Research Article

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

International Journal of Advanced Robotic Systems
July-August 2021: 1–13

DOI: 10.1177/17298814211027028
journals.sagepub.com/home/arx
SAGE

Asynchronous event feature generation and tracking based on gradient descriptor for event cameras

Ruoxiang Li[1], Dianxi Shi[2,3], Yongjun Zhang[2], Ruihao Li[2,3] and Mingkun Wang[1]

Abstract

Recently, the event camera has become a popular and promising vision sensor in the research of simultaneous localization and mapping and computer vision owing to its advantages: low latency, high dynamic range, and high temporal resolution. As a basic part of the feature-based SLAM system, the feature tracking method using event cameras is still an open question. In this article, we present a novel asynchronous event feature generation and tracking algorithm operating directly on event-streams to fully utilize the natural asynchronism of event cameras. The proposed algorithm consists of an event-corner detection unit, a descriptor construction unit, and an event feature tracking unit. The event-corner detection unit addresses a fast and asynchronous corner detector to extract event-corners from event-streams. For the descriptor construction unit, we propose a novel asynchronous gradient descriptor inspired by the scale-invariant feature transform descriptor, which helps to achieve quantitative measurement of similarity between event feature pairs. The construction of the gradient descriptor can be decomposed into three stages: speed-invariant time surface maintenance and extraction, principal orientation calculation, and descriptor generation. The event feature tracking unit combines the constructed gradient descriptor and an event feature matching method to achieve asynchronous feature tracking. We implement the proposed algorithm in C++ and evaluate it on a public event dataset. The experimental results show that our proposed method achieves improvement in terms of tracking accuracy and real-time performance when compared with the state-of-the-art asynchronous event-corner tracker and with no compromise on the feature tracking lifetime.

Keywords

Robotics, event camera, feature descriptor, feature tracking, SLAM

Date received: 16 February 2021; accepted: 27 May 2021

Topic Area: Robotics software Design and Engineering
Topic Editor: David Portugal
Associate Editor: D.J Lee

Introduction

Over the past several years, simultaneous localization and mapping (SLAM) has been widely studied and developed for augmented and virtual reality, self-driving cars, and unmanned aerial vehicles.[1] The combination of depth learning and SLAM[2,3] has also become a hot research topic at present. But, due to the complexity of the real environment, existing visual SLAM systems using single vision sensor are still faced with many problems, such as tracking

[1] National University of Defense Technology, Changsha, China
[2] Artificial Intelligence Research Center (AIRC), National Innovation Institute of Defense Technology (NIIDT), Beijing, China
[3] Tianjin Artificial Intelligence Innovation Center (TAIIC), Tianjin, China

Corresponding author:
Dianxi Shi, Artificial Intelligence Research Center (AIRC), National Innovation Institute of Defense Technology (NIIDT), Beijing 100166, China.
Email: dxshi@nudt.edu.cn

failure. To enhance the robustness of SLAM systems, many researchers fuse the data derived from two or more sensors, such as cameras, Lidar, GPS, IMU, and so on.[4,5] However, there are still many challenges to these systems when faced with challenging scenes, such as high-speed motion and high dynamic range. Recently, bioinspired vision sensors[6,7] have aroused many researchers' interest and have become a hot research topic for robotics and computer vision. Event cameras respond to local pixel-level brightness changes, transmitting asynchronous events only when brightness changes are detected rather than frames with a fixed time interval, intrinsically different from standard cameras. Each event is a tuple $[x, y, t, p]$, where (x, y) is the coordinate in the imaging plane, t is the triggered timestamp, and p is the sign of the brightness change. The advantages of event cameras include low latency, low power consumption, high dynamic range, high temporal resolution, and no motion blur. Therefore, event cameras have the potential to help SLAM systems to overcome the limitations in challenging environments. For example, owing to the natural ability of being sensitive to dynamic objects, event cameras could be used for object detection and tracking,[8] and have great potential for improving the performance of the SLAM pipeline in dynamic environments,[9] for which we might have to filter the dynamic moving objects from the raw image data for better SLAM performance.[10,11]

Unfortunately, the asynchronous events from event cameras are intrinsically different from the intensity images, so standard computer vision methods cannot be directly applied for event cameras.[12] Researchers have to explore new methods to bring event cameras' potential into full play. Until now, there have been a large amount of research efforts focused on event cameras in multiple directions, such as SLAM,[9,13,14] segmentation,[15,16] reconstruction for visual information,[17,18,19] and control for unmanned aerial vehicles.[20,21] More related research contents can be found from the survey articles[12,22] and the list of event-based vision resources (https://github.com/uzh-rpg/event-based_vision_resources).

As one of the basic methods in SLAM, feature-based SLAM methods extract features from intensity frames, and each feature is associated with a descriptor, such as scale-invariant feature transform (SIFT), speeded-up robust features (SURF), oriented FAST and rotated BRIEF (ORB), and so on. The extracted descriptor preserves the information of the local area around the feature point and provides a quantitative comparison with other feature points.[5] Then, data association is performed to associate similar features to complete feature tracking tasks. To the best of our knowledge, there is still no visual SLAM system using asynchronous feature tracking method, which is proposed for event cameras. Driven by the demand for an efficient asynchronous feature tracking method for subsequent SLAM system based on event cameras, we propose an asynchronous event feature generation and tracking algorithm working directly on event-streams. The proposed algorithm consists of an event-corner detection unit, a descriptor construction unit, and an event feature tracking unit. The results of asynchronous event feature tracking are shown in Figure 1. The main contributions of this article can be summarized as follows:

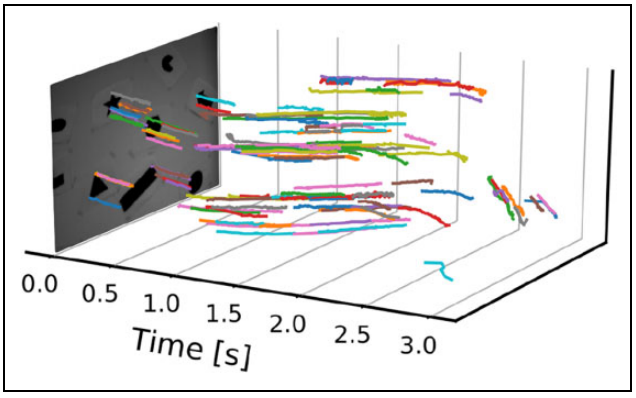

Figure 1. Our event feature generation and tracking algorithm works directly on asynchronous event-streams based on our proposed gradient descriptor. This figure shows the event feature tracking results in the spatiotemporal space with our proposed algorithm on shapes scene of the public event camera dataset.[23] Different colors indicate different tracked event features.

- We propose an asynchronous event feature generation and tracking algorithm, which can work directly on asynchronous event-streams. The proposed algorithm includes an event-corner detection unit, a descriptor construction unit, and an event feature tracking unit.
- We address a novel asynchronous event feature gradient descriptor. The descriptor can be constructed by speed-invariant time surface (SITS)[24] maintenance and extraction, principal orientation calculation, and descriptor generation. The descriptor is used to represent the distribution of the local gradient information for event-corners and help to achieve quantitative measurements of similarity between event feature pairs.
- We implement our proposed algorithm in C++ and evaluate it on the public dataset.[23] The experimental results show that our proposed method can improve the tracking accuracy and real-time performance when compared with the state-of-the-art asynchronous event-corner tracker and with no compromise on the feature tracking lifetime.

The rest of the article is organized as follows. The related works are given in the next section. Then, we give the overview of the presented algorithm, which is followed by the introduction of the proposed gradient descriptor. Later, the details of the event feature tracking method are outlined, and the following section presents the experimental results and the corresponding analysis. Finally, the conclusions are drawn and the future work is given.

Related works

In computer vision, a feature may be a specific structure, such as interest point, edge, block, or object, which differs from its immediate neighborhood in the image. The feature-based tracking method is widely applied for visual odometry, SLAM, and augmented reality. Feature-based tracking method generally consists of feature detection, feature description, feature matching, and feature tracking. In the whole process, feature description is one of the most significant steps for tracking.

