sensors

MDPI

Article

An Asynchronous Real-Time Corner Extraction and Tracking Algorithm for Event Camera

Jingyun Duo and Long Zhao *

School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China; JYDuo@buaa.edu.cn
* Correspondence: flylong@buaa.edu.cn

Abstract: Event cameras have many advantages over conventional frame-based cameras, such as high temporal resolution, low latency and high dynamic range. However, state-of-the-art event-based algorithms either require too much computation time or have poor accuracy performance. In this paper, we propose an asynchronous real-time corner extraction and tracking algorithm for an event camera. Our primary motivation focuses on enhancing the accuracy of corner detection and tracking while ensuring computational efficiency. Firstly, according to the polarities of the events, a simple yet effective filter is applied to construct two restrictive Surface of Active Events (SAEs), named as RSAE+ and RSAE−, which can accurately represent high contrast patterns; meanwhile it filters noises and redundant events. Afterwards, a new coarse-to-fine corner extractor is proposed to extract corner events efficiently and accurately. Finally, a space, time and velocity direction constrained data association method is presented to realize corner event tracking, and we associate a new arriving corner event with the latest active corner that satisfies the velocity direction constraint in its neighborhood. The experiments are run on a standard event camera dataset, and the experimental results indicate that our method achieves excellent corner detection and tracking performance. Moreover, the proposed method can process more than 4.5 million events per second, showing promising potential in real-time computer vision applications.

Keywords: asynchronous; corner event; corner extraction; corner tracking; event camera

Citation: Duo, J.; Zhao, L. An Asynchronous Real-Time Corner Extraction and Tracking Algorithm for Event Camera. *Sensors* **2021**, *21*, 1475. https://doi.org/10.3390/s21041475

Academic Editor: Leo Marco

Received: 24 January 2021
Accepted: 16 February 2021
Published: 20 February 2021

1. Introduction

In recent years, owing to the development of computer vision technology and the enhancement of computer information processing capability, conventional frame-based cameras show important application value in several fields such as unmanned robot system, intelligent security and virtual reality. While widely adopted, frame-based cameras are not always optimal in many circumstances. For instance, when up against high velocity motions (easily causing blur) or high dynamic range scenes, frame-based cameras can hardly show robust performance. Furthermore, in a time period without motion, the images captured by frame-based cameras contain the same redundant information, which leads to a huge waste of computing resources.

As a new type of vision sensor, event cameras [1,2] are inspired by biology and operate in a very different way from frame-based cameras. Instead of capturing images at a fixed rate, event cameras respond to pixel brightness changes, called "events", as they occur and output the time, locations and signs of the events asynchronously. Compared with frame-based cameras, event cameras show outstanding properties: very high temporal resolution (not suffering from motion blur) and low latency (both in the order of microseconds), very high dynamic range (140 dB vs. 60 dB of frame-based cameras) and low power consumption. In addition, event cameras only consider the information from "events", thereby eliminating information redundancy and reducing computation. Event cameras show great potentials to overcome the challenges of high speed and dynamic range faced

by conventional frame-based cameras, which have great application value in the field of virtual reality and robotics.

Since there are numerous advantages, several recent works [3–14] focus on processing the unconventional output of event cameras and unlocking their potentials. In event-based vision, the corner is one of the most fundamental features, and corner event tracking is usually used in many applications, such as target tracking [15,16], 3D Reconstruction [17,18] and motion estimation [19,20]. In 2015, the first corner event detection and tracking algorithm was proposed in Clady et al. [21]. In this paper, a nonlinear optimization method is used to fit planes from the Surface of Active Events (SAE), and corner events are labeled as the intersection of several planes. However, as the event signal contains much noise, the performance of plane fitting is less than satisfactory. Inspired by conventional frame-based Harris corner detector [22], Vasco et al. proposed an event-based Harris corner detector called "eHarris" [23]. In this method, event streams are used to construct an artificially binary frame, and they detect the Harris corners directly on this binary frame. Although this method provides satisfactory results, it has the poor computational efficiency due to using convolutions. In Mueggler et al. [24], an event-based Features from Accelerated Segment Test (FAST) corner detector, called "eFAST", was proposed, which is inspired by the original FAST [25] and detects the corner events on SAE using only comparison operations. Because of its simple design principle, this method has high computational efficiency, but the performance of corner detection is not as effective as that in Vasco et al. [23]. Alzugaray and Chli proposed an event-based corner detection and tracking algorithm [26]. The corner detection algorithm is inspired by eFAST but it is more accurate and efficient than eFAST. However, this detector has limited changes in fundamental principle, the detection performance is still not comparable to that of eHarris. Additionally, the corner tracking algorithm associates the current corner with the latest active corner in its neighborhood, and constructs a tree-like structure to represent the trajectory for the same corner in spatio-temporal space. However, the tracking performance is not ideal due to lack of effective restrictions. In order to improve the tracking accuracy, Alzugaray and Chli processed events individually as they generated and proposed an asynchronous patch-feature tracker [27], but because of the huge computing burden, this method runs about 30× slower than the tracker in [26], and it has poor real-time computing capability. Li et al. proposed an event-based corner detection algorithm called FA-Harris [28]; in this algorithm, corner candidates are first selected by an improved eFAST and then refined by an improved eHarris detector. Although the accuracy has been effectively improved, this algorithm consumes a large amount of computation time and hardly runs in real-time due to the tedious eHarris-based method.

Driven by the requirement for effective corner event processing, in this paper, we propose an asynchronous corner extraction and tracking algorithm for the event camera. Our primary motivation focuses on enhancing the repeatability of corner detection and the accuracy of corner tracking while ensuring computational efficiency. As shown in Figure 1, the proposed method consists of three units: an SAE filter unit (green part in Figure 1), a coarse-to-fine corner extractor unit (red part in Figure 1) and a corner tracker unit (blue part in Figure 1). The main contributions of this work are the following:

- Corner events are extracted by a new coarse-to-fine corner extractor. The coarse extractor with high calculation efficiency is used to extract corner candidates, and the fine extractor-based on a box filter only processes corner candidates. It significantly decreases the calculating amount and improves the efficiency without reducing the accuracy.
- A space, time and velocity direction constrained data association method is presented to realize corner event tracking, and we associate a new arriving corner event with the latest active corner that satisfies the velocity direction constraint in its neighborhood.

