Learning Optical Flow from Event Camera with Rendered Dataset

Xinglong Luo[1,3], Kunming Luo[2], Ao Luo[3], Zhengning Wang[1], Ping Tan[2], and Shuaicheng Liu[1,3*]

[1]University of Electronic Science and Technology of China
[2]The Hong Kong University of Science and Technology [3]Megvii Technology
{luoboom@std.,zhengning.wang@,liushuaicheng@}uestc.edu.cn
{kluoad@connect,pingtan@}.ust.hk luoao02@megvii.com

Abstract

We study the problem of estimating optical flow from event cameras. One important issue is how to build a high-quality event-flow dataset with accurate event values and flow labels. Previous datasets are created by either capturing real scenes by event cameras or synthesizing from images with pasted foreground objects. The former case can produce real event values but with calculated flow labels, which are sparse and inaccurate. The latter case can generate dense flow labels but the interpolated events are prone to errors. In this work, we propose to render a physically correct event-flow dataset using computer graphics models. In particular, we first create indoor and outdoor 3D scenes by Blender with rich scene content variations. Second, diverse camera motions are included for the virtual capturing, producing images and accurate flow labels. Third, we render high-framerate videos between images for accurate events. The rendered dataset can adjust the density of events, based on which we further introduce an adaptive density module (ADM). Experiments show that our proposed dataset can facilitate event-flow learning, whereas previous approaches when trained on our dataset can improve their performances constantly by a relatively large margin. In addition, event-flow pipelines when equipped with our ADM can further improve performances. Our code is available at https://github.com/boomluo02/ADMFlow.

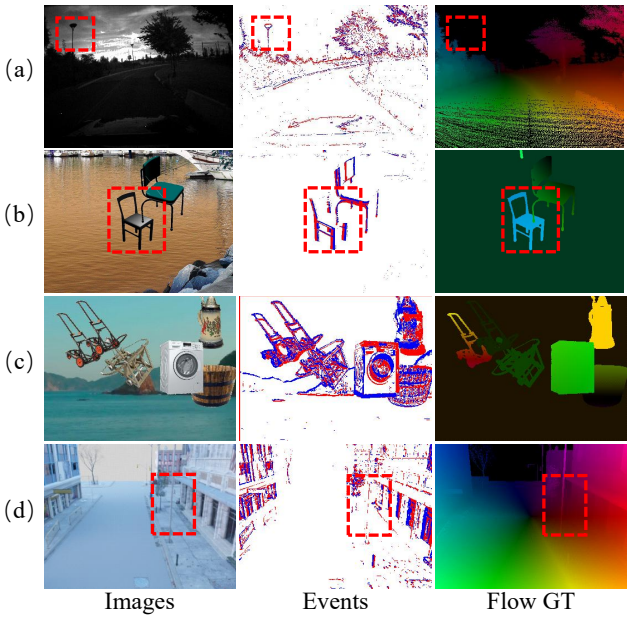

Figure 1: (a) the captured dataset from real event camera [55, 54]. (b) the synthesized dataset with flying chairs foreground [50]. (c) the synthesized dataset by moving a foreground image [42]. (d) Our synthesized dataset by graphics rendering, which not only reflects the real motions under correct scene geometries, but also produces accurate dense flow labels and events.

1. Introduction

Event cameras [29] record brightness changes at a varying framerate [4]. When a change is detected in a pixel, the camera returns an event in the form $e = (x, y, t, p)$ immediately, where x, y stands for the spatial location, t refers to the timestamp in microseconds, and p is the polarity of the change, indicating a pixel become brighter or darker. On the other hand, optical flow estimation predicts motions between two frames [44], which is fundamental and important for many applications [49, 52, 7]. In this work, we study the problem of estimating optical flow from event camera data, instead of from RGB frames. Different from traditional images, events are sparse and are often integrated in short intervals as the input for the prediction. As such, early works

*Corresponding author

can only estimate sparse flows at the location of events [1]. Recent deep methods can estimate dense flows but with the help of images, either as the guidance [54] or as the additional inputs [26, 38]. Here, we tackle a hard version of the problem, where dense flows are predicted based purely on the event values e. One key issue is how to create high quality event-based optical flow dataset to train the network.

Existing methods of event flow dataset creation can be classified into two types, 1) directly capturing from real event cameras [55, 54, 16]; 2) moving foregrounds on top of a background image to create synthesized flow motions [50, 42] and apply frame interpolation [14] to create events. For the first type, the ground-truth (GT) flow labels need to be calculated based on gray images acquired along with the event data. However, the optical flow estimations cannot be perfectly accurate [48, 43, 32, 24, 51], leading to the inaccuracy of GT labels. To alleviate the problem, additional depth sensors, such as LIDAR, have been introduced [55]. The flow labels can be calculated accurately when the depth values of LIDAR scans are available. However, LIDAR scans are sparse, and so do the flow labels, which are unfriendly for dense optical flow learning. Fig. 1 (a) shows an example, LIDAR points on the ground are sparse. Moreover, some thin objects are often missing, as indicated by the red box in Fig. 1 (a).

For the second category, the flow labels are created by moving foreground objects on top of a background image, similar to flying chairs [12] or flying things [36]. In this way, the flow labels are dense and accurate. To create events, intermediate frames are interpolated [23]. However, the frame interpolation is inaccurate due to scene depth disparities, where the occluded pixels cannot be interpolated correctly, leading to erroneous event values in these regions. To match high framerate of events, the large number of interpolated frames makes the problem even worse. Fig. 1 (b) shows an example, where the events are incorrect at the occluded chairs. Moreover, the motions are artificial, further decreasing the realism of the dataset (Fig. 1 (b) and (c)).

In this work, we create an event-flow dataset from synthetic 3D scenes by graphics rendering (Fig. 1 (d)). While there is a domain gap between rendered and real images, this gap is empirically found insignificant in event camera based classification [41] and segmentation [14] tasks. As noted by these works, models trained on synthetic events work very well for real event data. Because events contain only positive and negative polarities, no image appearances are involved. To this end, we propose a **M**ulti-**D**ensity **R**endered (MDR) event optical flow dataset, created by Blender on indoor and outdoor scenes with accurate events and flow labels. In addition, we design an **A**daptive **D**ensity **M**odule (ADM) based on MDR, which can adjust the densities of events, one of the most important factors for event-based tasks but has been largely overlooked.

Specifically, our MDR dataset contains $80,000$ samples from 50 virtual scenes. Each data sample is created by first rendering two frames and obtaining the GT flow labels directly from the engine. Then, we render $15 \sim 60$ frames in-between based on the flow magnitude. The events are created by thresholding log intensities and recording the timestamp for each spatial location. The density of events can be controlled by the threshold values. The ADM is designed as a plugin module, which further consists of two sub-modules, multi-density changer (MDC) and multi-density selector (MDS), where the MDC adjusts the density globally while the MDS picks the best one for every spatial location. Experiments show that previous event-flow methods, when trained on our MDR dataset, can improve their performances. Moreover, we train several recent representative flow pipelines, such as FlowFormer [21], KPA-Flow [32], GMA [24] and SKFlow [47], on our MDR dataset. When equipped with our ADM module, the performances can increase consistently.

Our contributions are summarized as:

- A rendered event-flow dataset MDR, with 80,000 samples created on 53 virtual scenes, which possess physically correct accurate events and flow label pairs, covering a wide range of densities.
- An adaptive density module (ADM), which is a plug-and-play module for handling varying event densities.
- We achieve state-of-the-art performances. Our MDR can improve the quality of previous event-flow methods. Various optical flow pipelines when adapted to the event-flow task, can benefit from our ADM module.

