Progressive Spatio-temporal Alignment for Efficient Event-based Motion Estimation

Xueyan Huang Yueyi Zhang * Zhiwei Xiong
University of Science and Technology of China
hxy2020@mail.ustc.edu.cn, {zhyuey, zwxiong}@ustc.edu.cn

Abstract

In this paper, we propose an efficient event-based motion estimation framework for various motion models. Different from previous works, we design a progressive event-to-map alignment scheme and utilize the spatio-temporal correlations to align events. In detail, we progressively align sampled events in an event batch to the time-surface map and obtain the updated motion model by minimizing a novel time-surface loss. In addition, a dynamic batch size strategy is applied to adaptively adjust the batch size so that all events in the batch are consistent with the current motion model. Our framework has three advantages: a) the progressive scheme refines motion parameters iteratively, achieving accurate motion estimation; b) within one iteration, only a small portion of events are involved in optimization, which greatly reduces the total runtime; c) the dynamic batch size strategy ensures that the constant velocity assumption always holds. We conduct comprehensive experiments to evaluate our framework on challenging high-speed scenes with three motion models: rotational, homography, and 6-DOF models. Experimental results demonstrate that our framework achieves state-of-the-art estimation accuracy and efficiency. The code is available at `https://github.com/huangxueyan/PEME`.

1. Introduction

Event cameras [25, 30, 33], also known as bio-inspired silicon retinas, are novel vision sensors that asynchronously respond to pixel-wise brightness changes. Event cameras have the properties of high temporal resolution and high dynamic range, which make event cameras appealing to tackle many computer vision tasks under challenging conditions, such as high-speed pose estimation [5, 18, 19], HDR video generation [27, 31, 32, 34], 3D reconstruction [9, 11, 26] and low-latency motion estimation [7, 10, 24].

Event-based motion estimation aims to find the ego-

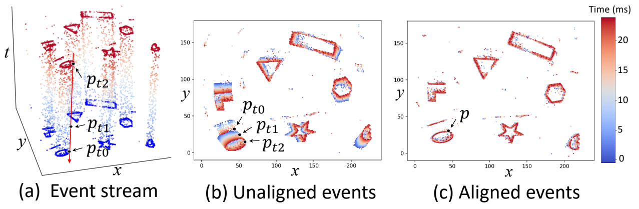


Figure 1. Event-based motion estimation. (a): A segment of the event cloud generated by rotational motion from the *shapes_rotation* dataset [20]. (b): An event frame generated from unaligned events. (c): An event frame generated from aligned events.

motion of the event camera. Since events can be triggered by the motion of the camera, the alignment of events is highly correlated with the camera's motion. The motion estimation problem is normally cast to an optimization problem based on the alignment of events [4]. With correct motion parameters, events triggered by the same world point can be aligned to the same pixel, forming an event frame with sharp edges. As for unaligned events, they generate blurred edges in the event frame. Fig. 1 shows a segment of the event cloud as well as two event frames with unaligned and aligned events.

Many approaches have been proposed for event-based motion estimation, such as contrast maximization, entropy minimization, and Poisson point process [6, 8, 22]. These methods follow the same procedure: slice an event cloud into batches with a fixed size or a fixed time interval, and then optimize a loss function with all the events in the batch. In practice, an event batch usually contains tens of thousands of events. It is very time-consuming and computationally redundant to involve such an amount of event data to optimize a motion transformation with 3 or 6 degrees of freedom (DOF). We observe that events approximately follow the same motion transformation in a short period; thus, it is probably not necessary to take all the events into account for motion estimation. From this point of view, we attempt to utilize sampled events to reduce the computational burden. To achieve this, we propose a distinct event-

*Corresponding author

to-map alignment scheme. In specific, we construct a time-surface (TS) map that maintains the timestamps of the former events at each pixel, and warp the later events to align the former events in the TS map. We measure the degree of alignment by a novel TS loss and update the motion parameters by minimizing the TS loss. We observe that in a short time interval, the events triggered by the same world point differ slightly in the timestamps and the coordinates, yielding almost identical residuals and gradient directions when evaluating the TS loss. Therefore, we can greatly reduce the computational burden by evaluating the TS loss with a small fraction of events.

In the alignment procedure, we attempt to warp the later events backward to the start timestamp t_{start} and align them with the former events. But few of the former events are triggered at t_{start} due to the spatial sparsity of the event camera. Moreover, these former events are normally triggered with a slight time shift from t_{start}, resulting in a slight drift in the coordinates. To fix this issue, we propose an iterative scheme to update the coordinates of the former events in the TS map with the latest motion parameters and progressively evaluate the TS loss based on the latest TS map.

In addition, the choice of event batch size has a significant impact on the accuracy. In practice, the batch size or the batch time interval is set manually, which is mainly determined by two constraints: one is that events in the batch should share the same motion parameters, *i.e.*, the time interval of the batch must be short enough to hold the constant velocity assumption, and the other is that the batch must contain sufficient events for the algorithm to execute normally. Essentially, these two constraints are mutually exclusive, making it difficult to determine the global batch size or time interval. To address this problem, we propose a dynamic batch strategy that can dynamically adjust the batch size to ensure that the constant velocity assumption always holds in this batch. We slice unprocessed events into event bundles with a small size and append these event bundles into the event batch if they meet the requirement that their overlap ratio reaches a threshold; otherwise, we stop merging and output an event batch with a certain number of bundles. With this strategy, our algorithm can adapt to the scenes under different conditions, such as different scene texture richness, camera motion speeds, and camera spatial resolutions, while for the fixed-size methods, they need to re-adjust the batch size to accommodate these scene changes.

We summarize the contributions of this work in the following.

- We present a unified event-based motion estimation framework that progressively aligns events using a novel event-to-map scheme with spatio-temporal information of sampled events.
- We also propose a dynamic batch size strategy to ensure that the constant velocity assumption always holds, which is more generalizable to different scene textures, camera speeds, and camera resolutions compared to the fixed batch strategy.
- Comprehensive experimental results demonstrate that our framework achieves state-of-the-art performance both in terms of accuracy and efficiency on publicly available datasets with three motion models.
- By utilizing a small number of sampled events in each iteration, our framework is able to achieve real-time implementation for the rotational model and the 6-DOF model with standard CPUs.

