Reliable Event Generation With Invertible Conditional Normalizing Flow

Daxin Gu, Jia Li, *Senior Member, IEEE*, Lin Zhu, Yu Zhang, and Jimmy S. Ren

***Abstract*—Event streams provide a novel paradigm to describe visual scenes by capturing intensity variations above specific thresholds along with various types of noise. Existing event generation methods usually rely on one-way mappings using hand-crafted parameters and noise rates, which may not adequately suit diverse scenarios and event cameras. To address this limitation, we propose a novel approach to learn a bidirectional mapping between the feature space of event streams and their inherent parameters, enabling the generation of reliable event streams with enhanced generalization capabilities. We first randomly generate a vast number of parameters and synthesize massive event streams using an event simulator. Subsequently, an event-based normalizing flow network is proposed to learn the invertible mapping between the representation of a synthetic event stream and its parameters. The invertible mapping is implemented by incorporating an intensity-guided conditional affine simulation mechanism, facilitating better alignment between event features and parameter spaces. Additionally, we impose constraints on event sparsity, edge distribution, and noise distribution through novel event losses, further emphasizing event priors in the bidirectional mapping. Our framework surpasses state-of-the-art methods in video reconstruction, optical flow estimation, and parameter estimation tasks on synthetic and real-world datasets, exhibiting excellent generalization across diverse scenes and cameras.**

***Index Terms*—Conditional normalizing flow, contrast threshold, event camera, event generation, event noise rate.**

I. Introduction

EVENT camera [1], [2] is a new type of vision sensor that measures brightness changes by the asynchronous event stream. With the advantage of high temporal resolution, high dynamic range, and low power consumption, the event camera has shown promising performance in many computer vision tasks, such as video reconstruction [3], [4], [5], [6], optical flow estimation [7], [8], [9], [10], semantic segmentation [11], [12], etc.

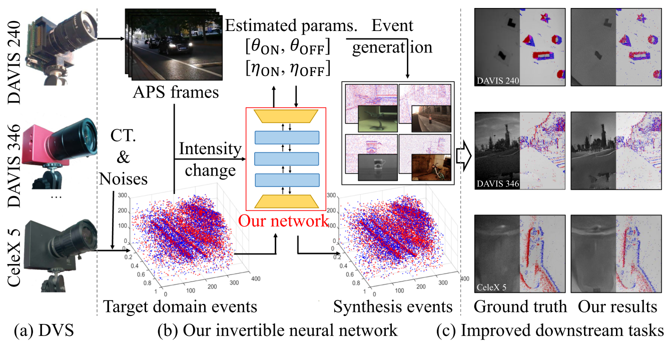

Fig. 1. Framework of our approach. With the sharing sampling principle of different dynamic vision sensors (DVS) shown in (a), the main factors affecting the event quality are contrast thresholds (CT mismatch) and event noises. To learn the inherent relationship between camera parameters and event spaces, we propose an invertible event generation network (shown in (b)) that can not only estimate camera parameters from target domain events but also produce high-quality synthetic event stream and its representation reversely. The visualization results in (c) show that our approach can adapt the target event distribution automatically, resulting in better reconstruction results (c).

Despite numerous advances in event-based vision, training the deep learning-based approaches requires a large amount of simulated event data, which is limited by the performance of the event simulator (e.g., ESIM [13], V2E [14]). As shown in Fig. 1, the realistic event streams can vary significantly when different event cameras and their settings are used. A high-quality simulation of event streams needs careful tuning of the event simulator according to different scenarios [15], [16], [17], which is often an arduous and time-consuming task. This brings us an open question in event-based vision field: *How to generate reliable event streams to fit all event cameras in diversified scenarios?*

To solve this problem, a class of event generation strategies has been proposed from the perspective of reducing the *sim-to-real* gap, which considers it as a domain adaption task and operates on the *learned features*. These strategies aim to transfer event representations to the target domain in order to directly improve the generalization capabilities of deep models. For example, Zhu et al. [18] propose an end-to-end manner that translates images to events directly via a CycleGAN [19] based framework. Planamente et al. [17] introduce domain adaptation techniques to event data to help reduce the sim-to-real gap.

Manuscript received 6 December 2022; revised 1 October 2023; accepted 18 October 2023. Date of publication 23 October 2023; date of current version 8 January 2024. This work was partially supported by the National Natural Science Foundation of China under Grant 62132002. Recommended for acceptance by C. Fermuller. *(Corresponding author: Lin Zhu; Jia Li.)*

Daxin Gu is with the State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing 100191, China (e-mail: daxingu@buaa.edu.cn).

Jia Li is with the State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing 100191, China, and also with Peng Cheng Laboratory, Shenzhen 518055, China (e-mail: jiali@buaa.edu.cn).

Lin Zhu is with the School of Computer Science, Beijing Institute of Technology, Beijing 100081, China (e-mail: linzhu@bit.edu.cn).

Yu Zhang is with SenseTime Research and Tetras.AI, Beijing 100080, China (e-mail: zhangyulb@gmail.com).

Jimmy S. Ren is with SenseTime Research and Tetras.AI, Beijing 100080, China, and also with Qing Yuan Research Institute, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: jimmy.sj.ren@gmail.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2023.3326538, provided by the authors.

Digital Object Identifier 10.1109/TPAMI.2023.3326538

Gu et al. [20] propose a learning-based event simulator that reduces the gap between synthetic and real data by learning distributions of event contrast thresholds from real events. This category of approaches can learn event distribution from target domain data automatically and efficiently. However, these deterministic generating frameworks are designed for transferring the synthetic events to a specific target domain, which needs to be re-trained or fine-tuned for modeling the various settings of event cameras and uncertainty of usage scenarios. In other words, this category of strategies has *limited generalization* for various scenarios or event cameras.

Another category aims to calibrate appropriate *hand-crafted generation parameters* and simulate target events via existing event simulators [13], [14]. In general, hand-crafted parameters are designed based on the physics process of event generating. For example, Brandli et al. [21] propose a parameter estimating approach that utilizes the difference between the intensity image and event cumulative map to update parameters iteratively. Stoffregen et al. [16] present a strategy to search a set of appropriate parameters for existing simulators (e.g., ESIM [13]) by measuring the *average events per pixel per second*. Wang et al. [22] estimate parameters by building and solving an overdetermined system of events number, parameters, and log intensity changes. Compared with methods in the first category, some heuristic metrics are utilized to compile statistical event stream information, which has a better generalization to a certain extent. Unfortunately, it is *difficult to fully characterize a realistic pixel model* (e.g., the contrast threshold mismatch and noise distribution of the event camera) if only relying on hand-crafted parameters. Moreover, the lengthy event simulating steps and complex parameter tuning strategies make it difficult to generalize to other domains.

In this paper, we start from a new perspective that both *learned features* and *hand-crafted generation parameters* should be utilized to generate a reliable event stream for diversified scenarios. Instead of considering the event generation as a unidirectional generation or estimation task, we attempt to model the event stream and the parameter features in a unified invertible framework. The main reasons can be briefly summarized as follows: 1) First, the deterministic generation frameworks (e.g., [17], [18], [20]) that learn features from event distributions are efficient for transferring the synthetic events to a specific target domain. However, the disadvantage is that the inherent randomness of event generation from noisy contrast threshold sampling is difficult to be modeled without additional guidance. 2) Second, incorporating the hand-crafted parameters of event generation is critical to model generalization. Although the hand-crafted parameters are usually simple [16], [21], they are developed to some extent based on the sensor characterization (e.g., contrast threshold, noise level). These parameters have a clear physical definition to guide event emitting and are common to different event cameras or scenarios. Thus, with the guidance of parameters, the event distribution learned by the model will be more realistic and more generalized.

Based on the above analysis, we propose a novel invertible event generation framework based on conditional normalizing flow [23], which efficiently exploits the inherent relationship between the event distribution space and the event parameter space. The event representation and event parameters can be generated and optimized in an invertible manner. Specifically, according to the physics-based event generation model, we first define several parameters of the event stream: intensity changes, contrast threshold, and temporal event noise. Thanks to the exact log-likelihood training mechanism of normalizing flow, our approach can build an explicit projection between event parameters space and event distribution space for each pixel. It allows our normalizing flow network to learn the calibration of precise event camera parameters consistent with the input event streams. The event parameters and event distribution features can be simultaneously optimized based on the proposed invertible framework. Unlike previous works [17], [18], [20], we do not need to re-train or fine-tune our network for new scenarios because the invertible framework can learn the relationship between event parameters and event distribution spaces. It also makes our framework generalize well to different types of event cameras (e.g., CeleX5 [24], DAVIS240 [2], DAVIS346 [25]) due to their event generation models being common.

The main contributions of this paper are summarized as:

1) We propose a novel invertible event generation framework, which has for the first time built the inherent relationship between event parameter space and event distribution space. The proposed framework generalizes well to multiple scenarios and event cameras without re-training or fine-tuning, while performing significantly better than recent state-of-the-art methods.
2) We design a novel intensity-guided conditional affine simulation module, which is able to build an invertible and bijective mapping from event to parameter under a certain intensity. We also explore multiple event-based losses, including a conditional loss, an event representation loss, and two flow losses, which are effective for optimizing our invertible network.
3) Experiments on both synthetic and real-world datasets demonstrate that our framework accomplishes superior performance for both threshold estimation and noise estimation. Moreover, the generated event streams improve over the state-of-the-art on two downstream vision tasks (i.e., video reconstruction and optical flow estimation) that are very sensitive to the distribution of events and noise, demonstrating the superiority of our framework.

The rest of this paper is organized as follows: Section II reviews related work. Section III analyzes the physics-based event generation model and discusses the effects of each parameter. Section IV presents the whole framework of our approach. Qualitative and quantitative experiments are reported and analyzed in Section V. Finally, Section VI concludes our work.

II. Related Work

This section briefly reviews the existing methods strongly related to our work, including dynamic vision sensor, event data generation, event parameter estimation, and normalizing flow. Table I summarizes the difference of existing event generation methods with the works related to event parameter estimation.

TABLE I
SUMMARY OF EXISTING ASYNCHRONOUS/SYNCHRONOUS EVENT GENERATION AND PARAMETER ESTIMATION APPROACHES

Method	Input	Optimization objectives			Type	Description
		Ae.	Sr.	Op.		
TAF [26]	Frames	✓			Non-learning	Event simulator without considering noise and asynchronous.
PIX2NVS [27]	Frames	✓			Non-learning	Simualting asynchronous readout with high *FPS* sequences.
ESIM [13]	Frames	✓			Non-learning	Event simulator with manual tuning parameters.
V2E [14]	Frames	✓			Non-learning	Event simulator with manual tuning parameters.
EGAN [18]	Frames		✓		Learning-based	Learning event representation for particular downstream tasks.
DA4E [17]	Event		✓		Learning-based	Adapting synthetic events to a given target domain.
EZoom [28]	Event		✓		Learning-based	Redistributing events without modeling the uncertain noise.
ELSTM [29]	Event		✓		Learning-based	Unsupervised learning-based representation not to solve *sim-to-real*.
JAER [21]	Frames & Event			✓	Non-learning	Estimating CT. by oversimplified strategy and lacking theoretic.
ECC [22]	Frames & Event			✓	Non-learning	Estimating CT. without considering noise.
S2R [16]	Frames & Event	✓		✓	Non-learning	Reduce the *sim-to-real* gap for a given target domain.
LGAN [20]	Frames	✓	✓	✓	Learning-based	Learning-based CT. estimation for a given target domain.
EVolt [30]	Frames	✓		✓	Non-learning	Simulating events by modeling parameters of specific DVS circuit.
Our	Frames & Event	✓	✓	✓	Learning-based	Generalization for different event cameras and different scenes.

Ae.: Asynchronous event stream. Sr. Synchronous representation. Op.: Optimized parameters. CT.: Contrast threshold.

A. Dynamic Vision Sensor

Dynamic vision sensor (DVS) captures intensity information in a considerably different manner from other conventional CMOS-based image sensor technology. It records intensity change instead of intensity absolute value, which causes its low latency and high dynamic range. With the development of DVS, there are serial of manufacturers (e.g., iniVation, Samsung, CelePixel, et al.) developing DVS in different circuit designs and parameter configurations [31]. A main trend of DVS development is that it has increasing spatial resolution, increasing readout speed, and adding features (e.g., grayscale output, color vision, inertial measurement unit (IMU), and optical flow). The first practical DVS is DVS128 [1], whose spatial resolution is 128×128 pixels. It can only output events without IMU and grayscale images. Then more DVSs are developed with larger spatial resolution, such as DAVIS240 [2] (240×180 pixels) and DAVIS346 [25] (346×260 pixels). Most of them are designed to record not only event streams but also grayscale images, IMU with more bandwidth (faster readout speed). Recent DVSs have developed more spatial resolution (CeleX5 [24], Gen4-CD [32], and DVS-Gen4 [33]), whose horizontal resolution can reach more than 1280 pixels, while the maximum bandwidths are larger than 100 Meps (compared with 1 Meps for DVS128). There are some other efforts to provide additional functions for DVS. For example, [34], [35], [36] attempt to offer color information with filter arrays. [24], [37] provide optical flow information when recording events.

