2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022)
October 23-27, 2022, Kyoto, Japan

Monocular Event Visual Inertial Odometry based on Event-corner using Sliding Windows Graph-based Optimization

Weipeng Guan and Peng Lu*

Abstract—**Event cameras are biologically-inspired vision sensors that capture pixel-level illumination changes instead of the intensity image at a fixed frame rate. They offer many advantages over the standard cameras, such as high dynamic range, high temporal resolution (low latency), no motion blur, etc. Therefore, developing state estimation algorithms based on event cameras offers exciting opportunities for autonomous systems and robots. In this paper, we propose monocular visual-inertial odometry for event cameras based on event-corner feature detection and matching with well-designed feature management. More specifically, two different kinds of event representations based on time surface are designed to realize event-corner feature tracking (for front-end incremental estimation) and matching (for loop closure detection). Furthermore, the proposed event representations are used to set mask for detecting the event-corner feature based on the raw event-stream, which ensures the uniformly distributed and spatial consistency characteristic of the event-corner feature. Finally, a tightly coupled, graph-based optimization framework is designed to obtain high-accurate state estimation through fusing pre-integrated IMU measurements and event-corner observations. We validate quantitatively the performance of our system on different resolution event cameras: DAVIS240C (240*180, public dataset, achieve state-of-the-art), DAVIS346 (346*240, real-test), DVXplorer (640*480 real-test). Furthermore, we demonstrate qualitatively the accuracy, robustness, loop closure, and re-localization performance of our framework on different large-scale datasets, and an autonomous quadrotor flight using our Event Visual-inertial Odometry (EVIO) framework. Videos of all the evaluations are presented on the project website.**

I. INTRODUCTION

State estimation is the most fundamental topic in the field of robotics, such as Simultaneous Localization and Mapping (SLAM) / Visual Odometry (VO), navigation, path planning, drone control, autonomous driving, etc. Recently, the approaches that assist the visual sensor (camera) with inertial sensor (inertial measurement unit, IMU), also known as Visual Inertial Odometry (VIO), have gained significant research interest and progress [1] [2]. However, due to the inherent limitations of the standard cameras, such as motion blur and low dynamic range, the VIO systems based on standard cameras might easily fail during the high-speed motions or in the high-dynamic-range (HDR) scenarios.

Event cameras, also called Dynamic Vision Sensors (DVS), offer a huge potential to overcome the aforementioned issues due to their extremely high temporal resolution and HDR property [3]. Event cameras constitute a new paradigm shift that operates asynchronously, transmitting only the information conveyed by brightness changes in individual pixel. Event cameras have many advantages over the standard cameras: (*i*) Extremely high temporal resolution and negligible latency on the order of a few milliseconds; (*ii*) HDR (140 dB for the event cameras, while 60 dB for standard cameras). (*iii*) Since all pixels only capture the brightness change asynchronously and independently, event cameras do not suffer from motion blur [3], and also remove the inherent redundancy of standard image. These properties allow the event cameras to confer robustness to vision-based localization in challenging scenarios. However, adopting the event camera into the SLAM/VO is a very challenging task, this is caused by the fact that the event streams are composed of asynchronous events which are fundamentally different from the synchronous intensity images. Thus, the traditional computer vision algorithms cannot be directly applied. Some works have addressed this challenge by reconstructing the intensity frame from the event data [4] [5], aggregating a group of events within a short period of time into event frame [6] [7] [8], or combining event cameras with additional sensors (e.g. depth sensors [9], standard cameras [10]). However, those would introduce bottlenecks and lose the natural advantages of the event camera.

