Received July 19, 2019, accepted July 28, 2019, date of publication August 6, 2019, date of current version August 22, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2933594

Feature Tracking Based on Line Segments With the Dynamic and Active-Pixel Vision Sensor (DAVIS)

KAIYUE LI[1], DIANXI SHI[2,3], YONGJUN ZHANG[2], RUOXIANG LI[1], WEI QIN[2,3], AND RUIHAO LI[2,3]

[1]College of Computer, National University of Defense Technology, Changsha 410073, China
[2]Artificial Intelligence Research Center (AIRC), National Innovation Institute of Defense Technology (NIIDT), Beijing 100166, China
[3]Tianjin Artificial Intelligence Innovation Center (TAIIC), Tianjin 300457, China

Corresponding authors: Dianxi Shi (dxshi@nudt.edu.cn) and Ruihao Li (liruihao2008@gmail.com)

This work was supported by the project Unmanaged Cluster System Software Architecture and Operational Support Platform, which is supported by the National Key Research and Development Program of China under Grant 2017YFB1001901. This work is also supported by the National Natural Science Foundation of China (Grant 61903377).

ABSTRACT As a novel vision sensor, the dynamic and active-pixel vision sensor (DAVIS) combines a standard camera and an asynchronous event-based sensor in the same pixel array. In this paper, we propose a novel asynchronous feature tracking method based on line segments with the DAVIS. The proposed method takes asynchronous events, synchronous image frames, and IMU data as the input. We first use the Harris detector to extract feature points and the Canny detector to extract line segment templates from image frames. Then we select spatio-temporal windows from asynchronous events and perform registration to estimate the optical flow. The registration is achieved by associating the extracted line segments with the events inside the window. Expectation maximization-iterative closest point (EM-ICP) is adopted for the registration. Afterward, we use the estimated optical flow and the IMU data to update the position of line segments, and take them as the new templates. We evaluate our method on the public event camera datasets. The results show that our method can achieve comparable performance to other methods in terms of accuracy and tracking time.

INDEX TERMS Feature tracking, event camera, EM-ICP, line segments, DAVIS.

I. INTRODUCTION

Feature tracking plays an important role in object tracking and Simultaneous Localization and Mapping (SLAM) [1], [2], and it is widely studied in robotics and computer vision. Images acquired from standard cameras easily produce motion blur when faced with fast movements, thus it is difficult for standard cameras to achieve feature tracking in this case. Besides, standard cameras provide no information to compute precise feature motions during the blind time between frames. What is more, standard cameras capture redundant information in static scenes, which not only causes waste of storage resources, but also consumes large amounts of additional computing resources in the processing procedure.

Bio-inspired event cameras, such as the Dynamic Vision Sensor (DVS) [3], overcome the above mentioned limitations of standard frame-based cameras. Different from standard cameras, event cameras only output pixel-level brightness changes. In the pixel array, each pixel independently generates an output (named "event") whenever the intensity change exceeds a threshold. Event cameras output spatio-temporal streams of asynchronous events. They have great advantages in fast movement scenes because of the low latency. Furthermore, they can also avoid recording redundant information in static or slowly changed scenes.

However, current frame-based feature tracking methods [4], [5] cannot be used directly to process the streams of asynchronous events, and new algorithms need to be developed to achieve feature tracking based on event cameras. Considering that event streams do not provide absolute brightness values, which makes it difficult to perform robust and long-term feature tracking, the combination of events and image frames provides a good solution to achieve fast and robust feature tracking. In this paper, we aim at the feature tracking problem based on the Dynamic and Active-pixel Vision Sensor

The associate editor coordinating the review of this manuscript and approving it for publication was Chao Tan.

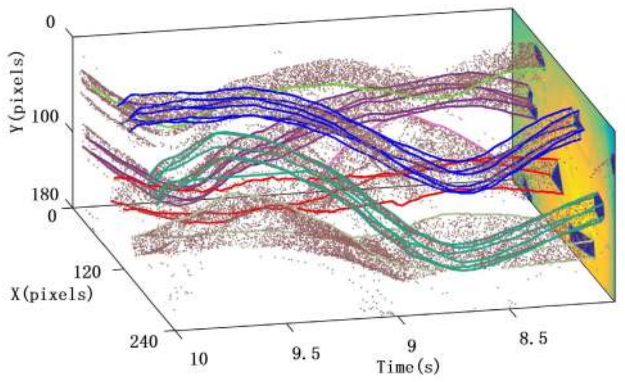


FIGURE 1. The tracked event features with our method on the shape_rotation sequence of Event Camera Dataset [7]. The start time for tracking is 8.0038*s*.

(DAVIS) [6]. Introduced in 2014, the DAVIS combines a standard camera with an asynchronous event-based sensor in the same pixel array. So it provides the possibility to solve the data association problem using both events and image frames.

Inspired by the fact that edges in scenes trigger events more easily because of their large gradients, we propose a novel feature tracking method based on line segments using DAVIS camera. As shown in Fig. 1, in our work, the tracked feature points are extracted using Harris *et al.* [8]. We adopt the Canny detector [9] to extract line segments from the image frame, and select spatio-temporal windows from asynchronous event streams based on the lifetime [10] of events. We then take line segments as the template to perform feature tracking for asynchronous event streams. EM-ICP [11] is used for the registration in the spatio-temporal coordinate. Finally, we use the estimated optical flow and the IMU data to update the position of line segments, and take the updated line segments as the new template. The proposed method combines events, image frames and IMU data for feature tracking, and this combination can yield accuracy improvement and achieve high-speed updates. The main contributions of this paper can be summarized as below:

- We present a novel asynchronous feature association method based on line segment templates. We detect Harris features and line segments from image frames, and track the Harris features in event streams by aligning the line segment templates with spatio-temporal event windows. EM-ICP is adopted for the alignment in order to estimate the optical flow.
- We combines the estimated optical flow and the IMU data to update line segment templates. This combination ensures the accuracy and high speed of the template correction, which can enhance the tracking performance.
- We implement our proposed tracking method with C++ and perform evaluational experiments on the public Event Camera Datasets [7]. The results show that our method demonstrates comparable performance to the previous event tracking methods.

The rest of this paper is organized as follows. Section II reviews the related works. The details of our proposed tracking method are described in Section III. Section IV gives our experimental results. Finally, our conclusion and future work are discussed in Section V.

II. RELATED WORK

In the past decade, visual SLAM has become a hot research topic in both robotics and computer vision community. With the emergence of novel event cameras, event-based SLAM methods [12]–[15] draw lots of attention from researchers, and as an important component of SLAM system, event-based feature detection and tracking are widely studied.