While all the aforementioned Dynamic Vision Sensors (DVSs) are designed using event generation models, the variations in circuit structures and manufacturing techniques result in slight differences in the distributions of their event streams. This poses challenges to the reliable generation of events in our task.

B. Event Generation and Parameter Estimation

Asynchronous Event Stream Simulation: The prior works [13], [14], [26], [27] try to simulate massive event streams by input video. For example, an event simulator [26] proposes to generate events with image sequences by mimicking real event camera imaging processes that record the difference in intensities between consecutive images. However, the asynchronous and noise of the event camera are not simulated. In order to address the problems, Bi et al. [27] propose to simulate events with high frame rate rendering images. At the same time, the approach mimics a real event camera to provide the asynchronous readout. The ESIM simulator [13] and v2e simulator [14] mimic several key steps of the real physical process of event cameras to generate event streams, including contrast threshold noise and temporal noise. These event simulators can provide high-quality event streams for other event-based tasks with the appropriate simulation parameters. It has been shown well generalization compared with real-world events [16], [38], [39]. However, these simulators need elaborative manual tuning of various parameters to bridge the sim-to-real domain gap.

Synchronous Event Representation Generation: To enhance the performance of event-based deep learning models, a series of research studies [28], [29] have focused on generating 2D deep event features directly using deep learning models. For instance, ELSTM [29] has been proposed to learn a 2D spatial grid of events, efficiently representing the event stream. EZoom [28] aims to denoise events and enhance resolution to improve downstream tasks. These works are capable of generating high-quality event features from existing event data. However, there is an inherent domain shift between the input 2D grids and the output features. Other approaches [17], [18] treat event simulation as a domain adaptation problem and generate event features to serve as inputs for event-based downstream tasks. For example, EventGan [18] proposes a CycleGAN-based framework [19] to convert image sequences into event representations directly. However, training CycleGAN with pairwise data fails to capture the inherent randomness involved in generating events for a specific image sequence. Planamente et al. [17] introduces an unsupervised domain adaptation framework to convert synthetic events generated by the ESIM simulator [13] into a specific target domain. The generative models [17] can automatically and efficiently generate event representations without requiring manual parameter tuning. However, the generation processes in these methods are deterministic and cannot model the uncertain event noise. Different from the above approaches, we propose a novel intensity-guided conditional affine simulation module that establishes an invertible and bijective mapping from events to parameters. This module is designed to address domain shifts in new datasets while providing sufficient generalization ability.

Event Parameter Estimation: A series of event parameter estimation works aim to estimate parameters with a simple model (e.g., contrast threshold) by analyzing the relationship between the event number occurring in a period of time and the intensity change at the corresponding time [21], [22]. However, those statistical methods are lack noise modeling of contrast threshold mismatch and limited by the quality of image sequences. Stoffregen et al. [16] propose a strategy to search a set of appropriate parameters for existing simulators (e.g., ESIM [13]) by aligning the *average events per pixel per second* of synthetic events with that of real ones. EVolt [30] models real DVS circuit and calibrates its parameters by calculating multivariable regression. LGAN [20] proposes a learning-based parameter estimating approach, which learns the contrast thresholds by distinguishing the events in different domains via several discriminators. Both of those approaches can narrow the gap between synthetic and real data with long-time training and amounts of real event streams.

C. Normalizing Flow

Normalizing flow based models [23], [40], [41] parameterize a distribution using an invertible neural network with another one. It has been widely used in point cloud generation [42], [43] and image conditional generation [44], [45]. The prior work NICE [40] proposes a simple additive coupling layer, which skillfully simplifies the calculation of reversible functions. After that, RealNVP [41] ulteriorly proposes an affine coupling layer to enrich the learning parameters of reversible functions. Besides, the proposed spatial-channel features division, shuffle, and squeeze strategy promote the use of convolution neural networks in normalizing flow. The latter works mainly adopt the widely successful Glow architecture [23], which first introduces the actnorm layer and invertible 1×1 convolutions, to conditional generation by encoding variables in the affine coupling layers.

Instead of parameter tuning design or fixed event domain translation, we propose a learning-based invertible framework that establishes a dependable relationship between parameters and events. Thanks to the merits of normalizing flow for invertible conversion of distribution, the proposed approach can not only build the reversible mapping between event parameters to events, but also generate event streams that align with target domain characteristics realistically. To the best of our knowledge, this is also the first event-based invertible neural network.

III. Physics-Based Event Generation Model

This section revisits the working principle of the event camera, including the fundamental parameters of the circuit components, and analyzes the effect of those parameters based on real events. It will provide guidance for designing our invertible network.

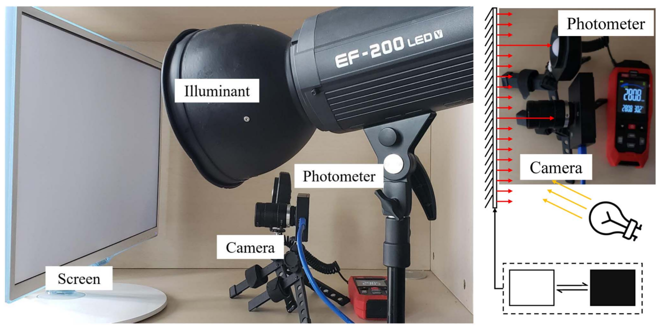

Fig. 2. Our event data collection scene. The illuminant is utilized to maintain constant light intensity, and the screen is used to display different colors. The light intensity can be measured by a photometer.

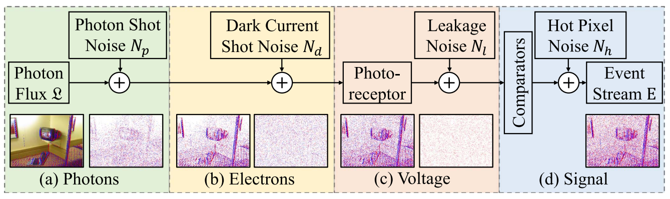

Fig. 3. Overview of event generation process and visualization of noise source of each stage.

A. Event Signal Model

Essentially, the event sampling process can be expressed as the following formula: $\Delta\log(\mathcal{I}) \geq \theta$, where θ is the contrast threshold of the event camera, $\mathcal{I}$ denotes the light intensity [1], [2]. An event $e = \{x, y, t, p\}$ occurs when the log intensity changes $\Delta\log(\mathcal{I})$ at position (x, y) over the set threshold θ at time t. In detail, the whole events generation process can be concluded into four stages, as shown in Fig. 3.

Photons Stage: When a photon flux signal $\mathfrak{L}$ reaches a pixel $\mathbf{p}$ on the photoreceptor at time t, it is transformed into a photocurrent $\mathcal{I}_p$ ($\mathcal{I}_p \propto \mathfrak{L}$). However, due to the quantal nature of photons, this process introduces a constant photon shot noise N_s [14]. Unlike conventional cameras that accumulate photons over a fixed integration time, DVS integrates a constant number of photons, triggering events when an intensity change exceeds a defined threshold θ. Therefore, the photon shot noise is influenced not only by the light intensities but also by the contrast thresholds.

Electrons Stage: The photodiode exhibits a dark current $\mathcal{I}_d$ at all times, even in the absence of light, which introduces noise N_d to the input photocurrent along with the photon shot noise [46]. The impact of this noise is determined by the contrast between the photocurrent and the dark current. In situations with low lighting intensities, the input signal photocurrent becomes comparable to the dark current, leading to a greater influence of the dark current.

Voltage Stage: In this stage, the input photocurrent $\mathcal{I}_v = \mathcal{I}_p + \mathcal{I}_d$ undergoes logarithmic conversion and amplification to produce a voltage change $\Delta\mathcal{V}(t_k)$, which is stored after the last event triggered at time t_{k-1}. DVS accomplishes this through a change detector reset switch that resets the voltage change once it reaches a certain threshold. However, the reset switch introduces an unavoidable junction leakage current $\mathcal{I}_l$, which affects the voltage change and contributes to leakage noise [46]. The voltage change can be described by $\Delta\mathcal{V}(t_k) = -\log(\frac{\mathcal{I}_v(t_k)}{\mathcal{I}_v(t_{k-1})}) + \int_{t_{k-1}}^{t_k} \mathcal{I}_l du$. Because the leakage current always reduces the voltage change $\Delta\mathcal{V}(t_k)$, the leak noise event is activated as an ON event.

Signal Stage: The event signal is generated by comparing the voltage change $\Delta\mathcal{V}(t_k)$ with the contrast threshold θ using comparators [2]. When the voltage change $\Delta\mathcal{V}(t_k)$ crosses the ON or OFF threshold, DVS outputs an ON or OFF event, respectively. This can be expressed as follows:

$$\begin{cases} \Delta\mathcal{V}_d(t_k) \leq -\theta_{\text{ON}}, & \text{ON events} \\ \Delta\mathcal{V}_d(t_k) \geq \theta_{\text{OFF}}, & \text{OFF events} \\ -\theta_{\text{ON}} < \Delta\mathcal{V}_d(t_k) < \theta_{\text{OFF}}, & \text{no events.} \end{cases} \quad (1)$$

In this stage, due to the abnormally low thresholds or reset switches with very high dark current [14], the comparators will records events continuously at a few hot pixels positions, even in the absence of input.

B. Parameters Analysis

Our objective is to establish the correlation among parameters, events, and brightness in order to synthesize events that align with the target domain under different event cameras and environmental conditions. From the event signal model, we can identify two factors that have a significant impact on event triggering: the contrast threshold θ and event noise N. The contrast threshold determines the sensitivity of the event camera to changes in intensity. On the other hand, noise events are inevitably generated during the event generation process, and they can significantly affect the total number of events, particularly in low-light scenarios. In this section, we introduce how to model the event generation process using two key factors, i.e., contrast threshold θ and event noise N. To better model the event stream, we first divide the events into two categories $\mathcal{E} = \{\mathcal{E}_A, \mathcal{E}_N\}$, where $\mathcal{E}_A = \{e_{A1}, \dots, e_{Ak}\}$ represents the events that are not affected by noise and $\mathcal{E}_N = \{e_{N1}, \dots, e_{Nk}\}$ denotes the noise events.

Contrast Threshold: The first category events $\mathcal{E}_A$ are related to the intensity changes of scenes and the contrast threshold θ. As discussed in [13], the contrast thresholds are various in different pixels and can be modeled with normal distribution $\mathcal{N}(\theta; \sigma_\theta)$, where σ_θ is the mismatch variance of thresholds. Furthermore, the ON thresholds θ_{ON} that tagger ON events also differ from the OFF thresholds θ_{OFF}. The contrast thresholds can be adapted independently in real event cameras with the right set of electronic biases. The mimicking of the threshold mismatch is also considered as a key step in recent event simulators [14].

To estimate ON or OFF contrast thresholds, we build an ideal laboratory environment (Fig. 2) to force the event camera to record the events with ON or OFF polarity, respectively. Since the contrast threshold cannot be directly obtained, we set the sensor configure parameter (SCP) to different values to control the contrast threshold of the event camera and capture bright and dark screens separately for analyzing ON (OFF) contrast thresholds. As shown in Table II and the first row of Fig. 4,

TABLE II
EVENTS CAPTURED BY CELEX5 WITH DIFFERENT THRESHOLDS

SCP	θ_{ON}	N_{ON}	θ_{OFF}	N_{OFF}
101	0.119 ± 0.082	2850599	0.203 ± 0.121	1751785
121	0.159 ± 0.089	1675849	0.249 ± 0.107	1110802
141	0.210 ± 0.096	1172399	0.302 ± 0.103	8564635
161	0.269 ± 0.103	890028	0.358 ± 0.105	713966
181	0.342 ± 0.113	729427	0.420 ± 0.105	637139
201	0.433 ± 0.140	558242	0.479 ± 0.109	545253

We can change the contrast threshold of Celex5 by setting different SCPs. The total number of ON (OFF) events N_{ON} (N_{OFF}) and the statistical ON (OFF) contrast thresholds θ_{ON} (θ_{OFF}) are shown in the table.

