low-light conditions, showcasing the advantages of event cameras.

Validation on Event-based Video Reconstruction. Event-based video reconstruction [32,56] is a typical task that effectively validates the effectiveness of synthetic data from various simulators. Specifically, we first use our PECS and two competitive simulators (e.g, ESIM [31] and V2E [12]) to generate large-scale synthetic datasets in the UE platform. Then, we select a representative events-to-video reconstruction algorithm (i.e., E2VID [32]) to be trained on these simulated datasets respectively. The training details align with those suggested in E2VID [32]. Finally, we test various trained models on the real-world HS-ERGB dataset [45]. We use three metrics (i.e., PSNR, SSIM [46], and LPIPS [51]) to measure the performance of event-based video reconstruction (see Table 4). Obviously, our PECS shows superior generalization capability compared to the top

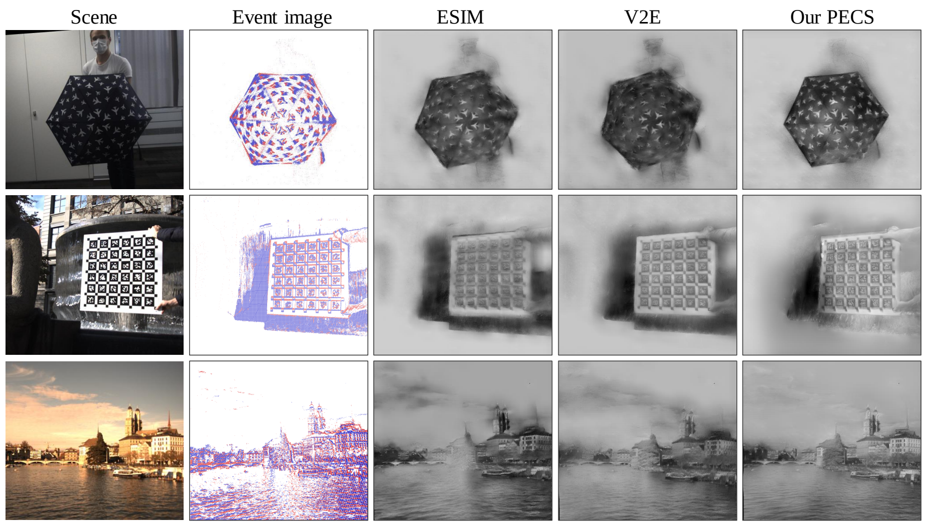


Fig. 6: Visual comparisons on event-based video reconstruction. The E2VID network [32] trains on simulated event data from each simulator and tests on the real-world HS-ERGB dataset [45] recorded by a Prophesee Gen4 camera and a RGB camera.

competitor V2E with an average increase of 0.681 in PSNR, 0.6% improvement in SSIM, and a 1.7% decrease in LPIPS. Furthermore, we present some visualization results on the real-world HS-ERGB dataset [45] recorded by a Prophesee Gen4 camera and an RGB camera. As shown in Fig. 6, our PECS-trained network produces visually pleasing images with finer details and fewer artifacts. As our PECS is designed from a physical-based perspective, it naturally promotes reconstructed images with natural image statistics. Hence, the utilization of our simulator proves more effective, enabling the learning of high-quality video reconstruction [11,23,27,47,48,50,55] even with limited or no real training samples.

6 Conclusion

This paper proposes a novel Physical-based Event Camera Simulator (PECS) that can generate a highly realistic event stream by directly interfacing with the 3D scene. To the best of our knowledge, our PECS is the first event camera simulator to design a realistic lens simulation block and a multispectral rendering module. Experiments demonstrate that Our PECS consistently outperforms four state-of-the-art simulators in some scenarios with various motion speeds and light changes, showing that it is currently the most realistic event camera simulator. Besides, integrating our PECS into the Unreal Engine (UE) platform allows the generation of extensive multi-task datasets, showcasing its utility in downstream vision tasks. We also believe that our PECS provides a proof-of-principle tool for the next generation of neuromorphic cameras.

Acknowledgements

This work is partially supported by the National Natural Science Foundation of China under Grant 20181310845.