Feature descriptors for standard images

As one of the most widely used features, SIFT[25] is invariant to image scale and rotation, and robust to changes in illumination and affine distortion. The generation of SIFT feature descriptor has four stages: scale-space extrema detection (based on the difference of Gaussian pyramid), keypoint localization, orientation assignment, and keypoint description. After the first two stages, keypoints will be selected including their locations and scales. In the step of the orientation assignment, one or more orientations will be assigned to each keypoint based on local image gradient information at the local patch region around the keypoint location. So, every keypoint can be assigned with the location, scale, and orientation. Finally, the descriptor with multidimensions can be calculated for each keypoint at the selected scale based on the local patch region around its location. The SURF descriptor[26] was proposed based on the idea similar to SIFT. SURF is faster than SIFT, and it is also scale invariant and rotation invariant. As a binary descriptor, the binary robust independent elementary feature (BRIEF) descriptor[27] allows very fast Hamming distance matching, but it is not scale invariant and rotation invariant. Another binary descriptor, called ORB,[28] combines the oriented features from accelerated segment test (FAST) detector[29] and rotated BRIEF descriptor. ORB is rotation invariant but not scale invariant. Compared to BRIEF and ORB, SIFT and SURF need significantly more computation effort. However, SIFT and SURF use binary strings as feature descriptions, which result in larger mismatch rates.

Event-based corner detection

In recent years, many asynchronous event-corner detection and tracking methods[30,31,32,33,34] have been proposed based on event-driven data. In detail, Vasco et al.[31] applied an adaptation of the original image-based Harris corner detector[35] for event-based data, while Mueggler et al.[32] presented a FAST-like event-based corner detector faster than the method proposed by Vasco et al.,[31] inspired by image-based FAST corner detection method. Li et al.[33] studied a fast and asynchronous event-based corner detection method, called FA-Harris, with a corner candidate selection and refinement strategy. Alzugaray and Chli[36] proposed a faster asynchronous event-corner detection method inspired by the method of Mueggler et al.[32] and a simple asynchronous event-corner tracker. The tracker utilizes a directed graph to record the tracks of event-corners. Then, Alzugaray and Chli[37] improved the asynchronous event-corner tracking algorithm by introducing the normalization descriptor for extracted event-corners. FA-Harris detector achieves better performance in terms of accuracy with moderate computation performance, compared with the other aforementioned corner detection methods.

All the above event-corner detection methods operate directly on asynchronous event-streams using the surface of active event (SAE)[38] (also called time surface.[39]). Time surface maps the position of the latest event to its timestamp. In other words, time surface keeps the absolute timestamps of the latest events triggered at the imaging plane. Manderscheid et al.[24] proposed the SITS, which is invariant to the motion speed of cameras or scene objects. SITS keeps the relative timestamps instead of absolute ones. They utilized the SITS to detect event-corners from event-streams by training a random forest.

Event-based feature tracking

Some event-based feature tracking methods work based on event frames (synthesized by events with a fixed number or in a fixed temporal window) or the absolute intensity information on images. In the study of Tedaldi et al.,[40] they first extracted Harris corners and Canny edge features on intensity images and then tracked the features on asynchronous event-streams. Kueng et al.[41] presented an event-based visual odometry method to track the six degrees of motion of the camera, and the proposed method is also based on corners and edges. Zhu et al.[42] accumulated events in a temporal window to integrate event frames. Based on the integrated event frames, they applied the original Harris corner detector and then tracked the detected corners with expectation–maximization scheme. Afterward, Zhu et al.[43] further introduced the inertial measurement into the system and proposed an event-based visual-inertial odometry method. Gehrig et al.[44] detected Harris corners on the intensity frames and tracked them on event-streams. Li et al.[45] proposed a feature tracking method using events, intensity frames, and IMU data. They first extracted Harris and Canny feature on intensity frames, and then, the feature templates are tracked using an expectation–maximization iterative closest point strategy. Besides, Alzugaray and Chli[46] addressed a method to track generic patch features event-by-event without the requirement for detecting event-corners and descriptors.

To fully utilize the natural asynchronism of event cameras, we propose a novel asynchronous event feature generation and tracking algorithm inspired by frame-based feature tracking techniques. The algorithm can work directly on event-streams without the requirement for

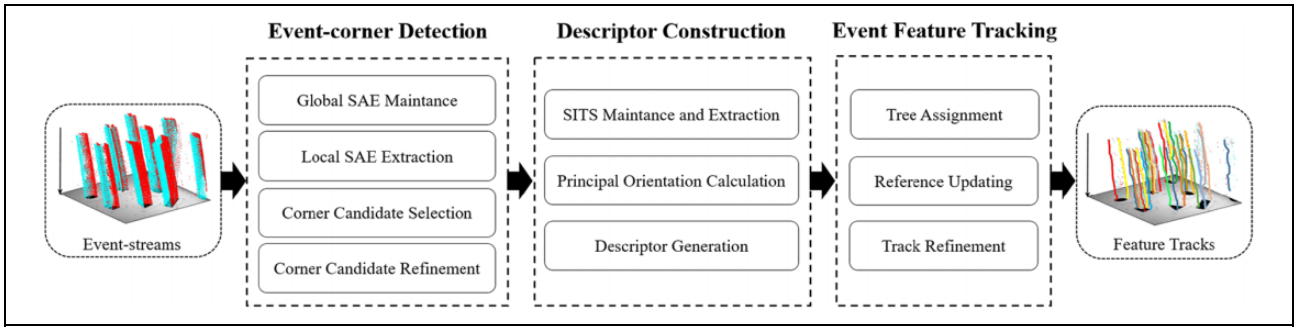

Figure 2. The overview of the proposed asynchronous event feature generation and tracking algorithm. The algorithm consists of an event-corner detection unit, a descriptor construction unit, and an event feature tracking unit. The input of the algorithm is the event-streams, and the output of the algorithm is the tracks of the tracked event features. The event-corner detection unit is based on a fast and asynchronous event-corner detection method. It extracts the event-corners through global SAE maintenance, local SAE extraction, corner candidate selection, and corner candidate refinement. The gradient descriptor is constructed by SITS maintenance and extraction, principal orientation calculation, and descriptor generation. The tracking unit is achieved using the constructed descriptor and an event feature matching method to achieve asynchronous feature tracking. SAE: surface of active event; SITS: speed invariant time surface.

intensity frames, artificially synthesized event frames, or other prior knowledge of scenes or camera motion. The proposed algorithm is based on a novel asynchronous event feature gradient descriptor inspired by the frame-based SIFT feature descriptor. The gradient descriptor represents the distribution of the local gradient information for event-corners, and it is used for feature matching during the asynchronous tracking process.

Overview

Inspired by standard computer vision tasks, we propose an asynchronous event feature generation and tracking algorithm in this article. As shown in Figure 2, the algorithm includes an event-corner detection unit, a descriptor construction unit, and an event feature tracking unit.

The event-corner detection unit is based on a fast and asynchronous event-corner detection method,[33] which is called FA-Harris. It detects event-corners directly on event-streams without using intensity images, which mainly consists of five steps, including event filter, global SAE maintenance, local SAE extraction, corner candidate selection, and corner candidate refinement. In the proposed event feature generation and tracking algorithm, the event-corner detection unit utilizes the FA-Harris detector to extract event-corners from event-streams, and the event filter included in FA-Harris detector is not used in our method here, which we found would not contribute to the performance improvement of the tracking method.

After detecting event-corners, we design a novel asynchronous event feature gradient descriptor for each event-corner based on the SITS.[24] The gradient descriptor can be constructed by SITS maintenance and extraction, principal orientation calculation, and descriptor generation. The descriptor can represent the distribution of the local gradient information for event-corners and help to achieve quantitative measurements of similarity between event feature pairs in the following event feature tracking unit. By introducing the gradient descriptor, we can define the event feature as a tuple $[x, y, t, d]$, where (x, y, t) is the spatiotemporal coordinate of the event feature and d is the gradient descriptor of the event feature. The details of the proposed gradient descriptor will be introduced in the following section.