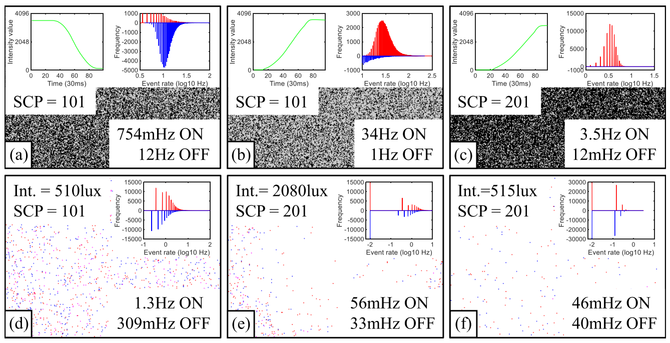

Fig. 4. Real events captured by CeleX5. We accumulate events in 30 ms and present the DVS histogram. The top-right inset in each sub-figure shows the frequency distribution of ON and OFF event rates. The axes represent the frequency of events across the pixel array plotted against the event rate in Hertz on a logarithmic scale (log_{10}). The top-left of each sub-figure shows the intensity change and threshold ($\theta \propto$ SCP). The bottom-right of each sub-figure illustrates the event rate. The three sub-figures in the first row show the events in a scene of decreasing brightness (a), increasing brightness (b), and increasing brightness with a larger contrast threshold (c), respectively. By comparing (b) and (c), it is evident that a larger contrast threshold reduces the number of events. The other three sub-figures show events captured in a fixed luminance of about 500 lx (d), 2000 lx (e), and 500 lx with a larger contrast threshold (f). Comparing (d) and (e), we observe that a brighter scene produces fewer noise events. Additionally, a larger contrast threshold reduces event noise compared (d) with (f).

we adjust the contrast thresholds in the range of $0.1 \sim 0.5$, which is a common threshold value range for event cameras [47]. Obviously, the number of ON and OFF events in CeleX5 differ in the same SCP set, and the variance of contrast thresholds is about 0.1. It means the mismatch of contrast thresholds is widespread, which deeply influencing the output events. It further motivated us to model ON and OFF contrast thresholds in our network.

Temporal Event Noise Rate: The number of noise events E_A determines the quality of the event streams, mainly affected by scene intensity and sensitivity of DVS. Typically, the influence factors of event noise can be divided into several types, such as photons shot noise, dark current, leak noise, and hot noise, as shown in Fig. 3, which have been modeled in [14]. Although there have been several existing methods [14], [30] that attempt to measure these noises individually, the complex and intertwined mechanisms of event noise make it difficult to isolate and precisely quantify each component independently. To model event noise in different event cameras, we follow [46] to utilize event noise rate (measured as the count of noise events per second per pixel) to quantify the different noise sources, respectively.

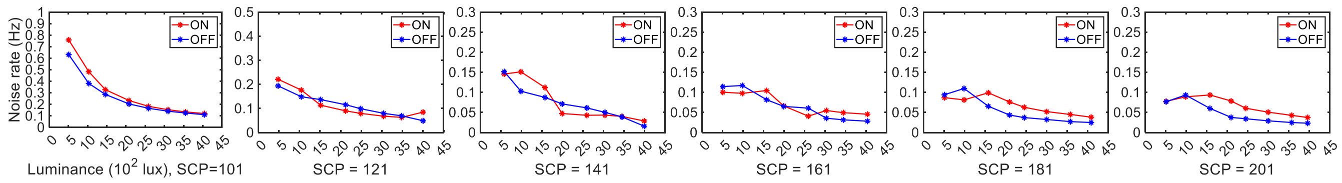

Fig. 5. Noise events captured in a fixed scene with different light intensities and contrast thresholds. We fix the screen display and then adjust the intensity of scene with the illuminator from 500 lx to 4000 lx. The contrast threshold of the event camera is changed with different SCP values. In each setting, we capture event streams for about 30 seconds. Under a fixed light intensity, all events should come from noise. The abscissa of the line chart represents the luminance ($\times$100lux), and the ordinate represents the noise event rate (Hz). The red line is the ON event noise rate, and the blue line is the OFF event noise rate. SCP is positively correlated to the contrast thresholds. Best viewed in electronic version.

Since there is currently a lack of quantitative analysis of the event noise for the CeleX camera, we conduct measurements of the noise level in terms of an event frequency per pixel (Hz) and analyze the noise under different brightness and threshold conditions. The statistical results are shown in Fig. 5. We collect and compute ON (OFF) noise event rate η_{ON} (η_{OFF}) under different intensities and contrast thresholds. Our analysis revealed that the noise event rate depends on the scene intensity and the event camera threshold, as these factors primarily influence the occurrence of noise events. The results in the second row of Fig. 4 clearly show the noise distribution in different settings. The number of noise events decreases with the increasing intensity and contrast threshold. To simplify the noise process, we utilize the temporal noise event rate η as a parameter to represent event noise (see Fig. 5). These findings serve as evidence for the subsequent validation framework's noise estimation capability. Furthermore, the visualization results in the second row of Fig. 4 clearly illustrate the noise distribution under different settings. It is evident that the number of noise events diminishes as the intensity and contrast threshold increase. This observation aligns with the findings of [46], indicating a comparable relationship between noise, threshold, and brightness in both CeleX and DAVIS cameras. This insight inspires us to use a unified framework for estimating the correlation between camera parameters (θ and η), brightness, and events across different cameras, aiming to improve generalizability.

IV. Approach

A. Motivation

The motivation of our framework is presented in Fig. 6. Our goal is to generate appropriate parameters that approximate the given target event domain (i.e., real captured event stream). It is an ill-posed problem to estimate camera parameters from the asynchronous and sparse event stream since the noise type and event distribution are diverse. To solve this problem, we introduce an intensity-guided conditional affine simulation to establish the reliable relationship between parameter and event space based on a large amount of simulated data. In this way, a bijective invertible mapping (Fig. 6(a)) can be built to ensure that the event parameters P and event streams E can transform each other with a specific intensity condition. Subsequently, the optimized parameters of the target event domain can be predicted using the real captured event and the invertible mapping (Fig. 6(b)), and the realistic event streams are obtained (Fig. 6(c)).

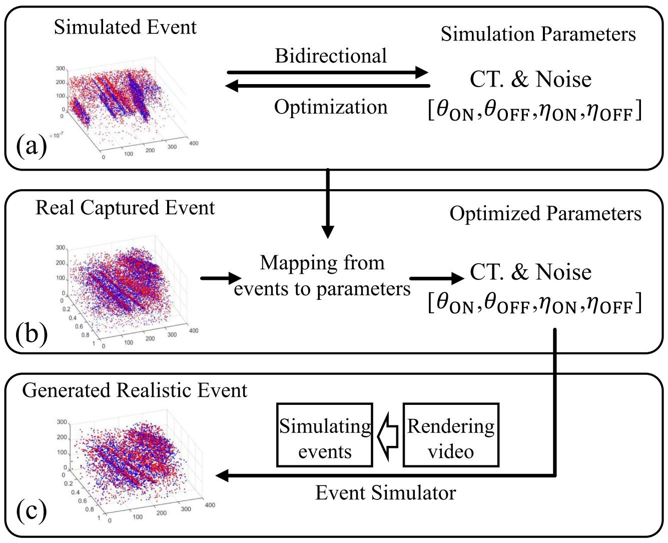
Simulated Event
Simulation Parameters
Bidirectional
Optimization
CT. & Noise
$[\theta_{\text{ON}},\theta_{\text{OFF}},\eta_{\text{ON}},\eta_{\text{OFF}}]$
(a)
Real Captured Event
Optimized Parameters
Mapping from events to parameters
CT. & Noise
$[\theta_{\text{ON}},\theta_{\text{OFF}},\eta_{\text{ON}},\eta_{\text{OFF}}]$
(b)
Generated Realistic Event
Simulating events
Rendering video
Event Simulator
(c)

Fig. 6. Motivation of our framework. (a) Our invertible network learns the inherent relationship between event and parameter spaces based on simulated data. (b) The optimized parameters of the target events can be obtained based on invertible mapping. (c) Realistic events can be generated by the event simulator with our optimized parameters.

Fig. 7 shows the sketch of our framework. As discussed in Section III, we simplify the parameters of the event camera into two main parameters, namely contrast threshold and temporal noise rate. Based on the event generation model, a given event streams $\mathcal{E}$ can be clearly defined by the intensity change of the scene ($\mathrm{I} = \Delta\log(\mathcal{I})$) and a group of event parameters ($\mathrm{P} = \{\theta_{\text{ON}}, \theta_{\text{OFF}}, \eta_{\text{ON}}, \eta_{\text{OFF}}\}$). On the basis of conditional normalizing flow [23], the distribution of the event streams can be expressed uniquely with the event parameters, and vice versa. The invertible neural network is trained with maximum likelihood estimation, using amounts of simulated event data and corresponding simulating parameters. With the help of well-trained flow model, the event noise rate (η_{ON} and η_{OFF}) and contrast thresholds (θ_{ON} and θ_{OFF}) can be adapted for different target domains of real event streams precisely.

B. Event-Based Invertible Neural Network Architecture

The general architecture of our event-based invertible neural network is composed of an intensity encoder and the stacked invertible normalizing flow layers. Each level of the invertible normalizing flow layer contains 16 flow steps, where each flow

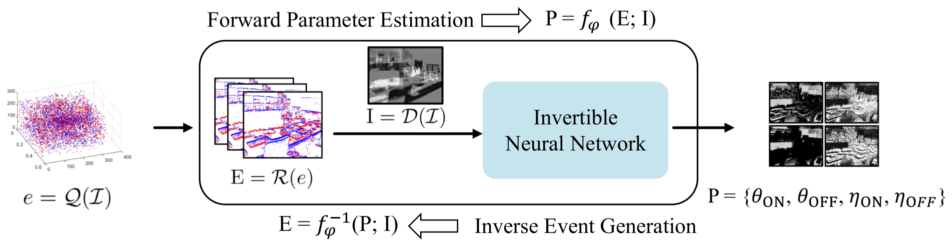
Forward Parameter Estimation
P = f_φ (E; I)
I = $\mathcal{D}(\mathcal{I})$
Invertible Neural Network
E = $\mathcal{R}(e)$
$e = \mathcal{Q}(\mathcal{I})$
P = {θ_{ON}, θ_{OFF}, η_{ON}, η_{OFF}}
E = f_φ^{-1}(P; I)
Inverse Event Generation

Fig. 7. Illustration of our invertible network. $\mathcal{Q}$, $\mathcal{R}$, and $\mathcal{D}$ denote the event sampling process, event representation generation process, and light intensity changes estimation, respectively. In the forward parameter estimation procedure, the event representation E combined the intensity changes I are transformed to event parameters P through a parameterized invertible function f_ϕ. In the inverse event generation procedure, the parameter P is transformed to event representation E through the inverse function f_ϕ^{-1}.

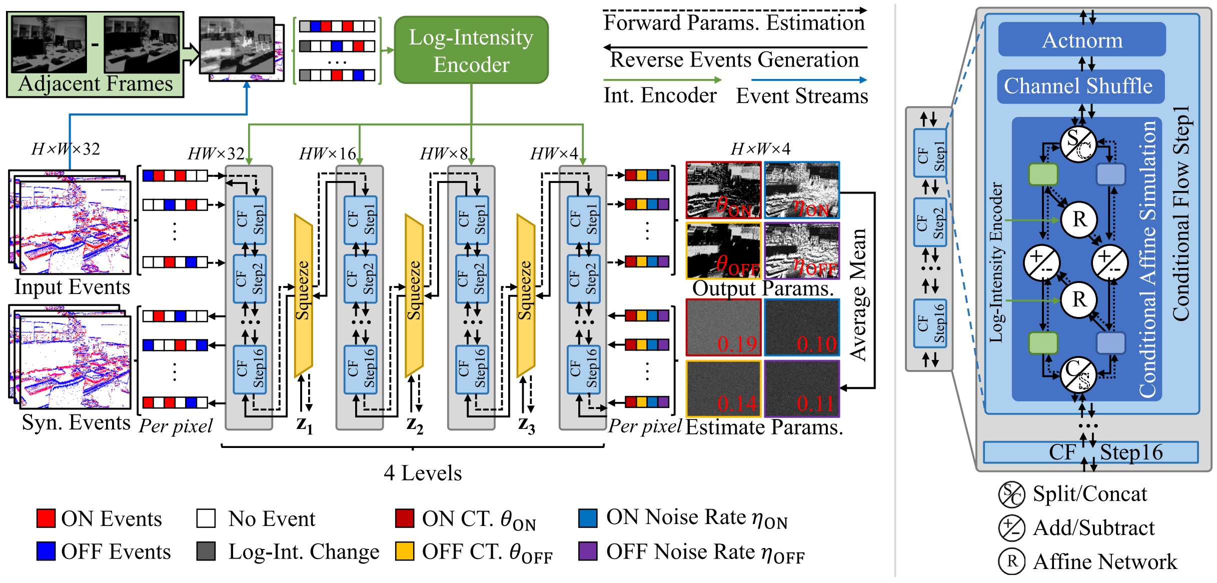

Fig. 8. Pipeline of our invertible neural network. The event stream is processed through a series of flow steps to estimate event parameters per pixel. Each level of the invertible normalizing flow layer contains 16 flow steps, each consisting of actnorm layer, channel shuffle layer, and conditional affine simulation. The details of the flow step are illustrated in the right portion of the figure. Best viewed in electronic version.

step consists of actnorm layer, channel shuffle layer, and conditional affine simulation, as illustrated in Fig. 8. We will introduce the invertible architecture in this subsection.