2. Related Work

Entropy based method. Nunes and Demiris [22, 24] proposed the entropy minimization framework (EMin) to obviate the need for an explicit intermediate image-based representation. They estimate the motion transformations by minimizing the dispersion of events, measured via a family of entropy functions. Since the entropy function has quadratic complexity, they extended EMin with an approximated version (AEMin) that searches events within a certain distance and an online version (IncEMin [23]) that incrementally estimates the motion parameters, achieving real-time rotational motion estimation. In contrast, our TS loss obviates the pair-wise calculation of events, showing a much lower computational complexity than the entropy loss. In addition, our approach is capable of conducting real-time estimations with sampled events and exhibiting superior accuracy compared to IncEMin.

3D Points based method. Liu *et al.* [14] considered events as 3D points in the spatio-temporal space. They proposed a spatio-temporal registration algorithm (STR) that estimates rotational motion transformations by splitting events into early and late parts and registering events one-by-one based on trimmed iterative closest points [3]. STR requires costly computation with a nearest-neighbor search strategy. In addition, it is prone to incorrect registrations with noise events. In our work, we handle the data association problem implicitly and we apply a denoising operation on the TS map to reduce the influence of the noise.

Image based method. To combine events with well-established frame-based vision tools, some works [6, 8, 13, 17] chose to transform events into an image-based representation. Gallego *et al.* [6, 7] proposed the Contrast Maximization (CMax) framework, which accumulates events to produce an image of warped events and maximizes the image contrast with respect to the motion parameters. Cheng *et al.* [8] developed a spatio-temporal Poisson point process (ST-PPP) that aligns events through a maximum likelihood approach. The hyper-parameter λ of the Poisson process is environment-specific and requires re-measurement when switching scenes or event cameras. Mitrokhin *et al.* [17] proposed a method to estimate similarity transformations with metrics in the time-image, where they minimize a time-image loss (*i.e.*, the sum of gradients of the time-image) with respect to the motion parameters. In [4,29], the CMax framework was evaluated with twenty more image-based loss metrics. The contrast loss [6] achieves the best performance in terms of accuracy and efficiency, while the time-image loss [17] reports the lowest accuracy. A distinct difference between our approach and [17] is that our approach constructs the TS map with the minimum timestamps of the events warped at the same pixel. Thus not all the event timestamps are involved in optimization.

3. Method

We define an event $e = \left(\mathbf{x}^\mathsf{T}, t, p\right)$ with its pixel coordinate $\mathbf{x} = (x, y)^\mathsf{T}$, the triggering time t and the polarity $p \in \{+1, -1\}$ indicating the increase or decrease of the brightness. Given an event batch $\xi = \{e_k\}_{k=1}^{N_e}$ recorded in a short time interval, we assume that all events in this batch share the same motion parameters $\boldsymbol{\theta}$ (*e.g.*, the velocity), with which warping an event backward to the start time t_{start} can be expressed as

$$e_k \to e'_k: \quad \mathbf{x}'_k = \mathcal{W}_b\left(\mathbf{x}_k, t_k; \boldsymbol{\theta}\right) \ , \tag{1}$$

where the warping function $\mathcal{W}_b$ is a coordinate transformation that is compatible with many models, including the rotational, homography, and 6-DOF motion models.

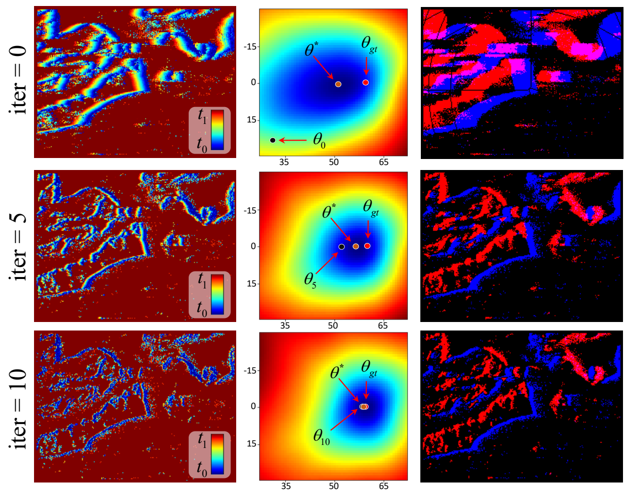
3.1. Time-Surface Map and Loss

Time-surface map. In our event-to-map alignment scheme, we adopt the TS map as the reference template for alignment. Different from the TS map defined in other works [12, 35], where they apply an exponential decay kernel on the pixels, our TS map directly sets the pixel value with the event timestamp and sets a penalty value in the background. We construct the TS map with the backward warped events and define the pixel value at the coordinate $\mathbf{x}$ as

$$\mathcal{I}_b(\mathbf{x}) = \min\{t_k \mid 1 \le k \le N_e,\ \mathbf{x}'_k = \mathbf{x}\} \ , \tag{2}$$

where $\mathbf{x}'_k$ is the coordinate of the warped event e'_k. Eq. (2) indicates that the value of the TS map $\mathcal{I}_b$ at the pixel coordinate $\mathbf{x}$ is equal to the smallest timestamp of the events warped to $\mathbf{x}$. For the pixel coordinates without any events warped to them, we set a high value (*e.g.*, t_{N_e}) as a penalty. Fig. 2 (a) provides several TS maps for illustration.

Time-surface loss. As aforementioned, with correct motion parameters, events triggered by the same world point should be warped to the same pixel, which means that the later events should closely align with the former events. In



(a) Time-surface map (b) Loss map (c) Event frame

Figure 2. Evolution of the TS map, loss map, and event frame with increasing iterations. Events are generated by horizontal sliding motion with velocity $\boldsymbol{\theta}_{gt}$ (pixels/s). We initialize the estimated parameter $\boldsymbol{\theta}_0$ with a velocity away from $\boldsymbol{\theta}_{gt}$, and plot the corresponding TS map, loss map, and event frame at different iterations.

the TS map, the sufficient and easy-to-compute image gradients provide the temporal information of the neighborhood, which can guide the later events toward the former events. Based on this observation, we propose an optimization loss of event alignment, *i.e.,* TS loss, which is measured in terms of temporal information. The proposed TS loss is expressed as

$$L(\boldsymbol{\theta}) = \sum_{e_k \in \xi_s} \mathcal{I}_b\left(\mathcal{W}_b\left(\mathbf{x}_k, t_k; \boldsymbol{\theta}\right)\right) \ , \tag{3}$$

where ξ_s is composed of the uniformly sampled events from ξ. Essentially, the TS loss is the sum of the timestamps acquired from the TS map $\mathcal{I}_b$ at the coordinates of the warped event samples. Since the TS map assigns the pixels warped by the former events with small timestamps and the background pixels with a penalty value, the value of our TS loss can implicitly reflect the degree of alignment between the later events and the former events. On the one hand, the motion parameters that warp events closer to the coordinates of the former events will be rewarded with a smaller loss value. On the other hand, the parameters that warp events far from the coordinates of the former events will obtain a large loss for the penalty value in the background of the TS map. In addition, the temporal information embedded in the gradient of the TS map provides the optimization direction to reduce loss. With the support of abundant and temporally continuous events in the TS map, the objective function exhibits a smooth loss map, as shown in Fig. 2 (b).