Finally, the generated event features will be tracked using the event feature tracking unit. The tracking unit is achieved using the constructed descriptor and an event feature matching method to achieve asynchronous event feature tracking. The proposed gradient descriptor is used to provide the similarity measurements between event feature pairs. The event feature matching method is implemented based on a directed graph, which is composed of multiple structured track trees.

Gradient descriptor

This section introduces our proposed gradient descriptor. The construction of the gradient descriptor can be divided into three stages: SITS maintenance and extraction, principal orientation calculation, and descriptor generation, which is mainly inspired by the last two stages of the frame-based SIFT descriptor. We firstly select the event-corners (keypoints) from the incoming events including their locations, timestamps, and polarities based on FA-Harris detector. Compared with the keypoints in the SIFT method, our event-corners do not contain the scale information. For the frame-based SIFT descriptor, the first stage is the scale-space extrema detection based on the difference of Gaussian pyramid before the keypoint localization step. For the consideration of simplicity, we did not utilize the scale space compared with the frame-based SIFT descriptor, which would be a feature direction for further research. As mentioned above, the keypoint localization for our method is achieved using the FA-Harris detector for event

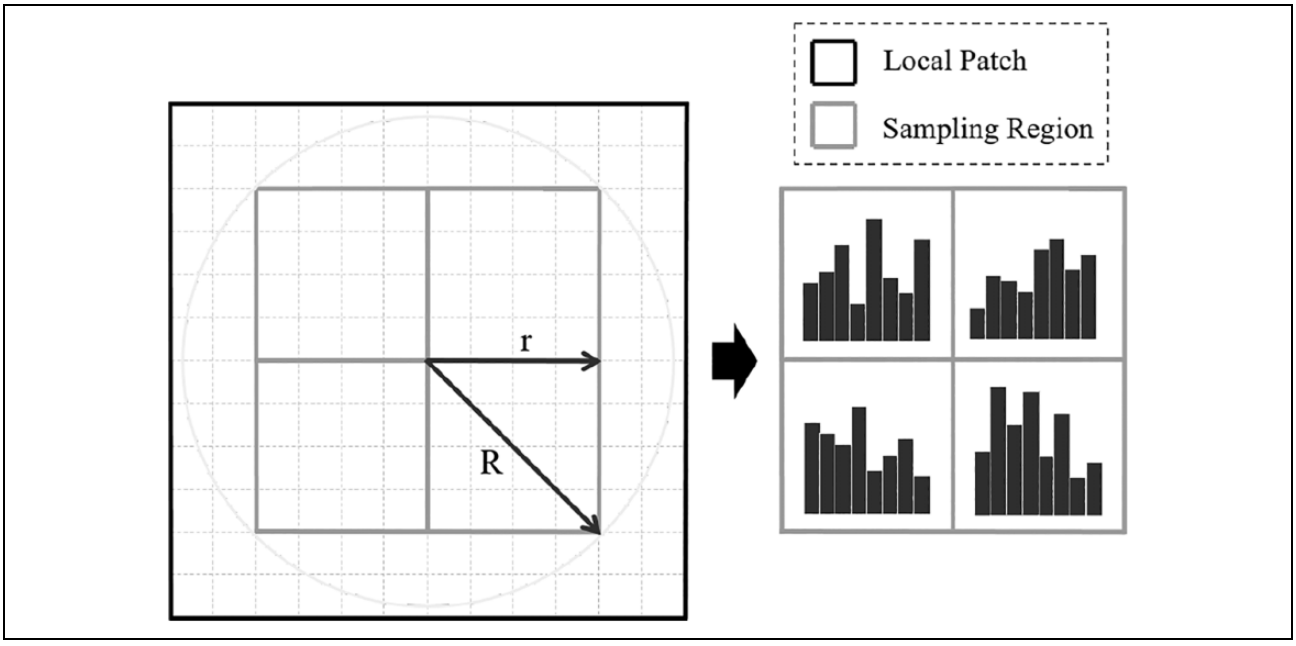

Figure 3. The generation of the proposed asynchronous event feature gradient descriptor. The local patch with size $(2\lfloor R \rfloor + 2) \times (2\lfloor R \rfloor + 2)$ is extracted from the global SITS. The sampling region with size $2r \times 2r$ is divided into 2×2 cells in this case. One gradient histogram with eight bins is computed for each cell based on the reoriented local patch. In this way, a 32-dimension gradient descriptor vector will be constructed. SITS: speed invariant time surface.

cameras rather than localizing the keypoint in the scale space. We apply the SITS method[24] to provide temporal information and gradient information of events. The global SITS structure will be maintained based on the incoming events. It has the same size width × height as the imaging plane, where each position is associated with the corresponding pixel position in the imaging plane and will be used to provide the gradient information for each event-corner. For each incoming event-corner, the gradient descriptor will be generated based on the local patch region extracted from the global SITS structure around its location. The construction process of the gradient descriptor is asynchronous, in other words, the algorithm generates a gradient descriptor once a new event-corner arrives.

Speed-invariant time surface maintenance and extraction

Since there is no concept of intensity images for event cameras and a single event does not bring any information to provide gradient information for descriptor construction, we choose the SITS[24] (updated asynchronously, with every incoming event) to store the temporal information of events and provide the gradient information, rather than using the intensity image as in the frame-based SIFT descriptor.[25] According to Manderscheid et al.,[24] SITS is invariant to the motion speed of cameras or objects in the environment, which can contribute to the speed-invariant property of event features. To distinguish event-corners from event-streams, SITS keeps relative values for timestamps rather than absolute ones. The method tries to maintain one SITS structure for each polarity of event, which stores a single value for each pixel location. Specifically, all the values in the SITS are initialized to 0. When a new event arrives, the values, which are larger than the value at the corresponding event pixel position (x, y), within the window $l \times l$ will be reduced by 1. According to the study of Manderscheid et al.,[24] l is set to 11. The value at the corresponding event pixel position (x, y) will be modified to l^2. SITS is invariant to the motion speed of cameras or objects in the environment.

In our proposed algorithm, we maintain one global SITS structure for each polarity same as in the method of Manderscheid et al.[24] For each incoming event, the global SITS structure corresponding to the polarity of the new coming event will be updated. When a new event-corner arrives, we extract the local patch P with size $(2\lfloor R \rfloor + 2) \times (2\lfloor R \rfloor + 2)$ around the pixel position of the event-corner on the global SITS structure corresponding to the polarity of the new event-corner. As shown in Figure 3, $R = \sqrt{2}r$ is decided by the radius r of the sampling region for the gradient descriptor. The extracted local patch P will be used for descriptor generation.

Principal orientation calculation

On the extracted local patch P extracted from the global SITS structure, we calculate the magnitude $m(x, y)$ and orientation $\theta(x, y)$ of gradient for every position as follows

$$m_{xy} = m(x,y) = \sqrt{d_x^2 + d_y^2} \tag{1}$$

$$\theta_{xy} = \theta(x,y) = \arctan\left(\frac{d_y}{d_x}\right) \tag{2}$$

where $d_x = P(x+1,y) - P(x-1,y)$ is the differential in x direction of the local patch (which is the same as the x direction of the imaging plane), $d_y = P(x,y+1) - P(x,y-1)$ is the differential in y direction of the local patch (which is the same as the y direction of the imaging plane), and $x, y = 1, 2, ..., 2\lfloor R \rfloor$. A gradient histogram with $n_1 = 36$ bins (according to Lowe[25]) will be generated for the local patch. The histogram is essentially a vector with 36 bins corresponding to angles $0, 10, \ldots, 350$. The magnitude values will be weighted using Gaussian weighting function w with spread $\sigma = 1$ pixel before adding to the gradient histogram. Hence, the gradient magnitude near the event-corner will have greater weight. For all magnitudes m_{xy} and orientation $\theta_{xy} \in [0, 360^\circ]$, where $x, y = 1, 2, ..., 2\lfloor R \rfloor$, the generation of a gradient histogram $\mathbf{H} = (H_0, H_1, ..., H_k, ..., H_{n_1-1})$, where $k = 0, 1, ..., n_1 - 1$, can be formalized as

$$H_k = \sum_{x,y \in \Psi} m_{xy} w_{xy} \tag{3}$$

where

$$\Psi = \{x, y | \left\lfloor \frac{n_1 \theta_{xy}}{360} \right\rfloor = k\} \tag{4}$$

$$w_{xy} = e^{\frac{-(x^2+y^2)}{2\sigma^2}} \quad x, y = 1, 2, ..., 2\lfloor R \rfloor \tag{5}$$

The orientation corresponding to the peak value in the gradient histogram represents the gradient orientation of the local patch, and it is also regarded as the principal orientation of the local patch. Since the orientation we get from a gradient histogram is essentially an interval of 10°, we apply the parabolic interpolation processing to get the specific orientation. More specially, the selected orientation and the orientations adjacent to it are used for parabolic interpolation.