A. EVENT-BASED FEATURE DETECTION

Event-based feature detection can mainly be divided into two kinds of methods: one relies on artificially synthesized event frames and the other is directly based on asynchronous event streams. For event frame-based feature detectors, the event frames are synthesized by events according to a temporal window, then the traditional feature detectors such as Harris, FAST [16] can be used for feature detection on these synthesized event frames. As for event stream based feature detectors, they do not require artificially synthesized event frames and can directly operate on asynchronous events. Mueggler *et al.* [17] presented an event-based FAST corner detector for event streams and improved the event-based Harris corner detector [18]. Later, [19] presented an speed-up version of [18]. Alzugaray and Chli [20] introduced a filter for event streams to remove redundant events before detecting, and the filter can enhance both accuracy and real-time performance.

B. EVENT-BASED FEATURE TRACKING USING EVENTS

For feature tracking methods using asynchronous events, Lagorce *et al.* [21] presented an approach which defines and tracks various kernels in event streams. These kernels are modeled with different mathematical tools, such as Gaussian, Gabor, the combination of Gabor functions and arbitrary user-defined kernels. The proposed tracker can achieve robust tracking even when faced with variations of position, scale, and orientation. Benosman *et al.* [22] computed the optical flow for events by using a local plane-fitting algorithm. They selected a spatio-temporal window with size $L \times L \times 2\Delta t$ centered on each event and used a plane to fit events. Then the optical flow of events was estimated by using a differential approach on the plane. Mueggler *et al.* [10] changed the spatio-temporal window from $2\Delta t$ to Δt to overcome a Δt latency. Clady *et al.* [23] extended the method of [22], and selected corner events which occur at the intersection of two or more fitting planes. The authors computed the velocities of corner events using the intersection of several geometric constraints and avoided the aperture problem. Bardow *et al.* [24] estimated optical flow and scene intensity simultaneously by minimizing a cost function. The cost function considers three constraints: the intensity changes acquired by events in a sliding time window, the photometric consistency, and the spatio-temporal smoothness. However, this method is time-consuming and needs GPU for accelerating the computational speed. Zhu *et al.* [25] proposed a method to track corners by computing optical flow using the EM algorithm [11]. This method can achieve long-time feature tracking. Later, [26] extended this method into a visual inertial odometry system, in which they used IMU to correct feature positions.

Rebecq *et al.* [14] computed optical flow by applying the Lukas-Kanade method [27] to motion-corrected event frames. The motion-corrected event frames were generated by projecting events to the same plane using rotation R and translation t. The depth of these 3D points was estimated by 2D linear interpolation of landmarks. Alzugaray and Chli [20] proposed an asynchronous event corner detector which was called 'Arc*'. Based on the proposed 'Arc*', the authors employed a graph and computed the Euclidean distance for feature tracking. Later, they [28] also proposed a novel descriptor for asynchronous events and used it for tracking, and they adopted descriptor distance instead of the Euclidean distance for tracking.

C. EVENT-BASED FEATURE TRACKING USING BOTH EVENTS AND IMAGE FRAMES

Tedaldi *et al.* [29] proposed a tracking method which used edges from image frames to perform registration. The authors took advantage of the fact that events were generated due to the brightness change and most events took place on edges. For each incoming asynchronous event, they performed registration via a transformation between edges and events.

Gehrig *et al.* [30] also proposed an event tracking method which combined image frames and events. They solved the data association problem by exploiting the strength of brightness gradients. The features were firstly extracted from standard image frames, and the brightness-increment images from events were generated. Then the warp and optical flow of features were estimated simultaneously by achieving data association between the brightness-increment images and intensity gradient of standard image frames.

III. METHODOLOGY

Inspired by [25], [29], our method adopts the line segments information extracted from image frames to compute optical flow. These line segments are used as the bridge to establish the correspondence between events. In order to enhance performance and achieve asynchrony, we also use the IMU data to correct the line segments during the tracking process.

An overview of our proposed feature tracking method is shown in Fig. 2. In the initialization period, the feature points and edge map are detected from image frames, and the edge map is divided into line segment templates for every detected feature point. In the tracking period, spatio-temporal windows are selected according to the lifetime of events, then the spatio-temporal windows and line segment templates are used for optical flow estimation by using EM-ICP algorithm. After getting the estimated optical flow, the feature points and line segment templates will be updated, which includes two steps. In step 1, we update feature positions using the estimated optical flow (see details in Section III-B). In step 2, we first use the estimated optical flow to update the position of line segment templates and then use the IMU data to correct the position of line segment templates (see details in Section III-C).

Algorithm 1 describes the implementation of the proposed tracking method. The time interval $[t_0, t_N]$ can be divided into a series of subintervals $[t_0, t_1], \ldots, [t_{N-1}, t_N]$. The spatio-temporal window is constructed by collecting asynchronous events. The temporal size of the window is determined by the lifetime of events, and the spatial size of the window is $s \times s$, in which s is 29.

Algorithm 1 Feature Tracking Based on Line Segments With DAVIS Camera

```
    Initialization:
1:  Detect features $\{\mathbf{f}_i\}_{i=1}^{n}$ from image frames using Harris detector.
2:  Detect line segments using Canny detector and select line segment templates $P_{\mathrm{f}_i} = \{\mathbf{p}_j\}_{j=1}^{m_i}$ for every detected Harris feature $\mathbf{f}_i$.
    Tracking:
3:  while every subinterval $\Delta t$ do
4:    for $i = 1 : n$ do
5:      Select spatio-temporal windows for $\mathbf{f}_i$.
6:      Estimate optical flow $\mathbf{v}_i$ of $\mathbf{f}_i$ by (6),(7).
7:      Update feature $\mathbf{f}_i \leftarrow \mathbf{f}_i + \mathbf{v}_i \Delta t$.
8:      Update the position of line segment templates $P_{\mathrm{f}_i}$ using the estimated $\mathbf{v}_i$.
9:      Correct the position of line segment templates $P_{\mathrm{f}_i}$ using IMU data.
10:   end for
11:   Compute the $\Delta t$ of next subinterval by lifetime of events.
12: end while
```

A. FEATURE AND EDGE DETECTION FROM FRAMES

We detect feature points $\{\mathbf{f}_i\}_{i=1}^{n}$ using Harris detector and extract the edge map using Canny detector from the image frame at time t_0. As shown in Fig. 3(b), the red points represent feature points and the blue points represent pixels on the edge map. For every feature point, we take the line segments (composed of the blue points) in the same red square patch as the templates $P_{\mathrm{f}_i} = \{\mathbf{p}_j \mid \mathbf{p}_j(t_0) \in H_i\}_{j=1}^{m_i}$. H_i represents the square patch around $\mathbf{f}_i$, $\mathbf{p}_j$ is the jth template point on the line segments in H_i, and m_i is the number of template points of $\mathbf{f}_i$.The patch size is determined by trial and error. It should not be too large for the reason that we assume the pixels in the same patch have the same optical flow. For simplicity, we set the same size for all patches.