Event Representation: To process the event stream using conditional normalizing flow, the asynchronous event streams should be converted into a fixed-size representation vector. We follow [38] to encode the events into a spatiotemporal voxel grid. In detail, we select ON (or OFF) events at a position p in fixed-length duration Δt and discretize the events into B bins by event timestamp t as :

$$\mathrm{E}(t_n) = \sum^{\Delta t} \max(0, 1 - |t_k - t_i^*|), \tag{2}$$

where $t_i^* = \frac{B-1}{\Delta t}(t_i - t_0)$ denotes the normalized event timestamp. In our work, the temporal bins B is set to 16.

Invertible Conditional Normalizing Flow: Our invertible network explicitly establishes the conditional distribution $P_{\mathrm{E|I}}(\mathrm{E|I}, \varphi)$ of event representations corresponding to the scene intensity. Due to the event representations are changing with the variety of parameter settings, we build a pixel-level model between the events space and parameters space in a specific intensity condition. The key point of our invertible network f_ϕ is to parameterize the distribution $P_{\mathrm{E|I}}(\mathrm{E|I}, \varphi)$ using conditional normalizing flow. In the conditional setting, f_φ maps an event representation in a specific condition scene to a group of parameters $\mathrm{P} = f_\varphi(\mathrm{E;I})$. We hope the network is invertible for mapping diverse event representations with corresponding parameters. In detail, for the same scene, the event representation can be reconstructed from corresponding parameters P as $\mathrm{E} = f_\varphi^{-1}(\mathrm{P;I})$. In this case, given the parameter distribution $P_{\mathrm{P}}(\mathrm{P})$, the probability density $P_{\mathrm{E|I}}$ can be explicitly represented via the conditional normalizing flows as,

$$P_{\mathrm{E|I}}(\mathrm{E|I}, \varphi) = P_{\mathrm{P}}(f_\varphi(\mathrm{E;I}))\left|\det \frac{\partial f_\varphi}{\partial \mathrm{E}}(\mathrm{E;I})\right|. \tag{3}$$

In order to enhance the expression ability of the conditional normalizing flow, we compose a stack of invertible and tractable bijective function $\{f_{\varphi_n}\}_n^N$ to build our invertible networks f_φ. By applying the chain rule along with the multiplicative property of the determinant [41], (3) can be further expressed as

$$P_{\mathrm{E|I}}(\mathrm{E}|\mathrm{I},\varphi) = P_{\mathrm{P}}(\mathrm{P}) \prod_{n=0}^{N} \left| \det \frac{\partial f_{\varphi n}}{\partial \mathbf{h}_n}(\mathbf{h}_n; g_\varphi(\mathrm{E};\mathrm{I})) \right|. \quad (4)$$

Equation (4) consists of a sequence of N invertible function $\mathbf{h}_{n+1} = f_\varphi^n(\mathbf{h}_n; g_\varphi(\mathrm{E};\mathrm{I}))$, where $\mathbf{h}_0 = \mathrm{E}$ and $\mathbf{h}_N = \mathrm{P}$. The scene intensity change information is first encoded with a lightweight network $g_\varphi(\mathrm{E};\mathrm{I})$ to extract a high frequency log-intensity change representation (described in Section IV-C), which is suitable for conditioning all flow-function $f_{\varphi n}$.

Conditional Flow Step: The conditional flow step is the basic block of our invertible conditional normalizing flow. As shown in Fig. 8, each flow step consists of three different layers. The actnorm layer is first, followed by the channel shuffle layer. At last, a conditional affine simulation layer is applied. Our network employs $K = 16$ flow steps at each level. $L = 4$ levels is set to transform parameter space (4 dimensions) from event representation space (32 dimensions). The networks $f_{\varphi n}$ in the conditional affine simulation layer are constructed via an MLP network. Details of these layers are described as follows, and the conditional affine simulation is described in Section IV-C.

Due to the different data dimensions between parameter space and event space, a squeeze operation between a neighboring flow step is introduced to the halves feature in the channel dimension. In this way, only half of the latent variables can be transmitted to the next flow step to estimate parameter space distribution. Another half of the latent variables are abandoned. In the reverse process, those parts of variables are sampled from a simple distribution (e.g., standard normal distribution).

For a more flexible combination of latent features in two different parts (event and parameter), we introduce the channel shuffle layer to achieve the whole latent layer update $\mathbf{h}_{n+1} = \mathcal{W} \cdot \mathbf{h}_n$. In detail, the latent features are shuffled in channel dimension via a given order $\mathcal{W}$. Moreover, the channel shuffle layer is invertible with efficient computing of determinants. Similar as [23], we introduce the actnorm layer to channel-wise normalizing via a learned scaling and bias.

C. Intensity-Guided Conditional Affine Simulation

Based on the analysis in Section IV-A, estimating camera parameters from the event stream is an ill-posed problem. Inspired by the event generation model, we introduce additional intensity information to solve this problem. An intensity-guided conditional affine simulation is proposed to establish a reliable bijective mapping between parameter and event space.

Intensity Change Conditional Encoder: Our intensity change encoder takes the combination of the above event representation with low-framerate log-intensity change value (i.e., the total log-intensity change in the corresponding duration) as input, and outputs a high-framerate log-intensity change vector. The network contains two stages. In the first stage, a two-layer MLP structure is utilized to mix the input information of changed log-intensity and event representation to generate a global intensity change feature. This MLP structure extracts the features of a continuous intensity change scene in a duration, which is considered as the scene intensity condition input to the invertible normalizing flow. In the second stage, to supervise learning the continuous intensity change feature, we divide the global intensity change feature into m pieces, where m is the intensity change sample frequency. Empirically, we set $m = 16$ to obtain sufficient temporal information. Subsequently, the m intensity change features are input to another MLP network to infer respective log-intensity change values. The MLP contains three linear layers, which produce the features with 96, 96, and 1 channels, respectively. Note that each linear layer is followed by a regularization layer (LayerNorm) and an activation function (ReLU) in our network.

Conditional Affine Simulation: To ensure efficient training and inference, the determinant of the Jacobian $\frac{\partial f_{\varphi n}}{\partial \mathbf{h}_n}$ needs to be computed efficiently and tractably. Based on the affine coupling layer proposed in [40], [41], we design our conditional affine simulating layer to provide a simple and powerful strategy for building an easily invertible and fastly computing flow layer. In contrast to the traditional affine coupling layer, our proposed affine simulating layer mixes the conditional encoding u and latent feature $\mathbf{h}_n$ with a nonlinear function $f_{\varphi n}$. In this way, a more adequate information aggregation from log-intensity change to the flow branch is achieved. The detailed affine simulating layer is conclusively expressed as follows,

$$\mathbf{h}_{\mathrm{A}n+1} = \mathbf{h}_{\mathrm{A}n}, \mathbf{h}_{\mathrm{B}n+1} = \mathbf{h}_{\mathrm{B}n} + f_{\varphi n}(\mathbf{h}_{\mathrm{A}n}; g_\varphi(\mathrm{E};\mathrm{I})), \quad (5)$$

where $\mathbf{h}_{\mathrm{A}n}, \mathbf{h}_{\mathrm{B}n}$ are divided from $\mathbf{h}_n$ in the channel dimension. Note that $f_{\varphi n}$ can represent an arbitrary neural network without influencing the invertible and bijective of the flow layer. Furthermore, the Jacobian of (5) is triangular, with a simple determinant computation result ($|\det \frac{\partial f_{\varphi n}}{\partial \mathbf{h}_n}| = 1$ in our work).

D. Optimization Objectives

Our approach for invertible event generation constructs a mapping that specifies a correspondence between the event representation E and event parameters P. The total optimization process contains two directions: The forward pass of our invertible takes event representations and intensity change encoding features as input and produces the parameter values. Otherwise, inputting parameters and intensity code, the reverse pass aims at recovering realistic event data. The details of loss functions and training strategy are discussed as follows.

1) Mapping From Event to Parameter: As discussed in Section IV-C, the proposed conditional affine simulation needs intensity change information as guidance to estimate an accurate mapping f_φ from event space E to parameter space P. To achieve this, we utilize conditional loss and parameter loss to guide the mapping from event to parameter.

Conditional Loss: It is easily observed that events often occur sparsely in real scenes, which means the intensity change of each pixel is slight and most of the intensity change ground truth label is zero. In other words, the training samples of the intensity encoder are imbalanced. To address the problems, we conduct bi-directional training with Quality Focal Loss, widely used in object detection [48], [49], to optimize our intensity encoding network.

$$\mathcal{L}_{\mathrm{I}} = -\alpha||\sigma_1(g_\varphi) - \mathrm{I}_{\mathrm{gt}}||^\beta \cdot [\mathrm{I}_{\mathrm{gt}} \log \sigma_1(g_\varphi) + (1 - \mathrm{I}_{\mathrm{gt}}) \log(1 - \sigma_1(g_\varphi))], \quad (6)$$

where I_{gt} is the intensity change ground truth, and we limit it to the range of $0 \sim 1$ with a linear scale factors $\log_{255} \mathrm{e}$. σ_1 is the sigmoid function, and we set $\alpha = 1$ and $\beta = 2$ in our experiment.

Parameter Loss: Due to the event parameters (contrast threshold and shot noise rate) are usually close to zero, we introduce L1 Charbonnier loss to optimize our network.

$$\mathcal{L}_{\mathrm{P}} = \sqrt{\lambda^2||\sigma_2(f_\varphi(\mathrm{E}, \mathrm{I})) - \mathrm{P}_{\mathrm{gt}}||^2 + \epsilon^2}, \quad (7)$$

where P_{gt} is the ground truth parameters, and $\epsilon = 1 \cdot 10^{-3}$ in our experiment. The existence of ϵ in the above function makes the gradient in the backpropagation not too small when the difference of parameters is close to zero. σ_2 is the tanh function to limit the value in the parameter space $(-1, 1)$. $\lambda = \frac{\mathrm{N}_{\mathrm{now}}}{\mathrm{N}_{\mathrm{total}}}$ is the confidence weight of the predicted value, which is decided by the event number of current sample $\mathrm{N}_{\mathrm{now}}$ and the total event number of the current batch $\mathrm{N}_{\mathrm{total}}$.

Forward Flow Loss: We follow [44] to train our network by minimizing the negative log-likelihood (NLL) $\mathcal{L}_{\mathrm{F}}$. The loss contains two parts: the log-determinant of the Jacobian and the negative log-likelihood to a simple distribution, as shown in (8).

$$\mathcal{L}_{\mathrm{F}} = -\sum_{i=0}^{L-1} \log P_{\mathbf{z}}(\mathrm{Squeeze}(f_{\varphi_{Ki}}(\mathrm{E}; \mathrm{I}))) - \sum_{n=0}^{N-1} \log \left|\det \frac{\partial f_{\varphi_n}}{\partial \mathbf{h}_n}(\mathbf{h}_n; g_\varphi(\mathrm{E}; \mathrm{I}))\right|, \quad (8)$$

where L is the level number of our flow network, and K is the number of flow steps for each level. Squeeze$(\cdot)$ operation is used to half the feature in the channel dimension. We constrain half of the output from the Squeeze$(\cdot)$ operation to a simple distribution $P_{\mathbf{z}}$ (e.g., Gaussian) with negative log-likelihood.

2) Mapping From Parameter to Event: To perform the reverse mapping f_φ^{-1} from parameter space to event space, we design an event representation loss that is inspired by the event distribution prior. The reverse mapping can be effectively achieved based on event representation loss, flow loss, and conditional loss.