It can be observed that our event-to-map scheme is different from the event-to-event scheme [14] and the map-to-map scheme [21]. The event-to-event scheme conducts the registration event-wisely and measures their alignment based on their geometric distance. The map-to-map scheme registers patches from two TS maps defined in [12] and measures the patch alignment based on the timestamp consistency. Our event-to-map scheme aligns events with the former events in the TS map according to the TS loss.

Sampling strategy. Due to the low latency, the event camera can depict a scene at a rate of millions of events per second. We observe that the events triggered by the same world point differ slightly in the timestamps and the coordinates in a short time interval, which leads to very close residuals and gradient directions when evaluating the TS loss. This observation gives us an inspiration that we do not need to evaluate the residuals and gradients of all events, but only a tiny fraction of them are enough to provide a good optimization direction. In our algorithm, we uniformly sample a small number of events in the event batch and evaluate their TS loss. Since the optimization process involves only a small number of events, the running time of our algorithm is greatly reduced while it still maintains a competitive accuracy. See the reported experimental results in Sec. 5.

Bidirectional warping strategy. In constructing the TS map, the later events are overridden by the early events. To further utilize these events, similar to the multi-reference strategy in [28], we utilize a bidirectional warping strategy that warps events both forward and backward. We denote $\mathcal{W}_f$ as the forward warping function that warps events to t_{end}. Then we construct the TS map $\mathcal{I}_f$ of the forward warped events by picking the maximum timestamps of the events warped at the same coordinates:

$$\mathcal{I}_f(\mathbf{x}) = \max\{t_k \mid 1 \leq k \leq N_e,\ \mathbf{x}'_k = \mathbf{x}\} \,. \tag{4}$$

Note that the background (*i.e.*, no warped events falling in it) of $\mathcal{I}_f$ is set with a penalty value of 0. The TS loss with the bidirectional warping strategy can be expressed as

$$L(\boldsymbol{\theta}) = \sum_{e_k \in \xi_s} \left[\mathcal{I}_b \left(\mathcal{W}_b(\mathbf{x}_k, t_k; \boldsymbol{\theta}) \right) - \mathcal{I}_f \left(\mathcal{W}_f(\mathbf{x}_k, t_k; \boldsymbol{\theta}) \right) \right] . \tag{5}$$

3.2. Iterative Alignment

We attempt to warp the later events backward to t_{start} and align them with the former events. However, few of the former events are triggered at t_{start} due to the spatial sparsity of the event camera. Moreover, these former events are normally triggered with a slight time shift from t_{start}, resulting in a slight drift in the coordinates. Therefore, we can only obtain rough estimations by aligning the later events with the slightly drifted former events. As shown in Fig. 2

```
Algorithm 1 Progressive Spatio-temporal Alignment
Input: an event batch $\xi$, sampling ratio $s$, iterations $T$.
 1: $i \leftarrow 0$ (initialize the iteration index).
 2: $\boldsymbol{\theta}_0 \leftarrow \mathbf{0}$ (initialize the motion parameters $\boldsymbol{\theta}_0$).
 3: while $i < T$ do
 4:     warp the event batch $\xi$ backward to $t_{start}$.
 5:     create the backward TS map $\mathcal{I}_b$ using Eq. (2).
 6:     warp the event batch $\xi$ forward to $t_{end}$.
 7:     create the forward TS map $\mathcal{I}_f$ using Eq. (4).
 8:     uniformly sample $sN_e$ events from $\xi$.
 9:     find the optimal $\boldsymbol{\theta}^*$ to minimize TS loss in Eq. (5)
        with $sN_e$ sampled events.
10:     $\boldsymbol{\theta}_{i+1} \leftarrow \boldsymbol{\theta}^*$.
11:     $i \leftarrow i + 1$.
12: end while
13: return motion parameters $\boldsymbol{\theta}_T$.
```

(b), the drift of the former events leads to inconsistency between the ground truth $\boldsymbol{\theta}_{gt}$ and the optimal motion parameters $\boldsymbol{\theta}^*$ at the smallest TS loss. To fix this issue, we propose an iterative scheme to progressively update the former events in the TS map. In specific, we correct the coordinate drift by warping the former events to t_{start} with the latest motion parameters. Then, the former events can serve as more accurate targets for the later events to align with. As the iteration increases, the optimal parameters $\boldsymbol{\theta}^*$ with the minimum loss value in the loss map get closer to the ground truth $\boldsymbol{\theta}_{gt}$, as shown in Fig. 2 (b). The event-based progressive spatio-temporal alignment algorithm is summarized in Algorithm 1.

3.3. Dynamic Batch Strategy

With a fixed-size batch strategy, the constant velocity assumption, *i.e.,* all the events in the batch share the same motion parameters, may not hold during the time span of the events in low-texture scenes. To fix this issue, Cheng *et al.* [8] proposed to estimate the motion parameters with an affine model, *i.e.*, a uniform acceleration model. However, a more expressive motion model comes at the cost of more computing resources and poorer robustness to noise events. Our algorithm keeps the constant velocity assumption but allows the batch size to change adaptively. In our dynamic batch strategy, we first slice the unprocessed events Ψ of size N_e into several event bundles:

$$\{\xi^b_i \mid 0 \leq i < \lfloor N_e / N_b \rfloor ; \xi^b_i = \{e_k\}^{(i+1)N_b}_{k=iN_b}\} \,, \tag{6}$$

where each event bundle ξ^b_i contains N_b events, and the floor function $\lfloor x \rfloor$ outputs the greatest integer less than or equal to x. Then, we create an event batch with the first event bundle ξ^b_0, and initialize the motion model with $\boldsymbol{\theta}_0$ using Algorithm 1. The event bundles that are consistent with