To enhance the robustness of feature matching, we choose the orientation corresponding to the maximum value in the histogram and orientations where the value is greater than 80% (same as in reference[25]) of the maximum value. In this way, one or more orientations are assigned to each event-corner on the local patch. The orientations belong to circular data rather than linear data, and the circular mean[47] can provide a more intuitive estimate of the "center" of the distribution for this kind of data compared with the arithmetic mean. So, to obtain the final principal orientation from the orientations assigned to each event-corner, we compute the circular mean of these orientations rather than the arithmetic mean. To be specific, for the orientations $(\phi_1, \phi_2, \ldots, \phi_n)$, the circular mean $\overline{\phi}$ is calculated as follows

$$\overline{\phi} = \arctan\left(\frac{1}{n}\sum_{i=1}^{n}\cos\phi_i, \frac{1}{n}\sum_{i=1}^{n}\sin\phi_i\right) \tag{6}$$

The circular mean $\overline{\phi}$ is regarded as the principal orientation of the local patch.

Descriptor generation

In this stage, we reorient the local patch to its principal orientation to generate the gradient descriptor vector, that is, we rotate the x direction of the local patch (which is the same as the x direction of the imaging plane) to coincide with its principal orientation. The neighborhood with size $2r \times 2r$ around the event-corner position (center of the local patch) is taken as the sampling region for descriptor generation. As shown in Figure 3, the sampling region is divided into $c \times c$ cells. We set c to 2 here. In the literature,[25] the author suggested setting the value of c to 4. However, according to our findings, the number of features tracked by the algorithm in this case is not enough. Therefore, with this in mind, we set the value of c to 2 to guarantee that we can track a sufficient number of features without too much impact on the other performance of the algorithm. We calculate the orientation and magnitude of the gradient for every pixel position on the local patch and then determine the reoriented pixel value as follows

$$\begin{cases} x_{rot} = x\cos\overline{\phi} - y\sin\overline{\phi} \\ y_{rot} = x\sin\overline{\phi} + y\cos\overline{\phi} \end{cases} \tag{7}$$

where $x, y = 0, 1, ..., 2\lfloor R \rfloor + 1$. After the reorientation, only pixels falling into the sampling region contribute to descriptor generation. Since the coordinate values of the reoriented pixels are not integers, we compute their contribution to each adjacent cell using trilinear interpolation same as in SIFT.[25] Then, a gradient histogram with $n_2 = 8$ bins (same as in the literature[25]) will be given for each cell based on the reoriented local patch. We generate a N-dimensional descriptor vector, where $N = c^2 \times n_2$. The descriptor vector will be normalized using L2 norm to remove the scale, and finally, we can get the gradient descriptor for an event-corner.

After we get the gradient descriptors for event-corners, we need to compute the descriptor distance between event feature pairs to measure the similarity between them. For two gradient descriptor vector d_1, d_2, we compute their distance D as follows

$$D = \sqrt{\sum_{i=0}^{N}(d_1^i - d_2^i)^2} \tag{8}$$

Event feature tracking

In the event feature tracking unit, we combine the constructed gradient descriptor and an event feature matching

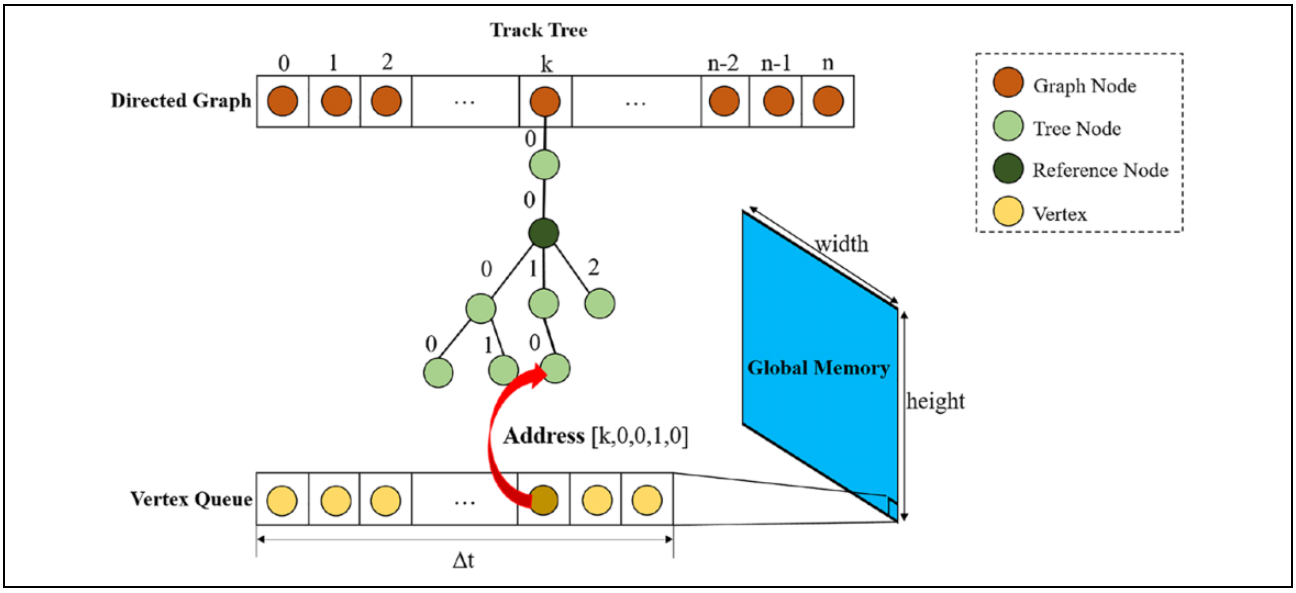

Figure 4. Implementation details of the event feature tracking unit. A global memory with size width × height saves the latest event features used for feature matching. The latest event features within Δt for each pixel position are saved in a vertex queue. The directed graph keeps all the possible tracks of the tracked event features. Each track tree keeps a reference tree node (dark green), which maintains a maximum number of $\rho_{\max}$ vertices from itself to the deepest vertex in the same tree. $\rho_{\max}$ is 2 in this case. An address tuple is assigned to each vertex (yellow), and the vertex (yellow) is used to find where the matching vertex (green) is in the directed graph.

method to achieve asynchronous event feature tracking. For each incoming event-corner, we assign a gradient descriptor to it based on the above-mentioned gradient descriptor construction method. Our proposed gradient descriptor is used to represent the distribution of the local gradient information within the event-corner neighborhood on the time surface space. We define the event feature as a tuple $[x, y, t, d]$, where (x, y, t) is the spatiotemporal coordinate of the event feature and d is the gradient descriptor of the event feature. The gradient descriptor is regarded as a quantitative measurement of similarity between event feature pairs for feature matching.