2. Related Work

2.1. Image-based Optical Flow

Optical flow estimates the per-pixel motion between two frames according to photo consistency. Traditional approaches minimize energies, leveraging both feature similarities and motion smoothness [13]. Deep methods train the networks that take two frames as input and directly output dense flow motions [12, 46]. Recent deep methods design different pipelines [30, 48, 24] as well as learning modules [35, 31, 34, 33] for performance improvements. The training often requires large labeled datasets, which can be synthesized by moving a foreground on top of a background image, such as FlyingChairs [12], and AutoFlow [45], or rendered from graphics such as Sintel [5] and FlyingThings [36], or created directly from real videos [19]. In this work, we use computer graphics techniques to render accurate and physically correct event and flow values.

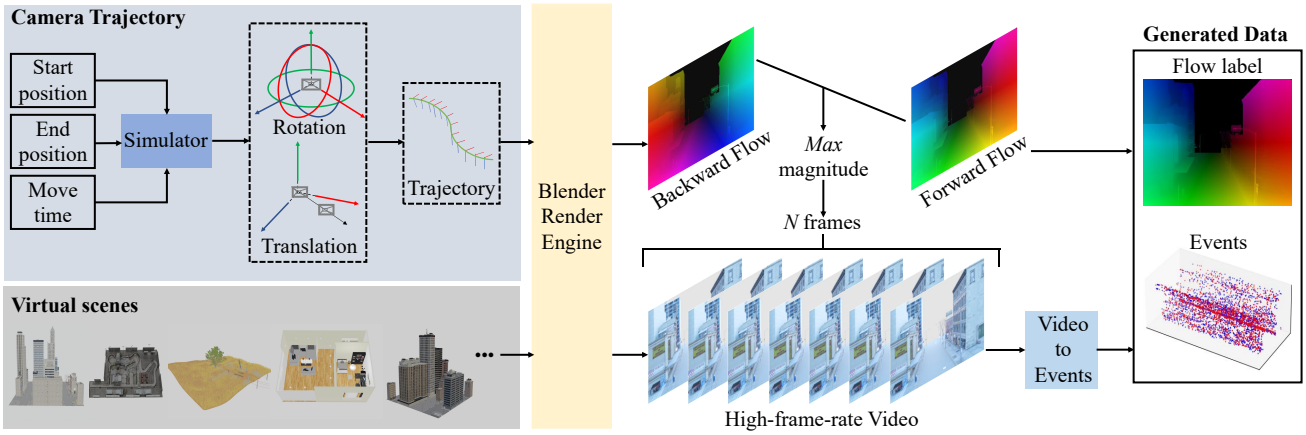

Figure 2: Our data generation pipeline. Given 3D scenes in the graphics engine, we generate camera trajectories and render high-frame-rate videos with forward and backward optical flow labels. Then, we build the event optical flow dataset by generating events using the videos.

2.2. Event-based Optical Flow

Benosman *et al.* [1] first proposed to estimate optical flow from events, which can only estimate sparse flows at the location of event values. Recent deep methods can estimate dense optical flows. EV-FlowNet [54] learns the event and flow labels in a self-supervised manner, which minimizes the photometric distances of grey images acquired by DAVIS [4]. Different event representations were explored, e.g., EST [15] and Matrix-LSTM [6], with various network structures, such as SpikeFlowNet [27], LIF-EV-FlowNet [18], STE-FlowNet [11], Li *et al.* [28], and E-RAFT [17]. Some works take both events and images as input for the flow estimation [26, 38]. In general, dense flows are more desirable than sparse ones but are more difficult to train. Normally, regions with events can produce more accurate flows than empty regions where no events are triggered. Moreover, supervised training can produce better results than unsupervised ones, as long as the training dataset can provide sufficient event-flow guidance.

2.3. Event Dataset

The applications to the event camera dataset were first explored in the context of classification [37, 2]. Early works generate events simply by applying a threshold on the image difference [25]. Frame interpolation is often adopted for high framerate [14]. The synthesized events often contain inaccurate timestamps. The DAVIS event camera can directly capture both images and events [4], based on which two driving datasets are captured, DDD17 [3] and MVSEC [55]. With respect to the event flow dataset, EV-Flownet [54] calculated sparse flow labels from LIDAR depth based on MVSEC [55]. Wan *et al.* [50] and Stoffregen *et al.* [42] created the foreground motions and interpolated the intermediate frames for events. The real captured dataset can only provide sparse labels while the synthesized ones contain inaccurate events. In this work, we propose to render a physically correct event dense flow dataset.

3. Method

3.1. The event-based optical flow dataset

In order to create a realistic event dataset for optical flow learning, we propose to employ a graphics engine with 3D scenes for data generation. Given 3D scenes, we first define camera trajectories, according to which we generate optical flow labels for timestamps at 60 FPS and 15 FPS. Then we render high-frame-rate videos based on the motion magnitude of the optical flow label between two timestamps. Finally, we generate event streams by rendering high-frame-rate videos and simulating the event trigger mechanism in the event camera using the v2e toolbox [20]. The overview of our data generation pipeline is shown in Fig. 2.

Virtual Scenes. To ensure that the generated event dataset has the correct scene structure, we utilize a variety of indoor and outdoor 3D scenes, including cities, streets, forests, ports, beaches, living rooms, bedrooms, bathrooms, kitchens, and parking lots. Totally, we obtain 53 virtual 3D scenes (31 indoor and 22 outdoor) that simulate real-world environments. Some examples are shown in Fig. 3.

Camera Trajectory. Given a 3D scene model, we first generate the 3D camera trajectory using PyBullet [8], an open-source physics engine, to ensure that the camera does not pass through the inside of the objects and out of the effective visible region of the scene during the motion. After setting the start position, end position, and moving speed of

Figure 3: Examples of our MDR training set. Each row shows images, events and flow labels from top to bottom.

the camera trajectory, we randomly add translation and rotation motions to create a smooth curve function $\Gamma(t)$ that outputs the location and pose $P(t) = [x(t), y(t), z(t), r(t)]^T$.

High-frame-rate Video and Optical Flow. After camera trajectory generation, we use the graphics engine to render a sequence of images $I(\boldsymbol{u}, t_i)$, where $\boldsymbol{u} = (x, y)$ is the pixel coordinates and t_i is the timestamp. We extract the forward and backward optical flow labels between every two timestamps ($\boldsymbol{F}_{t_i \to t_j}$, $\boldsymbol{F}_{t_j \to t_i}$). Then we need to generate the event data to construct the event optical flow dataset. Here, we render high-frame-rate videos $\{I(\boldsymbol{u}, \tau)\}, \tau \in [t_i, t_j]$ between timestamps t_i and t_j for events generation according to the camera trajectory $\Gamma(t)$.

Inspired by ESIM [40], we adopt an adaptive sampling strategy to sample camera locations from the camera trajectory for interval t_i to t_j, so that the largest displacement of all pixels between two successive rendered frames ($I(\boldsymbol{u}, \tau_k)$, $I(\boldsymbol{u}, \tau_{k+1})$) is under 1 pixel, we define the sampling time interval $\Delta\tau_k$ as follows:

$$\begin{aligned} \Delta\tau_k &= \tau_{k+1} - \tau_k \\ &= (\max_{\boldsymbol{u}} \max\{\left\|\boldsymbol{F}_{\tau_{k-1} \to \tau_k}\right\|, \left\|\boldsymbol{F}_{\tau_k \to \tau_{k-1}}\right\|\})^{-1}, \end{aligned} \tag{1}$$

where $|F| = \max_{\boldsymbol{u}} \max\{\left\|\boldsymbol{F}_{\tau_{k-1} \to \tau_k}\right\|, \left\|\boldsymbol{F}_{\tau_k \to \tau_{k-1}}\right\|\}$ is the maximum magnitude of the motion field between images $I(\boldsymbol{u}, \tau_{k-1})$ and $I(\boldsymbol{u}, \tau_k)$.