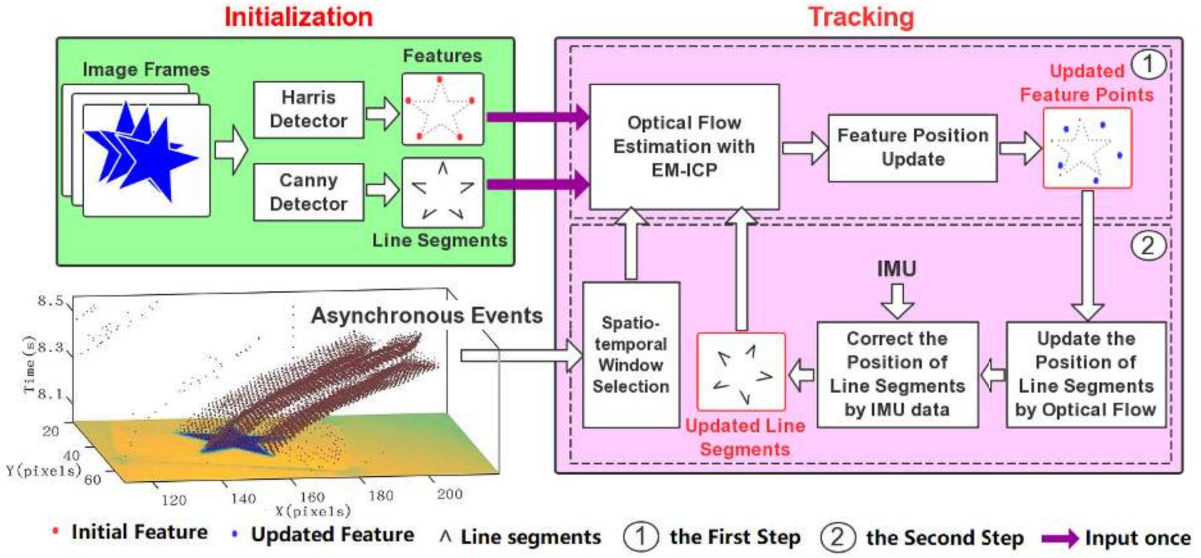


FIGURE 2. **The overview of the proposed feature tracking method based on line segments. The method can achieve asynchronous feature tracking for events. We extract line segment templates from image frames, and select spatio-temporal windows from asynchronous events. Then we estimate optical flow by associating line segment templates with spatio-temporal event windows by minimizing the distances between them using EM-ICP algorithm. We get the updated feature points by estimated optical flow. Finally, the estimated optical flow is used for updating the position of line segments and then the IMU data is used for correcting the position of line segments. The updated line segments are taken as the new templates for optical flow estimation.**

For scenes of complex texture, the detected edge map may obtain some useless edges which have a negative impact on establishing correspondences between template points and events. The filter can be used to reject useless edges and extract the main edges from the edge map.

B. OPTICAL FLOW ESTIMATION AND FEATURE UPDATE

Fig. 3(c), 3(d) show the process of estimating optical flow and updating feature positions. Considering the ith feature $\mathbf{f}_i$ detected from the image frame at time t_0. In the time subinterval $[t_0, t_1]$, the motion of $\mathbf{f}_i$ in image plane can be described as follow:

$$\mathbf{f}_i(t_1) = \mathbf{f}_i(t_0) + \int_{t_0}^{t_1} \dot{\mathbf{f}}_i(t)dt \tag{1}$$

$\dot{\mathbf{f}}_i(t)$ is the optical flow of $\mathbf{f}_i$ at time t. For the reason that the time subinterval $[t_0, t_1]$ is small, we assume $\mathbf{f}_i$ has the constant optical flow $\mathbf{v}$ during $[t_0, t_1]$. Based on this assumption, we compute $\mathbf{v}$ by following steps.

1) SELECT THE SPATIO-TEMPORAL WINDOW

For the event set W_1 during the tracking procedure in subinterval $[t_0, t_1]$, we define a subset of W_1 as:

$$W_1^{\mathbf{f}_i} = \{\mathbf{e}_k | t_0 < t_{e_k} < t_1, \mathbf{x}_k \in H_i\}_{k=1}^{l_i} \tag{2}$$

where $W_1^{\mathbf{f}_i}$ is the spatio-temporal window around feature $\mathbf{f}_i$, and $\mathbf{e}_k(\mathbf{x}_k, t_{e_k})$ is the kth event of $W_1^{\mathbf{f}_i}$ which happens at time t_{e_k} and its pixel location is $\mathbf{x}_k$. l_i represents the number of events in $W_1^{\mathbf{f}_i}$.

2) ASSOCIATING THE SPATIO-TEMPORAL WINDOW WITH THE LINE SEGMENT TEMPLATES

We assume that the events in $W_1^{\mathbf{f}_i}$ have the same optical flow considering the small size of H_i. Then we can get the updated position of $\mathbf{e}_k$ at time t_0 using the optical flow $\mathbf{v}$:

$$\mathbf{x}'_k = \mathbf{x}_k - \mathbf{v}(t_{e_k} - t_0) \tag{3}$$

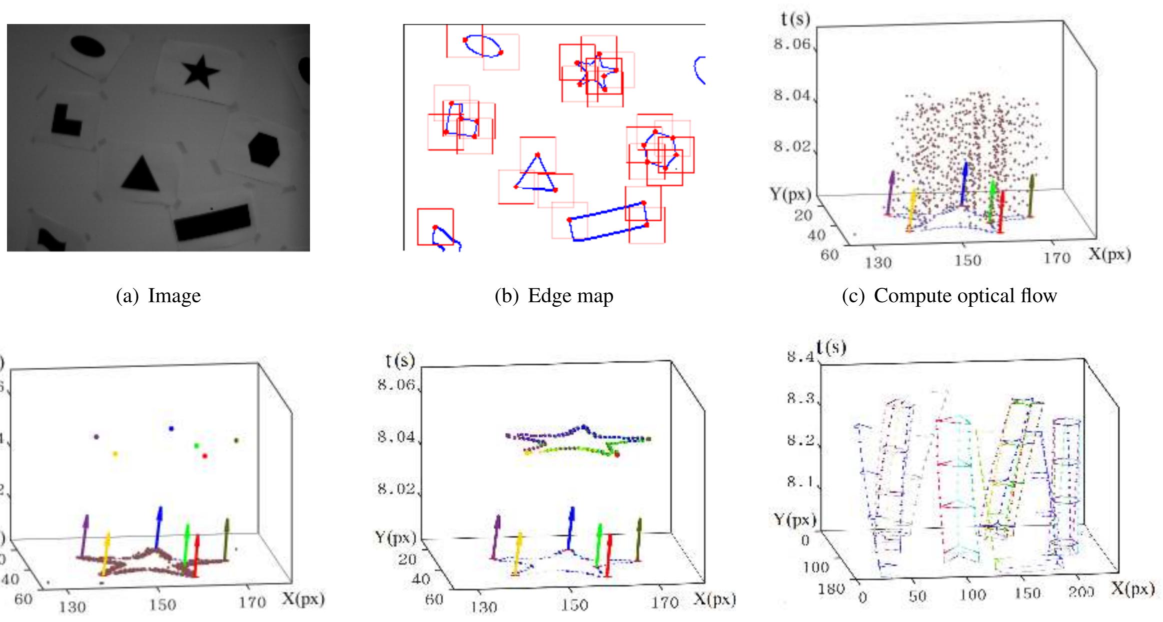
For convenience, we define $\bar{t}_{e_k} = t_{e_k} - t_0$. Assuming $\mathbf{e}_k$ is generated by the motion of $\mathbf{p}_j$, we have $\mathbf{e}_k = \pi(\mathbf{p}_j)$ and $\mathbf{x}'_k = \mathbf{p}_j(t_0)$. $\mathbf{p}_j$ is the jth template point of the line segment templates $P_{\mathbf{f}_i}$. And $\mathbf{p}_j(t_0)$ represents the position of the template point $\mathbf{p}_j$ at time t_0. In this way, we have associated the event $\mathbf{e}_k$ with the template point $\mathbf{p}_j$ by optical flow $\mathbf{v}$. Notice that one template point may correspond to multiple events. Then the following equation should hold for any pair (k, j):

$$\| (\mathbf{x}_k - \bar{t}_{e_k}\mathbf{v}) - \mathbf{p}_j(t_0) \|^2_{\mathbf{e}_k=\pi(\mathbf{p}_j)} = 0 \tag{4}$$

However the correspondence π between $\mathbf{e}_k$ and $\mathbf{p}_j$ is unknown. Referring to [25], we represent the correspondence between events and template points with probability $r_{kj} := P(\mathbf{e}_k = \pi(\mathbf{p}_j))$. The symbol ':=' means it is a definition. Then the optical flow $\mathbf{v}$ of $\mathbf{f}_i$ is computed by minimizing the following function:

$$min \sum_{k=1}^{l_i} (\sum_{j=1}^{m_i} r_{kj} \| (\mathbf{x}_k - \bar{t}_{e_k}\mathbf{v}) - \mathbf{p}_j(t_0) \|^2) \tag{5}$$

where we take the probability r_{kj} as the weight of the distance between the updated position of $\mathbf{e}_k$ at t_0 and $\mathbf{p}_j(t_0)$. r_{kj} can be



(a) Image (b) Edge map (c) Compute optical flow

(d) Update features (e) Update line segments (f) Tracking performance

FIGURE 3. The tracking process of our method. (a) The standard image frame. (b) The feature points(red points) and the line segment templates (blue line segments inside the red square patches). (c) Optical flow estimation by aligning the line segment template and the spatio-temporal event window. The color arrows represent the optical flow. The brown points represent events. (d) Feature position update. The color points on the top represent the updated features. (e) Line segment position update using both estimated optical flow and IMU data. The pentagram on the top composes of updated line segments. (f) The tracking performance. We draw the line segment templates during the tracking.

calculated by:

$$r_{kj} = \frac{\| (\mathbf{x}_k - \bar{t}_{e_k}\mathbf{v}) - \mathbf{p}_j(t_0) \|^2}{\sum_{j=1}^{m_i} \| (\mathbf{x}_k - \bar{t}_{e_k}\mathbf{v}) - \mathbf{p}_j(t_0) \|^2} \tag{6}$$

Since the calculation of r_{kj} is related to the optical flow $\mathbf{v}$, we use the EM-ICP algorithm [11] to iteratively calculate $\mathbf{v}$. The method are summarized in Algorithm 2. At the initial part of the algorithm, we initialize the optical flow $\mathbf{v}$ as $\mathbf{0}$.

In the E step, we update r_{kj} with (6) where $\mathbf{v}$ is given from the M step of last iteration.

In the M step, we update $\mathbf{v}$ by solving the linear least squares problem of (5):

$$\mathbf{v} = \frac{\sum_{k=1}^{l_i} \sum_{j=1}^{m_i} r_{kj}\bar{t}_{e_k}(\mathbf{x}_k - \mathbf{p}_j(t_0))}{\sum_{k=1}^{l_i} \sum_{j=1}^{m_i} r_{kj}\bar{t}_{e_k}^2} \tag{7}$$

We iteratively conduct E step and M step until the change $\Delta\mathbf{v}$ is below a threshold σ, and we can get the final estimated optical flow $\mathbf{v}$. We then update the position of feature $\mathbf{f}_i$ at time t_1 using optical flow:

$$\mathbf{f}_i(t_1) = \mathbf{f}_i(t_0) + \mathbf{v}(t_1 - t_0) \tag{8}$$

C. LINE SEGMENT TEMPLATES UPDATE

Considering the assumption that events in the same square patch have the same optical flow, we also use the estimated optical flow to update the position of the line segment templates.

Algorithm 2 Optical Flow Estimation for Feature $\mathbf{f}_i$

```
Input: The line segment templates $P_{f_i} = \{\mathbf{p}_j\}_{j=1}^{m_i}$, the patch
    $H_i$, the threshold $\sigma$, the time intervel $[t_0, t_1]$.
Output: Estimated optical flow $\mathbf{v}$.
 1: Initialize the optical flow $\mathbf{v} = \mathbf{0}$.
 2: Select the events set $W_1^{f_i} = \{\mathbf{e}_k | t_0 < t_{e_k} < t_1, \mathbf{x}_i \in H_i\}$.
 3: while $\Delta\mathbf{v} > \sigma$ do
 4:     for $k = 1 : l_i$ do
 5:         for $j = 1 : m_i$ do
 6:             if $\mathbf{p}_j$ is associated with $\mathbf{e}_k$ then
 7:                 Update $r_{kj}$ using (6).
 8:             else
 9:                 $r_{kj} = 0$.
10:             end if
11:         end for
12:     end for
13:     Update $\mathbf{v}$ using (7).
14:     Compute the change $\Delta\mathbf{v}$.
15: end while
```

However, the estimated optical flow can only help to update the translation of the line segment templates. Without rotation correction for line segment templates, the tracked position of feature points will deviate from ground truth after several iterations. To solve this problem, we introduce IMU to correct the position of the line segment templates in spatio-temporal event coordinate (as shown in Fig. 3(e)).