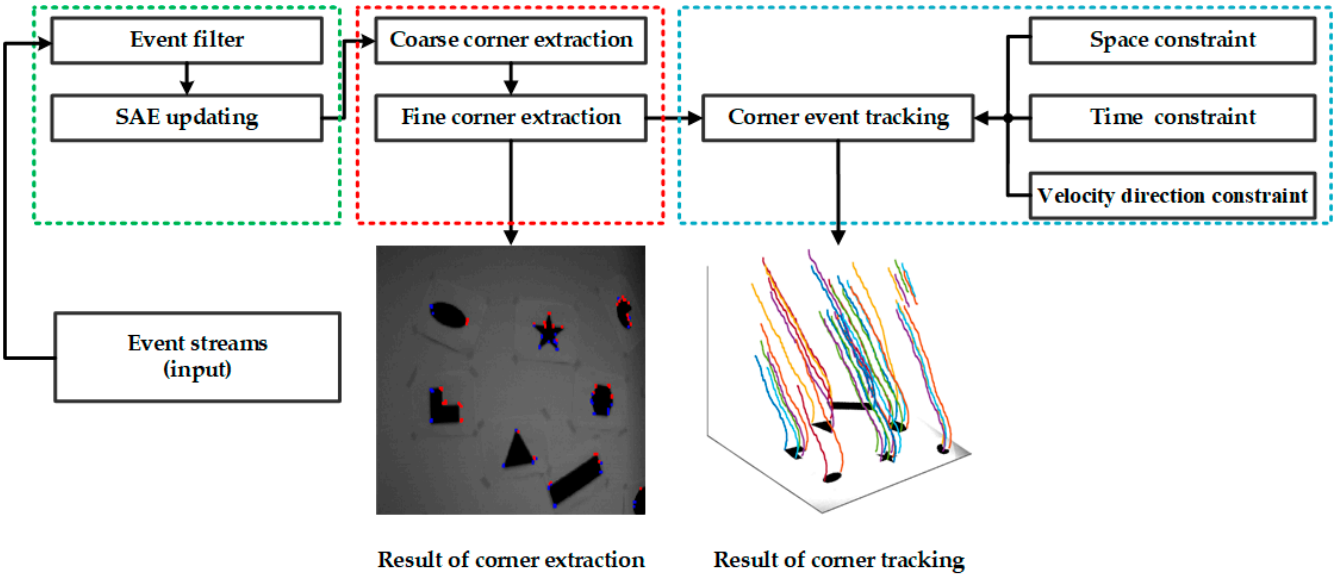


Figure 1. System overview of our proposed algorithm. The proposed algorithm consists of three units: a Surface of Active Events (SAEs) filter unit (green part), a coarse-to-fine corner extractor unit (red part) and a corner tracker unit (blue part).

The remainder of this paper is organized as follows. The proposed method is detailed in Section 2. In Section 3, we perform our method on a publicly available event camera dataset and present the experimental results. Finally, we draw some conclusions and shed light on future work in Section 4.

2. Materials and Methods

In this paper, we propose an asynchronous corner extraction and tracking algorithm for an event camera. The main work of this paper has the following. Firstly, a simple yet effective filter is applied to filter out noise and redundant events. Afterwards, we extract corner events by a new coarse-to-fine corner extractor. Finally, a space, time and velocity direction constrained data association is presented to realize corner events tracking. The proposed method is detailed in the following subsections.

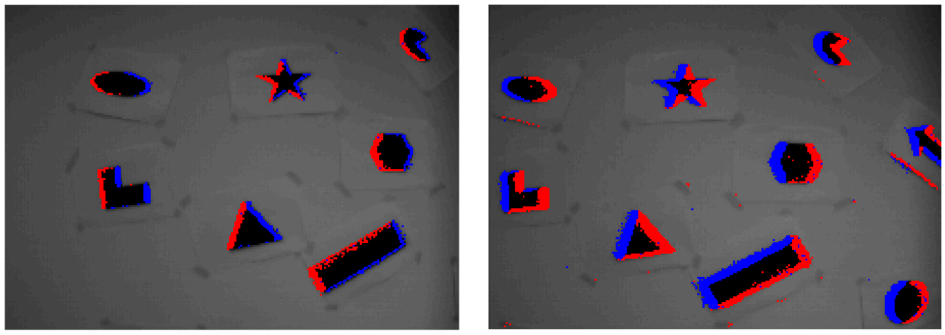
2.1. Filter of SAE

Event cameras operate in very differently from frame-based cameras; they only respond to pixel brightness changes, called "events", and output the time, locations and signs of the events asynchronously. As an implementation of the send-on-delta transmission scheme [29], an event $\boldsymbol{e} = (x, y, t, pol)$ is generated at the pixel location (x, y) at time t if the absolute difference of logarithmic brightness $I(x, y, t)$ reaches a threshold K, Formally

$$I(x, y, t) - I(x, y, t - \Delta t) = pol \cdot K \tag{1}$$

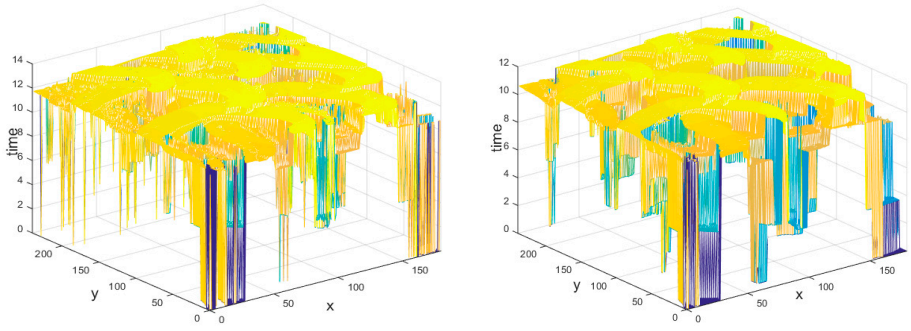
where $t - \Delta t$ denotes the time when the last event was generated at the pixel location (x, y), pol denotes the polarity of the event (the sign of the logarithmic brightness change that is either 1 or -1). Normally, when an event is generated, it will be appended asynchronously to the event streams. Since there is no concept of image frames for event cameras, a notion of SAE is proposed in [30], which is used to store the timestamp of the most recent event at each pixel. When an event $\boldsymbol{e} = (x, y, t, pol)$ arrives, the value of SAE at (x, y) is updated as $SAE(x, y) \leftarrow t$. Because of noises and hardware limitations, a sudden brightness change will generate several events at the same pixel almost instantly, so the latest timestamps stored in the SAE are not the exact time when the stimulus signals are generated. This will reduce the accuracy of corner event extraction and tracking. In order to solve this problem, a simple yet effective filter [26] is applied to construct a more restrictive SAE, named as RSAE. In the RSAE, events generated within a small time window k (typically

50 ms) at the same pixel will be ignored and not be used for updates. More precisely, when an event $\boldsymbol{e} = (x, y, t, pol)$ arrives, if $t\text{-}SAE(x, y) > k$, or if the polarity of the latest event at (x, y) differs from $\boldsymbol{e}$, the value of RSAE at (x, y) is updated as $RSAE(x, y) \leftarrow t$. Owing to the special operating principle, in the same scene, different camera motions might generate different event streams. As shown in Figure 2, the brightness of the edge pixels will change with a moving camera, and different moving directions might generate different events, especially reflecting in the polarities of events. For this reason, according to the polarities of the events, we construct two more precise RSAEs, named as RSAE+ (see Figure 3a) and RSAE− (see Figure 3b) to replace the original RSAE. The above operations have the following two main benefits. Firstly, high contrast patterns (that is, edges) can be accurately represented in both time and space when brightness changes. Secondly, the noises and redundant events can be effectively filtered, which saves considerable calculation time.



(**a**) (**b**)

Figure 2. Events generated by two different motions: (**a**) the camera moves to the right, (**b**) the camera moves to the left. The blue and red dots represent events with positive and negative polarities, respectively.



(**a**) RSAE+ (**b**) RSAE−

Figure 3. Two global Restrictive Surface of Active Events (RSAEs) constructed according to the polarities of the events: (**a**) RSAE+ constructed by the timestamp of the most recent positive event at each pixel, (**b**) RSAE− constructed by the most recent negative event at each pixel.

2.2. Corner Event Extraction

Among the existing corner event extractors, eHarris [23] provides satisfactory results but has poor computational efficiency due to the use of convolutions. Conversely, eFAST [24] has high computational efficiency, but the performance is not as effective as eHarris. FA-Harris [28] adopts a coarse-to-fine extraction strategy; in this algorithm, corner candidates are first selected by an improved eFAST detector and then refined by an improved eHarris detector. Although the accuracy has been effectively improved, this algorithm consumes a large amount of computation time and hardly runs in real-time due to the tedious eHarris-based method. For this reason, in this paper, a new coarse-to-fine corner event extraction method is proposed. We first adopt Arc* [26] to extract corner candidates from event streams and then develop a box filter-based extractor to refine

the corner candidates. Compared with the FA-Harris detector, our method significantly improves the efficiency without reducing the accuracy.