Event Generation from High-frame-rate Video. Given a high-frame-rate video $\{I(\boldsymbol{u}, \tau)\}, \tau \in [t_i, t_j]$ between timestamps t_i and t_j, we next generate event stream by simulating the event trigger mechanism. Similar to [40] and [14], we use linear interpolation to approximate the continuous intensity signal in time for each pixel between video frames. Events $\{(\boldsymbol{u_e}, t_e, p_e)\}$ are generated at each pixel $\boldsymbol{u_e} = (x_e, y_e)$ whenever the magnitude of the change in the log intensity values ($L(\boldsymbol{u_e}, t_e) = \ln(I(\boldsymbol{u_e}, t_e))$ exceeds the threshold C. This can be expressed as Eq. (2) and Eq. (3):

$$L(\boldsymbol{u_e}, t_e + \Delta t_e) - L(\boldsymbol{u_e}, t_e) \geq p_e C, \tag{2}$$

$$t_e = t_{e-1} + \Delta\tau_k \frac{C}{|L(\boldsymbol{u_e}, t_e + \Delta t_e) - L(\boldsymbol{u_e}, t_e)|}, \tag{3}$$

where t_{e-1} and t_e are the timestamps of the last triggered event and the next triggered event respectively, $p_e \in \{-1, +1\}$ is the polarity of the triggered event. We define it as $E(t_k, t_{k+1})$, witch is the sequence $\{(\boldsymbol{u_e}, t_e, p_e)^N, e \in [0, N]\}$ with N events between time t_k and t_{k+1}.

Multi-Density Rendered Events Dataset. Using the above data generation method, we can generate data with different event densities by using different threshold values C. Since event stream is commonly first transformed into event representation [56, 42, 17] and then fed into deep networks. In order to measure the amount of useful information carried by the event stream, we propose to calculate the density of the event stream using the percentage of valid pixels (pixels where at least one event is triggered) in the voxel representation:

$$V(\boldsymbol{u_e}, b) = \sum_{e=0}^{N} p_e \max(0, 1 - |b - \frac{t_e - t_0}{t_N - t_0}(B-1)|), \tag{4}$$

$$D = \frac{1}{HW} \sum_{e=1}^{N} \varepsilon(\sum_{b=0}^{B-1} |V(\boldsymbol{u_e}, b)|), \varepsilon(x) = \begin{cases} 1, x > 0 \\ 0, x \leq 0 \end{cases}, \tag{5}$$

where $V(\boldsymbol{u_e}) \in \mathbb{R}^{B \times H \times W}$ is the voxel representation [56] of the event stream $\{(\boldsymbol{u_e}, t_e, p_e)^N\}$ between t_0 and t_N, $b \in [0, B-1]$ indicates the temporal index , B (typically set to 5) donates temporal bins and D is the density of the input event representation V.The notation $|\cdot|$ denotes the absolute value operation. In practical applications, different event cameras may use different threshold values in different scenes, resulting in data with different event densities. Intuitively, event data with lower density is more difficult for optical flow estimation. In order to train models that can cover event data with various densities, in this paper, we propose to adaptively normalize the density of the input events to a certain density representation for optical flow estimation, so as to increase the generalization ability of the network.

3.2. Event-based Optical Flow Estimation

Event-based optical flow estimation involves predicting dense optical flow $\boldsymbol{F}_{k-1 \to k}$ from consecutive event sequences $E(t_{k-1}, t_k)$ and $E(t_k, t_{k+1})$. In this paper, we find that networks perform better on event sequences with appropriate density than on those with excessively sparse or

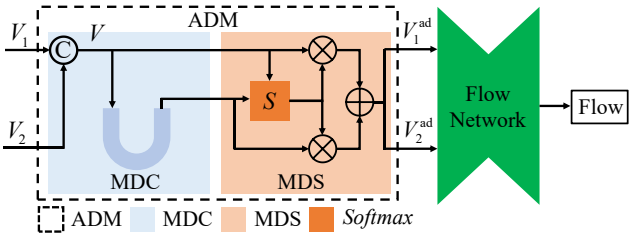

Figure 4: The structure of the proposed network. We design a plug-and-play Adaptive Density Module (ADM) to transform input event representations V_1 and V_2 into V_1^{ad} and V_2^{ad} with suitable density for optical flow estimation.

dense events, when given events from the same scene. Motivated by this, we propose a plug-and-play Adaptive Density Module (ADM) that normalizes the input event stream to a density suited for estimating optical flow. Our network architecture is shown in Fig. 4, where the ADM transforms input event representations V_1 and V_2 to justified event representations V_1^{ad} and V_2^{ad}, which are then used by an existing network structure to estimate optical flow.

Adaptive Density Module. As shown in Fig. 4, our ADM module consists of two sub-modules: the multi-density changer (MDC) module and the multi-density selector (MDS) module. The MDC module globally adjusts the density of the input event representations from multi-scale features, then the MDS module picks the best pixel-wise density for optical flow estimation.

The MDC module adopts an encoder-decoder architecture with three levels, as illustrated in Fig. 5(a). To generate multiscale transformed representations V_3^{MDC}, V_2^{MDC} and V_1^{MDC} (also noted as $V_{\text{out}}^{\text{MDC}}$) from the concatenated input event representations V, three encoding blocks are employed to extract multiple scale features, followed by three decoding blocks and two feature fusion blocks. It is worth noting that, to ensure the lightweightness of the entire module, we utilize only two 3×3 and one 1×1 convolutional layers in each encoding and decoding block.

To maintain the information in the input event representation and achieve density transformation, we adopt the MDS module for adaptive selection and fusion of $V_{\text{out}}^{\text{MDC}}$ and V, as depicted in Fig. 5(b). We first concatenate $V_{\text{out}}^{\text{MDC}}$ and V, and then use two convolutional layers to compare them and generate selection weights via softmax. Finally, we employ the selection weights to identify and fuse $V_{\text{out}}^{\text{MDC}}$ and V, producing the transformed event representation V_1^{ad} and V_2^{ad}, which are fed into an existing flow network for optical flow estimation. In this paper, we use KPA-Flow [32] for optical flow estimation by default.

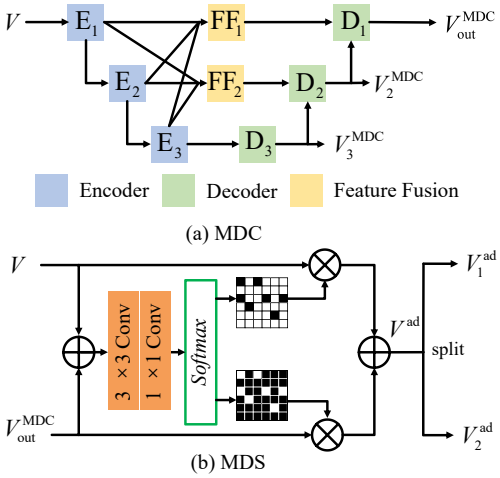

Figure 5: The detailed structure of sub-modules used in our proposed ADM model: (a) MDC, (b) MDS.