Algorithm 2 Event-based Progressive Spatio-temporal Alignment With Dynamic Batch Strategy

```
Input: unprocessed events $\Psi$.
 1: slice $\Psi$ equally into event bundles $\{\xi_0^b, \xi_1^b, ...\}$ using Eq. (6).
 2: set the current event bundle index $i$ to 0.
 3: initialize the dynamic batch $\xi$ with the first event bundle $\xi_i^b$.
 4: set the first overlap ratio $r_i$ to $r_{init}$.
 5: while True do
 6:     obtain the motion parameters $\boldsymbol{\theta}_i$ of $\xi$ using Algorithm 1.
 7:     warped $\xi$ with $\boldsymbol{\theta}_i$ and create TS map $\mathcal{I}_i$ using Eq. (2).
 8:     warp the next event bundle $\xi_{i+1}^b$ with $\boldsymbol{\theta}_i$.
 9:     calculate the overlap ratio $r_{i+1}$ between the TS map $\mathcal{I}_i$
        and the warped $\xi_{i+1}^b$ using Eq. (7).
10:     if $r_{i+1} > r_i$ then append $\xi_{i+1}^b$ to $\xi$.
11:     else break the loop.
12:     end if
13:     $i \leftarrow i + 1$.
14: end while
15: return parameters $\boldsymbol{\theta}_i$ corresponding to the dynamic batch $\xi$.
```

the current motion model will be appended to the current batch. The consistency is measured in terms of the overlap ratio r_{i+1} between the current event bundle ξ_{i+1}^b and the previous TS map $\mathcal{I}_i$, which can be expressed as

$$r_{i+1} = \frac{1}{N_b} \sum_{e_k \in \xi_{i+1}^b} |\ \mathcal{I}_i\left(\mathcal{W}(\mathbf{x}_k, t_k; \boldsymbol{\theta})\right) < th\ |\ , \tag{7}$$

where the threshold th is normally set to the median time of the event batch. If the overlap ratio of the current bundle ξ_{i+1}^b is higher than that of the previous bundle ξ_i^b, we append this bundle to the current event batch and finetune the motion model with Algorithm 1. Otherwise, we stop appending bundles and output the latest motion parameters. The event-based progressive spatio-temporal alignment with dynamic batch strategy is summarized in Algorithm 2. Our dynamic batch strategy is distinct from [15], where they update an event slice according to the area event counts and a histogram of average matching distance. In contrast, we adaptively append the unprocessed events according to their overlap ratio with the current event batch.

4. Motion Models

In this section, we provide the formulations of different motion models for our proposed framework with the forward and backward warping strategy.

Rotational Model. In the rotational model, the motion parameters $\boldsymbol{\theta} = (\omega_x, \omega_y, \omega_z)^\top$ are the angular velocities around the (x, y, z) axes. The backward warping function of the rotational model can be expressed as

$$\mathcal{W}_b\left(\mathbf{x}_k, t_k; \boldsymbol{\theta}\right) \propto \mathcal{R}^{-1}\left(t_k - t_{start}; \boldsymbol{\theta}\right) \overline{\mathbf{x}}_k = \exp\left(-\widehat{\boldsymbol{\theta}} \cdot \Delta t\right) \overline{\mathbf{x}}_k\ , \tag{8}$$

where $\mathcal{R}^{-1} \in SO(3)$ is the inverse rotational transform that warps events from t_k to t_{start}, $\overline{\mathbf{x}}_k = K^{-1}(\mathbf{x}_k^\top, 1)^\top$ is the calibrated coordinate of $\mathbf{x}_k$ with the intrinsic matrix K, $\exp : so(3) \rightarrow SO(3)$ is the matrix exponential map and $\widehat{\boldsymbol{\theta}}$ is the 3×3 cross-product matrix associated with $\boldsymbol{\theta}$. Similarly, the forward warping function that warps an event e_k from t_k to t_{end} can be parameterized by the same $\boldsymbol{\theta}$:

$$\mathcal{W}_f\left(\mathbf{x}_k, t_k; \boldsymbol{\theta}\right) \propto \mathcal{R}\left(t_{end} - t_k; \boldsymbol{\theta}\right) \overline{\mathbf{x}}_k\ . \tag{9}$$

See supplementary material for the mathematical derivation for Eq. (9).

Homography Model. In the homography model, the estimated parameters $\boldsymbol{\theta} = (\boldsymbol{v}^\top, \boldsymbol{w}^\top, \boldsymbol{n}^\top)^\top$ are composed of the translation velocity $\boldsymbol{v}$, the angular velocity $\boldsymbol{\omega}$, and the normalized normal vector $\boldsymbol{n}$ of the plane. The backward warping function can be parameterized by $\boldsymbol{\theta}$:

$$\mathcal{W}_b\left(\mathbf{x}_k, t_k; \boldsymbol{\theta}\right) \propto \mathcal{H}^{-1}(t_k - t_{start}; \boldsymbol{\theta})\ \overline{\mathbf{x}}_k\ , \tag{10}$$

where $\mathcal{H}^{-1}$ is the inverse homography transform that warps events from t_k to t_{start}. The forward warping function in the homography model can be expressed as

$$\mathcal{W}_f\left(\mathbf{x}_k, t_k; \boldsymbol{\theta}\right) \propto \mathcal{H}(t_{end} - t_{start}; \boldsymbol{\theta}) \cdot \mathcal{H}^{-1}(t_k - t_{start}; \boldsymbol{\theta})\ \overline{\mathbf{x}}_k\ . \tag{11}$$

6-DOF Model. With an additional depth value assigned to each event, the 6-DOF model can be applied to estimate the ego-motion of the camera in arbitrary scenes. In the 6-DOF model, the motion parameters $\boldsymbol{\theta} = (\boldsymbol{v}^\top, \boldsymbol{w}^\top)^\top$ are composed of translation velocity $\boldsymbol{v}$ and angular velocity $\boldsymbol{\omega}$ at time t_{start}. The backward warping function can be parameterized by $\boldsymbol{\theta}$:

$$\mathcal{W}_b\left(\mathbf{x}_k, \mathbf{z}_k, t_k; \boldsymbol{\theta}\right) \propto \mathcal{T}^{-1}(t_k - t_{start}; \boldsymbol{\theta}) P_k\ , \tag{12}$$

where P_k is the corresponding 3D point of the event e_k with augmented depth and $\mathcal{T}^{-1}$ is the inverse transform that warps P_k from t_k to t_{start}. The forward warping function in the 6-DOF model can be expressed as

$$\mathcal{W}_f\left(\mathbf{x}_k, \mathbf{z}_k, t_k; \boldsymbol{\theta}\right) \propto \mathcal{T}(t_{end} - t_{start}; \boldsymbol{\theta}) \cdot \mathcal{T}^{-1}(t_k - t_{start}; \boldsymbol{\theta}) P_k. \tag{13}$$

See supplementary material for the mathematical simplification of Eq. (13).