Event Representation Loss: As discussed in Section III, the events can be classified into two categories: noise events and others. To precisely model two kinds of events, we consider utilizing different losses to handle them. In detail, for noise events, we consider the distribution of noise as a simple distribution (e.g., a Gaussian distribution) and utilize the negative log-likelihood (NLL) between the training sample pairs to optimize the network. For other events, we choose l1 Charbonnier loss to supervise the network. Moreover, we observe that intensity change sharply often occurs around the edges of moving scenes. Therefore, a multi-scale loss is proposed to optimize the network, as shown in Fig. 9.

$$\mathcal{L}_{\mathrm{E}} = \sum_{\mathbf{m} \in \mathbb{M}} [(1 - \gamma)\mathcal{L}_{\mathrm{l1c}}(\sigma_3(\mathbf{m})) + \gamma \mathcal{L}_{\mathrm{nll}}(\sigma_3(\mathbf{m}))], \quad (9)$$

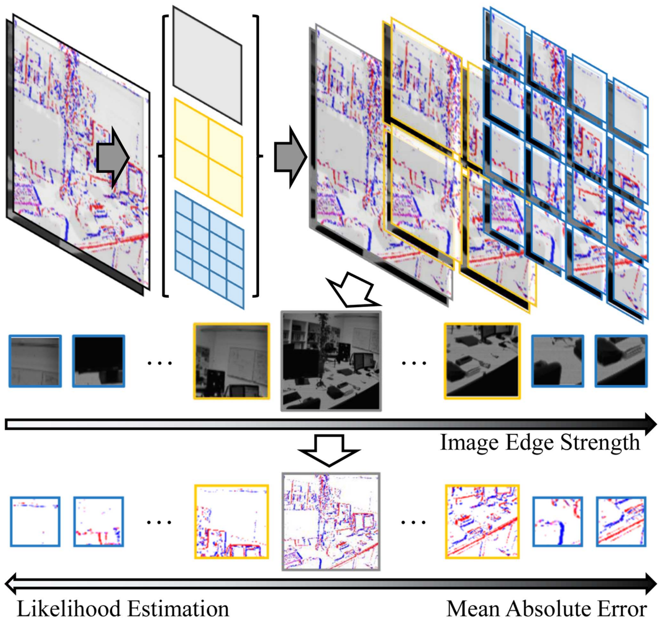

Fig. 9. Proposed event representation loss which partitions feature into multi-scale local patches and weights them by edge motion attribute. After that, different patches are handled with weighted L1 Charbonnier loss and negative log-likelihood loss.

where $\mathcal{L}_{\mathrm{l1c}}$ denotes the l1 Charbonnier loss, $\mathcal{L}_{\mathrm{nll}}$ denotes the NLL loss, and σ_3 is the ELU function. The generated event representations are partitioned into different size local patches, $\mathbb{M}$ is the set of local patches. For each local patch $\mathbf{m}$, we follow [50] to analyze the edge strength of the patch with its corresponding image gradients. Such attribute is tightly related to scene edges, and we employ it as the weights of two kinds of loss. For example, if there is no intensity change in a patch, the events in this patch are mainly generated from noise. Then we set a large weight to utilize NLL loss to constrain the event number in this patch.

Inverse Flow Loss: Similar to the forward process, we train the reverse mapping f_φ^{-1} with NLL loss $\mathcal{L}_{\mathrm{F}'}$ similarly. Different from (8), in each *squeeze* step, we should sample half of the latent variables from a simple distribution (e.g., standard normal distribution). There is no need to compute these parts of loss. We only require to compute the log-determinant for each flow step f_{φ_n}, as shown in (10).

$$\mathcal{L}_{\mathrm{F}'} = -\sum_{n=0}^{N-1} \log \left|\det \frac{\partial f_{\varphi}^{-1}{}_n}{\partial \mathbf{h}_n}(\mathbf{h}_n; g_\varphi(\mathrm{E}; \mathrm{I}))\right|. \quad (10)$$

3) Bidirectional Training: Loss Weight Adaption: Because the distribution of the event stream is usually unbalanced (e.g., the event density is mainly related to scene texture and motion), the loss may vary extremely with different event streams, which causes the training process of the invertible network to be difficult. Instead of training with a set of fixed weights, we propose to balance the effect of each loss into the same magnitude to effectively train our network. In the experiment, each loss value is normalized to $0 \sim 10$. For example, if a loss is 2.3×10^3, we adjust it to 2.3 to avoid the whole network converging to a local optimum prematurely.

Training Details: Our network is trained in both forward and inverse directions via $\mathcal{L}_{\text{forward}}$ and $\mathcal{L}_{\text{inverse}}$, which are defined as:

$$\mathcal{L}_{\text{forward}} = \omega_1 \mathcal{L}_{\text{I}} + \omega_2 \mathcal{L}_{\text{P}} + \omega_3 \mathcal{L}_{\text{F}},$$
$$\mathcal{L}_{\text{inverse}} = \omega_1 \mathcal{L}_{\text{I}} + \omega_2 \mathcal{L}_{\text{E}} + \omega_3 \mathcal{L}_{\text{F}'}. \tag{11}$$

During the training process, we use a patch size of 64×64, Adam optimizer with warmup strategy in the first 10 epochs. Then, the learning rate increases to $1 \cdot 10^{-4}$, which decreases via cosine annealing and reduces to zero after 300 epochs. Our network takes about 3 days to train on a single NVIDIA GTX 1080ti GPU.

E. Parameter-Event Paired Data Generation

Paired Data Generation for Training Phase: Training the proposed invertible network needs a large amount of event and parameter data. Formally, we define the event streams as $\mathbb{E}^S = \{\text{E}_1, \ldots, \text{E}_T\}$, the corresponding high framerate intensity change as $\mathbb{I}^S = \{\text{I}_1, \ldots, \text{I}_T\}$, and the corresponding event camera parameters as $\mathbb{P}^S = \{\text{P}_1, \ldots, \text{P}_T\}$. Our goal is to generate a large dataset of parameter-event pairs $\{\mathbb{E}^S, \mathbb{I}^S, \mathbb{P}^S\}$. However, there is not exist such large-scale event dataset that have ground-truth parameters and intensity value together. In our work, we train the network using synthetic events by generating random parameters to simulate extensive event streams across various scenes.

We use the event simulator V2E [14] to generate events with the rendering video sequences and the given simulation parameters. To render a video, we first sample one or multiple foreground images and a background image from the unlabeled subset of the MS COCO dataset [51]. Then, we crop the foreground objects from the corresponding annotated instance masks. We set a synthetic motion trajectory for each foreground object and the background image, which allows them to translate, scale, and rotate in a 2D plane with different speeds. Finally, we warp those foreground objects and the background image with their motion trajectories, and render the video sequence at a high framerate (1000 fps). During generating the events, we sample a different set of ON and OFF event parameters (i.e., contrast threshold and temporal event noise rate) for each rendering video sequence to enrich the training data. For the contrast threshold, we sample it according to a normal distribution with mean values from 0.1 to 0.5 and a standard deviation of 0.15. For event noise rate, we define it as follows:

$$\eta_{\text{ON}} = ((\text{F} - 1) \times \text{N}_{01}(\mathcal{I}) + 1) \times \text{R}(\theta_{\text{ON}}),$$
$$\eta_{\text{OFF}} = ((\text{F} - 1) \times \text{N}_{01}(\mathcal{I}) + 1) \times \text{R}(\theta_{\text{OFF}}). \tag{12}$$

where $\mathcal{I}$ denotes the light intensity. Similar to [14], (12) describes a linear decreasing process of noise with the intensity $\mathcal{I}$. N_{01} is the normalization function, which normalizes the $\mathcal{I}$ into $(0, 1]$. $\text{R}(\cdot)$ is defined as the relationship between noise event rate η and the contrast threshold θ. In our simulation process, we set $\text{R}(\theta) = \frac{4}{90\theta - 5}$ to constrain it in the range 0.1 to 1. F is the noise reducing factor in bright parts, and we set $\text{F} = 0.25$ as recommended in [14].

Asynchronous Event Stream Generation in Inference Phase: In the inference phase, the proposed network can estimate optimized parameters, including contrast threshold and temporal noise rate. Compared to other parameter estimation methods, such as [21] and [22], our estimated parameters are compatible with V2E [14] by replacing the corresponding components. In this way, we can utilize V2E [14] with our estimated parameters to generate asynchronous event stream readily. For example, V2E initials the contrast threshold from a Gaussian distribution with the given mean and variance. We adopt the initial value of each pixel with our estimated thresholds. For the noise generation, we set the original temporal noise rate and leaky noise rate to zero. Note that our event noise rate has contained temporal noise and noisy events triggered by leakage current and dark current. Thus, the temporal noise is directly added to the event stream in V2E. For hot pixels, we follow V2E to minimize the contrast threshold to 0.01 to prevent too many hot pixel events.

V. Experiment

A. Experimental Settings

In our experiments, we evaluate our invertible network on three different dataset settings, including a synthetic dataset, three public datasets, and our real captured dataset. Both parameter estimation (contrast threshold and temporal event noise rate) and event stream generation tasks are evaluated on the above datasets.

1) Datasets: Synthetic Dataset: First, as proposed in Section IV-E, we generate 300 event sequences with different parameters for training and 30 sequences for evaluating the validity of our network. We refer this dataset as the synthetic dataset, in which we can easily get a group of parameters for each event stream. The parameter-event pairs provide bi-directional ground truths, which ensure the parameter estimation performance can be evaluated quantitatively.

Public Dataset: To effectively evaluate the quality of the generated event stream, we conduct experiments on IJRR [52], MVSEC [53], and EVIMO2v2 [54] datasets. The former two datasets are widely used in event-based downstream tasks (i.e., video reconstruction [16], [20], [38] and optical flow estimation [9], [16], [20]). Both of them contain minutes to hours of event streams with corresponding video frame sequences, which are captured by real event cameras. The EVIMO2v2 [54] dataset comprises event streams acquired by the Samsung Gen3 sensor, along with corresponding image sequences captured by the Flea3 camera. We select 21 sequences with high-quality reference images as our test set. The details of the selected sequences can be found in supplementary materials. In our experiment, we test the video reconstruction and optical flow estimation performances by training related models on our generated event data and testing on these datasets. Note that our generation model is trained based on the training set of the datasets.

Real Captured Dataset: Moreover, in order to verify the generalization and robustness of our invertible network, we further conduct parameter estimation experiments on the real captured dataset. We capture several groups of event streams in different intensity conditions via different event cameras (DAVIS346 and CeleX5). The experiment on the real capturing events can future demonstrate the advantage of our proposed approach.

2) Evaluation Metrics: Our invertible network aims to estimate precise parameters and generate realistic event streams to improve the model training in downstream tasks. In our experiment, we evaluate our network from two aspects: parameter estimation and event generation.

Parameters Estimation: For parameters estimation, we evaluate the accuracy of contrast thresholds and noise rate, respectively, via Mean Absolute Error (MAE) on multiple datasets. To the best of our knowledge, there are only three statistical methods (ECC [22], S2R [16], JAER [21]) and only one deep-learning model (LGAN [20]) in literature that can estimate event contrast threshold, while [14] and [30] are the only two methods that are able to model the noise rate under different intensity.[1]

Event Stream Generation: In order to validate the performance of event stream generation, we follow the evaluation protocols of recent work [16], which evaluates the simulation methods by training the downstream task model with the simulated event data. Similar to [16], we evaluate our network on two tasks: event-based video reconstruction and optical flow estimation. We choose E2V [38] network as the baseline video reconstruction model to train and evaluate the video reconstruction quality with three metrics: Mean Square Error (MSE), Structural Similarity Index (SSIM), and the perceptual metric LPIPS [55]. Note that we evaluate all reconstruction results of each method in the log domain instead of the linear domain to perform the advantage of event-based video reconstruction in a high dynamic range. Meanwhile, EV-FlowNet [9] is employed as the baseline optical flow estimation model to train, and Flow Warp Loss (FWL) [16] is used to measure the quality of optical flow. For a fair comparison, we train E2V and EV-FlowNet with the same settings as [16].

B. Comparisons of Parameters Estimation

In this section, we validate the performance of the parameters estimation, including contrast threshold estimation and temporal noise rate estimation. We conduct experiments on synthetic dataset, public dataset, and real capturing dataset. We can get the parameters labels for synthetic dataset from their events generation configure files. Meanwhile, for the real-world datasets, we can statistics the corresponding parameters by controlling the captured scenes (as in Section III-B) to produce ON events, OFF events, or noise events, respectively.

Thresholds Estimation on Synthetic and Real Captured Datasets: Table III provides a summary of the quantitative

TABLE III
CONTRAST THRESHOLD ESTIMATION RESULTS ON SYNTHETIC AND REAL CAPTURED DATASETS

Method / Dataset	ECC [22]		S2R [16]		JAER [21]		LGAN [20]		Ours	
	θ_{ON}	θ_{OFF}	θ_{ON}	θ_{OFF}	θ_{ON}	θ_{OFF}	θ_{ON}	θ_{OFF}	θ_{ON}	θ_{OFF}
Synthetic data[1]	.043	.049	.133	.131	.055	.055	.117	.128	**.023**	**.022**
Real captured data[2]	.053	.089	.078	.063	.043	.149	.089	.108	**.018**	**.024**
Mean	.048	.069	.106	.097	.049	.102	.103	.118	**.021**	**.023**

[1] The ground truth of synthetic data is the given simulation parameters, which can be found in the supplementary materials.
[2] The ground truth of real data is obtained from Table 2.
The Mean Absolute Error (MAE) metric is used to compare the estimated ON and OFF contrast thresholds with the ground truth. Best results are highlighted in bold.

TABLE IV
NOISE RATE ESTIMATION RESULTS ON DIFFERENT EVENT CAMERAS

Camera type	V2E [14]		EVolt [30]		Ours	
	η_{ON}	η_{OFF}	η_{ON}	η_{OFF}	η_{ON}	η_{ON}
CeleX5	0.077	0.069	-	-	**0.037**	**0.033**
DAVIS346	0.087	0.081	0.083	0.107	**0.026**	**0.030**
Mean	0.032	0.075	0.083	0.107	**0.031**	**0.031**

The Mean Absolute Error (MAE) metric is used to evaluate the performance. Best results are highlighted in bold.

results for our framework compared with four related contrast threshold estimation methods (ECC [22], S2R [16], JAER [21], and LGAN [20]) on both synthetic and real captured datasets.