Loss Function. Based on our MDR dataset, we use the event representation with a moderate density as the ground truth (noted as V^{GT}) to train our ADM module. For the MDC module, we use a multi-scale loss as follows:

$$L_{\text{MDC}} = \sum_{k=1}^{3} \sqrt{(V_k^{\text{MDC}} - V_k^{\text{GT}})^2 + \xi^2}, \tag{6}$$

where $\xi = 10^{-3}$ is a constant value, V_k^{MDC} is the output of the k-th level in MDC, and V_k^{GT} is downsampled from V^{GT} to match the spatial size.

For the MDS module, we use the distance between the density of V^{ad} and V^{GT} as the guidance:

$$L_{\text{MDS}} = \left\| D(V^{\text{ad}}) - D(V^{\text{GT}}) \right\|_1, \tag{7}$$

where D means to calculate the density as in Eq. (5).

For the flow network, we use L1 loss (denoted as L_{Flow}) between flow prediction and ground truth as the guidance.

The final loss function for training the whole pipeline in Fig. 4 is determined as follows:

$$L_{total} = \lambda_1 L_{\text{MDC}} + \lambda_2 L_{\text{MDS}} + L_{\text{Flow}}, \tag{8}$$

where we empirically set $\lambda_1 = 0.1$ and $\lambda_2 = 10$.

4. Experiments

4.1. Datasets

MDR: We create the MDR dataset using the graphics engine blender. We use various 3D scenes for data generation. There are $80,000$ training samples and $6,000$ validation samples with accurate dense optical flow ground truth. Each sample has event sequences with different densities produced by using different thresholds C. By default, for

Method ($dt = 1$)	Train D.Type	Train D.Set	indoor flying1 EPE	indoor flying1 %Out	indoor flying2 EPE	indoor flying2 %Out	indoor flying3 EPE	indoor flying3 %Out	outdoor day1 EPE	outdoor day1 %Out	Avg EPE	Avg %Out
EST$_S$ [15]	E	M	0.97	0.91	1.38	8.20	1.43	6.47	-	-	1.26	5.19
EV-Flownet$_S$ [54]	I$_1$, I$_2$, E	M	1.03	2.20	1.72	15.1	1.53	11.9	0.49	0.20	1.19	7.35
Deng et al.$_S$ [10]	E	M	0.89	0.66	1.31	6.44	1.13	3.53	-	-	1.11	3.54
Paredes et al.$_S$ [39]	E	M	0.79	1.20	1.40	10.9	1.18	7.40	0.92	5.40	1.07	6.22
Matrix-LSTM$_S$ [6]	I$_1$, I$_2$, E	M	0.82	0.53	1.19	5.59	1.08	4.81	-	-	1.03	3.64
LIF-EV-FlowNet$_S$ [18]	E	FPV	0.71	1.41	1.44	12.8	1.16	9.11	0.53	0.33	0.96	5.90
Spike-FlowNet$_S$ [27]	I$_1$, I$_2$, E	M	0.84	-	1.28	-	1.11	-	0.49	-	0.93	-
Fusion-FlowNet$_D$ [26]	I$_1$, I$_2$, E	M	0.62	-	0.89	-	0.85	-	1.02	-	0.84	-
Fusion-FlowNet$_S$ [26]	I$_1$, I$_2$, E	M	0.56	-	0.95	-	0.76	-	0.59	-	0.71	-
E-RAFT$_D$ [17]	E	M	1.10	5.72	1.54	9.79	1.36	8.24	0.27	0.00	1.07	5.94
E-RAFT$_S$ [17]	E	M	0.87	1.32	1.13	6.20	1.03	4.70	0.24	0.00	0.82	3.06
Zhu et al.$_S$ [56]	E	M	0.58	0.00	1.02	4.00	0.87	3.00	0.32	0.00	0.69	1.75
DCEIFlow$_D$ [50]	I$_1$, I$_2$, E	M	0.56	0.28	0.64	0.16	0.57	0.12	0.91	0.71	0.67	0.31
DCEIFlow$_S$ [50]	I$_1$, I$_2$, E	C2	0.57	0.30	0.70	0.30	0.58	0.15	0.74	0.29	0.64	0.26
Stoffregen et al.$_S$ [42]	E	ESIM	0.56	1.00	0.66	1.00	0.59	1.00	0.68	0.99	0.62	0.99
STE-FlowNet$_S$ [11]	I$_1$, I$_2$, E	M	0.57	0.10	0.79	1.60	0.72	1.30	0.42	0.00	0.62	0.75
ADM-Flow$_D$(ours)	E	MDR	0.48	0.11	0.56	0.40	0.47	0.02	0.52	0.00	0.51	0.14
ADM-Flow$_S$(ours)	E	MDR	0.52	0.14	0.68	1.18	0.52	0.04	0.41	0.00	0.53	0.34
Method ($dt = 4$)	Train D.Type	Train D.Set	indoor flying1 EPE	indoor flying1 %Out	indoor flying2 EPE	indoor flying2 %Out	indoor flying3 EPE	indoor flying3 %Out	outdoor day1 EPE	outdoor day1 %Out	Avg EPE	Avg %Out
LIF-EV-FlowNet$_S$ [18]	E	FPV	2.63	29.6	4.93	51.1	3.88	41.5	2.02	18.9	3.36	35.3
EV-Flownet$_S$ [54]	I$_1$, I$_2$, E	M	2.25	24.7	4.05	45.3	3.45	39.7	1.23	7.30	2.74	29.3
E-RAFT$_D$ [17]	E	M	3.26	35.4	4.32	48.6	4.14	48.2	0.84	1.70	3.14	33.5
E-RAFT$_S$ [17]	E	M	2.81	30.6	3.99	40.9	3.78	40.5	0.72	1.12	2.83	28.3
Zhu et al.$_S$ [56]	E	M	2.18	24.2	3.85	46.8	3.18	47.8	1.30	9.70	2.62	32.1
Spike-FlowNet$_S$ [27]	I$_1$, I$_2$, E	M	2.24	-	3.83	-	3.18	-	1.09	-	2.58	-
Fusion-FlowNet$_D$ [26]	I$_1$, I$_2$, E	M	1.81	-	2.90	-	2.46	-	3.06	-	2.55	-
Fusion-FlowNet$_S$ [26]	I$_1$, I$_2$, E	M	1.68	-	3.24	-	2.43	-	1.17	-	2.13	-
STE-FlowNet$_S$ [11]	I$_1$, I$_2$, E	M	1.77	14.7	2.52	26.1	2.23	22.1	0.99	3.90	1.87	16.7
DCEIFlow$_D$ [50]	I$_1$, I$_2$, E	C2	1.49	8.14	1.97	17.4	1.69	12.3	1.87	19.1	1.75	14.2
DCEIFlow$_S$ [50]	I$_1$, I$_2$, E	C2	1.52	8.79	2.21	22.1	1.74	13.3	1.37	8.54	1.71	13.2
ADM-Flow$_D$(ours)	E	MDR	1.39	7.33	1.63	11.5	1.51	9.34	1.91	19.2	1.61	11.8
ADM-Flow$_S$(ours)	E	MDR	1.42	7.78	1.88	16.7	1.61	11.4	1.51	10.2	1.60	11.5

Table 1: Quantitative comparison of our method with previous methods on the MVSEC dataset [55]. Subscripts S and D donate the *sparse* and *dense* evaluation, respectively. We mark the best results in red and the second best results in blue.

training on MDR, we use the combination of all these samples with different densities to train flow networks. For the learning of our ADM module, we choose events with density between 0.45 and 0.55 as the label for L_{MDC} and L_{MDS}.