2.2.1. Coarse Corner Extraction

When an event $\boldsymbol{e} = (x, y, t, pol)$ arrives, we first update RSAE* (RSAE+ or RSAE−) according to the polarity of $\boldsymbol{e}$, by using the method in Section 2.1. Then Arc* [26] is adapted to extract corner candidates on the corresponding RSAE*. As shown in Figure 4, a moving corner pattern can generate a local RSAE* with two markedly distinct regions, and two different moving directions will create entirely different local RSAEs for the same corner pattern. Therefore, the corner candidates are extracted by searching for a continuous region of the local RSAE* with higher value than all other elements. More specifically, a 9×9 pixel-sized patch around $\boldsymbol{e}$ is selected on the RSAE*. For convenience, we only consider the pixels on two centered concentric circles with radius 3 and 4. For each circle, we search for a continuous arc with higher timestamps than all other pixels on the circle. On the inner circle (blue), the arc length l_{inner} should be within the interval of [3,6], and on the outer circle (yellow), the arc length l_{outer} should be within the interval of [4,8] (see Figure 5a). Alternatively, the arc length on the inner and outer circle should be within the interval of [10,13] and [12,16], respectively (see Figure 5b). In either case, if such an arc can be searched on both circles, the event is considered to be a corner candidate.

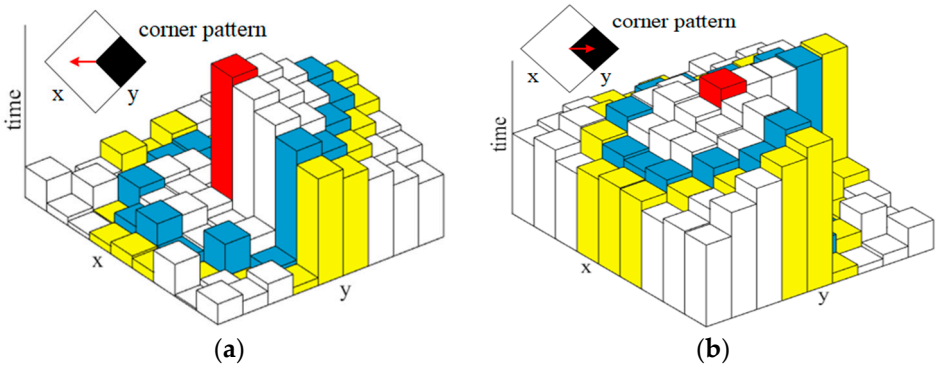


Figure 4. Illustrative examples of two entirely different local RSAEs generated by the same corner pattern with two different moving directions: (**a**) the corner pattern moves to the left, (**b**) the corner pattern moves to the right. The local RSAE is constructed by the 9×9 pixel-sized patch around $\boldsymbol{e}$ (the red element).

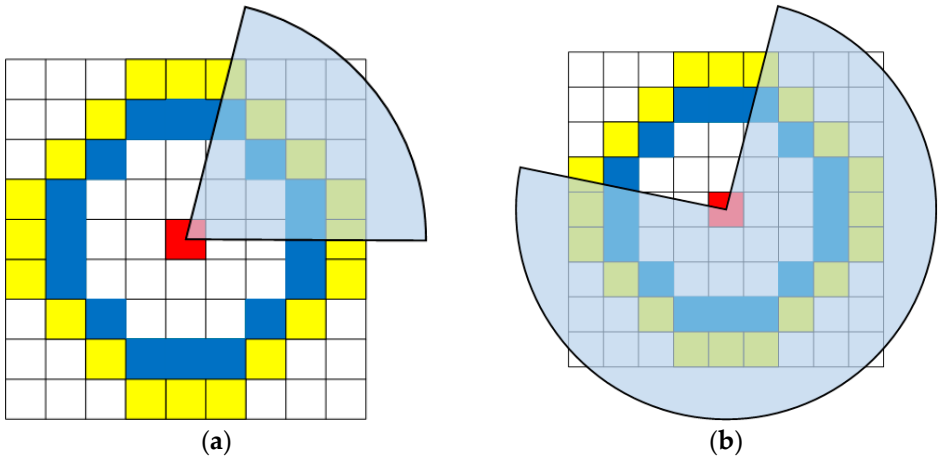


Figure 5. Schematic diagrams of coarse corner extraction in the two dimensional plane: (**a**) corresponding to the case in Figure 4a and (**b**) corresponding to the case in Figure 4b.

2.2.2. Fine Corner Extraction

The method in Section 2.2.1 can efficiently extract corner candidates and remove invalid events. In order to further enhance the accuracy, a box filter-based method is developed to refine the corner candidates. For a corner candidate, the corresponding local RSAE* is used to construct a 9×9 pixel-sized local binary patch T. We search for the largest n numbers (the latest n events) in the local RSAE* and set the corresponding elements in patch T to 1; meanwhile, the rest of the elements in patch T are set to 0. As shown in Figure 5, the total number of elements in the 9×9 pixel-sized local patch is 81, and the total number of elements on the inner circle (blue) is 16. Therefore, in our work we set

$$n = \text{round}(l_{inner} \times 81/16) \tag{2}$$

where l_{inner} is the arc length on the inner circle, which we have detailed in Section 2.2.1 and round$(\cdot)$ denotes the operation of rounding to the nearest integer. Similar to eHarris [23], we construct a 2×2 sized Hessian matrix $\boldsymbol{H}$ as follows:

$$\boldsymbol{H} = \begin{bmatrix} A & B \\ B & C \end{bmatrix} \tag{3}$$

where

$$A = \sum_{x,y} w(x,y) \cdot T_x^2(x,y) = L_{xx}(x,y) \otimes T(x,y) \tag{4}$$

$$B = \sum_{x,y} w(x,y) \cdot T_x(x,y) T_y(x,y) = L_{xy}(x,y) \otimes T(x,y) \tag{5}$$

$$C = \sum_{x,y} w(x,y) \cdot T_y^2(x,y) = L_{yy}(x,y) \otimes T(x,y) \tag{6}$$

where $T(x,y)$ is the element value of the patch T at the pixel location (x, y), $T_x(x,y)$ and $T_y(x,y)$ are the gradients of $T(x,y)$ in the x and y directions, respectively, $w(x,y)$ is a Gaussian convolution kernel with a standard deviation of 1.2. $L_{xx}(x,y)$,$L_{yy}(x,y)$ and $L_{xy}(x,y)$ are Gaussian second-order differential templates in the x-x, y-y and x-y directions, respectively, and $\otimes$ denotes the operation of convolution operation. For each corner candidate, a score R can be calculated as

$$R = \det(\boldsymbol{H}) = AC - B^2 \tag{7}$$

then we compare the score R with a threshold to determine whether this candidate is a corner event.