We compute the relative positions between the template points and corresponding feature point and correct the relative positions by IMU.

For the jth template point $\mathbf{p}_j \in P_{\mathrm{f}_i}$, we update the position of $\mathbf{p}_j$ as follows.

1) UPDATE THE POSITION OF p_j BY OPTICAL FLOW

$$\mathbf{p}_j(t_1) = \mathbf{p}_j(t_0) + \mathbf{v}(t_1 - t_0) \tag{9}$$

$\mathbf{v}$ is the optical flow of feature $\mathbf{f}_i$ computed from Section III-B.

2) CORRECT THE POSITION OF p_j USING IMU DATA

we correct the position of $\mathbf{p}_j$ by recalculating the relative position between the template point $\mathbf{p}_j$ and feature point $\mathbf{f}_i$ at time t_1. Define the relative position $\mathbf{p}_j^{relative}(t) = \mathbf{p}_j(t) - \mathbf{f}_i(t)$. We use IMU data to calculate the relative position $\mathbf{p}_j^{relative}(t_1)$ at time t_1.

We use P, F to represent the 3D points corresponding to $\mathbf{p}_j$, $\mathbf{f}_i$ at time t_0. Template point $\mathbf{p}_j$ and feature point $\mathbf{f}_i$ in 2D pixel coordinate are represented by homogeneous coordinates $(u, v, 1)$, and the 3D points P and F in 3D camera coordinate are represented by (x, y, z). Then we have below equations at time t_0 and t_1:

$$\begin{cases} s^0_{\mathrm{p}_j}\mathbf{p}_j(t_0) = KP \\ s^0_{\mathrm{f}_i}\mathbf{f}_i(t_0) = KF \end{cases} \tag{10}$$

$$\begin{cases} s^1_{\mathrm{p}_j}\mathbf{p}_j(t_1) = K(RP + t) \\ s^1_{\mathrm{f}_i}\mathbf{f}_i(t_1) = K(RF + t) \end{cases} \tag{11}$$

where R, t are the rotation matrix and translation vector of event camera which are obtained by integrating the IMU data. K is the camera projection matrix. $s^0_{\mathrm{p}_j}$, $s^0_{\mathrm{f}_i}$ are the depth of P, F in camera coordinate at time t_0, and $s^1_{\mathrm{p}_j}$, $s^1_{\mathrm{f}_i}$ are the depth of P, F in camera coordinate at time t_1. Then we get the relative position $\mathbf{p}_j^{relative}(t_1)$ by (11):

$$\begin{aligned} \mathbf{p}_j^{relative}(t_1) &= \mathbf{p}_j(t_1) - \mathbf{f}_i(t_1) \\ &= (\frac{1}{s^1_{\mathrm{p}_j}}KRP - \frac{1}{s^1_{\mathrm{f}_i}}KRF) + (\frac{1}{s^1_{\mathrm{p}_j}}Kt - \frac{1}{s^1_{\mathrm{f}_i}}Kt) \end{aligned} \tag{12}$$

Assuming $s^1_{\mathrm{p}_j} = s^1_{\mathrm{f}_i}$ which means the template point P and feature point F have the same depth in event camera coordinate at time t_1. Then we have:

$$\mathbf{p}_j^{relative}(t_1) = \frac{1}{s^1_{\mathrm{p}_j}}KRP - \frac{1}{s^1_{\mathrm{f}_i}}KRF \tag{13}$$

Substituting (10) into (13), we get:

$$\mathbf{p}_j^{relative}(t_1) = \frac{s^0_{\mathrm{p}_j}}{s^1_{\mathrm{p}_j}}KRK^{-1}\mathbf{p}_j(t_0) - \frac{s^0_{\mathrm{f}_i}}{s^1_{\mathrm{f}_i}}KRK^{-1}\mathbf{f}_i(t_0) \tag{14}$$

Considering the homogeneous representation of point $\mathbf{p}_j$ and $\mathbf{f}_i$, the final formula for calculating the relative position is:

$$\mathbf{p}_j^{relative}(t_1) = Nor(KRK^{-1}\mathbf{p}_j(t_0)) - Nor(KRK^{-1}\mathbf{f}_i(t_0)) \tag{15}$$

where $Nor(\cdot)$ represents the normalization operation which normalizes the vector (x, y, z) to $(\frac{x}{z}, \frac{y}{z}, 1)$. The coefficients $\frac{s^0_{\mathrm{p}_j}}{s^1_{\mathrm{p}_j}}$, $\frac{s^0_{\mathrm{f}_i}}{s^1_{\mathrm{f}_i}}$ can be omitted by the normalization operation.

After we get the relative position $\mathbf{p}_j^{relative}(t_1)$ between the template point $\mathbf{p}_j$ on line segment and feature point $\mathbf{f}_i$ through IMU data, the final position of the template point $\mathbf{p}_j$ at time t_1 is updated as below:

$$\mathbf{p}_j(t_1) = \mathbf{p}_j^{relative}(t_1) + \mathbf{f}_i(t_1) \tag{16}$$

IV. EXPERIMENTS

We implement our proposed feature tracking method using C++, and evaluate it on the public Event Camera Datasets [7]. The datasets contain a series of sequences acquired by the event camera in different scenes. We generate the ground truth using Lukas-Kanade method [31]. The feature positions between image frames are obtained by linear interpolation. For parameter settings in spatio-temporal window, the spatial patch size is set to 29×29 pixels, and the temporal size is set according to the lifetime of events. In experiments, we set the temporal size to 3×lifetime [10](three times lifetime), which means the event moves about 3 pixels. All experiments are implemented on a laptop equipped with an Intel Core i7 2.8GHz CPU, and 8G RAM.

A. QUALITATIVE EVALUATION OF OUR TRACKING METHOD

We test our feature tracker on sequences of shapes, poster, boxes and outdoors from the public Event Camera Datasets. As shown in Fig. 4, we draw the trajectories of the tracked asynchronous event features over a time interval on image frames. Fig. 4(a) shows the tracking result of our method in the shapes sequence which contains rotation of event camera. For the poster sequence, as shown in Fig. 4(b), we select a subsequence that the event camera moves towards an opposite direction suddenly. The inflection of the trajectories in Fig. 4(b) occurs at the time that the velocity of the event camera is 0. The result shows our method can also track the features when the event camera changes its motion direction. Fig. 4(c) shows the 3D textured scene with boxes. Fig. 4(d) shows the trajectories of feature tracking on the sequence of outdoors. This sequence was obtained by a hand-held event camera in an outdoor environment. The event camera sometimes shook in the hand during recording.