MVSEC: The MVSEC dataset [55] is a real-world dataset collected in indoor and outdoor scenarios with sparse optical flow labels. As a common setting, $28,542$ data pairs of the 'outdoor day2' sequence are used as the train set, and $8,410$ data pairs of the other sequences are used as the validation set. The density ranges of MVSEC train set and validation set are $[0.0003, 0.47]$ and $[0.001, 0.31]$, respectively.

DSEC: The DSEC dataset [16] is also collected using actual event cameras and lidar sensors on outdoor scenes. The dataset consists of $7,800$ training samples and $2,100$ test samples, divided into 24 sequences that include both day and night scenarios.

Previous methods have trained on other datasets, including synthetic datasets like C2 [50] and ESIM [42], as well as real-world captured datasets such as FPV [9]. To assess the performance of these methods, comparisons are made on the validation set of the MVSEC dataset. Both MVSEC and MDR datasets have two input settings for training and testing. These settings involve using the events between successive images within a time interval of $dt = 1$ and $dt = 4$. On the other hand, for the DSEC dataset, there is a public benchmark available specifically designed to evaluate flow networks. However, this benchmark only provides one input setting for testing. We denote MVSEC, DSEC, and MDR datasets by "M", "D", and "MDR".

4.2. Implementation details

We use PyTorch to implement our method and train all networks using the same setting. The networks are trained with the AdamW optimizer, with a batch size of 6 and learning rate of 4×10^{-4} for 150k iterations. Since the MVSEC dataset lacks multiple density event streams required for the learning of our ADM module, we disable L_{MDC} and L_{MDS}

when training on the MVSEC dataset.

4.3. Evaluation metrics

The evaluation metrics used in flow prediction models are the average End-point Error (EPE) and 1PE, 2PE, and 3PE (the percentage of pixels with EPE greater than 1,2,3 pixels) for DSEC dataset, and %Out (the percentage of points with EPE greater than 3 pixels and 5% of the ground truth flow magnitude) for MVSEC dataset. There are two types of evaluations: *dense* and *sparse*. In emphdense evaluation, errors are calculated only at pixels with valid flow annotations. *Sparse* evaluation focuses on pixels with valid flow annotations that have triggered at least one event. Experiments are performed on both MVSEC and MDR, considering both *dense* and *sparse* evaluations. However, only dense *dense* is conducted for DSEC.

4.4. Comparison with State-of-the-Arts

Results on MVSEC. In Table 1, we compare our model trained on the MDR train set with previous methods on the MVSEC evaluation set, and report detailed results for each sequence. We provide information on the data types (Train D.Type) and the training sets (Train D.Set) used in the training process for each method. Specifically, '$\mathrm{I}_1, \mathrm{I}_2, \mathrm{E}$' indicates that both image data and event data are used in the training and inference processes of the model, while 'E' indicates that only event data is used. As shown in Table 1, our model trained on the MDR dataset achieves state-of-the-art performance for both EPE and %Out metrics in settings of $dt = 1$ and $dt = 4$. Notably, our method demonstrates a 23.9% improvement (reducing EPE from 0.67 to 0.51) for dense optical flow estimation in $dt = 1$ settings, and an 8.0% improvement (reducing EPE from 1.75 to 1.61) in $dt = 4$ settings, surpassing previous methods. Qualitative comparison results are shown in row 1 and row 2 in Fig. 6.

Results on MDR. In Table 3, we compare our method with previous methods for training on the MVSEC dataset and testing on the MDR dataset for cross-training and validation to avoid over-fitting. We use different thresholds C to generate test data with different density ranges for evaluation. For the average EPE error of dense optical flow estimation, our model obtains the best result, which is a 17.1% improvement (reducing EPE from 0.82 to 0.68) in $dt = 1$ settings, and a 16.7% improvement (reducing EPE from 1.98 to 1.65) in $dt = 4$ settings. We also show some qualitative comparison results in row 3 and row 4 in Fig. 6.

Results on DSEC. In Table 2, we report the results on the DSEC test set for different models that are trained on DSEC, MDR, and a combination of DSEC and MDR. As can be seen, MDR-trained models produce comparable results to in-domain learned models on the DSEC train set.

Methods	Train D.Set	1PE	2PE	3PE	EPE
EV-FlowNet [54]	D	55.13	28.43	16.71	2.22
	MDR	42.52	24.40	16.32	1.86
	D+MDR	35.06	15.41	7.84	1.31
E-RAFT [17]	D	16.19	6.22	3.59	0.90*
	MDR	18.85	7.44	4.37	1.00
	D+MDR	13.16	5.19	2.93	0.82
ADM-EVFLOW	D	42.89	25.21	15.80	1.90
	MDR	40.02	14.88	12.34	1.67
	M+D	23.46	10.32	5.61	1.12
ADM-Flow(ours)	D	13.96	5.32	3.18	0.88
	MDR	14.53	5.78	3.59	0.92
	D+MDR	12.52	4.67	2.65	0.78

Table 2: Results on the DSEC test set. * denotes our reproduced results of E-RAFT, since the official training code is not released. All experiments are conducted on the same training code for fair comparison.

Moreover, training E-RAFT on the combined DSEC/MDR datasets yields improved performance, demonstrating the effectiveness of MDR. Our ADMFlow model achieves improved performance on all training datasets, highlighting the effectiveness and versatility of ADM. Some qualitative results are shown in Fig. 7.

Analysis of the MDR dataset. To demonstrate the effectiveness of our proposed MDR dataset, we train several optical flow networks [46, 17, 53, 21, 47, 24, 32] on both MDR and MVSEC train sets using identical training settings. For training, we use a combination of data samples from the MDR dataset with a density range of $[0.09, 0.69]$. We evaluate the trained networks on the MVSEC validation set, and the results, presented in Table 4, demonstrate that all networks trained on the MDR dataset outperform those trained on the MVSEC dataset.

4.5. Ablation study.

Training with different densities. We examine the impact of input event sequence density on optical flow learning, as our MDR dataset contains event data with various densities and corresponding dense flow labels. We train SKFlow [47], GMA [24], FlowFormer [21] and KPA-Flow [32] on the same sequence from our MDR dataset but with different average densities produced by using different threshold C, and then test them on the MVSEC dataset. Figure 8 shows the results in $dt = 4$ setting, indicating that these models perform better as the average density of the training set increases. However, their performance diminishes as the average density continues to increase. This phenomenon highlights the importance of selecting an appropriate density for the training set when learning event optical flow.