5. Experiments

5.1. Datasets

For the rotational model and the homography model, we adopt the Event-Camera dataset [20], which was acquired by a DAVIS240 camera with 240×180 resolution. This dataset contains many real-world sequences with fast motions (up to 1800 deg/s of angular velocity), a high event rate (up to 8 million events per second) and different scene textures. Therefore, it has strict requirements on accuracy,

Method	*poster_rotation*			*dynamic_rotation*			*shapes_rotation*			*classroom*		
	e_w	RMS$_w$	Time	e_w	RMS$_w$	Time	e_w	RMS$_w$	Time	e_w	RMS$_w$	Time
CMax [6]	8.14	11.67	119.6*	4.56	6.64	118.8*	29.90	48.90	118.9*	4.76	6.09	140.3*
ST-PPP [8]	7.26	10.37	99.9*	3.84	5.19	100.2*	38.16	76.50	102.1*	4.51	5.77	117.3*
AEMin [22]	8.52	12.19	28.1	4.50	6.40	32.5	26.25	55.80	49.0	3.85	5.19	25.3
IncEMin [24]	13.96	19.82	$\underline{4.4}$	12.26	22.39	$\underline{4.9}$	115.02	165.73	$\underline{8.3}$	16.61	24.59	$\underline{3.3}$
Ours-1k	$\underline{6.87}$	$\underline{10.11}$	**4.3**	$\underline{3.67}$	$\underline{4.87}$	**4.0**	$\underline{7.17}$	$\underline{10.85}$	**3.1**	$\underline{2.47}$	$\underline{3.36}$	**3.2**
Ours-30k	**6.73**	**9.98**	35.0	**3.59**	**4.78**	34.3	**7.15**	**10.62**	27.2	**2.01**	**2.61**	29.7
RT-CMax [10]	25.59	66.20	1.8	$\underline{9.70}$	$\underline{15.92}$	$\underline{1.6}$	$\underline{27.93}$	64.57	$\underline{2.1}$	$\underline{3.80}$	$\underline{4.72}$	1.2
RT-IncEMin [23]	$\underline{20.64}$	$\underline{30.45}$	$\underline{1.7}$	13.15	19.67	2.5	30.40	$\underline{43.66}$	2.4	18.57	25.89	$\underline{1.1}$
RT-Ours	**8.44**	**12.96**	**0.5**	**4.90**	**6.50**	**0.7**	**8.33**	**11.47**	**2.0**	**2.84**	**3.86**	**0.3**

*The time is measured in the the Python environment with GPU acceleration.

Table 1. Accuracy and processing time in rotational motion estimation on the Event-Camera dataset [20] and the DVSMOTION20 dataset [2]. The mean angular velocity error e_w over three axes (x, y, z) and the RMS are presented in deg/s. The time in microseconds (μs) is the average processing time for each event. The best and second best values are in bold and underlined, respectively.

processing speed and generalizability for motion estimation algorithms. In addition, we also evaluate the rotational model with the DVSMOTION20 dataset [2], which was acquired by a DAVIS346 camera with 346×260 resolution.

For the 6-DOF model, we adopt the MVSEC dataset [36], which collects event streams from stereo event cameras (DAVIS346), accompanied by IMU, LiDAR and GPS data. In addition, it also provides the depth images generated from LiDAR data, from which we assign each event a depth value according to its pixel coordinates.

5.2. Implementation Details

Our experiments can be divided into a real-time track and a non-real-time track. For the non-real-time track, we set the fixed-size models of related works with a batch size $N_e = 30k$ (normally contains a $5ms \sim 25ms$ event stream). In our framework with the dynamic batch size strategy, we apply the bidirectional warping strategy and set the iteration number $T = 2$ in Algorithm 1, the event bundle size $N_b = 5000$, and the initial overlap ratio $r_{init} = 50\%$ in Algorithm 2. In the quantitative comparison, we report our results with $1k$ event samples (the minimum processing time) and $30k$ event samples (the highest accuracy), respectively. For the real-time track, we fill the batch with the latest and unprocessed events (normally contains $1k \sim 150k$ events) and optimize $1k$ event samples with iteration number $T = 3$ using the unidirectional warping strategy.

In all experiments, we apply Gaussian smoothing for the TS maps with $\sigma = 0.5$ and kernel size of 5. The TS map is updated every ten optimization steps of the motion parameters. The polarities of events are considered equally in our algorithm. To reduce noise interference, we only sample events whose 8-connected neighborhood contains at least four events. The framework is implemented with C++ on a standard computer with a 2.9 GHz and 16-Core Intel i7 CPU. For the optimization part, we adopt the open-source library Ceres [1] with Powell's dog leg method [16].

5.3. Results

5.3.1 Rotational model

We compare our rotational models with the most recent works of angular velocity estimation [6, 8, 10, 22–24] in terms of accuracy and processing time. These two models are named with Ours-30k and Ours-1k. As for Ours-30k, we sample $30k$ events and apply the bidirectional warping strategy and the dynamic batch size strategy. As for Ours-1k, we sample $1k$ events in each optimization iteration. Following the evaluation protocol [8], we utilize the angular velocity from the IMU as the ground truth. The accuracy is measured in three error metrics: the mean angular velocity error e_w over three axes and the root-mean-square (RMS) error. We report the average processing time required for each event in microseconds (μs). We test these works with the same batch size $N_e = 30k$ in the non-real-time track, except for IncEMin [24], which maintains the most recent $10k$ events for the Event-Camera dataset [20] and $20k$ for the DVSMOTION20 dataset [2]. Since EMin [22] requires the expensive computation of entropy loss, we only report the results of AEMin [22] and IncEMin [24]. Note that we adopt the code of CMax and ST-PPP released in [8], which is implemented with Python and GPU acceleration. For a fair comparison of processing time with these two methods, we provide the computational complexity analysis in Sec. 5.4.