For the synthetic dataset, as described in Section V-A1, we evaluate the generated thresholds using ground truth provided by the simulator across 30 synthetic sequences. In the case of the real captured dataset, consistent with the settings in Section III-B, we employ a CeleX5 camera to capture 12 event sequences with varying sensor control parameters (i.e., different contrast thresholds). The contrast threshold for each pixel is determined by calculating the light intensity change divided by the number of events. The total statistics, obtained by counting the threshold of each pixel, serve as the ground truth for quantitative comparison.

In the metric of MAE, the proposed approach achieves better accuracy both in positive contrast θ_{ON} and negative θ_{OFF} contrast threshold estimation on both synthetic and real captured dataset. In particular, our network can estimate accuracy parameter results in once forward computation instead of the repeated iteration inference process as S2R [16] and JAER [21]. The invertible framework makes our network can effectively estimate the mapping from event space to parameter space, thus achieving better results. Moreover, compared with LGAN [20] that needs to fine-tune the network for different target domains, our invertible network can adapt to various domains without re-training or fine-tuning. To sum up, the results on MAE metric demonstrate that our network has better generalization than other methods. More experimental results on synthetic and real captured datasets can be found in supplementary materials.

Noise Rate Estimation on Real Captured Dataset: In Table IV, we compare the noise rate estimation results with V2E [14] and

[1]We evaluate the accuracy of contrast thresholds and noise rate via MAE $= \frac{1}{N_s}\frac{1}{N_f}\frac{1}{H\cdot W}\sum_{i=1}^{N_s}\sum_{j=1}^{N_f}\sum_{\mathbf{p}\in\mathbb{P}^{H\times W}}|y_i - \hat{y}_{i,j}^{\mathbf{p}}|$. In a dataset with N_s sequences, for each event sequence i with N_f frames and resolution of $H \times W$, we first calculate absolute value error between the predicted value $\hat{y}_{i,j}^{\mathbf{p}}$ and ground true value y_i for each pixel $\mathbf{p}$ of each frame j. Then the Mean Absolute Error (MAE) for overall dataset is used as a measure of the accuracy of contrast threshold and noise rate estimation.

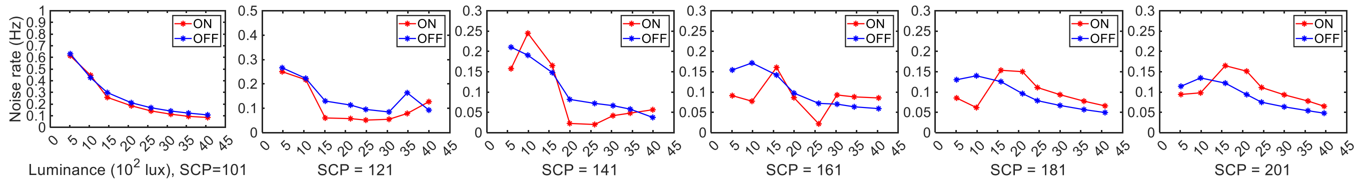

Fig. 10. Noise rate estimating results of CeleX5 data in different intensity and contrast thresholds. The abscissa is the luminance ($\times$100lux) and the ordinate is the noise event rate (Hz). The red line and blue lines represent the ON and OFF event noise rate, respectively. SCP is a DVS parameter, which is related to the contrast thresholds. Best viewed in electronic version.

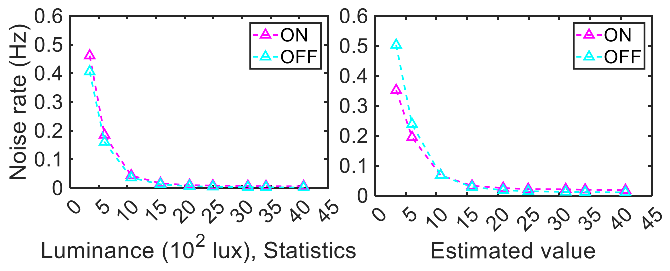

Fig. 11. Noise rate estimating results of DAVIS346 data in different intensities. The abscissa is the luminance ($\times$100lux), and the ordinate is the noise event rate (Hz). The pink and blue lines represent the ON and OFF event noise rates, respectively. Best viewed in electronic version.

DVS-Voltmeter [30] on two real datasets captured by CeleX5 and DAVIS346, respectively. V2E designs an event noise model to estimate noise based on the light intensity and contrast threshold, and uses the estimated noise level to generate event streams. In this experiment, we can directly use the temporal noise model of the V2E simulator to predict the noise rate values of this method. DVS-Voltmeter simulator [30] generates event streams by calibrating with the internal parameters of the DAVIS346 camera. Its noise function has no explicit expression. We can generate noise events by inputting the intensity obtained by CeleX5 into this simulator, and calculate the temporal noise through the generated noise events. Moreover, DVS-Voltmeter only calibrates the the parameters of the DAVIS346 camera, which does not apply to CeleX5. Thus we only test this method on DAVIS346. The MAE metric results on the real captured dataset from two event cameras are shown in Table IV, each of which contains various luminance. As shown in Table IV, our proposed approach can obtain more precise noise rate estimation on different cameras.

In Fig. 10, we show the noise rate estimation results under different luminance and contrast thresholds. The experiment is conducted on the real event dataset captured by the CeleX5 camera. The estimated noise-luminance curves shown in Fig. 10 are similar to those of Fig. 5, which proves the proposed network can effectively model the relationship between noise rate and luminance/contrast threshold.

Furthermore, the noise rate estimation results on DAVIS346 are shown in Fig. 11. The left figure shows the statistical values (ground truth), and the right one is the estimated noise rate of our network. The similar curve between statistics and estimation values shown in Figs. 5, 10, and 11 on two different event cameras demonstrates the realism and reliability of our network.

C. Comparisons on Downstream Tasks

To demonstrate the effectiveness of our network, we conduct experiments on downstream tasks. Table V summarizes the quantitative comparison results of event-based video reconstruction on IJRR [52], MVSEC [53], and EVIMO2v2 [54] datasets, respectively. The original E2V [38] is trained on the simulated events and optical flow. To compare the performance of different event generation methods, we utilize the generated event stream (GE) to train the E2V model and assess the performance of the reconstructed image. Our network demonstrates significant improvements in SSIM compared to the image reconstructed from raw events, indicating minimal distortion of scene structures. Furthermore, when compared to other event simulation pipelines, our reconstruction results outperform them across all metrics. Fig. 12 shows the video reconstruction results generated by different approaches. The results of the original E2V and LGAN present indistinct details on edges, while the results of S2R and EGAN present distorted local details and artifacts. The reconstruction model training on our generated event streams performs better on the details of edges and the contrast, which contains abundant details and a high dynamic range.

In addition, we conduct the optical estimation experiment on three public datasets: IJRR [52], MVSEC [53], and EVIMO2v2 [54] datasets. The results of different methods are summarized in Table VI. We use FWL metric, as defined in [16], where a higher value indicates better performance. We utilize the generated event stream (GE) to train the EVFLOW [9] model and assess the performance of the estimated flow. Our proposed network outperforms other approaches on all three public datasets, significantly improving the performance compared to both the raw events and other simulation pipelines. As shown in Fig. 13, the optical flow visualization results and the image of warped events intuitively demonstrate the improvement of our generated events.

In summary, the results of downstream tasks demonstrate the proposed network can generate better domain-adapted event streams, which boosts the state-of-the-art performances on event-based video reconstruction and optical flow estimation up to 18.6% and 6.4%, respectively. Detailed video reconstruction and optical flow estimation results for each sequence in various datasets can be found in the supplementary materials.

D. Analysis and Discussion

1) Ablation Study. Network Depths: In order to study the effect of network depth, we set different numbers of flow steps

TABLE V
VIDEO RECONSTRUCTION RESULTS TRAINED WITH DIFFERENT EVENT SIMULATION PIPELINES ON DIFFERENT DATASETS

Method / Dataset	Original E2V [38]			GE (S2R [16])			GE (EGAN [18])			GE (LGAN [20])			GE (Ours)		
	MSE↓	SSIM↑	LPIPS↓	MSE↓	SSIM↑	LPIPS↓	MSE↓	SSIM↑	LPIPS↓	MSE↓	SSIM↑	LPIPS↓	MSE↓	SSIM↑	LPIPS↓
IJRR [52]	.117	.633	.425	.057	.697	**.264**	.069	.613	.325	.071	.656	.311	**.052**	**.704**	.266
MVSEC [53]	.103	.429	.583	.110	.505	.538	.109	.495	.550	.114	.491	.521	**.085**	**.510**	**.455**
EVIMO2v2 [54]	.010	.907	.114	.011	.909	.111	.010	.907	.115	.012	.908	.119	**.008**	**.918**	**.105**
Mean	.077	.656	.374	.059	.704	.304	.063	.672	.330	.066	.685	.317	**.048**	**.711**	**.275**

We utilize the generated event stream (GE) to train the E2V [38] model and assess the performance of the reconstructed image. Best results are highlighted in bold.

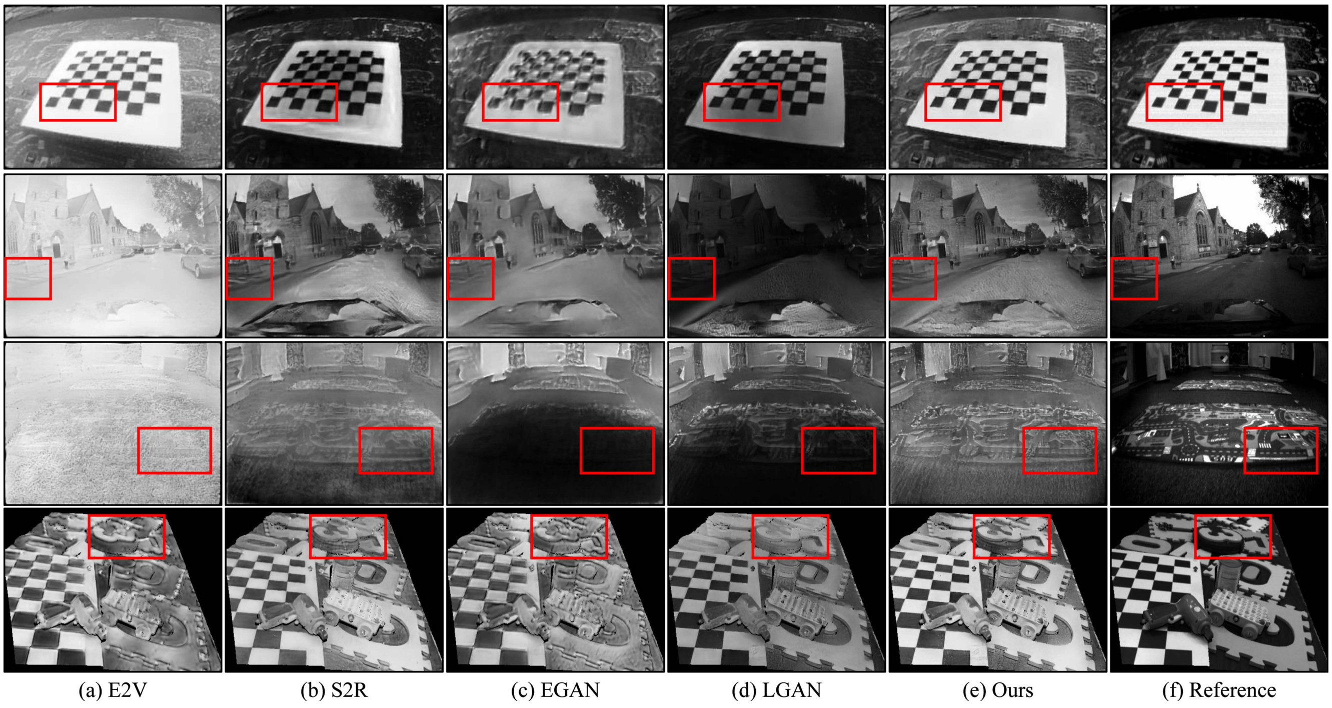

Fig. 12. Video reconstruction results of different approaches on IJRR [52], MVSEC [53], and EVIMO2v2 [54] datasets. The APS images are taken as ground truth. Best viewed with zoom.

TABLE VI
OPTICAL FLOW ESTIMATION RESULTS TRAINED WITH DIFFERENT EVENT SIMULATION PIPELINES ON DIFFERENT DATASETS

Scene	RE	GE (S2R)	GE (LGAN)	GE (EGAN)	GE (Ours)
IJRR [52]	1.32	1.45	1.49	1.43	**1.66**
MVSEC [53]	1.12	1.30	1.34	1.32	**1.36**
EVIMO2v2 [54]	1.02	1.66	1.81	1.71	**1.93**
Mean	1.15	1.47	1.55	1.49	**1.65**

We utilize the raw events (RE) and the generated event stream (GE) to train the EVFLOW [9] model, respectively. FWL metric [16] is utilized to assess the performance of the estimated flow. Best results are highlighted in bold.