References

1. Annamalai, L., Ramanathan, V., Thakur, C.S.: Event-lstm: An unsupervised and asynchronous learning-based representation for event-based data. IEEE RAL **7**(2), 4678–4685 (2022)
2. Baek, S., Eshraghian, J.K., Thio, W., Sandamirskaya, Y., Iu, H.H., Lu, W.D.: A real-time retinomorphic simulator using a conductance-based discrete neuronal network. In: AICAS. pp. 79–83 (2020)
3. Bellekens, B., Spruyt, V., Berkvens, R., Penne, R., Weyn, M.: A benchmark survey of rigid 3d point cloud registration algorithms. Int. J. Adv. Intell. Syst **8**(5), 118–127 (2015)
4. 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 (2023)
5. Bi, Y., Andreopoulos, Y.: Pix2nvs: Parameterized conversion of pixel-domain video frames to neuromorphic vision streams. In: ICIP. pp. 1990–1994 (2017)
6. Bocca, A., Baek, D.: Automated driving systems: Key advantages, limitations and risks. In: AEIT AUTOMOTIVE. pp. 1–6 (2019)
7. Gallego, G., Delbrück, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A.J., Conradt, J., Daniilidis, K., et al.: Event-based vision: A survey. IEEE TPAMI **44**(1), 154–180 (2020)
8. Gehrig, D., Gehrig, M., Hidalgo-Carrió, J., Scaramuzza, D.: Video to events: Recycling video datasets for event cameras. In: CVPR. pp. 3586–3595 (2020)
9. Greenspan, M., Yurick, M.: Approximate kd tree search for efficient icp. In: 3DIM. pp. 442–448 (2003)
10. Gu, D., Li, J., Zhang, Y., Tian, Y.: How to learn a domain-adaptive event simulator? In: ACM MM. pp. 1275–1283 (2021)
11. He, W., You, K., Qiao, Z., Jia, X., Zhang, Z., Wang, W., Lu, H., Wang, Y., Liao, J.: Timereplayer: Unlocking the potential of event cameras for video interpolation. In: CVPR. pp. 17804–17813 (2022)
12. Hu, Y., Liu, S.C., Delbruck, T.: v2e: From video frames to realistic dvs events. In: CVPRW. pp. 1312–1321 (2021)
13. Hwang, I., Kim, J., Kim, Y.M.: Ev-nerf: Event based neural radiance field. In: WACV. pp. 837–847 (2023)
14. 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)
15. Kang, Z., Li, J., Zhu, L., Tian, Y.: Retinomorphic sensing: A novel paradigm for future multimedia computing. In: ACM MM. pp. 144–152 (2021)
16. Kolb, C., Mitchell, D., Hanrahan, P.: A realistic camera model for computer graphics. In: CGIT. pp. 317–324 (1995)
17. Li, D., Li, J., Tian, Y.: Sodformer: Streaming object detection with transformer using events and frames. IEEE TPAMI **45**(11), 14020–14037 (2023)

18. Li, J., Li, J., Zhu, L., Xiang, X., Huang, T., Tian, Y.: Asynchronous spatio-temporal memory network for continuous event-based object detection. IEEE TIP **31**, 2975–2987 (2022)
19. Li, W., Saeedi, S., McCormac, J., Clark, R., Tzoumanikas, D., Ye, Q., Huang, Y., Tang, R., Leutenegger, S.: Interiornet: Mega-scale multi-sensor photo-realistic indoor scenes dataset. arXiv (2018)
20. 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)
21. Liu, P., Chen, G., Li, Z., Tang, H., Knoll, A.: Learning local event-based descriptor for patch-based stereo matching. In: ICRA. pp. 412–418 (2022)
22. Liu, X., Li, J., Fan, X., Tian, Y.: Event-based monocular dense depth estimation with recurrent transformers. In: arXiv (2022)
23. Mei, H., Wang, Z., Yang, X., Wei, X., Delbruck, T.: Deep polarization reconstruction with pdavis events. In: CVPR. pp. 22149–22158 (2023)
24. 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)
25. Mueggler, E., Rebecq, H., Gallego, G., Delbruck, T., Scaramuzza, D.: The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam. IJRR **36**(2), 142–149 (2017)
26. Muglikar, M., Gehrig, M., Gehrig, D., Scaramuzza, D.: How to calibrate your event camera. In: CVPR. pp. 1403–1409 (2021)
27. Pan, L., Hartley, R., Scheerlinck, C., Liu, M., Yu, X., Dai, Y.: High frame rate video reconstruction based on an event camera. IEEE TPAMI **44**(5), 2519–2533 (2020)
28. 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)
29. Peng, X., Wang, Y., Gao, L., Kneip, L.: Globally-optimal event camera motion estimation. In: ECCV. pp. 51–67 (2020)
30. Pharr, M., Jakob, W., Humphreys, G.: Physically based rendering: From theory to implementation (2023)
31. Rebecq, H., Gehrig, D., Scaramuzza, D.: Esim: An open event camera simulator. In: CoRL. pp. 969–982 (2018)
32. Rebecq, H., Ranftl, R., Koltun, V., Scaramuzza, D.: Events-to-video: Bringing modern computer vision to event cameras. In: CVPR. pp. 3857–3866 (2019)
33. Rebecq, H., Ranftl, R., Koltun, V., Scaramuzza, D.: High speed and high dynamic range video with an event camera. IEEE TPAMI **43**(6), 1964–1980 (2019)
34. Rizzo, C., Schuman, C., Plank, J.: Event-based camera simulation wrapper for arcade learning environment. In: ICNS. pp. 1–5 (2022)
35. Rogalski, A., Razeghi, M.: Narrow-gap semiconductor photodiodes. In: Photodetectors: Materials and Devices III. vol. 3287, pp. 2–13 (1998)
36. Sasián, J.: Introduction to lens design. Cambridge University Press (2019)
37. Segal, A., Haehnel, D., Thrun, S.: Generalized-icp. In: RSS. vol. 2, p. 435. Seattle, WA (2009)
38. Sengupta, J., Liu, S., Andreou, A.: Retinosim: an event-based data synthesis tool for neuromorphic vision architecture exploration. In: ICNS. pp. 1–9 (2022)
39. Su, Q., Chou, Y., Hu, Y., Li, J., Mei, S., Zhang, Z., Li, G.: Deep directly-trained spiking neural networks for object detection. In: ICCV. pp. 6555–6565 (2023)