In practical applications, it is a complicated and time-consuming process to calculate the Hessian Matrix $\boldsymbol{H}$ through the above formulas. In order to improve the efficiency of the algorithm, we introduce box filter templates [31] to approximate Gaussian second-order differential templates. The box filter templates and the corresponding Gaussian second-order differential templates are shown in Figure 6. The elements in Gaussian second-order differential templates have different values, however, the box filter templates are only composed of several rectangular regions, and each rectangular region is filled with the same value. Therefore, elements in the Hessian matrix $\boldsymbol{H}$ can be approximated as

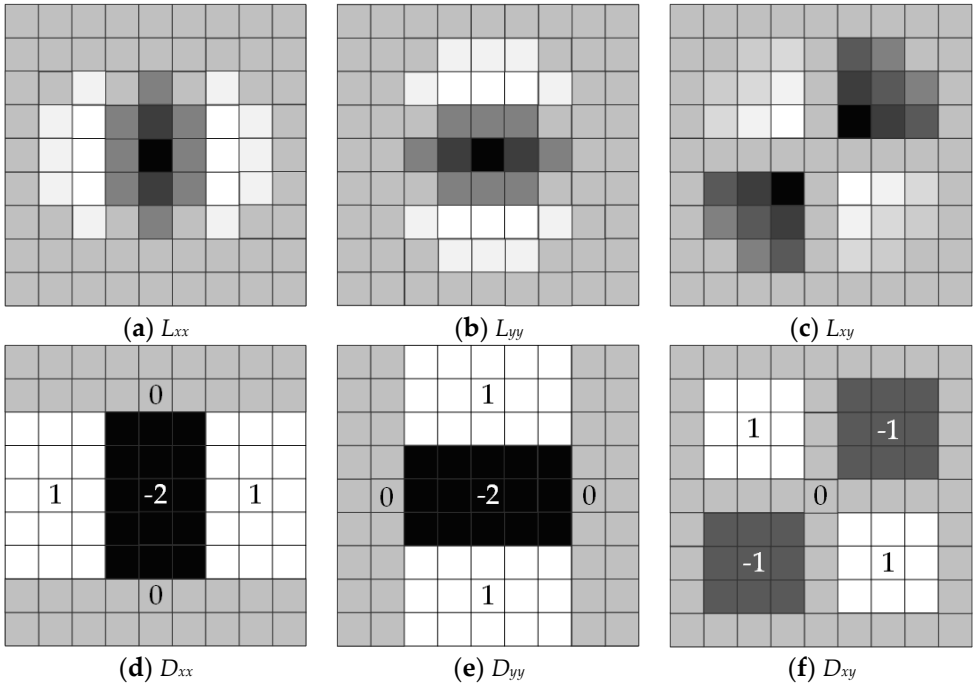
$$A \approx D_{xx}(x,y) \otimes T(x,y) \tag{8}$$

$$B \approx D_{xy}(x,y) \otimes T(x,y) \tag{9}$$

$$C \approx D_{yy}(x,y) \otimes T(x,y) \tag{10}$$

where $D_{xx}(x,y)$,$D_{xy}(x,y)$ and $D_{yy}(x,y)$ are the box filter templates corresponding to the templates in Figure 6d,e,f, respectively. The box filter templates can transform a convolution operation into the simple addition of values between different rectangular regions. Com-

pared with the FA-Harris detector, our method greatly reduces the computation amount and improves computational efficiency.



(**a**) L_{xx} (**b**) L_{yy} (**c**) L_{xy}

(**d**) D_{xx} (**e**) D_{yy} (**f**) D_{xy}

Figure 6. The 9 × 9 pixel-sized Gaussian second-order differential templates and the corresponding box filter templates. (**a**–**c**) are Gaussian second-order differential templates in the *x*-*x*, *y*-*y* and *x*-*y* directions, respectively, and (**d**−**f**) are the box filter templates corresponding to (**a**−**c**), respectively. In these templates, the elements are within the interval of [–2, 1], and dark colors correspond to small values, whereas light colors correspond to large values. The elements in Gaussian second-order differential templates have different values; for the sake of convenience, we have not labeled the exact values. The box filter templates are only composed of several rectangular regions, and each rectangular region is filled with the same value. We have labeled the exact value for each rectangular region.

2.3. Corner Event Tracking

When corner events are extracted asynchronously, how to establish data associations between these corners is a big challenge. Alzugaray and Chli [26] associate the current corner with the latest active corner in its neighborhood and construct a tree-like structure to represent the trajectory for the same corner in spatio-temporal space. However, the tracking performance of this method is not ideal due to lack of effective restrictions. In order to enhance the tracking accuracy, we improve on Alzugaray's method and propose a space, time and velocity direction constrained corner event tracking method in this paper.

When the above event $\boldsymbol{e} = (x, y, t, pol)$ is determined as a corner, as shown in Figure 7, we define the corresponding local RSAE* as a function $S_e(\cdot)$ which projects the position $\boldsymbol{p}(x, y)$ to time t

$$S_e(\boldsymbol{p}) = t \tag{11}$$

where time t is an increasing quantity. The first partial derivative of $S_e(\cdot)$ can be written as

$$\frac{\partial S_e}{\partial x}(x, y_0) = \frac{dS_e|_{y_0}}{dx}(x) = \frac{1}{v_x(x, y_0)} \tag{12}$$

$$\frac{\partial S_e}{\partial y}(x_0, y) = \frac{dS_e|_{x_0}}{dy}(y) = \frac{1}{v_y(x_0, y)} \tag{13}$$

where $S_e|_{x_0}$ and $S_e|_{y_0}$ denote $S_e(\cdot)$ restricted to $x = x_0$ and $y = y_0$, respectively. The gradient of $S_e(\cdot)$ can be written as

$$\nabla S_e = \left(\frac{1}{v_x}, \frac{1}{v_y}\right)^T \tag{14}$$

the gradient vector ∇S_e measures the change of time versus position, and it is also the inverse of the velocity vector. In order to estimate the gradient vector ∇S_e robustly, we assume that the local velocity vector is constant over a very small time on the local RSAE*. This is also equal to the assumption that the elements in the local RSAE* with higher timestamps (yellow in Figure 7) is a local plane, because the first partial derivative of $S_e(\cdot)$ is the inverse of the velocity vector, and a constant local velocity vector produces a constant change in $S_e(\cdot)$. Assuming the corner event $\boldsymbol{e} = (x, y, t, pol)$ belongs to a local plane with parameters $N = (a, b, c, d)^T$, it satisfies

$$(a\ b\ c\ d)\begin{pmatrix} x \\ y \\ t \\ 1 \end{pmatrix} = 0 \tag{15}$$

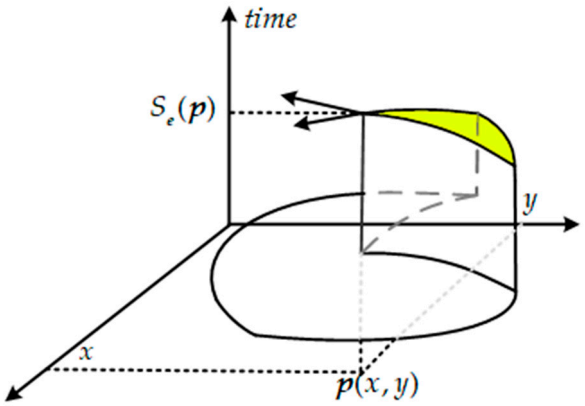


Figure 7. Schematic diagram of the velocity direction computation. The velocity vector is equal to the inverse of the first partial derivative of RSAE.