In Tab. 1, Ours-30k achieves the best accuracy among all the sequences. When the scene changes (*shape_rotation*) or the camera resolution changes (*classroom*), other methods with the fixed-size strategy suffer from a large drop in accuracy. Our method can dynamically adjust the batch size to adapt to these environmental changes and achieve higher accuracy. For the processing time comparison, Ours-1k reports the minimum average processing time of 3.66 μs per

	Method	ARPE	ARRE	RMS$_w$	Time
translation	CMax [6]	39.23	0.0108	17.14	118.3*
	AEmin [22]	27.24	0.0140	24.92	33.1
	Ours-1k	26.80	0.0117	16.59	**5.9**
	Ours-30k	**20.59**	**0.0097**	**13.28**	42.1
6dof	CMax [6]	48.58	0.0150	51.47	118.0*
	AEmin [22]	46.80	0.0147	43.82	32.2
	Ours-1k	28.48	0.0157	22.51	**5.4**
	Ours-30k	**22.62**	**0.0138**	**22.54**	36.7
hdr	CMax [6]	45.51	0.0124	22.08	107.6*
	AEmin [22]	50.04	0.0243	46.20	33.5
	Ours-1k	25.44	0.0138	26.55	**5.5**
	Ours-30k	**22.86**	**0.0122**	**17.08**	38.1

*The time is measured in the Python environment with GPU acceleration.

Table 2. Accuracy and processing time of motion estimation in planar scenes from the poster sequences of the Event-Camera dataset [20]. The time in microseconds (μs) is the average processing time for each event. The best results are in bold.

event. Meanwhile, it maintains high accuracy and outperforms IncEMin [24] (with the second minimum processing time) by a large margin. For the real-time track, we compare it with other methods that can achieve real-time processing [10, 23], as shown at the bottom of Tab. 1. In the real-time implementation, the optimization stops when the next batch arrives, which can decrease the accuracy for insufficient optimization time. Our model shows an improvement of approximately 50% in RMS compared with the second best performance method.

5.3.2 Homography model

We compare our homography model with CMax [6] and AEMin [22] in three planar scenes. We evaluate the accuracy in three error metrics: RMS$_w$, relative rotation error (RRE = $\left\|\operatorname{logm}\left(\mathcal{R}_{est}^{\top}\mathcal{R}_{gt}\right)\right\|$), where logm denotes the matrix log, and relative pose error (RPE = $\arccos\frac{\mathbf{t}_{est}\cdot\mathbf{t}_{gt}}{\|\mathbf{t}_{est}\|\cdot\|\mathbf{t}_{gt}\|}$). Since IncEMin [24] is not yet applicable in planar scenes, we only report the results of AEMin. Note that the CMax with homography model is based on the PyTorch implementation from [8], and we only compare its accuracy.

In Tab. 2, Ours-30k shows a significant improvement and achieves a new state-of-the-art accuracy based on these three metrics. Even with $1k$ events involved in each iteration, Ours-1k still outperforms AEMin in the *translation* and *hdr* sequences with about 18% processing time of AEMin.

5.3.3 6-DOF model

We quantitatively compare our 6-DOF model with CMax [6], AEMin [22] and IncEMin [24] in terms of accuracy and processing time. The accuracy is measured with five metrics: RMS$_w$ and RRE to measure the rotation error, RMS$_v$, RPE and endpoint error (EE $= \|\mathbf{t}_{est} - \mathbf{t}_{gt}\|$) to measure translation error. Note that CMax with the 6-DOF model is implemented on the PyTorch implementation from [8]. We list its processing time for reference and only compare it with its accuracy. We set the batch size $N_e = 30k$ for AEMin and CMax, except for IncEMin, which maintains $15k$ and $30k$ events for the *indoor_flying1* and the *outdoor_driving1* sequences.

	Method	RMS$_w$	RMS$_v$	ARPE	ARRE	AEE	Time
indoor_flying1	CMax [6]	1.75	0.072	11.72	0.0077	0.11	135.0*
	AEmin [22]	1.38	0.069	10.86	0.0062	0.11	712.8
	IncEMin [24]	1.65	0.085	13.75	0.0075	0.13	4.7
	Ours-1k	1.07	0.040	7.74	0.0061	**0.06**	2.9
	Ours-30k	**1.05**	**0.039**	**7.49**	**0.0060**	**0.06**	26.9
	RT-Ours	1.45	0.053	7.86	0.0067	0.08	**1.3**
outdoor_driving1	CMax [6]	14.19	1.225	15.41	0.0228	1.55	143.3*
	AEmin [22]	4.00	0.677	9.39	0.0137	0.98	424.1
	IncEMin [24]	4.05	1.035	15.75	0.0096	1.49	52.2
	Ours-1k	2.73	0.606	7.96	0.0077	0.93	3.9
	Ours-30k	**2.72**	**0.598**	**7.83**	**0.0076**	**0.92**	36.5
	RT-Ours	2.85	0.630	8.26	0.0080	0.96	**1.2**

*The time is measured in the Python environment with GPU acceleration.

Table 3. Accuracy and processing time of motion estimation of the 6-DOF model in the MVSEC dataset [36]. The time in microseconds (μs) is the average processing time for each event. The best results are in bold.

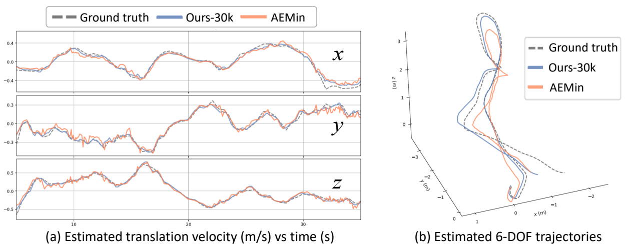


Figure 3. Estimation results of the 6-DOF model with the *indoor_flying1* sequence [36]. (a): estimated translation velocity. (b): the corresponding accumulated translation trajectories.

In Tab. 3, Ours-30k consistently outperforms the others in all metrics. Even the real-time implementation of our 6-DOF model, which involves $1k$ events with 3 iterations using the unidirectional warping strategy, achieves state-of-the-art accuracy with real-time processing, showing a promising method in event-based motion estimation.

In Fig. 3 (a), we qualitatively compare the estimated translation velocity with the ground truth on the *indoor_flying1* sequence for the first 35 seconds. Then, we integrate these velocities with their batch duration and plot the 3D trajectories in Fig. 3 (b). Since the estimated velocities are based on event batches, the drifts can be accumulated with increasing time. Even then, our trajectory follows the ground truth closely and exhibits small drifts than AEMin.