40. Sun, L., Sakaridis, C., Liang, J., Jiang, Q., Yang, K., Sun, P., Ye, Y., Wang, K., Gool, L.V.: Event-based fusion for motion deblurring with cross-modal attention. In: ECCV. pp. 412–428 (2022)
41. Sundar, V., Ardelean, A., Swedish, T., Bruschini, C., Charbon, E., Gupta, M.: Sodacam: Software-defined cameras via single-photon imaging. In: ICCV. pp. 8165–8176 (2023)
42. Ta, K., Bruggemann, D., Brödermann, T., Sakaridis, C., Van Gool, L.: L2e: Lasers to events for 6-dof extrinsic calibration of lidars and event cameras. In: ICRA. pp. 11425–11431 (2023)
43. Taverni, G., Moeys, D.P., Li, C., Cavaco, C., Motsnyi, V., Bello, D.S.S., Delbruck, T.: Front and back illuminated dynamic and active pixel vision sensors comparison. IEEE Trans. Circuits Syst. II: Exp. Briefs **65**(5), 677–681 (2018)
44. Tsuji, Y., Yatagawa, T., Kubo, H., Morishima, S.: Event-based camera simulation using monte carlo path tracing with adaptive denoising. arXiv (2023)
45. Tulyakov, S., Bochicchio, 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: CVPR. pp. 17755–17764 (2022)
46. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: From error visibility to structural similarity. IEEE TIP **13**(4), 600–612 (2004)
47. Weng, W., Zhang, Y., Xiong, Z.: Event-based video reconstruction using transformer. In: ICCV. pp. 2563–2572 (2021)
48. Yu, L., Zhang, X., Liao, W., Yang, W., Xia, G.S.: Learning to see through with events. IEEE TPAMI **45**(07), 8660–8678 (2023)
49. Yu, W., Li, J., Zhang, S., Ji, X.: Learning scale-aware spatio-temporal implicit representation for event-based motion deblurring. In: ICML (2024)
50. Zhang, H., Zhang, L., Dai, Y., Li, H., Koniusz, P.: Event-guided multi-patch network with self-supervision for non-uniform motion deblurring. IJCV **131**(2), 453–470 (2023)
51. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR. pp. 586–595 (2018)
52. Zhang, Z., Cui, S., Chai, K., Yu, H., Dasgupta, S., Mahbub, U., Rahman, T.: V2ce: Video to continuous events simulator. arXiv (2023)
53. Zheng, X., Liu, Y., Lu, Y., Hua, T., Pan, T., Zhang, W., Tao, D., Wang, L.: Deep learning for event-based vision: A comprehensive survey and benchmarks. arXiv (2023)
54. Zhu, A.Z., Wang, Z., Khant, K., Daniilidis, K.: Eventgan: Leveraging large scale image datasets for event cameras. In: ICCP. pp. 1–11 (2021)
55. Zhu, L., Li, J., Wang, X., Huang, T., Tian, Y.: Neuspike-net: High speed video reconstruction via bio-inspired neuromorphic cameras. In: ICCV. pp. 2400–2409 (2021)
56. Zhu, L., Wang, X., Chang, Y., Li, J., Huang, T., Tian, Y.: Event-based video reconstruction via potential-assisted spiking neural network. In: CVPR. pp. 3594–3604 (2022)
57. Ziegler, A., Teigland, D., Tebbe, J., Gossard, T., Zell, A.: Real-time event simulation with frame-based cameras. In: ICRA. pp. 11669–11675 (2023)