In addition, the elements in the local RSAE* with higher timestamps also satisfy Equation (15), and the parameters $N = (a, b, c, d)^T$ can be solved by a nonlinear least squares problem as

$$\underset{N}{\operatorname{argmin}} \sum_i \left| N^T \begin{pmatrix} x_i \\ y_i \\ t_i \\ 1 \end{pmatrix} \right|^2 \tag{16}$$

then the velocity vector can be calculated by

$$v^T = (v_x, v_y)^T = \left(-\frac{c}{a}, -\frac{c}{b}\right)^T \tag{17}$$

As shown in Figure 8, we associate a new corner event $\boldsymbol{e} = (x, y, t, pol)$ with the latest active corner that satisfies the velocity direction constraint in its neighborhood. The neighborhood is defined as a region with a maximum range of d_{con} pixels centered around $\boldsymbol{e}$; in our work we set $d_{con} = 5$. We also introduce a time constraint, so that corners more than t_{con} seconds later than $\boldsymbol{e}$ in the neighborhood will not be considered, which can effectively prevent data association with older corners. In our work we set $t_{con} = 0.1$. In addition to the time and space constraints, we add a velocity direction constraint for accurate tracking. More precisely, for a new arriving corner event $\boldsymbol{e}$, we firstly search for the corners that satisfy

the above time and space constraints in its neighborhood, then sort them in chronological order and store them in a container C_{neigh}. For each corner event $\boldsymbol{e}' = (x', y', t', pol')$ in C_{neigh}, the connection vector $\boldsymbol{c}'$ between $\boldsymbol{e}$ and $\boldsymbol{e}'$ can be calculated as $\boldsymbol{c}' = (x - x', y - y')^T$. Suppose we denote the velocity vector of $\boldsymbol{e}'$ as $\boldsymbol{v}'$, the angle between $\boldsymbol{v}'$ with $\boldsymbol{c}'$ can be calculated as

$$\theta = \arccos \frac{\boldsymbol{v}' \cdot \boldsymbol{c}'}{|\boldsymbol{v}'||\boldsymbol{c}'|} \tag{18}$$

in our work, $\boldsymbol{e}'$ is considered to satisfy the velocity vector constraint if $\theta < 5$ degrees. Then we associate $\boldsymbol{e}$ with the latest active corner event that satisfies the above constraint in C_{neigh} and realize an accurate tracking of corner events.

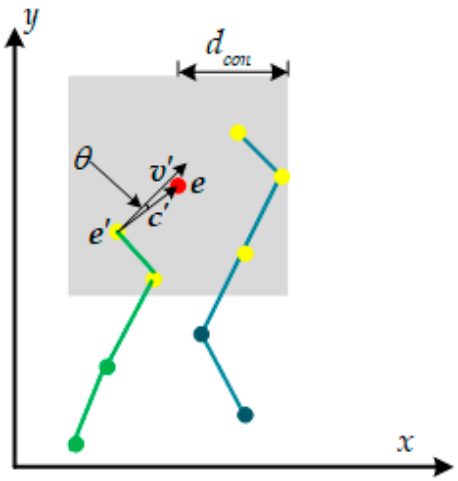


Figure 8. Schematic diagram of the proposed data association method. For a new arriving corner event $\boldsymbol{e}$ (red dot), we firstly search for the corners that satisfy the time and space constraints (yellow dots) and then search for the corner associated with the latest active corner that satisfies the velocity direction constraint. Green lines and blue lines indicate trajectories of two different corners.

3. Results

The proposed algorithm is performed on the publicly available event camera dataset [32]. This dataset is recorded by a DAVIS240C, which captures both events and intensity images with the resolution of 240 × 180 pixels. Our algorithm only processes the event streams, and the intensity images are adopted to collect ground truth. Because of space limitations, we select four representative scenes (shapes, dynamic, poster and boxes) with increasing complexity and event rate. Note that in order to ensure the fairness of the experiments, the selected scenes used in our experiments are same as those in Alzugaray and Chli [26]. We compare our algorithm with other advanced algorithms in term of extraction and tracking accuracy, and computational efficiency. In addition, the experimental results are analyzed. The following section details the ground truth collection method and experiment results.

3.1. Ground Truth

Since there is no data directly reflecting the actual behavior of the DAVIS240C in this dataset, how to collect ground truth for evaluating the performance is one of the difficulties in our work. In references [21,23], corner events are manually determined and labelled in event streams, however, these methods are only applicable to clear corners in simple scenes due to tedious manual work. In Mueggler et al. [24], an automatic determining and labelling method is proposed to calculate the spatiotemporal coordinates of the corner events, however, this method suffers from tremendous missed detections. Similar to the method in Alzugaray and Chli [26], the available intensity images are used to collect ground truth in this paper, and we use original Harris [22] to extract corners and track the corners by KLT [33]. In order to match the temporal resolution of the corner events, we

adopt a cubic splines method to interpolate corresponding coordinates in the image plane. In our accuracy metrics, we only concern the events which are not filtered in Section 2.1. As shown in Figure 9, corners only generated within 5 pixel-sized neighborhood of the KLT-based tracking are concerned in our metrics. Since the selected dataset is recorded with increasing velocity, we just select the data of the first 10 s, which can reduce the influence on ground truth caused by high speed motion blur.

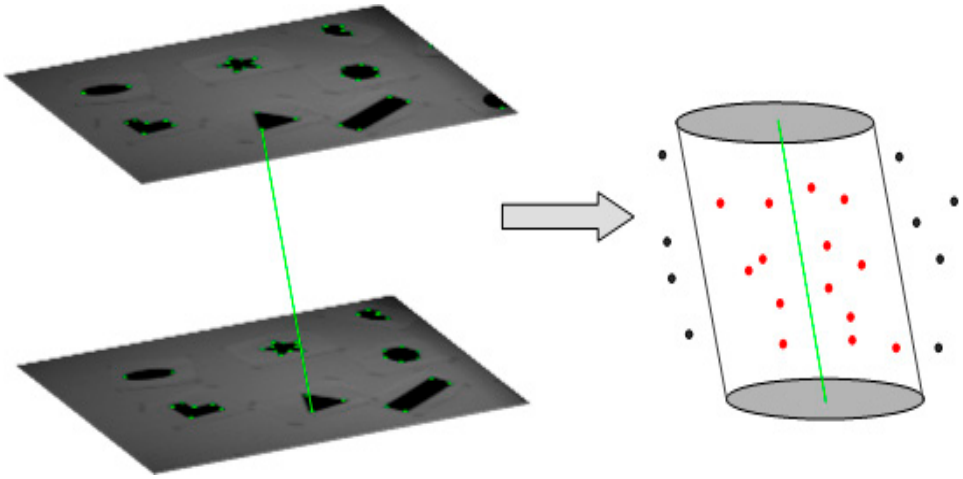


Figure 9. Schematic diagram of the ground truth collection method. The green corners are extracted by the original Harris method and tracked by KLT on intensity images. The solid green line is the trajectory of one of the corners. The 5 pixel-sized neighborhood of the trajectory is enlarged and represented as an oblique cylinder. In our metrics, we only concern the events within 5 pixel-sized neighborhood of the KLT-based tracking (events in the oblique cylinder).