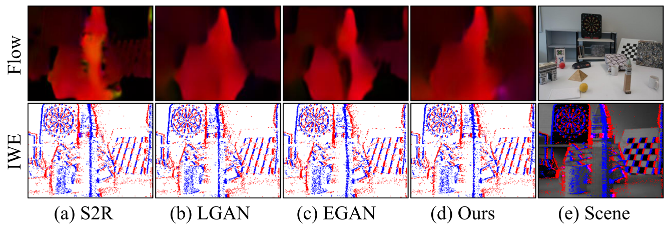

Fig. 13. Optical flow estimation results generated by different approaches. Note that the more precise of optical flow results, the edge thinner for image of warped events (IWE). Best viewed with zoom.

(K values) for each level. The parameter estimation results on our real captured event data are shown in Fig. 14. The results show that our network achieves the best performance with 16 flow steps. To investigate the impact of various flow steps on downstream tasks, we conduct experiments to assess the performance of video reconstruction and flow optical estimation. The results presented in Table VII demonstrate that our network achieves optimal performance when configured with 16 flow steps. With fewer flow steps, the features of the event stream and parameter are difficult to be effectively extracted, thus causing performance degradation. In contrast, more flow steps may lead to gradient disappearance of backpropagation in the training phase since the event stream is usually sparse.

Loss Functions: As shown in Tables VIII and IX, we study the effect of different losses as proposed in Section IV-D. The conditional loss $\mathcal{L}_{\text{I}}$ is essential for supervising our network in the training phase. Except for $\mathcal{L}_{\text{I}}$, we analyze different settings

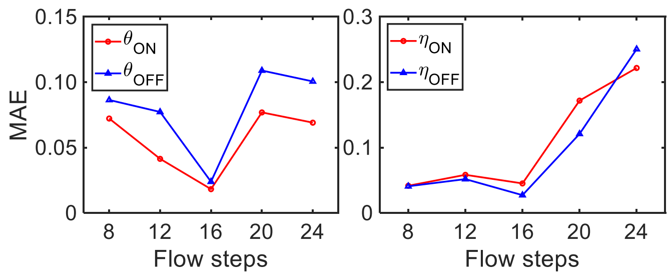

Fig. 14. Parameter estimation performance on different flow steps settings. Best viewed in electronic version.

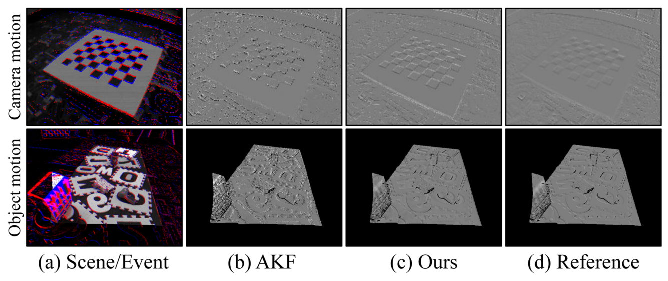

Fig. 15. Qualitative results of intensity change estimation. Best viewed in electronic version.

TABLE VII
ABLATION STUDY OF DIFFERENT FLOW STEPS ON REAL DATASET. BEST RESULTS ARE HIGHLIGHTED IN BOLD

Flow steps	Video reconstruction			Optical flow
	MSE↓	SSIM↑	LPIPS↓	FWL↑
8	0.093	0.630	0.394	1.611
12	0.078	0.676	0.323	1.626
16 (final)	**0.052**	**0.703**	**0.266**	**1.663**
20	0.075	0.680	0.312	1.612
24	0.079	0.672	0.328	1.633

Best results are highlighted in bold.

TABLE VIII
ABLATION STUDY OF LOSS ADAPTION SCHEME ON SYNTHETIC DATASET

Setting	MAE↓			
	θ_{ON}	θ_{OFF}	η_{ON}	η_{OFF}
$\mathcal{L}_{\mathrm{P}}$	.074	.064	.038	.046
$\mathcal{L}_{\mathrm{P}}+\mathcal{L}_{\mathrm{F}}+\mathcal{L}_{\mathrm{F}'}$	.055	.075	**.037**	.054
$\mathcal{L}_{\mathrm{P}}+\mathcal{L}_{\mathrm{E}}$	.037	.073	.054	.048
$\mathcal{L}_{\mathrm{P}}+\mathcal{L}_{\mathrm{F}}+\mathcal{L}_{\mathrm{F}'}+\mathcal{L}_{\mathrm{E}}$	.032	.046	.066	.065
Weight $(\mathcal{L}_{\mathrm{P}},\mathcal{L}_{\mathrm{F}},\mathcal{L}_{\mathrm{F}'},\mathcal{L}_{\mathrm{E}})$	**.018**	**.024**	.045	**.027**

Weight(·) denotes the loss adaption scheme. Best results are highlighted in bold.

TABLE IX
ABLATION STUDY OF LOSS ADAPTION SCHEME ON REAL DATASET

Setting	Video reconstruction			Optical flow
	MSE↓	SSIM↑	LPIPS↓	FWL↑
$\mathcal{L}_{\mathrm{P}}$	0.079	0.665	0.340	1.590
$\mathcal{L}_{\mathrm{P}}+\mathcal{L}_{\mathrm{F}}+\mathcal{L}_{\mathrm{F}'}$	0.074	0.674	0.326	1.599
$\mathcal{L}_{\mathrm{P}}+\mathcal{L}_{\mathrm{E}}$	0.071	0.688	0.305	1.610
$\mathcal{L}_{\mathrm{P}}+\mathcal{L}_{\mathrm{F}}+\mathcal{L}_{\mathrm{F}'}+\mathcal{L}_{\mathrm{E}}$	0.074	0.684	0.313	1.620
Weight $(\mathcal{L}_{\mathrm{P}},\mathcal{L}_{\mathrm{F}},\mathcal{L}_{\mathrm{F}'},\mathcal{L}_{\mathrm{E}})$	**0.052**	**0.703**	**0.266**	**1.663**

Weight(·) denotes the loss adaption scheme. Best results are highlighted in bold.

of $\mathcal{L}_{\mathrm{P}}$, $\mathcal{L}_{\mathrm{F}}$, and $\mathcal{L}_{\mathrm{E}}$. As shown in the table, without the forward flow loss $\mathcal{L}_{\mathrm{F}}$ and the inverse flow loss $\mathcal{L}_{\mathrm{F}'}$, our network cannot converge due to the lack of correct optimization direction. Meanwhile, both $\mathcal{L}_{\mathrm{F}}$ and $\mathcal{L}_{\mathrm{E}}$ can improve the parameter estimation results compared with training network using $\mathcal{L}_{\mathrm{P}}$ alone. By introducing the proposed losses, the performance of our network can be further boosted. Furthermore, the results presented in the last row of Table VIII and IX demonstrate that our adaptation scheme (as described in Section IV-D3) effectively enhances the performance of event generation across various tasks. The introduction of weight adaptation also prevents gradient explosion in practical applications.

2) The Necessity of Intensity Change Estimation: In Table XI, we compare the intensity change estimation results of our proposed conditional encoder network with the recent works (i.e., AKF [56]) on IJRR [52] and EVIMO2v2 [54] datasets. We utilize the difference between adjacent APS frames as reference images to measure the accuracy of intensity change estimation. To achieve this, we compare the results obtained from both the AKF algorithm and our network with these reference images and subsequently calculate the mean absolute error (MAE). The visualization results are shown in Fig. 15. In camera motion scenes from IJRR dataset, the events occur across the whole screen, posing a challenge in distinguishing noise events from those triggered by actual intensity changes. In object motion scenes from EVIMO2v2 dataset, the events mainly occur in localized regions. In these areas, the potential for significant errors in intensity change estimation is high. However, the visual comparison results demonstrate that our conditional encoder network exhibits robustness to event noise and performs well even under conditions involving numerous events. Our condition encoder network can effectively extract high-quality scene intensity information, which helps guide the bidirectional mapping of our invertible network.

3) The Event Representation With Various Parameters: Benefiting from the reversible property of our invertible network, the reliable event streams of diverse scenes can be effectively generated. In fact, our invertible network builds a bidirectional mapping between the event representation and event parameters. Our network can generate controllable event representations based on the inverse mapping from the event parameters to representations. Fig. 16 shows the generated event representations with different parameters. The original event representation and the representation generated with our estimated parameters are shown in Fig. 16(a) and (d), respectively. By adjusting positive contrast thresholds $+\theta_{\mathrm{ON}}$, negative contrast thresholds $-\theta_{\mathrm{OFF}}$, and the contrast threshold $-\theta_{\mathrm{ON}}$ and $+\theta_{\mathrm{OFF}}$, we can obtain different event representations. The results show that our invertible network has the potential to flexibly and directly generate event representation according to different scenarios. Furthermore, this experiment provides a visual demonstration of the ability to generate events that align with the target domain by inversely estimating the parameters within the framework. It illustrates that manual adjustments to parameters like threshold and noise can be utilized to modify the characteristics of the generated events. These findings affirm that our method excels not only

TABLE X
GENERALIZATION CAPABILITY ON DIFFERENT EVENT CAMERAS

Method / Camera type	ECC [22] RMSE↓	ECC [22] PSNR↑	ECC [22] SSIM↑	S2R [16] RMSE↓	S2R [16] PSNR↑	S2R [16] SSIM↑	JAER [21] RMSE↓	JAER [21] PSNR↑	JAER [21] SSIM↑	LGAN [20] RMSE↓	LGAN [20] PSNR↑	LGAN [20] SSIM↑	Ours RMSE↓	Ours PSNR↑	Ours SSIM↑
DAVIS240	37.56	16.82	0.41	32.37	18.10	**0.45**	35.10	17.39	0.42	32.30	18.15	0.42	**32.24**	**18.17**	0.42
DAVIS346	54.81	13.75	0.33	50.55	14.45	0.35	57.72	13.42	0.34	45.73	15.29	0.36	**39.76**	**16.37**	**0.37**
CeleX5	15.34	25.59	0.88	15.67	25.61	0.91	13.52	26.58	0.92	10.51	28.41	0.94	**8.12**	**30.78**	**0.95**
Mean	35.90	18.72	0.54	32.86	19.39	0.57	35.45	19.13	0.56	29.51	20.62	0.57	**26.71**	**21.77**	**0.58**

We utilize the estimated event parameters with the corresponding event streams to generate the direct integrated image [22] and assess the performance of the generated image. Best results are highlighted in bold.

TABLE XI
THE QUANTITATIVE RESULTS OF CONDITIONAL INTENSITY CHANGE ENCODER

Dataset	Scene	AKF [56]	Ours
IJRR [52]	Camera motion	0.10	**0.06**
EVIMO2v2 [54]	Object motion	0.18	**0.13**

Best results are highlighted in bold. More details can be found in the supplementary materials.

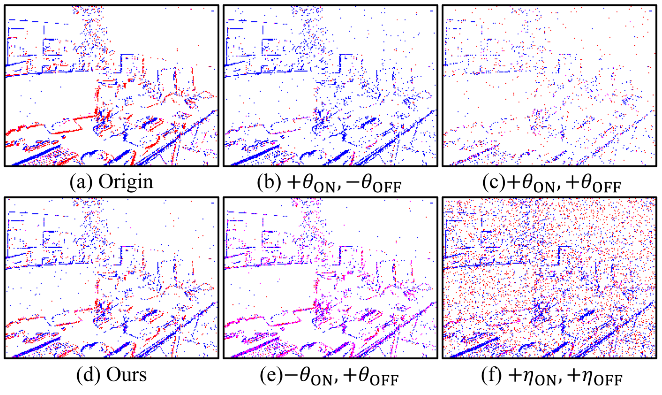
(a) Origin
(b) $+\theta_{\mathrm{ON}}, -\theta_{\mathrm{OFF}}$
(c)$+\theta_{\mathrm{ON}}, +\theta_{\mathrm{OFF}}$
(d) Ours
(e)$-\theta_{\mathrm{ON}}, +\theta_{\mathrm{OFF}}$
(f) $+\eta_{\mathrm{ON}}, +\eta_{\mathrm{OFF}}$

Fig. 16. Event representation generation results with different parameter settings. Best viewed in electronic version.

in event generation but also holds great potential for various applications, including event representation, event denoising, and related tasks.

4) Generalization Capability on Different Event Cameras: To evaluate the generalization capability of our proposed method across different event cameras, we conduct experiments using real datasets captured with various event cameras: DAVIS240 [2], DAVIS346 [25], and CeleX5 [24]. The sequences from DAVIS240 [2] and DAVIS346 [25] are obtained from publicly available datasets, namely IJRR [52] and MVSEC [53], respectively. Additionally, we capture event streams using the CeleX5 [24] camera specifically for evaluation purposes. We follow [22] to utilize direct integrated images to evaluate the per-pixel contrast threshold calibration performance. This involves comparing the estimated intensity images with the referenced ground truth intensity image. We compare our framework against four other methods: ECC [22], S2R [16], JAER [21], and LGAN [20]. The results of these experiments are presented in Table X, clearly demonstrating that our proposed approach outperforms the other methods across all three camera types. This finding highlights the superior generalization ability of our network when applied to different event cameras.