The event feature matching method[37] used in our presented algorithm is based on a directed graph. The implementation details of the event feature matching method are summarized in Figure 4. For each new incoming event feature, the algorithm generates a new vertex v_{new}. The global vertex memory with size width × height saves the latest event features corresponding to pixels within the temporal window Δt. The latest event features within Δt for each pixel position are saved in the vertex queue. The directed graph is composed of multiple structured track trees, and each tree represents a set of multiple possible tracks for the same event feature. An address is assigned for every event feature. The address is a tuple encoding the path to find where the event feature is in the directed graph. Therefore, a vertex is associated with an individual event feature and encodes the information about an event feature and the associated address. Each tree node encodes the information about a vertex, the depth of the node and the state (active or inactive), and its children nodes. The edge between two nodes represents the association between event feature pairs. Each graph node keeps a hypothetical track tree and encodes the information about the tree, which includes the tree depth, the reference tree node (the reference vertex), and the pointer to the tree.

For every new generated vertex, it can be assigned to an existing tree or become the root of a new tree. Once the depth of a tree increases, the proposed algorithm will perform the reference updating operation.

Tree assignment

Considering the descriptor distance introduced in the above section, the closest vertex compared with the new vertex in the spatiotemporal window $w \times w \times \Delta t$ will be selected as the matching vertex v_{match}. The tree that v_{match} belongs to is regarded as the matching tree. For the vertices on the matching tree, the newest vertex within the spatial window $w \times w$ is regarded as the parent vertex of v_{new}, and the new vertex v_{new} will be associated to the parent vertex by adding a new edge from the parent vertex to v_{new}. The descriptor distance between v_{new} and v_{match} must be smaller than the threshold $d_{\max}$. Otherwise, v_{new} will be identified as the root of a new tree.

Reference updating

A reference vertex v_{ref} for each tree maintains a maximum number of $\rho_{\max}$ vertices from itself to the deepest vertex in the same tree. The vertex in the tree whose depth is smaller than v_{ref} is regarded as the inactive vertex, and the vertex whose depth is larger than v_{ref} is regarded as the active

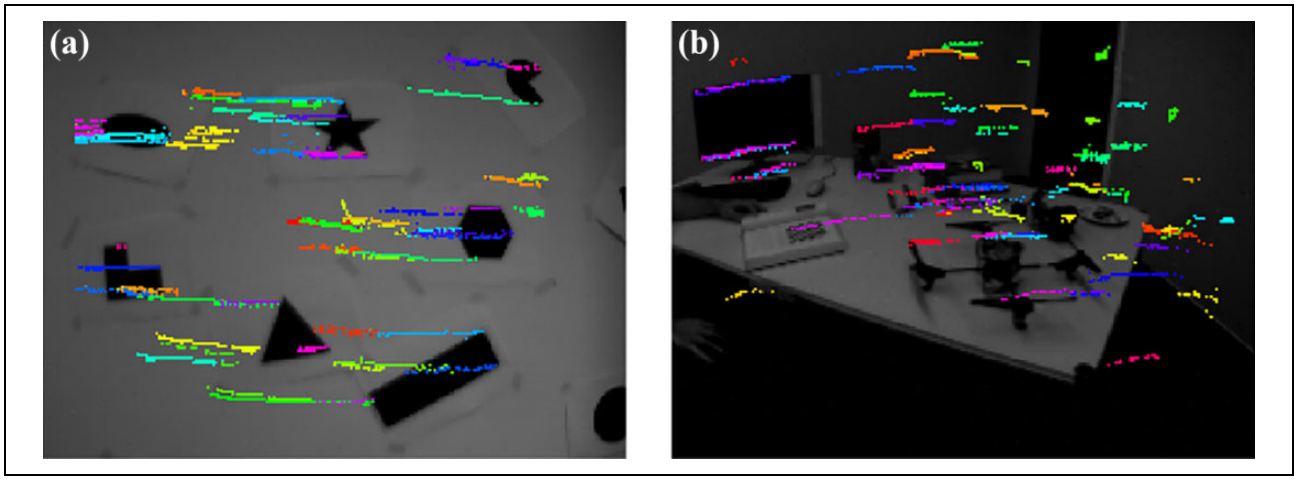

Figure 5. Asynchronous event feature tracks on (a) shapes and (b) dynamic scenes. The figures show the different feature tracks using our method over the last 0.5 s. The intensity frame is used for visualization in the figure.

vertex. Once the depth of a tree increases, v_{ref} will be updated according to the descriptor distance between the previous v_{ref} and its children vertices. The children vertices of v_{ref} can be divided into weak vertices and strong vertices depending on their descriptor distances with v_{ref}. A child vertex, whose descriptor distance with v_{ref} is larger than the threshold $d_{\min}$, is regarded as the weak vertex. Otherwise, the child vertex will be regarded as the strong vertex. The newest strong child vertex will be considered as the new v_{ref} and the new parent vertex of the other strong children vertices as well. If there is no strong child vertex, the weak child vertex with the smallest distance will be regarded as the new v_{ref}, and the other weak vertices together with their children trees will disconnect from the tree to generate new trees. These weak vertices disconnected from the tree will become the roots of new trees.

Track refinement

To get smoother tracks, the event feature tracks are smoothed using a simple interpolation operation. For each vertex, its pixel coordinate will be interpolated using its s predecessors and s successors in the same track. Only the event feature track which contains at least m refined vertices is used to filter out the short and noisy tracks.

Experiments

This section introduces the experimental results of our proposed algorithm, including accuracy and computational performance evaluation for feature tracking. We compare our proposed method with the tracking method[42] (referred as EOF tracker), the ACE tracker,[37] and the tracking method of reference[46] (referred as AMH tracker). The public event camera dataset,[23] generated using a DAVIS240C with a spatial resolution of 240 × 180 pixels, is adopted to perform the comparison. The data consist of events, intensity images, IMU measurements, and ground truth from a motion-capture system. In the experiments, we choose multiple scenes with different complexity of textures from the dataset, including `shapes`, `dynamic`, `poster`, `boxes` and `poster`, `boxes` with high dynamic range of illumination. Only the first 10 s of the dataset are used for evaluation.

To perform the comparison, we implement the normalization descriptor and the ACE tracker[37] in C++. The same parameters and values are used as those presented in the article. All methods are implemented in C++ and evaluated on a laptop equipped with an Intel i7-7700HQ CPU with 2.80 GHz and RAM with 16 GB. For the event feature tracking unit, we employ the spatiotemporal window $w \times w \times \Delta t$, where $w = 4$, $\Delta t = 0.5$ s. We set the threshold of the gradient descriptor $d_{\max} = 100$ for the matching vertex selecting, and $d_{\min} = 50$ for distinguishing the strong and weak vertex. $\rho_{\max}$ is set to 8. s is set to 14. m is set to $50, 30, 12, 12$ for `shapes`, `dynamic`, `poster`, and `boxes`, respectively. Note that the suitable values for above parameters are chosen using the trial and error method.

Tracking performance

We use the event-based feature tracking evaluation code[48] for tracking performance analysis. The ground truth feature tracks are collected using KLT-based feature tracking method on frames. The positions of the initial features on frames are interpolated from the event-based features close to the time of the frames. The initial features are tracked using KLT tracking method until they are lost, and the tracker updates the tracked features for each frame. The asynchronous event feature tracks for our proposed algorithm on `shapes` and `dynamic` scenes are depicted in Figure 5. The figures show the different feature tracks using our method with different colors over the last 0.5 s.