We also give the tracking results on a subsequence from the shapes_translation sequence which contains high-speed motion of event camera. Fig. 5 shows the trajectories of tracked features. The colored points represent the positions of features over time. We show five consecutive image frames for reference. Our method updates the position of the event features about 10 times between two consecutive image frames. Fig. 6 shows the case with high-speed rotation of event camera. In this figure, the event camera changed its movement direction between the third and fourth image

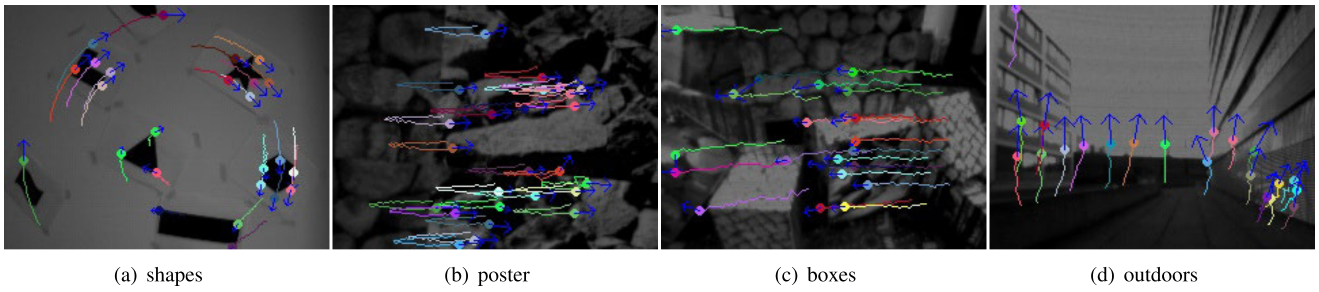


FIGURE 4. **The trajectories of the tracked event features in different scenes: (a) the shapes sequence with the rotational motion. (b) the poster sequence with an event camera moving in two opposite directions. (c) the boxes sequence in a 3D environment. (d) the outdoors sequence which was recorded with a hand-held event camera. The colored lines represent the trajectories of features during a time interval. The blue arrows represent the velocities of features.**

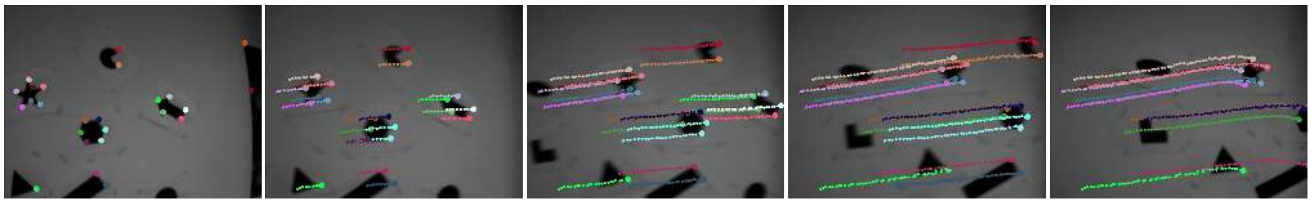


FIGURE 5. **Feature tracking performance on shapes_translation sequence. The figure contains five consecutive image frames, and the colored points represent the position of event features.**

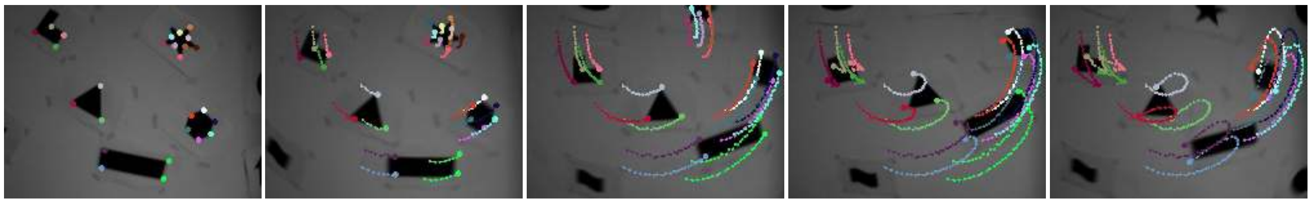


FIGURE 6. **Feature tracking performance on shapes_rotation sequence. The figure contains five consecutive image frames, and the colored points represent the position of event features.**

frames, and it is difficult for feature tracking. As shown in Fig. 6, our method still works well and the asynchronous event features are tracked accurately. We do not require to extract line segments from every image frame, and in the tracking process, the line segment templates can be updated continuously by the IMU data and the estimated optical flow, so our method can still work in the case of motion blur.