5.4. Computational Complexity

The computational complexity in our framework mainly consists of two parts: generating the TS map in Eq. (2),

	CMax [6]	ST-PPP [8]	EM [22]	AEM [22]	Ours-1k	Ours-30k
Complexity	N_e	N_e	N_e^2	$N_e k^d$	$0.26N_e$	$2.2N_e$
Time	1.4	5.1	891	8.4	0.4	5.4

Table 4. Comparison of computational complexity and evaluation time with N_e events. For comparability, all algorithms are implemented in C++, and the evaluation time in milliseconds (ms) is the duration for one loss evaluation. For simplicity, we show the specific value of the coefficients of our framework.

which involves N_e events; optimizing the loss function in Eq. (3), which involves sN_e events. In our experiments, for every 10 optimizing steps on the motion parameters, we update the TS map. On average, the computational complexity for each loss evaluation of our framework is $\mathcal{O}\left((0.1+s)2N_e\right)$, where the factor 2 is due to the bidirectional warping strategy. The sampling ratio s is typically set to 0.03, *i.e.*, sampling $1k$ events from a batch with $N_e = 30k$. In Tab. 4, we demonstrate the computational complexity of the related works in the rotational model [6, 8, 22] and compare the evaluation time for each loss evaluation with $30k$ events. For comparison, we reimplement the loss function of CMax [6] and ST-PPP [8] in C++. The results show that our approach with $1k$ event samples achieves the minimum computational complexity as the least evaluation time.

5.5. Ablation Study

In Tab. 5, we present ablation studies on the rotational model with the sampling strategy, bidirectional warping strategy and dynamic batch size strategy. For the ablation of the sampling strategy, we apply different event sampling numbers in Algorithm 1. Their results are reported at the top of Tab. 5, where the symbol 'Ours-1k' denotes our method with $1k$ event samples. It shows that the accuracy increases with more sampling events. However, the slight increase in accuracy comes at the cost of several times the computation effort. Surprisingly, our algorithm still retains competitive performance, even with $0.1k$ event samples involved in the optimization. This result also confirms our judgement that we do not need to evaluate the residuals and gradients of all events. Only a tiny fraction of them are enough to provide a good optimization direction. The sampling strategy may be the key to implementing event-based motion estimation algorithms in some computation-limited and time-critical applications.

At the bottom of Tab. 5, we present ablation studies of the bidirectional warping strategy and the dynamic batch size strategy. For a fair comparison, we fix the same event sampling number in these experiments. Under both the dynamic batch size strategy and the fixed-size strategy, the results of the bidirectional warping strategy are better than those of the unidirectional warping strategy in accuracy, but it comes at the cost of nearly twice the averaging process-

Method	warp	batch	*poster_rotation*			*shapes_rotation*		
			e_w	RMS$_w$	Time	e_w	RMS$_w$	Time
Ours-0.1k	b	d	7.22	10.23	**1.9**	7.39	10.92	**1.4**
Ours-1k	b	d	6.87	10.11	4.3	7.17	10.85	3.1
Ours-10k	b	d	6.76	9.99	11.9	**7.11**	**10.55**	12.2
Ours-30k	b	d	**6.73**	**9.98**	35.7	7.15	10.62	27.4
Ours-30k	u	d	7.91	11.05	15.9	10.62	15.08	15.4
Ours-30k	b	f	7.47	10.21	29.0	11.85	18.39	25.6
Ours-30k	u	f	8.15	11.10	16.2	14.51	22.41	12.0

Table 5. Ablations on the rotational model with the sampling strategy, bidirectional warping strategy and dynamic batch size strategy on the Event-Camera dataset [20]. The symbols 'u' and 'b' denote the unidirectional and the bidirectional warping strategies, respectively. The symbols 'f' and 'd' denote the fixed-size and the dynamic batch size strategies, respectively.

ing time. To balance the tradeoff between the accuracy and the computation cost with the bidirectional strategy, we can apply a small event sampling number to reduce the computational burden and achieve higher accuracy.

In the experiments with the dynamic batch size strategy, the batch size can be adaptively change from $10k$ to $150k$. As shown in Tab. 5, the results of the dynamic batch size strategy report higher accuracy than the fixed-size strategy. Especially in the *shape_rotation* sequence, the time span of a batch with $30k$ events can be dozens of milliseconds so that all the events may not share the same motion parameters. Our dynamic batch size strategy obviates such globally fixed parameters and has better generalizability to the scene textures and the camera speeds. The processing time of the dynamic batch size strategy is similar to that of the fixed-size strategy in these two sequences.

6. Conclusion

In this work, we propose a unified event-based motion estimation framework for various motion models. With a novel event-to-map alignment scheme, our framework can efficiently perform motion estimations with a small number of sampled events, which allows us to achieve real-time performance for the rotational model and the 6-DOF model with standard CPUs. We also propose a dynamic batch size strategy to make our algorithm more generalizable to different scenes textures, camera speeds, and camera resolutions. Comprehensive experimental results demonstrate that our framework achieves state-of-the-art performance in terms of accuracy and efficiency. We believe that the proposed framework is a promising approach for computation-limited and time-demanding applications such as SLAM and robotics.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China under Grants 62131003 and 62021001.

References

[1] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org. 6

[2] Mohammed Almatrafi, Raymond Baldwin, Kiyoharu Aizawa, and Keigo Hirakawa. Distance surface for event-based optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 6

[3] Dmitry Chetverikov, Dmitry Stepanov, and Pavel Krsek. Robust euclidean alignment of 3d point sets: the trimmed iterative closest point algorithm. *Image and Vision Computing*, 23:299–309, 2005. 2

[4] Guillermo Gallego, Mathias Gehrig, and Davide Scaramuzza. Focus is all you need: Loss functions for event-based vision. In *Computer Vision and Pattern Recognition*, 2019. 1, 3

[5] Guillermo Gallego, Jon E. A. Lund, Elias Mueggler, Henri Rebecq, Tobi Delbruck, and Davide Scaramuzza. Event-based, 6-dof camera tracking from photometric depth maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:2402–2412, 2018. 1

[6] Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. In *Computer Vision and Pattern Recognition*, 2018. 1, 2, 3, 6, 7, 8

[7] Guillermo Gallego and Davide Scaramuzza. Accurate angular velocity estimation with an event camera. In *International Conference on Robotics and Automation*, 2017. 1, 2