Table 1 summarizes the average pixel error and average feature lifetime for event feature tracks on several scenes

Table 1. Average pixel error and feature lifetime of event feature tracks on different scenes using the AMH tracker, the EOF tracker, the ACE tracker, and our proposed method.[a]

	Average tracking error (px)				Average feature lifetime (s)		
Datasets	AMH	EOF	ACE	Ours	EOF	ACE	Ours
shapes_rotation	1.84	2.23	1.64	**1.04**	0.08	0.16	**0.43**
shapes_translation	**1.10**	1.93	1.50	1.12	0.15	0.24	**0.44**
shapes_6dof	1.52	2.14	1.71	**1.14**	0.20	0.66	**1.16**
dynamic_rotation	—	—	0.71	**0.65**	—	0.20	**0.22**
dynamic_translation	—	—	1.51	**0.81**	—	0.13	**0.37**
dynamic_6dof	—	—	1.61	**0.93**	—	0.17	**0.42**
poster_rotation	2.11	2.32	1.08	**0.50**	**0.19**	0.04	0.13
poster_translation	1.81	2.35	1.60	**0.80**	**0.41**	0.07	0.16
poster_6dof	2.11	2.44	1.84	**0.77**	**0.30**	0.16	0.21
boxes_rotation	2.09	2.34	1.42	**0.94**	0.07	0.06	**0.16**
boxes_translation	1.39	1.89	1.89	**1.09**	**0.29**	0.14	0.25
boxes_6dof	1.64	2.00	1.79	**0.99**	0.23	0.26	**0.31**
hdr_poster	1.79	2.26	1.47	**0.84**	0.35	**0.41**	0.29
hdr_boxes	1.68	2.03	1.98	**1.43**	0.33	0.34	**0.48**

[a]The best results are highlighted in bold.

with different textural complexity. In our experimental evaluation, if the pixel error for a tracked feature is above 5 pixels, the tracked feature is regarded as invalid, and only the valid tracked feature is considered in the evaluation. The best results are made in bold in the table, and the results indicate that our proposed method can achieve better performance in terms of accuracy when compared with EOF tracker and ACE tracker. Besides, we report the average tracking error from reference[46] (the authors did not explicitly report tracking lifetime numerically), compared with which our tracking method also performs better with significant improvement, except one case, `shapes_translation`. When considering the average feature tracking lifetime, our method also works well on the scenes with simple or moderate texture, such as `shapes`, `dynamic`, and so on. The results on the scenes `hdr_poster` and `hdr_boxes` demonstrate that our method can perform well when faced with the scenes with high dynamic range. However, while working on the scene `poster`, EOF tracker performs well compared with both ACE tracker and our method.

Figure 6 shows the average tracking pixel error and the percentage of the tracked surviving features over time for the ACE tracker and our proposed method on four different scenes. The results demonstrate that our proposed method achieves better performance in terms of tracking accuracy. What is more, the band around the central line is wider with our method, which indicates that our method is more robust. However, the ACE tracker performs better on the scenes with complex texture when considering the feature tracking lifetime.

Computational performance

In this section, we compare the computational performance of our proposed event feature tracking method with the ACE tracker.

The ACE tracker uses the normalization descriptor as a quantitative measurement of similarity between event-corners. The normalization descriptor is implemented based on the simple sorting operation of the events' timestamps in a local patch. The sorted timestamps are normalized into the range $[0, 1]$. They measured the descriptor distance by calculating the amount of overlap between the two normalization descriptors.

Table 2 presents the real-time performance of our proposed descriptor construction and the event feature matching method. We report the total time, the time spent on descriptor construction and event feature matching, and their ratios to the total time. As given in Table 2, the descriptor construction ratio with our method is larger than that of ACE. This is because our method needs gradient computation and Gaussian weighting, while the normalization descriptor used in ACE only needs a sort operation and a normalization operation. However, the matching time with our method is much shorter than the matching time with ACE, which contributes to a better real-time performance in terms of the total time for feature generation and tracking.

Also, we report the metric, real-time factor for real-time performance analysis, which indicates the total time spent processing the events of each scene with respect to the duration of dataset (10 s for each scene). For this metric, the smaller result indicates better real-time performance, and the results under 1 indicate the performance above real time. According to Table 2, our method achieves better real-time performance when facing all of the scenes compared with the ACE method. However, there are still some scenes, such as `dynamic_rotation`, `poster_rotation`, and `hdr_poster`, that the real-time factors are slightly greater than 1. It

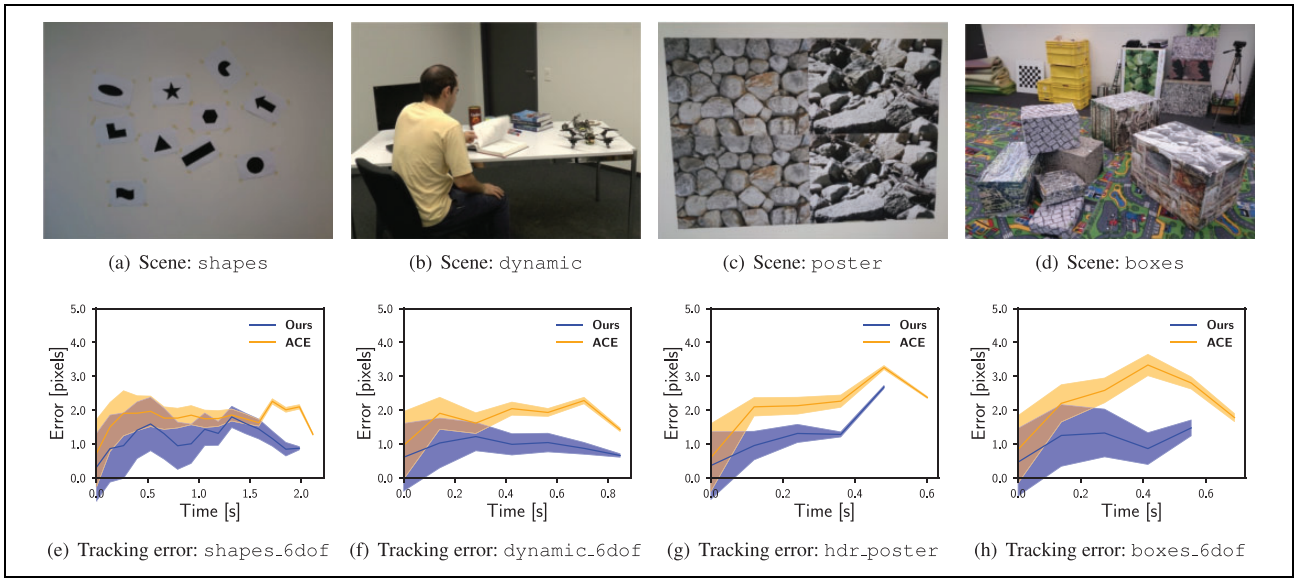

Figure 6. (a–d) The performance of feature tracking on different scenes. The figures in the second row show the average tracking pixel error (central line) on the corresponding scenes for the ACE tracker and our method. The band around the central line represents the percentage of the tracked surviving features over time. The wider the band is, the more robust the tracking method is.
(a) Scene: shapes, (b) scene: dynamic, (c) scene: poster, (d) scene: boxes, (e) tracking error: shapes_6dof, (f) tracking error: dynamic_6dof, (g) tracking error: hdr_poster, (h) tracking error: boxes_6dof.

Table 2. The real-time performance of our proposed descriptor construction and event feature matching method compared with those of the ACE tracker, including the total time, the time for descriptor construction (des. time), the time for event feature matching (matching time), the ratio of time spent on descriptor construction to the total time (des. ratio), the ratio of time spent on event feature matching to the total time (matching ratio), and the real-time factor.[a]