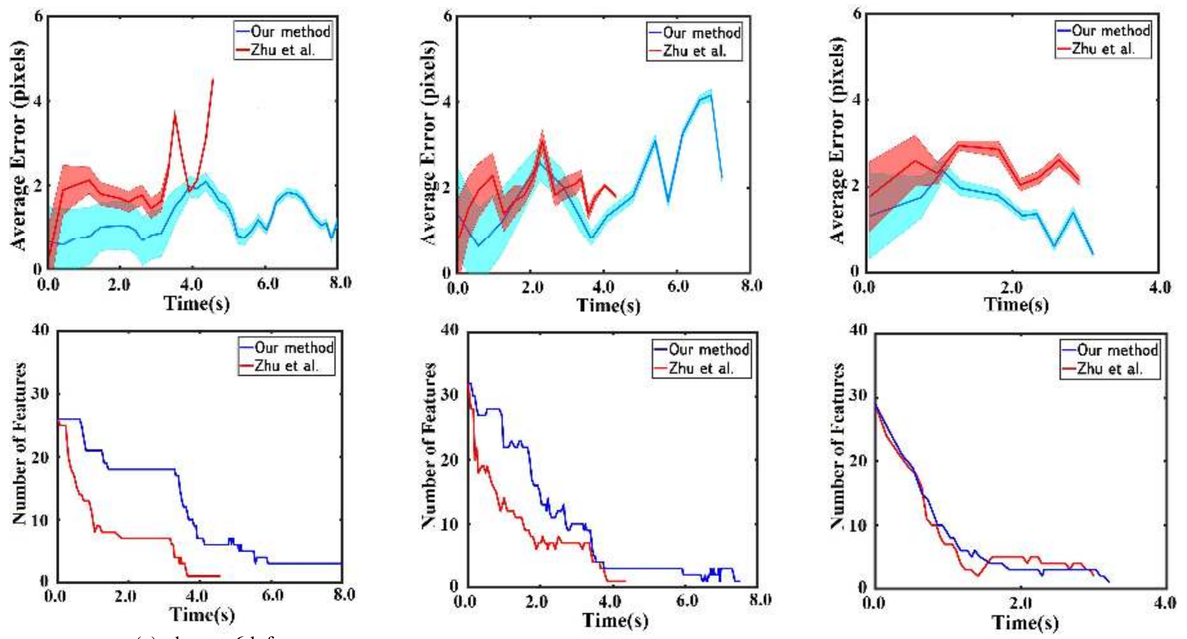
B. QUANTITATIVE EVALUATION OF OUR TRACKING METHOD

To evaluate our tracking method quantitatively, we test the average tracking error and average tracking time of our method. Meanwhile, we compare our tracking method with different tracking methods [12], [26]. For the method proposed by Zhu *et al.* [26], we use the public implementation[1] of [26] and test it on the same datasets. In addition, in order to maintain consistency, we use the same initial features detected from the image frames in our method with [26]. For the method proposed by Kueng *et al.* [12], we use the experimental results of [12] reported in [30].

[1]https://github.com/daniilidis-group/event_feature_tracking

Fig. 7 shows the details of average tracking error and number of features over time of our method and the proposed method in [26] tested on the sequences of shapes_6dof, poster_6dof and boxes_6dof. As shown in Fig. 7(a), our method is more accurate and can track features for a longer time than [26] on the shapes_6dof sequence. The average tracking error of our method is below 2 pixels. At the first 3 seconds, most features are tracked well. At the 3rd second, for the reason that some features move out of the camera view, the number of available features reduce rapidly. For the poster_6dof sequence in Fig. 7(b), our method achieves longer tracking when compared with the proposed method in [26]. In Fig. 7(c), our method have better accuracy than the method proposed in [26], but from the 2nd second, our method has fewer available features.

Table 1 shows the performances of average tracking error and average tracking time with different methods: our method, [26] and [12]. Besides, in order to evaluate the contribution of IMU to tracking performance, we perform our method with and without IMU. We only get the result of [12] on sequences of shapes_6dof, poster_6dof and boxes_6dof.



(a) shapes_6dof (b) poster_6dof (c) boxes_6dof

FIGURE 7. The average tracking error and the number of features over time of our method and [26]. We used the sequences of shapes_6dof(a), poster_6dof(b) and boxes_6dof(c) for testing. For figures in the first row, the center lines represent the average pixel error of feature positions over time. The width of the band around the center line represents the proportion of available feature tracks. For figures in the second row, the lines represent the number of feature points over time.

TABLE 1. Average tracking error and average tracking time with different methods tested on several sequences of the public Event Camera Datasets [7]. The best results are made in bold.

Sequences	Average tracking error[px]				Average tracking time[s]			
	Ours (with IMU)	Ours (without IMU)	Zhu [26]	Kueng [12]	Ours with IMU)	Ours (without IMU)	Zhu [26]	Kueng [12]
shapes_translation	0.9339	**0.7414**	1.7583	×	**3.6018**	3.4911	1.4904	×
shapes_rotation	**1.0692**	1.4766	1.3065	×	**3.7994**	1.7499	0.2910	×
shapes_6dof	1.0991	**0.8154**	1.7480	1.75	**4.1130**	3.3706	1.4587	1.53
poster_translation	**0.8762**	1.0282	1.4949	×	**2.0151**	1.7250	1.8623	×
poster_rotation	**1.2331**	2.0937	1.7334	×	**2.5980**	0.5127	1.1671	×
poster_6dof	**1.5797**	1.7042	1.8692	2.86	**2.5066**	0.9367	2.4901	0.65
boxes_translation	1.4811	**1.4259**	1.6566	×	1.5132	0.9169	**1.6931**	×
boxes_rotation	**1.5545**	1.9986	2.2872	×	**2.0689**	1.0337	0.8028	×
boxes_6dof	1.7779	**1.6167**	2.3572	3.10	1.2101	0.6266	**1.7256**	0.78
overall	**1.2894**	1.4334	1.8013	2.57	**2.6029**	1.5959	1.4433	0.99

For the performance of average tracking error, our method with IMU has smaller average tracking error than [26] and [12] as shown in Table 1. On the sequence of shapes, the average tracking error of our method with IMU is about 1 pixel. There is about a 0.5-pixel reduction compared with the method proposed in [26]. For high complexity sequences, the average tracking error of our method with IMU also reduces. In total, the average tracking error of our method with IMU is below 2 pixels. For the performance of average tracking time, our tracking method with IMU achieves longer time feature tracking than [26] and [12] on the sequences of shapes and poster. Specially, for the shapes sequence, the average tracking time of our method with IMU is about 3 times in comparison with [26]. This proves the robustness of our proposed method. Different from the method proposed in [26] which has obvious shorter tracking time on the rotation sequences than that on the translation sequences, our method with IMU shows good performance in both translational and rotational cases. The last line shows the overall average tracking error and average tracking time among different sequences with different methods. It shows our method with IMU and without IMU both have smaller average tracking error and longer average tracking time compared with the other two methods.

To evaluate the contribution of IMU to our method, we also compare the results of our method with and without IMU.

For the performance of average tracking error, our method with IMU has smaller error than the case without IMU on the rotation sequences. However, for the sequences of shapes_translation and shapes_6dof, the average tracking error without IMU is smaller while the average tracking time is close to the case with IMU. As to the performance of average tracking time, our method with IMU achieves longer time for feature tracking than the case without IMU. Besides, on the rotation sequences, the average tracking time is very short when not using IMU.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we present a novel asynchronous feature tracking method which is based on image frames, events, and IMU data. We extract line segments from image frames and take them as templates to estimate the optical flow. The optical flow is estimated by minimizing the distance between spatio-temporal event windows and the templates. Then we introduce IMU data to correct the position of line segments and take them as the new templates. We test our method on several sequences of the public Event Camera Datasets [7] and it shows comparable performance in terms of accuracy and robustness in comparison with other methods.

In the future, we will improve our tracking method by adopting effective filter and outlier rejecter. Additionally, based on our proposed tracking method, We will also achieve semi-dense map construction based on asynchronous events and 6-DoF pose estimation with DAVIS.

REFERENCES

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, Jun. 2006.

[2] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping: Part II," *IEEE Robot. Autom. Mag.*, vol. 13, no. 3, pp. 108–117, Sep. 2006.

[3] P. Lichtsteiner, C. Posch, and T. Delbrück, "A 128×128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," *IEEE J. Solid State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008.