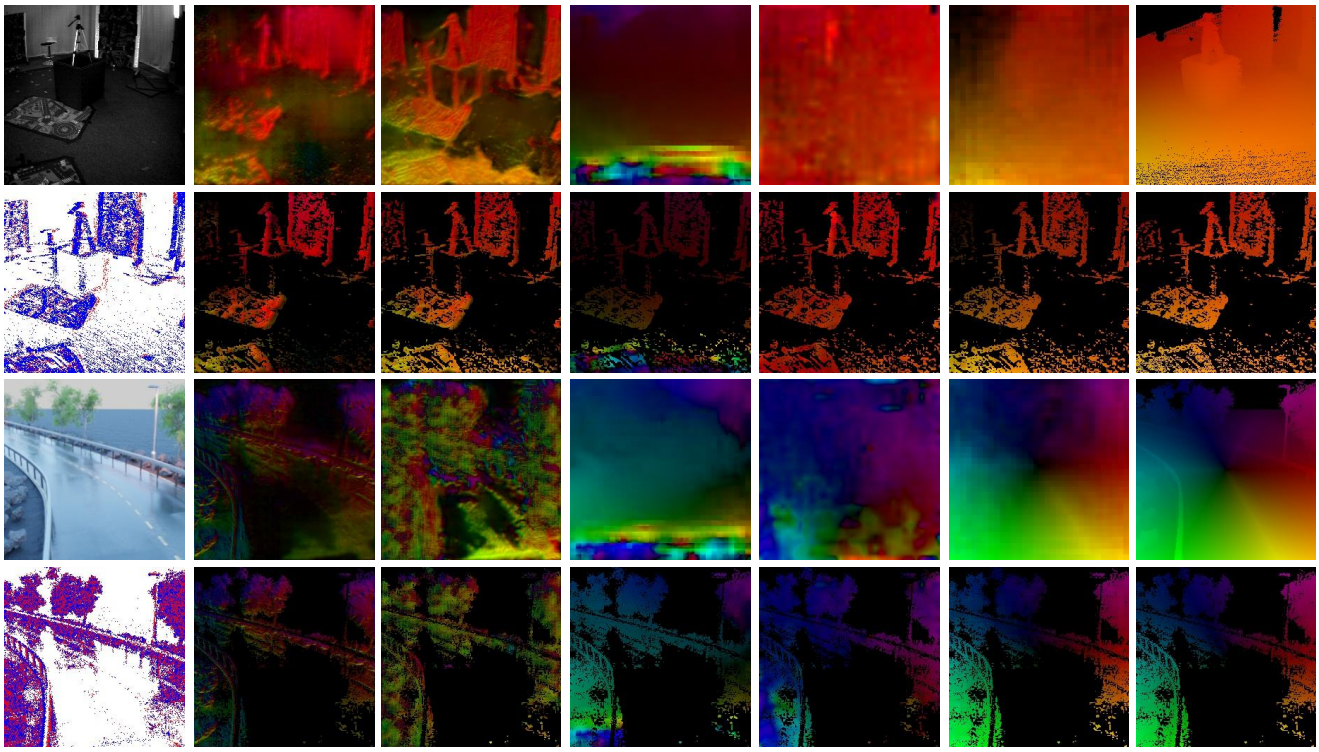

Figure 6: Qualitative comparisons compared with existing event-based methods. Row 1 and 2 are from MVSEC, whereas row 3 and 4 are from MDR. Row 1 and 3 visualize the dense predictions, whereas row 2 and 4 show the sparse.

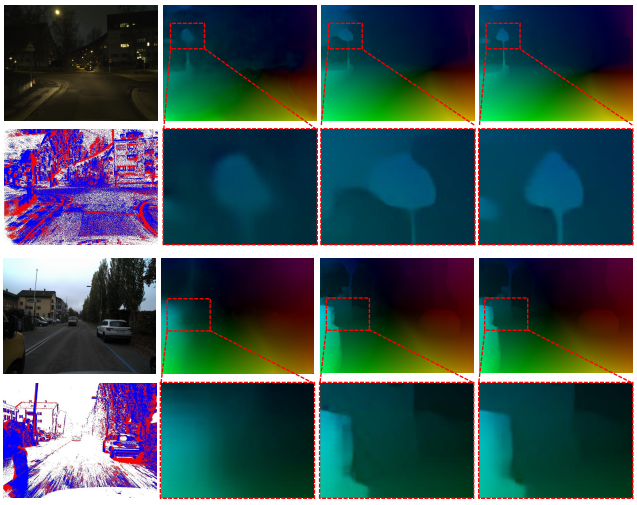

Figure 7: Qualitative comparisons on the DSEC test set. We visualize the dense predictions and zoom in the areas where are apparent differences.

Method ($dt = 1$)	density range				Avg
	0.09-0.24	0.24-0.39	0.39-0.54	0.54-0.69	
Spike-Flow$_S$ [27]	1.93	1.20	1.33	2.01	1.62
STE-Flow$_S$ [11]	1.17	1.11	0.82	1.30	1.10
DCEIFlow$_D$ [50]	1.28	1.37	1.27	1.46	1.35
DCEIFlow$_S$ [50]	1.00	0.84	0.75	0.82	0.85
E-RAFT$_D$ [17]	0.93	0.74	0.69	0.92	0.82
E-RAFT$_S$ [17]	0.82	0.65	0.66	0.82	0.74
ADM-Flow$_D$(ours)	0.77	0.69	0.50	0.74	0.68
ADM-Flow$_S$(ours)	0.69	0.64	0.49	0.74	0.64
Method ($dt = 4$)	density range				Avg
	0.09-0.24	0.24-0.39	0.39-0.54	0.54-0.69	
Spike-Flow$_S$ [27]	3.95	1.96	2.09	2.87	2.72
STE-Flow$_S$ [11]	2.90	2.27	1.98	2.17	2.33
DCEIFlow$_D$ [50]	3.84	2.68	2.99	4.27	3.45
DCEIFlow$_S$ [50]	2.25	1.32	1.19	2.34	1.78
E-RAFT$_D$ [17]	2.64	1.42	1.35	2.51	1.98
E-RAFT$_S$ [17]	2.18	1.31	1.24	2.29	1.76
ADM-Flow$_D$(ours)	2.12	1.20	1.24	2.02	1.65
ADM-Flow$_S$(ours)	2.03	1.13	1.09	1.90	1.54

Table 3: Quantitative evaluation on our MDR dataset. The methods in the table are all trained on the MVSEC dataset for cross-training and validation to avoid over-fitting. S and D donate the *sparse* and *dense* evaluation, respectively. We use EPE as the evaluation metric.

Ablation for ADM. In order to verify the impact of our proposed ADM module, we conduct ablation experiments by plugging the ADM module into several optical flow networks to selectively adjust the densities of the input events. We train these networks on both MDR and MVSEC datasets with the same setting except that the ADM module is dis-

Method	Train D.Set	$dt = 1$ EPE	$dt = 1$ %Out	$dt = 4$ EPE	$dt = 4$ %Out
PWCNet [22]	M	1.25	5.41	4.03	51.48
	MDR	1.14	3.48	2.92	38.62
RAFT [48]	M	1.19	4.90	3.33	39.78
	MDR	0.59	0.51	2.57	30.24
GMFlowNet [53]	M	1.00	3.75	3.61	42.31
	MDR	0.82	1.66	2.70	31.53
FlowFromer [21]	M	0.87	3.08	3.38	41.04
	MDR	0.61	0.40	2.49	28.83
SKFlow [47]	M	1.07	3.97	3.41	40.87
	MDR	0.59	0.33	2.46	27.64
GMA [24]	M	0.88	4.05	2.99	34.80
	MDR	0.58	0.44	2.19	23.00
KPAFlow [32]	M	0.86	2.86	3.19	38.32
	MDR	0.58	0.39	2.33	26.10

Table 4: Comparison of training on MVSEC vs. MDR. Models are evaluated on MVSEC for dense optical flow estimation.

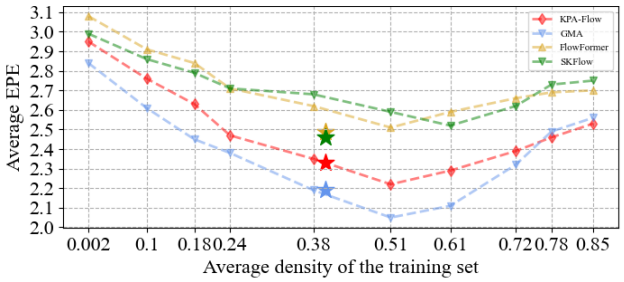

Figure 8: The performance of some supervised optical flow networks with different densities of the training set in $dt = 4$ setting. X-axis is the average density of events in the training set, and y-axis is their average EPE on MVSEC validation set. Different colors represent the performance of different networks, and the pentagons show the performance of networks trained on the MDR training set with a widely density range of $[0.09, 0.69]$, and the average density is 0.41.

abled or not, and test them on the MVSEC dataset. The experiment results are shown in Table 5, where we can notice that our ADM module can bring performance improvement for all supervised methods.