Datasets	Total time (s)		Des. time (s)		Des. ratio (%)		Matching time (s)		Matching ratio (%)		Real-time factor	
	ACE	Ours	ACE	Ours	ACE	Ours	ACE	Ours	ACE	Ours	ACE	Ours
shapes_rotation	11.38	**10.04**	1.48	2.88	13.03	28.63	8.27	4.36	72.62	43.44	1.138	**1.004**
shapes_translation	11.88	**10.32**	1.45	2.95	12.25	28.58	8.71	4.30	73.28	41.64	1.188	**1.032**
shapes_6dof	10.44	**9.31**	1.21	2.79	11.55	29.96	7.90	3.82	75.63	40.98	1.044	**0.931**
dynamic_rotation	13.28	**11.28**	1.49	2.70	11.24	23.91	8.99	3.48	67.66	30.81	1.328	**1.128**
dynamic_translation	10.10	**9.79**	1.15	2.19	11.42	22.38	6.29	2.64	62.25	26.92	1.010	**0.979**
dynamic_6dof	10.12	**9.43**	1.15	2.14	11.34	22.65	6.26	2.56	61.85	27.15	1.012	**0.943**
poster_rotation	16.84	**12.81**	1.86	2.94	11.05	22.98	10.88	3.17	64.65	24.78	1.684	**1.281**
poster_translation	13.65	**11.75**	1.59	2.66	11.64	22.67	8.26	2.53	60.49	21.58	1.365	**1.175**
poster_6dof	12.07	**10.70**	1.47	2.44	12.20	22.79	7.17	2.39	59.45	22.36	1.207	**1.07**
boxes_rotation	10.88	**10.28**	1.39	2.31	12.82	22.43	6.17	2.13	56.73	20.76	1.088	**1.028**
boxes_translation	10.00	**9.65**	1.27	2.18	12.74	22.63	5.62	2.00	56.22	20.75	1.000	**0.965**
boxes_6dof	10.83	**10.01**	1.32	2.15	12.18	21.49	5.93	1.92	54.80	19.19	1.083	**1.001**
hdr_poster	13.63	**11.35**	1.46	2.56	10.74	22.59	8.81	2.62	64.69	23.12	1.363	**1.135**
hdr_boxes	11.72	**10.38**	1.30	2.04	11.12	19.63	6.49	1.84	55.34	17.69	1.172	**1.038**

[a]Both ACE and our method are performed on the same laptop. The better results are highlighted in bold.

means that our method has a potential in improving the calculating performance, especially in refining the matching method to reduce more processing time for better real-time performance for utilization in real-time applications.

Table 3 gives the event processing ability of the ACE tracker and our proposed event feature generation and tracking algorithm. The table includes the mean rate of events, the mean rate of the event-corners, and the mean time for a single feature matching. According to Table 3, our proposed method achieves faster event-rate, corner-rate, and speed for every single feature matching than the ACE tracker, which contributes to the improvement of the real-time performance of our method.

Table 3. The event processing ability of the ACE tracker and our proposed event feature generation and tracking algorithm, including the mean rate of events (mean event-rate), the mean rate of the event-corners (mean corner-rate), and the mean time for a single feature matching (time per feature).[a]

Datasets	Mean event-rate (ev/s)		Mean corner-rate (ev/s)		Time per feature (μs/ev)	
	ACE	Ours	ACE	Ours	ACE	Ours
shapes_rotation	65236	**113012**	2855	**5995**	278.04	**80.76**
shapes_translation	70667	**111170**	3203	**5884**	264.54	**83.18**
shapes_6dof	64997	**95172**	3125	**5360**	269.62	**88.04**
dynamic_rotation	179366	**282281**	3250	**5544**	257.27	**70.05**
dynamic_translation	255690	**316904**	3256	**4755**	243.11	**71.93**
dynamic_6dof	251609	**332970**	3285	**5305**	238.45	**64.44**
poster_rotation	183331	**306422**	3503	**5987**	262.50	**60.94**
poster_translation	236475	**326620**	3844	**5675**	231.68	**57.60**
poster_6dof	258344	**375016**	3796	**6002**	220.85	**53.06**
boxes_rotation	295339	**394796**	4327	**6301**	196.16	**50.59**
boxes_translation	300323	**384416**	4078	**5918**	199.69	**51.77**
boxes_6dof	317866	**430091**	4121	**6201**	208.21	**48.85**
hdr_poster	212706	**322970**	3567	**5451**	254.51	**61.17**
hdr_boxes	316174	**437711**	3801	**5711**	229.65	**49.64**

[a]The better results are highlighted in bold.

Conclusion

In this article, we present a novel asynchronous event feature generation and tracking algorithm operating directly on event-streams for event cameras. The algorithm consists of an event-corner detection unit, a descriptor construction unit, and an event feature tracking unit. An asynchronous gradient descriptor is developed for the quantitative measurement of similarity between event feature pairs, and it is constructed through SITS maintenance and extraction, principal orientation calculation, and descriptor generation. The experimental evaluation demonstrates that our proposed algorithm performs better in terms of tracking accuracy and real-time performance when compared with the state-of-the-art asynchronous event-corner tracker and with no compromise on the feature tracking lifetime.

In the future, there are still some works for us to do, such as improving the tracking accuracy and lifetime performance, such as by adding scale-invariant property to the descriptor, so that the feature tracking algorithm could fulfill the demand of visual odometry pipeline and even a SLAM system. And also, learning-based methods for event feature generation may be a great direction for further research.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed the receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the National Key Research and Development Program of China [2017YFB1001901] and by National Natural Science Foundation of China [Grant No. 61903377].

ORCID iDs

Ruoxiang Li https://orcid.org/0000-0002-6684-2011
Ruihao Li https://orcid.org/0000-0002-9839-1489

References

1. Cadena C, Carlone L, Carrillo H, et al. Past, present, and future of simultaneous localization and mapping: toward the robust-perception age. *IEEE Trans Robot* 2016; 32(6): 1309–1332.
2. Zhou T, Brown M, Snavely N, et al. Unsupervised learning of depth and ego-motion from video. In: *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, Honolulu, HI, USA, 21 July 2017–26 July 2017, pp. 6612–6619. Los Alamitos, CA, USA: IEEE.
3. Li R, Liu Q, Gui J, et al. Indoor relocalization in challenging environments with dual-stream convolutional neural networks. *IEEE Trans Auto Sci Eng* 2018; 15: 651–662.
4. Tong Q, Li P, and Shen S. VINS-Mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Trans Robot* 2017; PP(99): 1–17.
5. Younes G, Asmar D, Shammas E, et al. Keyframe-based monocular SLAM: design, survey, and future directions. *Robot Auto Sys* 2017; 98: 67–88.
6. Lichtsteiner P, Posch C, and Delbruck T. A 128×128 120 db 15μ s latency asynchronous temporal contrast vision sensor. *IEEE J Solid-State Circuits* 2008; 43(2): 566–576.
7. Brandli C, Berner R, Yang M, et al. A 240×180 130 db 3μ s latency global shutter spatiotemporal vision sensor. *IEEE J Solid-State Circuits* 2014; 49(10): 2333–2341.
8. Mitrokhin A, Fermüller C, Parameshwara C, et al. Event-based moving object detection and tracking. In: *2018 IEEE/*

RSJ international conference on intelligent robots and systems (IROS) (eds T Maciejewski, C Monje, D Tsai, et al.), Madrid, Spain, 1–5 October 2018, pp. 1–9. Los Alamitos, CA, USA: IEEE.