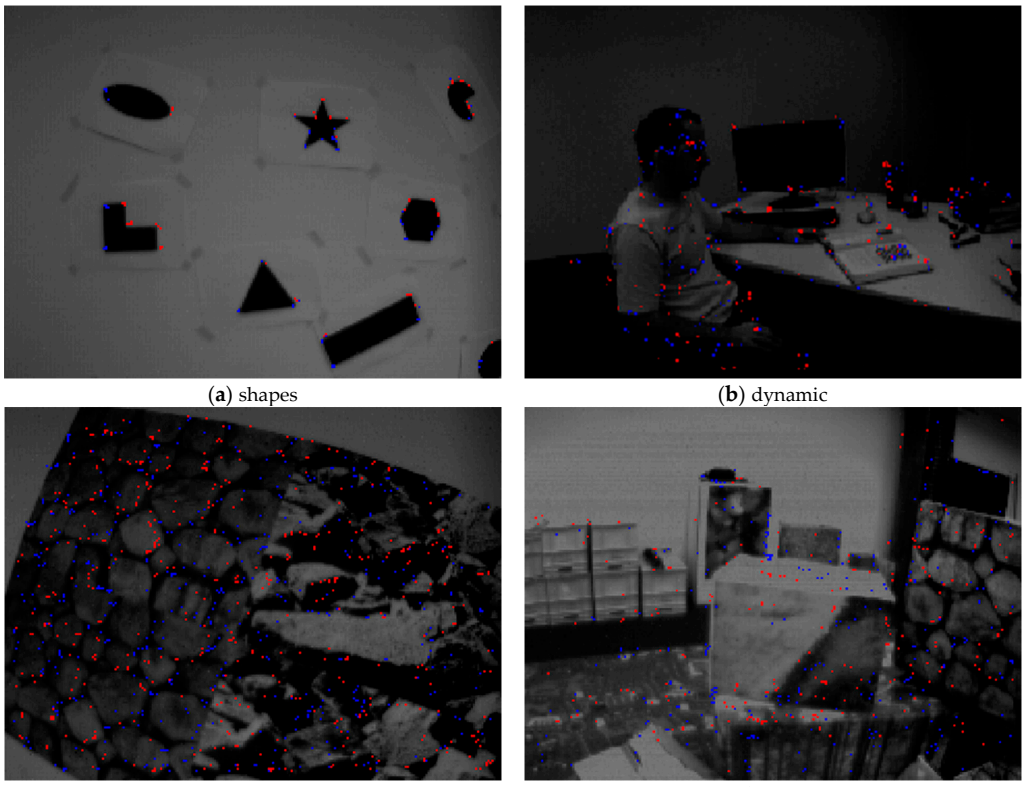
3.2. Experimental Evaluation

3.2.1. Corner Extraction Performance

Illustrative examples of corner event extraction by our method are shown in Figure 10. For each scene, we synthesize all corner events within 100 milliseconds and display on an intensity image. In Figure 10, different colors represent different polarities of the corner events; red and blue represent negative and positive respectively. Similar to [26], we adopt True Positive Rate (TPR) and False Positive Rate (FPR), a standard framework of binary classification, to evaluate the performance of the different extractors. A corner event is marked as True Positive (TP) if it is within 3.5 pixels of the KLT-based tracking, or False Positive (FP) if it is within the interval of [3.5, 5] pixels. Conversely, an event which is not considered as a corner but is within 3.5 pixels of the KLT-based tracking is marked as False Negative (FN), or True Negative (TN) if it is within the interval of [3.5, 5] pixels. TPR is calculated as $TPR = TP/(TP + FN)$, and FPR is calculated as $FPR = FP/(FP + TN)$. The TPR and FPR of different corner event extractors for different scenes are shown in Tables 1 and 2, respectively. Table 3 reports the Corner Event Rate (CER) of different extractors in four different scenes. CER is defined as the total number of corner events versus the total number of events.

The statistical results show that in a simple scene (shapes) all five methods have higher CER, TPR and FPR scores than that in complex scenes (dynamic, poster and boxes). It is because that there are more events in complex scenes than that in simple scene, and numerous events are considered as non-corner events in complex scenes. It can be seen from Table 3 that Arc* has higher CER scores than other extractors, that is to say, Arc* considers more events as corners. Additionally, Arc* has lower TPR and higher FPR scores, so it can be inferred that Arc* slightly enhances the CER at great sacrifice of accuracy; therefore, the corner events extracted by Arc* contain more incorrect detections. eFAST has the lowest CER scores because it only considers the condition in Figure 5a and ignores the condition in Figure 5b; therefore, many corners are not detected by eFAST. Owing to

the coarse-to-fine corner extraction strategy, the CER scores of our method are comparable to the FA-Harris method but are slightly lower than the eHarris method; note that our method achieves results about 20× faster than eHarris. As shown in Tables 1 and 2 that the TPR scores of our method are comparable to eHarris and FA-Harris, and they are obviously better than the other two algorithms. Moreover, our method and the FA-Harris method have lower FPR scores than the other three extractors. That is to say, our method and the FA-Harris method have better precision performance then the above five extractors, which is largely due to the coarse-to-fine corner extraction strategy. Because we develop a box filter-based method to replace the tedious eHarris-based method in fine corner extraction process, our method achieves results about 3× faster than the FA-Harris method. Compared with FA-Harris, our method significantly decreases the calculating amount and improves the efficiency without reducing the accuracy.



(**a**) shapes (**b**) dynamic

(**c**) poster (**d**) boxes

Figure 10. Illustrative examples of corner event extraction by our method in four different scenes. We synthesize all corner events within 100 milliseconds and display on an intensity image. Red and blue dots represent negative and positive polarities of the corner events, respectively.

Table 1. The True Positive Rate (%) of different extractors in four different scenes. The best performance across each experiment and metric is highlighted in bold.

Scenes	Algorithm				
	eFAST [24]	**Arc * [26]**	**eHarris [23]**	**FA-Harris [28]**	**Our Method**
shapes	33.21	45.31	57.52	57.64	**57.83**
dynamic	21.74	25.70	**46.18**	45.85	45.97
poster	15.10	22.82	42.72	**43.31**	43.17
boxes	12.29	17.65	39.36	**40.03**	39.97

Table 2. The False Positive Rate (%) of different extractors in four different scenes. The best performance across each experiment and metric is highlighted in bold.

Scenes	Algorithm				
	eFAST [24]	**Arc* [26]**	**eHarris [23]**	**FA-Harris [28]**	**Our Method**
shapes	10.08	12.47	9.39	5.42	**5.21**
dynamic	5.57	7.61	5.43	**3.06**	3.19
poster	5.41	7.08	5.10	**1.87**	1.96
boxes	4.40	6.17	4.25	1.68	**1.54**

Table 3. The Corner Event Rate (%) of different extractors in four different scenes.

Scenes	Algorithm				
	eFAST [24]	**Arc * [26]**	**eHarris [23]**	**FA-Harris [28]**	**Our Method**
shapes	8.82	13.49	11.28	10.17	10.26
dynamic	5.91	8.65	7.36	6.29	6.16
poster	5.72	9.43	7.74	6.80	6.62
boxes	5.16	8.12	6.67	6.02	6.17

3.2.2. Corner Tracking Performance

Illustrative examples of corner event tracking by our method are shown in Figure 11. Similar to [26], we adopt Mean Absolute Error (MAE), Valid Track Rate (VTR) and Mean Track Lifetime (MTL) to evaluate the performance and summarize the results in Table 4. MAE is defined as the mean absolute distance between event-based tracking and KLT- based tracking. We consider event-based tracking as valid tracking if the MAE is within 5 pixels, otherwise we consider it as invalid tracking. For each scene, VTR is defined as the total number of valid tracking versus the total number of corner event tracking. Our tracking algorithm associates a new arriving corner event with the latest active corner that satisfies the velocity direction constraint in its neighborhood, so it is based on the assumption that the corner events are continuously and steadily detected. If the assumption is broken, the new arriving corner event will be considered as a new corner or even generate an incorrect data association. MTL is defined as the mean duration for the corner event tracking which matches the same intensity-based tracking validly. As shown in Table 4, in four different scenes, our method achieves less MAE and higher VTR than the algorithm in [26]. This is because our method adds a velocity direction constraint on Alzugaray's method, which can effectively eliminate incorrect data associations. Owing to this strict constraint, the MTL of our method is slightly less than that in [26]. Note that our method effectively enhances the tracking accuracy at slight sacrifice of MTL.