Ablation for the design of ADM. We conduct ablation experiments to verify the effectiveness of each component in our ADM module, including MDC, MDS, and the two training loss functions L_{MDC} and L_{MDS}. We train models using the same settings on the MDR dataset and evaluate them on the MVSEC dataset to show the individual impact of each component in our ADM module. The results are presented in Table 6. The comparison of (a)&(b) shows that adding only the MDC plugin results in a slight performance gain. Comparison of (b)&(c) reveals that enabling the density selection function through the MDS module brings a significant improvement. Comparing (c)&(d) and (d)&(e), we notice that with the guidance of two loss functions, ADM can learn to selectively choose the best density for optical flow estimation, resulting in a relatively significant improvement.

Method	M($dt = 1$) EPE	M($dt = 1$) %Out	MDR($dt = 1$) EPE	MDR($dt = 1$) %Out	M($dt = 4$) EPE	M($dt = 4$) %Out	MDR($dt = 4$) EPE	MDR($dt = 4$) %Out
PWCNet [22]	1.25	5.41	1.14	3.48	4.03	51.48	2.92	38.62
ADM-PWCNet	1.07	4.52	0.76	1.48	2.95	33.31	1.94	18.74
RAFT [48]	1.19	4.90	0.59	0.51	3.33	39.78	2.57	30.24
ADM-RAFT	0.82	3.03	0.56	0.24	2.73	30.91	1.72	13.83
GMFlowNet [53]	1.00	3.75	0.82	1.66	3.61	42.31	2.70	31.53
ADM-GMFlowNet	0.87	3.05	0.58	0.32	2.78	31.26	1.81	14.45
FlowFromer [21]	0.87	3.08	0.61	0.40	3.38	41.04	2.49	28.83
ADM-FlowFromer	0.78	2.87	0.53	0.15	2.56	26.57	1.67	12.78
SKFlow [47]	1.07	3.97	0.59	0.33	3.41	40.87	2.46	27.64
ADM-SKFlow	0.84	3.18	0.53	0.14	2.67	28.17	1.69	12.61
GMA [24]	0.88	4.05	0.58	0.44	2.99	34.80	2.19	23.00
ADM-GMA	0.76	2.65	0.54	0.22	2.45	25.75	1.63	11.95
KPAFlow [32]	0.86	2.86	0.58	0.39	3.19	38.32	2.33	26.10
ADM-KPAFlow	0.80	2.67	0.51	0.14	2.59	28.42	1.61	11.83

Table 5: Quantitative comparison of whether using ADM. Models are trained on MVSEC and MDR, and evaluated on MVSEC for dense optical flow estimation in $dt = 1$ and $dt = 4$ settings.

Method	MDC	MDS	L_{MDC}	L_{MDS}	Param. (M)	$dt = 1$ EPE	$dt = 1$ %Out	$dt = 4$ EPE	$dt = 4$ %Out
(a)	×	×	×	×	6.01	0.58	0.39	2.33	26.10
(b)	✓	×	×	×	7.71	0.57	0.33	2.20	23.78
(c)	✓	✓	×	×	7.72	0.54	0.26	1.92	18.26
(d)	✓	✓	✓	×	7.72	0.52	0.16	1.66	13.29
(e)	✓	✓	✓	✓	7.72	0.51	0.14	1.61	11.83

Table 6: Ablation study. Models are trained on the MDR training set, and evaluated on the MVSEC validation set for dense optical flow estimation in $dt = 1$ and $dt = 4$ settings.

5. Conclusion

In this work, we have created a rendered dataset for event-flow learning. Indoor and outdoor virtual scenes have been created using Blender with rich scene contents. Various camera motions are placed for the capturing of the virtual world, which can produce frames as well as accurate flow labels. The event values are generated by rendering high frame rate videos between two frames. In this way, the flow labels and event values are physically correct and accurate. The rendered dataset can adjust density of events by modifying the event trigger threshold. We have introduced a novel adaptive density module (ADM), which has shown its effectiveness by plugin into various event-flow pipelines. When trained on our dataset, previous approaches can improve their performances constantly.

Acknowledgements This work was supported by Sichuan Science and Technology Program of China under grants Nos. 2023NSFSC0462, 2023NSFSC1972, 2021YFG0001, 2022YFG0011 and 2022YFG0050.

References

[1] Ryad Benosman, Sio-Hoi Ieng, Charles Clercq, Chiara Bartolozzi, and Mandyam Srinivasan. Asynchronous frameless event-based optical flow. *Neural Networks*, 2012. 2, 3

[2] Yin Bi, Aaron Chadha, Alhabib Abbas, Eirina Bourtsoulatze, and Yiannis Andreopoulos. Graph-based object classification for neuromorphic vision sensing. In *Proc. ICCV*, 2019. 3

[3] Jonathan Binas, Daniel Neil, Shih-Chii Liu, and Tobi Delbruck. Ddd17: End-to-end davis driving dataset. *arXiv:1711.01458*, 2017. 3

[4] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240×180 130 db 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 2014. 1, 3

[5] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *Proc. ECCV*, 2012. 2

[6] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci. A differentiable recurrent surface for asynchronous event-based data. In *Proc. ECCV*, 2020. 3, 6

[7] Linda Capito, Umit Ozguner, and Keith Redmill. Optical flow based visual potential field for autonomous driving. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020. 1

[8] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016. 3

[9] Jeffrey Delmerico, Titus Cieslewski, Henri Rebecq, Matthias Faessler, and Davide Scaramuzza. Are we ready for autonomous drone racing? the UZH-FPV drone racing dataset. In *ICRA*, 2019. 6

[10] Yongjian Deng, Hao Chen, Huiying Chen, and Youfu Li. Learning from images: A distillation learning framework for event cameras. *IEEE Trans. on Image Processing*, 2021. 6

[11] Ziluo Ding, Rui Zhao, Jiyuan Zhang, Tianxiao Gao, Ruiqin Xiong, Zhaofei Yu, and Tiejun Huang. Spatio-temporal recurrent networks for event-based optical flow estimation. In *Proc. AAAI*, 2022. 3, 6, 8

[12] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proc. ICCV*, 2015. 2

[13] Denis Fortun, Patrick Bouthemy, and Charles Kervrann. Optical flow modeling and computation: A survey. *Computer Vision and Image Understanding*, 2015. 2

[14] Daniel Gehrig, Mathias Gehrig, Javier Hidalgo-Carrió, and Davide Scaramuzza. Video to events: Recycling video datasets for event cameras. In *Proc. CVPR*, 2020. 2, 3, 4

[15] Daniel Gehrig, Antonio Loquercio, Konstantinos G Derpanis, and Davide Scaramuzza. End-to-end learning of representations for asynchronous event-based data. In *Proc. ICCV*, 2019. 3, 6

[16] Mathias Gehrig, Willem Aarents, Daniel Gehrig, and Davide Scaramuzza. Dsec: A stereo event camera dataset for driving scenarios. *IEEE Robotics and Automation Letters*, 2021. 2, 6

[17] Mathias Gehrig, Mario Millhäusler, Daniel Gehrig, and Davide Scaramuzza. E-raft: Dense optical flow from event cameras. In *3DV*, 2021. 3, 4, 6, 7, 8