9. Vidal AR, Rebecq H, Horstschaefer T, et al. Ultimate SLAM? Combining events, images, and IMU for robust visual SLAM in HDR and high-speed scenarios. *IEEE Robot Auto Lett* 2018; 3(2): 994–1001.
10. Sun Y, Liu M, and Meng MQH. Improving RGB-D slam in dynamic environments: a motion removal approach. *Robot Auto Sys* 2017; 89: 110–122.
11. Sun Y, Liu M, and Meng MQH. Motion removal for reliable RGB-D slam in dynamic environments. *Robo Auto Sys* 2018; 108: 115–128.
12. Gallego G, Delbrück T, Orchard G, et al. Event-based vision: a survey. *IEEE Trans Pattern Anal Mach Intell* 2020. DOI: 10.1109/TPAMI.2020.3008413.
13. Gehrig M, Shrestha SB, Mouritzen D, et al. Event-based angular velocity regression with spiking networks. In: *2020 IEEE international conference on robotics and automation (ICRA)* (eds A Howard, H Admoni, K Althoefer, et al.), NODUS, Paris, France, 31 May 2020–31 August 2020, pp. 4195–4202. Los Alamitos, CA, USA: IEEE.
14. Zhou Y, Gallego G, and Shen S. Event-based stereo visual odometry. *IEEE Transactions on Robotics*, DOI: 10.1109/TRO.2021.3062252. 2021.
15. Stoffregen T, Gallego G, Drummond T, et al. Event-based motion segmentation by motion compensation. In: *Proceedings of the IEEE/CVF international conference on computer vision*, Seoul, Korea (South), 27 October-2 November 2019, pp. 7244–7253. Los Alamitos, CA, USA: IEEE.
16. Mitrokhin A, Hua Z, Fermuller C, et al. Learning visual motion segmentation using event surfaces. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*. Seattle, WA, USA, 13-19 June 2020, pp. 14402–14411. Los Alamitos, CA, USA: IEEE.
17. Rebecq H, Ranftl R, Koltun V, et al. High speed and high dynamic range video with an event camera. *IEEE Trans Pattern Anal Mach Intell* 2021; 43(6): 1964–1980. DOI: 10.1109/TPAMI.2019.2963386.
18. Pan L, Scheerlinck C, Yu X, et al. Bringing a blurry frame alive at high frame-rate with an event camera. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, Long Beach, CA, USA, 15-20 June 2019, pp. 6813–6822. Los Alamitos, CA, USA: IEEE.
19. Choi J, Yoon KJ, et al. Learning to super resolve intensity images from events. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, Seattle, WA, USA, 13-19 June 2020, pp. 2765–2773. Los Alamitos, CA, USA: IEEE.
20. Falanga D, Kleber K, and Scaramuzza D. Dynamic obstacle avoidance for quadrotors with event cameras. *Sci Robot* 2020; 5(40): eaaz9712.
21. Hagenaars JJ, Paredes-Vallés F, Bohté SM, et al. Evolved neuromorphic control for high speed divergence-based landings of MAVs. *IEEE Robot Auto Lett* 2020; 5(4): 6239–6246.
22. Sun R, Shi D, Zhang Y, et al. Data-driven technology in event-based vision. *Complexity* 2021; 2021: 6689337.
23. Mueggler E, Rebecq H, Gallego G, et al. The event-camera dataset and simulator: event-based data for pose estimation, visual odometry, and SLAM. *Int J Robot Res* 2017; 36(2): 142–149.
24. Manderscheid J, Sironi A, Bourdis N, et al. Speed invariant time surface for learning to detect corner points with event-based cameras. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Long Beach, CA, USA, 15-20 June 2019, pp. 10237–10246. Los Alamitos, CA, USA: IEEE.
25. Lowe DG. Distinctive image features from scale-invariant keypoints. *Int J Comput Vis* 2004; 60(2): 91–110.
26. Bay H, Ess A, Tuytelaars T, et al. Speeded-up robust features (SURF). *Comput Vis Image Und* 2008; 110: 346–359.
27. Calonder M, Lepetit V, Strecha C, et al. BRIEF: binary robust independent elementary features. In: *European conference on computer vision*, Heraklion, Crete, Greece, 5-11 September, 2010, pp. 778–792. Berlin, Heidelberg: Springer.
28. Rublee E, Rabaud V, Konolige K, et al. ORB: an efficient alternative to SIFT or SURF. In: *2011 international conference on computer vision*, Barcelona, 6 November 2011–13 November 2011, pp. 2564–2571. Los Alamitos, CA, USA: IEEE.
29. Rosten E and Drummond T. Machine learning for high-speed corner detection. In: *European conference on computer vision (ECCV)*, Graz, Austria, May 7-13, 2006, pp. 430–443. Berlin, Heidelberg: Springer.
30. Clady X, Ieng SH, and Benosman R. Asynchronous event-based corner detection and matching. *Neural Networks* 2015; 66: 91–106.
31. Vasco V, Glover A, and Bartolozzi C. Fast event-based Harris corner detection exploiting the advantages of event-driven cameras. In: *2016 IEEE/rsj international conference on intelligent robots and systems (IROS)* (eds W Burgard, T Arai, F Arai, et al.), Daejeon, Korea, 9–14 October 2016, pp. 4144–4149. Los Alamitos, CA, USA: IEEE.
32. Mueggler E, Bartolozzi C, and Scaramuzza D. Fast event-based corner detection. In: *28th british machine vision conference (BMVC)* (eds TK Kim, S Zafeiriou, Gl Brostow, et al.), London, 4-7 September 2017, pp. 1–8. Durham, UK: BMVA Press.
33. Li R, xi Shi D, Zhang Y, et al. FA-Harris: a fast and asynchronous corner detector for event cameras. In: *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (eds T Maciejewski, C Monje, D Tsai, et al.), The Venetian Macao, Macau, 3–8 November 2019, pp. 6223–6229. Los Alamitos, CA, USA: IEEE.
34. Scheerlinck C, Barnes N, and Mahony R. Asynchronous spatial image convolutions for event cameras. *IEEE Robot Auto Lett* 2019; 4(2): 816–822.
35. Harris C and Stephens M. A combined corner and edge detector. In: *Alvey vision conference*, pp. 10–5244. Princeton, New Jersey, USA: Citeseer.

36. Alzugaray I and Chli M. Asynchronous corner detection and tracking for event cameras in real time. *IEEE Robot Auto Lett* 2018; 3(4): 3177–3184.
37. Alzugaray I and Chli M. ACE: an efficient asynchronous corner tracker for event cameras. In: *2018 international conference on 3D vision (3DV)*, Verona, Italy, 5–8 September 2018, pp. 653–661. Los Alamitos, CA, USA: IEEE.
38. Benosman R, Clercq C, Lagorce X, et al. Event-based visual flow. *IEEE Trans Neural Netw Learn Syst* 2014; 25(2): 407–417.
39. Lagorce X, Orchard G, Galluppi F, et al. HOTS: a hierarchy of event-based time-surfaces for pattern recognition. *IEEE Trans Pattern Anal Mach Intell* 2016; 39(7): 1346–1359.
40. Tedaldi D, Gallego G, Mueggler E, et al. Feature detection and tracking with the dynamic and active-pixel vision sensor (DAVIS). In: *2016 2nd international conference on event-based control, communication, and signal processing (EBCCSP)*, Krakow, Poland, 13–15 June 2016, pp. 1–7. Los Alamitos, CA, USA: IEEE.
41. Kueng B, Mueggler E, Gallego G, et al. Low-latency visual odometry using event-based feature tracks. In: *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (eds W Burgard, T Arai, F Arai, M Bennewitz, et al.), Daejeon, Korea, 9–14 October 2016, pp. 16–23. Los Alamitos, CA, USA: IEEE.
42. Zhu AZ, Atanasov N, and Daniilidis K. Event-based feature tracking with probabilistic data association. In: *2017 IEEE international conference on robotics and automation (ICRA)* (eds A Okamura, F Arai, F Arrichiello, et al.), Singapore, 29 May–3 June 2017, pp. 4465–4470. Los Alamitos, CA, USA: IEEE.
43. Zhu AZ, Atanasov N, and Daniilidis K. Event-based visual inertial odometry. *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, Honolulu, HI, USA, 21–26 July 2017, pp. 5816–5824. Los Alamitos, CA, USA.
44. Gehrig D, Rebecq H, Gallego G, et al. Asynchronous, photometric feature tracking using events and frames. In: *Proceedings of the european conference on computer vision (ECCV)* (eds V Ferrari, M Hebert, C Sminchisescu and Yair Weiss), pp. 750–765. Berlin, Heidelberg: Springer.
45. Li K, Shi D, Zhang Y, et al. Feature tracking based on line segments with the dynamic and active-pixel vision sensor (DAVIS). *IEEE Access* 2019; 7: 110874–110883.
46. Alzugaray I and Chli M. Asynchronous multi-hypothesis tracking of features with event cameras. In: *2019 international conference on 3D vision (3DV)*, Québec City, QC, Canada, 16–19 September 2019, pp. 269–278. Los Alamitos, CA, USA: IEEE.
47. Jammalamadaka SR and Sengupta A. *Topics in circular statistics*, volume 5. Singapore: World Scientific, 2001.
48. Gehrig D, Rebecq H, Gallego G, et al. EKLT: Asynchronous photometric feature tracking using events and frames. *Int J Comp Vis* 2019; 128: 601–618.