Table 4. The tracking performance evaluation of our method. The best performance across each experiment and metric is highlighted in bold.

Scenes	Algorithm	MAE (Pixels)	VTR (%)	MTL (s)
shapes	method in [26]	2.25	82.39	**1.13**
	our method	**1.57**	**83.11**	1.07
dynamic	method in [26]	2.64	60.21	**0.77**
	our method	**1.61**	**78.28**	0.71
poster	method in [26]	2.82	54.64	**0.64**
	our method	**1.51**	**74.43**	0.62
boxes	method in [26]	2.88	53.25	**0.67**
	our method	**1.74**	**75.72**	0.64

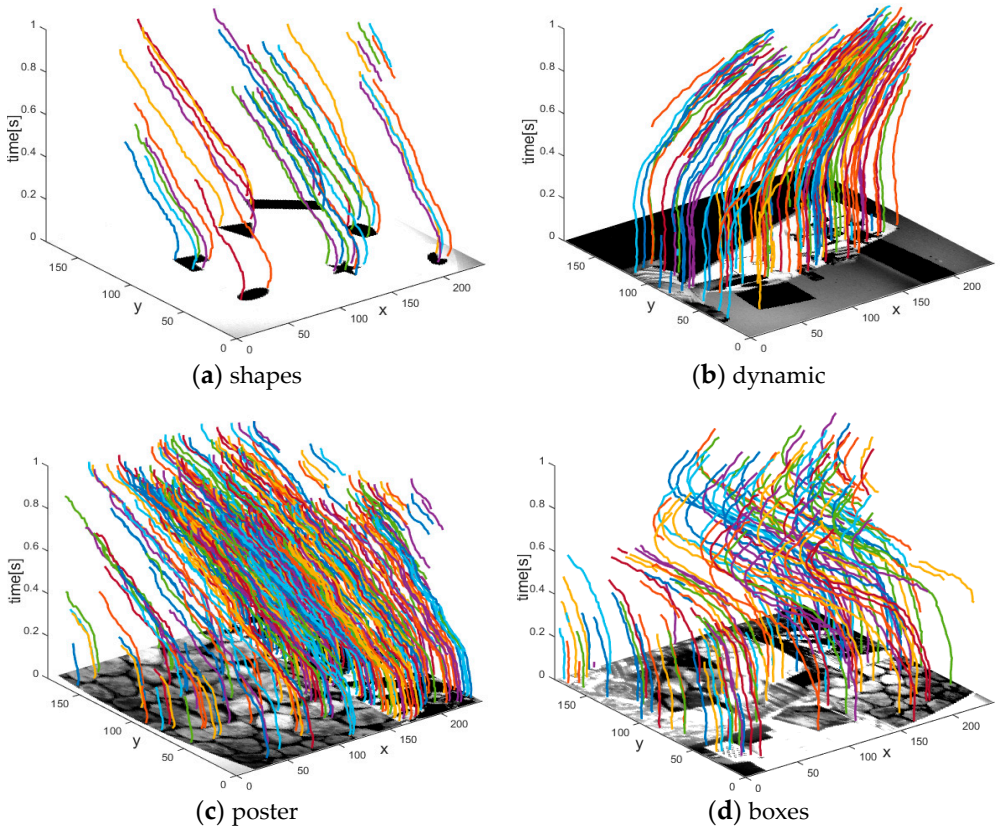


Figure 11. Illustrative examples of corner event tracking by our method in four different scenes. Date associations for different corner events are marked as solid lines in different colors. We synthesize the trajectories on the corresponding intensity image for viewing convenience.

3.2.3. Computational Performance

All the above algorithms are implemented by single threaded C++ programs and run on an Intel(R) Core(TM) i7 CPU with 2.80 GHz and 16 GB of RAM. Table 5 presents the average time consumption of a single event and the maximum processing rate in millions of events per second (Mev/s) for each algorithm. Intuitively, our method has a good performance in computational consumption. On average, our proposed extractor achieves results about 20× faster than the eHarris method, but about 1.5× slower than Arc*. This improvement is because we propose a coarse-to-fine corner extraction method; the coarse extraction with high calculation efficiency is used to extract corner candidates, and the fine extraction only processes corner candidates, not all events. Although our extractor is more computationally expensive than Arc*, the accuracy performance is significantly better. Our proposed extractor also achieves results about 3× faster than the FA-Harris method, because we develop a box filter-based method to replace the tedious eHarris-based method in the fine corner extraction process. Compared with FA-Harris, our method significantly decreases the calculating amount and improves the efficiency without reducing the accuracy. Table 5 also reports the average time consumption of corner event tracking. Superficially, the proposed tracking algorithm takes much more time than extraction; note that only corner events, not all events, are concerned in the tracking process, in fact the time consumption of the tracking process accounts for less than 45% of the total computation time. Furthermore, a single corner tracking time of our method is slower than that in [26], but because of the fine extraction process, our algorithm extracts relatively less but more accurate corner events than Arc*, which also saves a lot of computing time in the whole tracking process. Therefore, our work has great potential in real-time computer vision applications.

Table 5. The computational performance of different corner event extraction and tracking algorithms.

Algorithm		Time per Event (μs/event)	Max. Event Rate (Mev/s)
Extraction method	eHarris [23]	5.36	0.17
	eFAST [24]	0.16	6.21
	Arc * [26]	0.13	7.63
	FA-Harris [28]	0.68	1.49
	our method	0.22	4.52
Tracking method	method in [26]	2.14	0.44
	our method	3.98	0.25

4. Conclusions

We propose an asynchronous real-time corner extraction and tracking algorithm for event cameras and show its excellent accuracy performance and good computational efficiency. In our algorithm, corner events are asynchronously extracted by a coarse-to-fine extractor and associated with the latest active corners that satisfy the velocity direction constraints in their neighborhood. Compared with the method in [26], our method effectively enhances the accuracy at slight sacrifice of computational efficiency. Experiment results also indicate that the proposed method can process more than 4.5 million events per second, showing great potential in real-time computer vision applications. Our further interest lies in applying our work to SLAM in challenging environments (high velocity movements or high dynamic range scenes) while existing frame-based SLAM algorithms have limitations.