[4] C. Leng, H. Zhang, B. Li, G. Cai, Z. Pei, and L. He, "Local feature descriptor for image matching: A Survey," *IEEE Access*, vol. 7, pp. 6424–6434, 2019.

[5] Z. Yang, T. Dan, and Y. Yang, "Multi-temporal remote sensing image registration using deep convolutional features," *IEEE Access*, vol. 6, pp. 38544–38555, 2018.

[6] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240×180 130 db 3 μs latency global shutter spatiotemporal vision sensor," *IEEE J. Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, Oct. 2014.

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

[8] C. G. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. Alvey Vis. Conf.*, vol. 15, no. 50, pp. 5210–5244, 1988.

[9] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.

[10] E. Mueggler, C. Forster, N. Baumli, G. Gallego, and D. Scaramuzza, "Lifetime estimation of events from dynamic vision sensors," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2015, pp. 4874–4881.

[11] S. Granger and X. Pennec, "Multi-scale EM-ICP: A fast and robust approach for surface registration," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Apr. 2002, pp. 418–432.

[12] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza, "Low-latency visual odometry using event-based feature tracks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 16–23.

[13] H. Rebecq, T. Horstschäfer, G. Gallego, and D. Scaramuzza, "EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real time," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 593–600, Apr. 2017.

[14] H. Rebecq, T. Horstschäefer, and D. Scaramuzza, "Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, Sep. 2017, pp. 1–12.

[15] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate SLAM? Combining events, images, and IMU for robust visual SLAM in HDR and high-speed scenarios," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 994–1001, Apr. 2018.

[16] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, May 2006, pp. 430–443.

[17] E. Mueggler, C. Bartolozzi, and D. Scaramuzza, "Fast event-based corner detection," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, Sep. 2017, pp. 1–11.

[18] V. Vasco, A. Glover, and C. Bartolozzi, "Fast event-based Harris corner detection exploiting the advantages of event-driven cameras," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 4144–4149.

[19] R. Li, D. Shi, Y. Zhang, K. Li, and R. Li, "FA-Harris: A fast and asynchronous corner detector for event cameras," 2019, *arXiv:1906.10925*. [Online]. Available: https://arxiv.org/abs/1906.10925

[20] I. Alzugaray and M. Chli, "Asynchronous corner detection and tracking for event cameras in real time," *IEEE Robot. Automat. Lett.*, vol. 3, no. 4, pp. 3177–3184, Oct. 2018.

[21] X. Lagorce, C. Meyer, S.-H. Ieng, D. Filliat, and R. Benosman, "Asynchronous event-based multikernel algorithm for high-speed visual features tracking," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 8, pp. 1710–1720, Aug. 2015.

[22] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi, "Event-based visual flow," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 2, pp. 407–417, Feb. 2013.

[23] X. Clady, S.-H. Ieng, and R. Benosman, "Asynchronous event-based corner detection and matching," *Neural Netw.*, vol. 66, pp. 91–106, Jun. 2015.

[24] P. Bardow, A. J. Davison, and S. Leutenegger, "Simultaneous optical flow and intensity estimation from an event camera," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 884–892.

[25] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based feature tracking with probabilistic data association," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May/Jun. 2017, pp. 4465–4470.

[26] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based visual inertial odometry," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, Jul. 2017, pp. 5816–5824.

[27] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. Artif. Intell. (IJCAI)*, Aug. 1981, pp. 674–679.

[28] I. Alzugaray and M. Chli, "ACE: An efficient asynchronous corner tracker for event cameras," in *Proc. Int. Conf. 3D Vis. (3DV)*, Sep. 2018, pp. 653–661.

[29] D. Tedaldi, G. Gallego, E. Mueggler, and D. Scaramuzza, "Feature detection and tracking with the dynamic and active-pixel vision sensor (DAVIS)," in *Proc. 2nd Int. Conf. Event-Based Control, Commun., Signal Process. (EBCCSP)*, Jun. 2016, pp. 1–7.

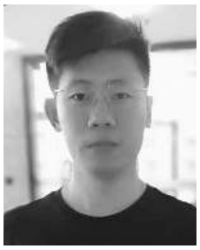
[30] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza, "Asynchronous, photometric feature tracking using events and frames," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 750–765.

[31] S. Baker and I. Matthews, "Lucas–Kanade 20 years on: A unifying framework," *Int. J. Comput. Vis.*, vol. 56, no. 3, pp. 221–255, 2004.

KAIYUE LI received the B.S. degree in computer science from Northeastern University, Shenyang, China, in 2017. He is currently pursuing the degree with the National University of Defense Technology. His research interests focus on computer vision, event camera, and SLAM.

DIANXI SHI received the B.S., M.S., and Ph.D. degrees from the National University of Defense Technology, Changsha, China, in 1989, 1996, and 2000, respectively, all in computer science. From 2003 to 2011, he served as an Associate Researcher and a Master Tutor of the Institute of Network and Information Security, College of Computer, National University of Defense Technology. From 2011 to 2018, he was a Researcher and a Doctoral Supervisor of the Parallel and Distributed Processing Laboratory, National University of Defense Technology. He is currently a Researcher of the Artificial Intelligence Research Center, National Innovation Institute of Defense Technology and the Deputy Director of the Tianjin Artificial Intelligence Innovation Center. His research interests include distributed object middleware technology, software component technology, adaptive software technology, and intelligent unmanned cluster system software architecture.

YONGJUN ZHANG received the Ph.D. degree in computer science from the National University of Defense Technology, in 2000. He has participated in the National High Technology Research and Development Program of China and the National Natural Science Foundation of China, and has published more than 20 papers.



RUOXIANG LI received the B.S. degree in Internet of Things engineering from Northeastern University, Shenyang, China, in 2017. He is currently pursuing the degree with the National University of Defense Technology. His research interests focus on computer vision, visual SLAM, and event-based SLAM.

WEI QIN received the M.S. degree from the National University of Defense Technology, in 2008. He is currently an Assistant Researcher with the Artificial Intelligence Research Center, National Innovation Institute of Defense Technology and the Tianjin Artificial Intelligence Innovation Center. His research interests include information technology project evaluation, information construction project management, and science and technology development strategy.

RUIHAO LI received the B.Sc. degree in automation from the Beijing Institute of Technology, Beijing, China, in 2012, the M.Sc. degree in control science and engineering from the National University of Defense Technology, Changsha, China, in 2014, and the Ph.D. degree in robotics from the University of Essex, U.K., in 2018. He is currently an Assistant Professor with the Artificial Intelligence Research Center, National Innovation Institute of Defense Technology and the Tianjin Artificial Intelligence Innovation Center. His research interests include robotics, SLAM, deep learning, and semantic scene understanding.