VI. CONCLUSION

We have presented a novel event-based invertible neural network for reliable event generation. By introducing the conditional normalizing flow, the proposed network can be trained end-to-end to learn a bidirectional mapping between event space and parameter space. Our network is totally invertible, which can learn the inherent relationship of events and parameters. To take the advantage of the intensity prior, we propose an intensity-guided conditional affine simulation, which can lead to a better alignment of the features. Moreover, multiple losses, such as event representation loss and flow losses, are proposed to ensure the network can be effectively trained in both forward and inverse directions. Extensive experiments demonstrated that our invertible network is extremely helpful to the event generation task. We further show that the proposed framework generalizes well to multiple scenarios and event cameras without re-training or fine-tuning, while performing significantly better than recent state-of-the-art methods. Overall, it is promising to study the correlations between the event distribution and its parameters to generate reliable event stream. Moreover, extending the invertible network to other event-based vision tasks (e.g., event representation estimation, intensity reconstruction, and event denoising) is also an interesting direction worthy to be explored.

REFERENCES

[1] P. Lichtsteiner, C. Posch, and T. Delbrück, "A 128x128 120 db 15 s latency asynchronous temporal contrast vision sensor," *IEEE J. Solid State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008μ.
[2] C. Brandli, R. Berner, M. Yang, S. Liu, and T. Delbrück, "A 240 x 180 130 dB 3 μs latency global shutter spatiotemporal vision sensor," *IEEE J. Solid State Circuits*, vol. 49, no. 10, pp. 2333–2341, Oct. 2014.
[3] H. Kim, A. Handa, R. Benosman, S. Ieng, and A. J. Davison, "Simultaneous mosaicing and tracking with an event camera," in *Proc. Brit. Mach. Vis. Conf.*, 2014, pp. 1–12.
[4] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza, "High speed and high dynamic range video with an event camera," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 6, pp. 1964–1980, Jun. 2021.
[5] C. Scheerlinck, N. Barnes, and R. E. Mahony, "Asynchronous spatial image convolutions for event cameras," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 816–822, Apr. 2019.
[6] C. Scheerlinck, H. Rebecq, D. Gehrig, N. Barnes, R. E. Mahony, and D. Scaramuzza, "Fast image reconstruction with an event camera," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2020, pp. 156–163.
[7] M. Almatrafi and K. Hirakawa, "Davis camera optical flow," *IEEE Trans. Comput. Imag.*, vol. 6, pp. 396–407, 2020.

[8] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "Unsupervised event-based learning of optical flow, depth, and egomotion," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 989–997.

[9] A. Z. Zhu, L. Yuan, K. Daniilidis, and K. Chaney, "EV-FlowNet: Self-supervised optical flow estimation for event-based cameras," *Robot.: Sci. Syst.*, pp. 1–9, 2018.

[10] C. Ye, A. Mitrokhin, C. Fermüller, J. A. Yorke, and Y. Aloimonos, "Unsupervised learning of dense optical flow, depth and egomotion with event-based sensors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5831–5838.

[11] I. Alonso and A. C. Murillo, "EV-SegNet: Semantic segmentation for event-based cameras," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1624–1633.

[12] J. Zhang, K. Yang, and R. Stiefelhagen, "ISSAFE: Improving semantic segmentation in accidents by fusing event-based data," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 1132–1139.

[13] H. Rebecq, D. Gehrig, and D. Scaramuzza, "ESIM: An open event camera simulator," in *Proc. 2nd Conf. Robot Learn.*, 2018, pp. 969–982.

[14] Y. Hu, S. Liu, and T. Delbrück, "V2E: From video frames to realistic DVS events," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2021, pp. 1312–1321.

[15] D. Gehrig, M. Gehrig, J. Hidalgo-Carrió, and D. Scaramuzza, "Video to events: Recycling video datasets for event cameras," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 3583–3592.

[16] T. Stoffregen et al., "Reducing the sim-to-real gap for event cameras," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 534–549.

[17] M. Planamente et al., "DA4Event: Towards bridging the sim-to-real gap for event cameras using domain adaptation," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 6616–6623, Oct. 2021.

[18] A. Z. Zhu, Z. Wang, K. Khant, and K. Daniilidis, "EventGAN: Leveraging large scale image datasets for event cameras," in *Proc. IEEE Int. Conf. Comput. Photography*, 2021, pp. 1–11.

[19] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2242–2251.

[20] D. Gu, J. Li, Y. Zhang, and Y. Tian, "How to learn a domain-adaptive event simulator?," in *Proc. 29th ACM Int. Conf. Multimedia*, 2021, pp. 1275–1283.

[21] C. Brandli, L. Müller, and T. Delbrück, "Real-time, high-speed video decompression using a frame- and event-based DAVIS sensor," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2014, pp. 686–689.

[22] Z. Wang, Y. Ng, P. van Goor, and R. E. Mahony, "Event camera calibration of per-pixel biased contrast threshold," 2020, *arXiv: 2012.09378*.

[23] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 10 236–10 245.

[24] S. Chen and M. Guo, "Live demonstration: Celex-v: A 1m pixel multi-mode event-based sensor," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2019, pp. 1682–1683.

[25] G. Taverni et al., "Front and back illuminated dynamic and active pixel vision sensors comparison," *IEEE Trans. Circuits Syst. II Exp. Briefs*, vol. 65-II, no. 5, pp. 677–681, May 2018.

[26] J. Kaiser et al., "Towards a framework for end-to-end control of a simulated vehicle with spiking neural networks," in *Proc. IEEE Int. Conf. Simul. Model. Program. Auton. Robots*, 2016, pp. 127–134.

[27] Y. Bi and Y. Andreopoulos, "PIX2NVS: Parameterized conversion of pixel-domain video frames to neuromorphic vision streams," in *Proc. IEEE Int. Conf. Image Process.*, 2017, pp. 1990–1994.

[28] P. Duan, Z. W. Wang, X. Zhou, Y. Ma, and B. Shi, "EventZoom: Learning to denoise and super resolve neuromorphic events," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 12 824–12 833.

[29] A. Lakshmi, V. Ramanathan, and C. S. Thakur, "Event-LSTM: An unsupervised and asynchronous learning-based representation for event-based data," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 4678–4685, Apr. 2022.

[30] S. Lin, Y. Ma, Z. Guo, and B. Wen, "DVS-voltmeter: Stochastic process-based event simulator for dynamic vision sensors," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 578–593.

[31] G. Gallego et al., "Event-based vision: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 154–180, Jan. 2022.

[32] T. Finateu, A. Niwa, D. Matolin, K. Tsuchimoto, and C. Posch, "5.10 A 1280x720 back-illuminated stacked temporal contrast event-based vision sensor with 4.86m pixels, 1.066GEPS readout, programmable event-rate controller and compressive data-formatting pipeline," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2020, pp. 112–114.

[33] Y. Suh, S. Choi, M. Ito, J. Kim, and Y. Park, "A 1280x960 dynamic vision sensor with a 4.95-m pixel pitch and motion artifact minimization," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2020, pp. 1–5.

[34] C. Li et al., "Design of an RGBW color VGA rolling and global shutter dynamic and active-pixel vision sensor," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2015, pp. 718–721.

[35] D. P. Moeys et al., "Color temporal contrast sensitivity in dynamic vision sensors," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2017, pp. 1–4.

[36] C. Scheerlinck, H. Rebecq, T. Stoffregen, N. Barnes, R. E. Mahony, and D. Scaramuzza, "CED: Color event camera dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1684–1693.

[37] M. Guo, J. Huang, and S. Chen, "Live demonstration: A 768 x 640 pixels 200Meps dynamic vision sensor," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2017, pp. 1–1.

[38] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza, "Events-to-video: Bringing modern computer vision to event cameras," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3857–3866.

[39] L. Wang, S. M. M.I. Y. Ho, and K. Yoon, "Event-based high dynamic range image and very high frame rate video generation using conditional generative adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10 081–10 090.

[40] L. Dinh, D. Krueger, and Y. Bengio, "NICE: Non-linear independent components estimation," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–12.

[41] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–10.

[42] A. Pumarola, S. Popov, F. Moreno-Noguer, and V. Ferrari, "C-flow: Conditional generative flow models for images and 3D point clouds," in *Proc. IEEE/CVF IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 7946–7955.

[43] G. Yang, X. Huang, Z. Hao, M. Y. Liu, and B. Hariharan, "PointFlow: 3D point cloud generation with continuous normalizing flows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 4540–4549.

[44] A. Lugmayr, M. Danelljan, L. V. Gool, and R. Timofte, "SRFlow: Learning the super-resolution space with normalizing flow," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 715–732.

[45] H. Li, J. Li, D. Zhao, and L. Xu, "DehazeFlow: Multi-scale conditional flow network for single image dehazing," in *Proc. 29th ACM Int. Conf. Multimedia*, 2021, pp. 2577–2585.

[46] Y. Nozaki and T. Delbruck, "Temperature and parasitic photocurrent effects in dynamic vision sensors," *IEEE Tran. Electron Devices*, vol. 64, no. 8, pp. 3239–3245, Aug. 2017.

[47] T. Delbruck, R. Graca, and M. Paluch, "Feedback control of event cameras," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2021, pp. 1324–1332.

[48] X. Li et al., "Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 21002–21012.

[49] X. Li, W. Wang, X. Hu, J. Li, J. Tang, and J. Yang, "Generalized focal loss V2: Learning reliable localization quality estimation for dense object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 11 632–11 641.

[50] Y. Romano, J. Isidoro, and P. Milanfar, "RAISR: Rapid and accurate image super resolution," *IEEE Trans. Comput. Imag.*, vol. 3, no. 1, pp. 110–125, Mar. 2017.

[51] T. Lin et al., "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

[52] E. Mueggler, H. Rebecq, G. Gallego, T. Delbrück, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM," *Int. J. Robot. Res.*, vol. 36, no. 2, pp. 142–149, 2017.

[53] A. Z. Zhu, D. Thakur, T. Özaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, "The multivehicle stereo event camera dataset: An event camera dataset for 3D perception," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2032–2039, Jul. 2018.

[54] L. Burner, A. Mitrokhin, C. Ye, C. Fermüller, Y. Aloimonos, and T. Delbruck, "EVIMO2: An event camera dataset for motion segmentation, optical flow, structure from motion, and visual inertial odometry in indoor scenes with monocular or stereo algorithms," 2022.

[55] L. Zhang and S. Rusinkiewicz, "Learning to detect features in texture images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6325–6333.

[56] Wang, Ziwei and Ng, Yonhon and Scheerlinck, Cedric and Mahony, Robert, "An asynchronous Kalman filter for hybrid event cameras," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 448–457.

Daxin Gu received the BE degree from Beihang University, Beijing, China, in 2018. He is currently working toward the PhD degree with the State Key Laboratory of Virtual Reality Technology and System, School of Computer Science and Engineering, Beihang University. His research interests include computer vision and image understanding.

Jia Li (Senior Member, IEEE) received the BE degree from Tsinghua University, in 2005, and the PhD degree from the Institute of Computing Technology, Chinese Academy of Sciences, in 2011. He is currently a full professor with the State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University. Before he joined Beihang University in 2014, he used to work with Nanyang Technological University, Shanda Innovations, and Peking University. His research is focused on computer vision, multimedia and artificial intelligence, especially the visual computing in extreme environments. He has co-authored more than 120 articles in peer-reviewed top-tier journals and conferences. He also has one Monograph published by Springer and more than 70 patents issued from U.S. and China. He is a fellow of IET, a distinguished member of CCF, and senior members of IEEE/ACM/CCF/CIE.

Lin Zhu received the BS degree in computer science from the Northwestern Polytechnical University, China, in 2014, the MS degree in computer science from the North Automatic Control Technology Institute, China, in 2018, and the PhD degree with the School of Electronics Engineering and Computer Science, Peking University, China, in 2022. He is currently an assistant professor with the School of Computer Science, Beijing Institute of Technology, China. His current research interests include neuromorphic computing and spiking neural network.

Yu Zhang received the BE degree and PhD degrees from the School of Computer Science and Engineering, Beihang University, in 2012 and 2018, respectively. He is currently a researcher with SenseTime Research, Beijing China. His research interests include computer vision and image/video processing.

Jimmy S. Ren received the PhD degree from the City University of Hong Kong, in 2013. He is currently a senior research director with SenseTime where he leads a team to build high impact computational photography products. He also holds an adjunct faculty position in Qing Yuan Research Institute, Shanghai Jiao Tong University. His research interests include computational photography, image processing, and computer vision.