Author Contributions: Conceptualization, J.D. and L.Z.; methodology, J.D. and L.Z.; software, J.D.; validation, J.D. and L.Z.; formal analysis, J.D. and L.Z.; investigation, J.D. and L.Z.; resources, L.Z.; data curation, L.Z.; writing—original draft preparation, J.D.; writing—review and editing, J.D.; visualization, J.D.; supervision, L.Z.; project administration and funding acquisition, L.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China (Grant No. 41874034), the National Science and Technology Major Project of the National Key R&D Program of China (Grant No. 2016YFB0502102), and the Beijing Natural Science Foundation (Grant No. 4202041), the Aeronautical Science Foundation of China.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: http://rpg.ifi.uzh.ch/davis_data.html.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Lichtsteiner, P.; Posch, C.; Delbruck, T. A 128× 128 120 dB 15 µs latency asynchronous temporal contrast vision sensor. *IEEE J. Solid State Circuits* **2008**, *42*, 566–576.
2. Brandli, C.; Berner, R.; Yang, M.; Liu, S.-C.; Delbruck, T. A 240 × 180 130 dB 3 µs latency global shutter spatiotemporal vision sensor. *IEEE J. Solid State Circuits* **2014**, *49*, 2333–2341.
3. Tapiador-Morales, R.; Maro, J.-M.; Jimenez-Fernandez, A.; Jimenez-Moreno, G.; Benosman, R.; Linares-Barranco, A. Event-Based Gesture Recognition through a Hierarchy of Time-Surfaces for FPGA. *Sensors* **2020**, *20*, 3404.
4. He, W.; Huang, J.; Wang, T.; Lin, Y.; He, J.; Zhou, X.; Li, P.; Wang, Y.; Wu, N.; Shi, C. A High-Speed Low-Cost VLSI System Capable of On-Chip Online Learning for Dynamic Vision Sensor Data Classification. *Sensors* **2020**, *20*, 4715.
5. Savran, A.; Bartolozzi, C. Face Pose Alignment with Event Cameras. *Sensors* **2020**, *20*, 7079.
6. Feng, Y.; Lv, H.; Liu, H.; Zhang, Y.; Xiao, Y.; Han, C. Event density based denoising method for dynamic vision sensor. *Appl. Sci.* **2020**, *10*, 2024.

7. Khan, N.; Martini, M.G. Bandwidth modeling of silicon retinas for next generation visual sensor networks. *Sensors* **2019**, *19*, 1751.
8. Brandli, C.; Strubel, J.; Keller, S.; Scaramuzza, D.; Delbruck, T. ELiSeD—an event-based line segment detector. In Proceedings of the 2th International Conference on Event-Based Control, Communication and Signal Processing, Krakow, Poland, 13–15 June 2016; pp. 41–77.
9. Glover, A.; Bartolozzi, C. Event-driven ball detection and gaze fixation in clutter. In Proceedings of the 29th IEEE/RSJ International Conference on Intelligent Robots and Systems, Daejeon, Korea, 9–14 October 2016; pp. 2203–2208.
10. Valeiras, D.R.; Lagorce, X.; Clady, X.; Bartolozzi, C.; Ieng, S.-H.; Benosman, R. An asynchronous neuromorphic event-driven visual part based shape tracking. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 3045–3059.
11. Ramesh, B.; Zhang, S.; Lee, Z.W.; Gao, Z.; Orchard, G.; Xiang, C. Long-term object tracking with a moving event camera. In Proceedings of the 29th British Machine Vision Conference, Newcastle, UK, 3–6 September 2018; pp. 1781–1792.
12. Linares-Barranco, A.; Perez-Pena, F.; Moeys, D.P.; Gomez-Rodriguez, F.; Jimenez-Moreno, G.; Liu, S.-C.; Delbruck, T. Low latency event-based filtering and feature extraction for dynamic vision sensors in real-time FPGA applications. *IEEE Access* **2019**, *7*, 134926–134942.
13. Liu, S.-C.; Delbruck, T.; Indiveri, G.; Whatley, A.; Douglas, R. *Event-Based Neuromorphic Systems*; John Wiley & Sons Ltd.: Hoboken, NJ, USA, 2016.
14. Miskowicz, M. *Event-Based Control and Signal Processing*; CRC Press Inc.: Boca Raton, FL, USA, 2016.
15. Li, K.; Shi, D.; Zhang, Y.; Li, R.; Qin, W.; Li, R. Feature tracking based on line segments with the DAVIS. *IEEE Access* **2019**, *7*, 110874–110883.
16. Mitrokhin, A.; Fermüller, C.; Parameshwara, C.; Aloimonos, Y. Event-based moving object detection and tracking. In Proceedings of the 31th IEEE/RSJ International Conference on Intelligent Robots and Systems, Madrid, Spain, 1–5 October 2018; pp. 241–248.
17. Rebecq, H.; Gallego, G.; Mueggler, E.; Scaramuzza, D. EMVS: Event-based multi-view stereo-3D reconstruction with an event camera in real-time. *Int. J. Comput. Vision* **2018**, *126*, 1394–1414.
18. Kim, H.; Leutenegger, S.; Davison, A.J. Real-time 3D reconstruction and 6-DoF tracking with an event camera. In Proceedings of the 14th European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 349–364.
19. Mueggler, E.; Gallego, G.; Scaramuzza, D. Continuous-time trajectory estimation for event-based vision sensors. In Proceedings of the 11th International Conference on Robotics: Science and Systems, Rome, Italy, 13–17 July 2015; pp. 196–205.
20. Gallego, G.; Scaramuzza, D. Accurate angular velocity estimation with an event camera. *IEEE Robot. Autom. Lett.* **2017**, *2*, 632–639.
21. Clady, X.; Ieng, S.-H.; Benosman, R. Asynchronous event-based corner detection and matching. *Neural Netw.* **2015**, *66*, 91–106.
22. Harris, C.; Stephens, M. A combined corner and edge detector. In Proceedings of the 4th Alvey Vision Conference, Manchester, UK, 31 August–2 September 1988; pp. 147–151.
23. Vasco, V.; Glover, A.; Bartolozzi, C. Fast event-based harris corner detection exploiting the advantages of event-driven cameras. In Proceedings of the 29th IEEE/RSJ International Conference on Intelligent Robots and Systems, Daejeon, Korea, 9–14 October 2016; pp. 4144–4149.
24. Mueggler, E.; Bartolozzi, C.; Scaramuzza, D. Fast event-based corner detection. In Proceedings of the 28th British Machine Vision Conference, London, UK, 4–7 September 2017; pp. 1–12.
25. Rosten, E.; Drummond, T. Machine learning for high-speed corner detection. In Proceedings of the 9th European Conference on Computer Vision, Graz, Austria, 7–13 May 2016; pp. 430–443.
26. Alzugaray, I.; Chli, M. Asynchronous corner detection and tracking for event cameras in real time. *IEEE Robot. Autom. Mag.* **2018**, *3*, 3177–3184.
27. Alzugaray, I.; Chli, M. Asynchronous multi-hypothesis tracking of features with event cameras. In Proceedings of the 7th International Conference on 3D Vision, Quebec, QC, Canada, 16–19 September 2019; pp. 269–278.
28. Li, R.; Shi, D.; Zhang, Y.; Li, K.; Li, R. FA-Harris: A fast and asynchronous corner detector for event cameras. In Proceedings of the 32th IEEE/RSJ International Conference on Intelligent Robots and Systems, Macau, China, 3–8 November 2019; pp. 1372–1378.
29. Miskovicz, M. Send-on-delta concept: An event-based data reporting Strategy. *Sensors* **2006**, *6*, 49–63.
30. Benosman, R.; Clercq, C.; Lagorce, X.; Ieng, S.-H.; Bartolozzi, C. Event-based visual flow. *IEEE Trans. Neural Netw. Learn. Syst.* **2014**, *25*, 407–417.
31. Bay, H.; Ess, A.; Tuytelaars, T.; Van Gool, L. Speeded up robust features (SURF). *Comput. Vis. Image. Und.* **2008**, *110*, 346–359.
32. Mueggler, E.; Rebecq, H.; Gallego, G.; Delbruck, T.; Scaramuzza, D. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *Int. J. Robot. Res.* **2017**, *36*, 142–149.
33. Lucas, B.D.; Kanade, T. An iterative image registration technique with an application to stereo vision. In Proceedings of the 7th International Joint Conference on Artificial Intelligence, Vancouver, BC, Canada, 24–28 August 1981; pp. 674–679.