[8] Cheng Gu, Erik Learned-Miller, Daniel Sheldon, Guillermo Gallego, and Pia Bideau. The spatio-temporal poisson point process: A simple model for the alignment of event camera data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13495–13504, 2021. 1, 2, 4, 6, 7, 8

[9] Xueyan Huang, Yueyi Zhang, and Zhiwei Xiong. High-speed structured light based 3d scanning using an event camera. *Optics Express*, 29(22):35864–35876, 2021. 1

[10] Haram Kim and H. Jin Kim. Real-time rotational motion estimation with contrast maximization over globally aligned events. In *International Conference on Robotics and Automation*, 2021. 1, 6, 7

[11] Hanme Kim, Stefan Leutenegger, and Andrew J Davison. Real-time 3d reconstruction and 6-dof tracking with an event camera. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI 14*, pages 349–364. Springer, 2016. 1

[12] Xavier Lagorce, Garrick Orchard, Francesco Galluppi, Bertram E. Shi, and Ryad Benosman. Hots: A hierarchy of event-based time-surfaces for pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1346–1359, 2017. 3, 4

[13] Daqi Liu, Alvaro Parra, and Tat-Jun Chin. Globally optimal contrast maximisation for event-based motion estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6349–6358, 2020. 2

[14] Daqi Liu, Alvaro Parra, and Tat-Jun Chin. Spatiotemporal registration for event-based visual odometry. In *Computer Vision and Pattern Recognition*, 2021. 2, 3

[15] Min Liu and Tobi Delbruck. Adaptive time-slice block-matching optical flow algorithm for dynamic vision sensors. *british machine vision conference*, 2018. 5

[16] M.L.A. Lourakis and A.A. Argyros. Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustment? In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1526–1531 Vol. 2, 2005. 6

[17] Anton Mitrokhin, Cornelia Fermüller, Chethan M. Parameshwara, and Yiannis Aloimonos. Event-based moving object detection and tracking. In *Intelligent Robots and Systems*, 2018. 2, 3

[18] Elias Mueggler, Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. Continuous-time visual-inertial odometry for event cameras. *IEEE Transactions on Robotics*, 34:1425–1440, 2018. 1

[19] Elias Mueggler, Basil Huber, and Davide Scaramuzza. Event-based, 6-dof pose tracking for high-speed maneuvers. In *Intelligent Robots and Systems*, 2014. 1

[20] Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam:. *The International Journal of Robotics Research*, 36:142–149, 2017. 1, 5, 6, 7, 8

[21] Jun Nagata, Yusuke Sekikawa, and Yoshimitsu Aoki. Optical flow estimation by matching time surface with event-based cameras. *Sensors*, 21:1150, 2021. 4

[22] Urbano Miguel Nunes and Yiannis Demiris. Entropy minimisation framework for event-based vision model estimation. In *European Conference on Computer Vision*, 2020. 1, 2, 6, 7, 8

[23] Urbano Miguel Nunes and Yiannis Demiris. Live demonstration: Incremental motion estimation for event-based cameras by dispersion minimisation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1322–1323, 2021. 2, 6, 7

[24] Urbano Miguel Nunes and Yiannis Demiris. Robust event-based vision model estimation by dispersion minimisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 1, 2, 6, 7

[25] Christoph Posch, Daniel Matolin, and Rainer Wohlgenannt. A qvga 143 db dynamic range frame-free pwm image sensor with lossless pixel-level video compression and time-domain cds. *IEEE Journal of Solid-state Circuits*, 46:259–275, 2011. 1

[26] Henri Rebecq, Guillermo Gallego, Elias Mueggler, and Davide Scaramuzza. Emvs: Event-based multi-view stereo—3d reconstruction with an event camera in real-time. *International Journal of Computer Vision*, 126(12):1394–1414, 2018. 1

[27] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. High speed and high dynamic range video with an event camera. *IEEE transactions on pattern analysis and machine intelligence*, 43(6):1964–1980, 2019. 1

[28] Shintaro Shiba, Yoshimitsu Aoki, and Guillermo Gallego. Secrets of event-based optical flow. In *European Conference on Computer Vision*, pages 628–645. Springer, 2022. 4

[29] Timo Stoffregen and Lindsay Kleeman. Event cameras, contrast maximization and reward functions: An analysis. In *Computer Vision and Pattern Recognition*, 2019. 3

[30] Yunjae Suh, Choi Seungnam, Ito Masamichi, Jeong-Seok Kim, Young-Ho Lee, Jong-Seok Seo, Jung Hee-Jae, Yeo Dong-Hee, Seol Namgung, Bong Jongwoo, Sehoon Yoo, Seung-Hun Shin, Doo-Won Kwon, Pil-Kyu Kang, Seok-Ho Kim, Hoon joo Na, Ki-Hyun Hwang, Chang-Woo Shin, Junseok Kim, Paul K. J. Park, Joon-Seok Kim, Hyunsurk Ryu, and Yongin Park. A 1280×960 dynamic vision sensor with a 4.95-μm pixel pitch and motion artifact minimization. In *International Symposium on Circuits and Systems*, 2020. 1

[31] Lin Wang, Yo-Sung Ho, Kuk-Jin Yoon, et al. Event-based high dynamic range image and very high frame rate video generation using conditional generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10081–10090, 2019. 1

[32] Wenming Weng, Yueyi Zhang, and Zhiwei Xiong. Event-based video reconstruction using transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2563–2572, October 2021. 1

[33] Wenming Weng, Yueyi Zhang, and Zhiwei Xiong. Boosting event stream super-resolution with a recurrent neural network. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VI*, pages 470–488. Springer, 2022. 1

[34] Zeyu Xiao, Wenming Weng, Yueyi Zhang, and Zhiwei Xiong. Eva^2: Event-assisted video frame interpolation via cross-modal alignment and aggregation. *IEEE Transactions on Computational Imaging*, 8:1145–1158, 2022. 1

[35] Yi Zhou, Guillermo Gallego, Henri Rebecq, Laurent Kneip, Hongdong Li, and Davide Scaramuzza. Semi-dense 3d reconstruction with a stereo event camera. In *European Conference on Computer Vision*, 2018. 3

[36] Alex Zihao Zhu, Dinesh Thakur, Tolga Ozaslan, Bernd G. Pfrommer, Vijay Kumar, and Kostas Daniilidis. The multi vehicle stereo event camera dataset: An event camera dataset for 3d perception. In *International Conference on Robotics and Automation*, 2018. 6, 7