[18] Jesse Hagenaars, Federico Paredes-Vallés, and Guido De Croon. Self-supervised learning of event-based optical flow with spiking neural networks. In *Proc. NeurIPS*, 2021. 3, 6

[19] Yunhui Han, Kunming Luo, Ao Luo, Jiangyu Liu, Haoqiang Fan, Guiming Luo, and Shuaicheng Liu. Realflow: Em-based realistic optical flow dataset generation from videos. In *Proc. ECCV*, 2022. 2

[20] Yuhuang Hu, Shih-Chii Liu, and Tobi Delbruck. v2e: From video frames to realistic dvs events. In *Proc. CVPR*, 2021. 3

[21] Zhaoyang Huang, Xiaoyu Shi, Chao Zhang, Qiang Wang, Ka Chun Cheung, Hongwei Qin, Jifeng Dai, and Hongsheng Li. FlowFormer: A transformer architecture for optical flow. In *Proc. ECCV*, 2022. 2, 7, 9

[22] Junhwa Hur and Stefan Roth. Iterative residual refinement for joint optical flow and occlusion estimation. In *Proc. CVPR*, 2019. 9

[23] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proc. CVPR*, 2018. 2

[24] Shihao Jiang, Dylan Campbell, Yao Lu, Hongdong Li, and Richard Hartley. Learning to estimate hidden motions with global motion aggregation. In *Proc. ICCV*, 2021. 2, 7, 9

[25] Jacques Kaiser, J Camilo Vasquez Tieck, Christian Hubschneider, Peter Wolf, Michael Weber, Michael Hoff, Alexander Friedrich, Konrad Wojtasik, Arne Roennau, Ralf Kohlhaas, et al. Towards a framework for end-to-end control of a simulated vehicle with spiking neural networks. In *SIMPAR*, 2016. 3

[26] Chankyu Lee, Adarsh Kumar Kosta, and Kaushik Roy. Fusion-flownet: Energy-efficient optical flow estimation using sensor fusion and deep fused spiking-analog network architectures. In *ICRA*, 2022. 2, 3, 6

[27] Chankyu Lee, Adarsh Kumar Kosta, Alex Zihao Zhu, Kenneth Chaney, Kostas Daniilidis, and Kaushik Roy. Spike-flownet: event-based optical flow estimation with energy-efficient hybrid neural networks. In *Proc. ECCV*, 2020. 3, 6, 8

[28] Zhuoyan Li, Jiawei Shen, and Ruitao Liu. A lightweight network to learn optical flow from event data. In *ICPR*, 2021. 3

[29] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128 $times$ 128 120 db 15 mus latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits*, 2008. 1

[30] Shuaicheng Liu, Kunming Luo, Ao Luo, Chuan Wang, Fanman Meng, and Bing Zeng. Asflow: Unsupervised optical flow learning with adaptive pyramid sampling. *IEEE Trans. on Circuits and Systems for Video Technology*, 2021. 2

[31] Shuaicheng Liu, Kunming Luo, Nianjin Ye, Chuan Wang, Jue Wang, and Bing Zeng. Oiflow: Occlusion-inpainting op-

tical flow estimation by unsupervised learning. *IEEE Trans. on Image Processing*, 2021. 2

[32] Ao Luo, Fan Yang, Xin Li, and Shuaicheng Liu. Learning optical flow with kernel patch attention. In *Proc. CVPR*, 2022. 2, 5, 7, 9

[33] Ao Luo, Fan Yang, Xin Li, Lang Nie, Chunyu Lin, Haoqiang Fan, and Shuaicheng Liu. Gaflow: Incorporating gaussian attention into optical flow. In *Proc. ICCV*, 2023. 2

[34] Ao Luo, Fan Yang, Kunming Luo, Xin Li, Haoqiang Fan, and Shuaicheng Liu. Learning optical flow with adaptive graph reasoning. In *Proc. AAAI*, 2022. 2

[35] Kunming Luo, Chuan Wang, Shuaicheng Liu, Haoqiang Fan, Jue Wang, and Jian Sun. Upflow: Upsampling pyramid for unsupervised optical flow learning. In *Proc. CVPR*, 2021. 2

[36] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proc. CVPR*, 2016. 2

[37] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. *Proc. NeurIPS*, 2016. 3

[38] Liyuan Pan, Miaomiao Liu, and Richard Hartley. Single image optical flow estimation with an event camera. In *Proc. CVPR*, 2020. 2, 3

[39] Federico Paredes-Vallés and Guido CHE de Croon. Back to event basics: Self-supervised learning of image reconstruction for event cameras via photometric constancy. In *Proc. CVPR*, 2021. 6

[40] Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza. Esim: an open event camera simulator. In *Conference on robot learning*. PMLR, 2018. 4

[41] Amos Sironi, Manuele Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad Benosman. Hats: Histograms of averaged time surfaces for robust event-based object classification. In *Proc. CVPR*, 2018. 2

[42] Timo Stoffregen, Cedric Scheerlinck, Davide Scaramuzza, Tom Drummond, Nick Barnes, Lindsay Kleeman, and Robert Mahony. Reducing the sim-to-real gap for event cameras. In *Proc. ECCV*, 2020. 1, 2, 3, 4, 6

[43] Xiuchao Sui, Shaohua Li, Xue Geng, Yan Wu, Xinxing Xu, Yong Liu, Rick Goh, and Hongyuan Zhu. Craft: Cross-attentional flow transformer for robust optical flow. In *Proc. CVPR*, 2022. 2

[44] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *Proc. CVPR*, 2010. 1

[45] Deqing Sun, Daniel Vlasic, Charles Herrmann, Varun Jampani, Michael Krainin, Huiwen Chang, Ramin Zabih, William T Freeman, and Ce Liu. Autoflow: Learning a better training set for optical flow. In *Proc. CVPR*, 2021. 2

[46] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proc. CVPR*, 2018. 2, 7

[47] Shangkun Sun, Yuanqi Chen, Yu Zhu, Guodong Guo, and Ge Li. Skflow: Learning optical flow with super kernels. *Proc. NeurIPS*, 2022. 2, 7, 9

[48] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Proc. ECCV*, 2020. 2, 9

[49] Mikko Vihlman and Arto Visala. Optical flow in deep visual tracking. In *Proc. AAAI*, 2020. 1

[50] Zhexiong Wan, Yuchao Dai, and Yuxin Mao. Learning dense and continuous optical flow from an event camera. *IEEE Trans. on Image Processing*, 2022. 1, 2, 3, 6, 8

[51] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, and Dacheng Tao. Gmflow: Learning optical flow via global matching. In *Proc. CVPR*, 2022. 2

[52] Rui Xu, Xiaoxiao Li, Bolei Zhou, and Chen Change Loy. Deep flow-guided video inpainting. In *Proc. CVPR*, 2019. 1

[53] Shiyu Zhao, Long Zhao, Zhixing Zhang, Enyu Zhou, and Dimitris Metaxas. Global matching with overlapping attention for optical flow estimation. In *Proc. CVPR*, 2022. 7, 9

[54] Alex Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Ev-flownet: Self-supervised optical flow estimation for event-based cameras. In *Proceedings of Robotics: Science and Systems*, 2018. 1, 2, 3, 6, 7

[55] Alex Zihao Zhu, Dinesh Thakur, Tolga Özaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. *IEEE Robotics and Automation Letters*, 2018. 1, 2, 3, 6

[56] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised event-based learning of optical flow, depth, and egomotion. In *Proc. CVPR*, 2019. 4, 6