In this paper, different from existing methods that reconstruct or aggregate intensity-image from the event data, we directly use the asynchronous raw events for feature detection (named as event-corner feature). To this end, we propose two kinds of event representations based on the time surface (TS) for assisting uniformly distributed feature detection, front-end feature tracking, and loop closure matching. We investigate the proposed event-corner feature tracker and matcher into monocular event visual-inertial odometry (EVIO) based on sliding windows graph-based optimization to estimate arbitrary 6 DoF (degree-of-freedom) motion. Our contributions are summarized as follows:

1) Instead of extracting the features on the intensity-image generated from events, we propose a steady and uniformly distributed event-corner feature detector that directly works on the raw events.
2) We design two different event representations, the TS with the polarity and the normalized TS without polarity, to perform robust feature tracking and loop closure matching using the previously detected event-corner features.
3) The event-corner features tracker and matcher are integrated into a keyframe-based visual-inertial system that tightly fuses the event-corner features with IMU data to update the state. Furthermore, our EVIO framework can

The authors are with Faculty of Engineering, Department of Mechanical Engineering, The University of Hong Kong, Hong Kong SAR, China. (*Corresponding author: lupeng@hku.hk). The supplemental dataset and video demos are available in: https://b23.tv/75eT25Z

bootstrap from unknown initial states, and also ensure global consistency thanks to the loop closure.

4) We evaluate the proposed method on different resolution event cameras: DAVIS240c (240*180, publicly available dataset, achieve state-of-the-art), DAVIS346 (346*240, real-test), DVXplorer (640*480, real-test), and DVXplorer-Mini (640*480, quadrotor flighting). It is our knowledge that this is the first EVIO framework can realize real-time performance in high resolution event cameras with the loop closure to reduce the drift.

The remainder of the paper is organized as follows: Section II introduces the related works in event-based SLAM, VO, and VIO. Section III introduces the principle of our methods. Section IV and V present the experiments and results. Finally, conclusion is given in Section VI.

II. Related Works

A. Event-based SLAM/VO

Event-based ego-motion estimation and perception have gained increasing interest for the task of SLAM/VO in challenging scenarios where the performance of traditional cameras is compromised such as in HDR scenarios or aggressive motions. Weikersdorfer et al. [11] proposed the first event-based 2D SLAM system, which tracks a ground robot pose while reconstructing the 2D ceiling map with an upward-looking event camera. Ref. [10] claimed that they developed the first event-based VO to track the 6-DoF motion. However, this method is not purely event-based, since the features are first detected in the grayscale frames, and then tracked asynchronously using event stream. The first purely event-based 6-DoF VO was presented in [4], which performed real-time event-based SLAM through three decoupled probabilistic filters that jointly estimate the 6-DoF camera pose, 3D map of the scene, and image intensity. However, it is computationally expensive and requires GPU to achieve real-time performance. EVO [8] proposed to solve the SLAM problem without recovering image intensity, thus reducing computational complexity, and it can run in real-time on a standard CPU. It performs a geometric approach which combines a tracking approach based on image-to-model alignment and semi-dense 3D reconstruction algorithm [12] in parallel. However, the algorithm is needed to run in the scene that is planar to the sensor, up to several seconds, for bootstrapping the system. ESVO [13] is the first stereo event-based VO method, which follows a parallel tracking-and-mapping scheme to estimate the ego-motion and the semi-dense 3D map of the scene. However, it barely operates real-time in DAVIS346 (346*240) and is limited by rigorous and unreliable initialization. Furthermore, the ESVO needs to perform re-initialization in the case of too few events boosting. Ref. [14] proposed stereo VO for event cameras based on features. The pose estimation is done by re-projection error minimization, while the features are stereo and temporally matched through the consecutive left and right event TS. It solves the problems of ESVO mentioned above. However, it still cannot operate in real-time for high-resolution event cameras (640*480).

B. Event-based Visual-Inertial Odometry (EVIO)

Most of the event cameras, e.g. DAVIS or DVXplorer, have the IMU readily integrated. The first EVIO method is proposed in Ref. [15] which fuses a purely event-based tracking algorithm with IMU through Extended Kalman Filter. A similar method was proposed in Ref. [6], which generated motion-compensated edge images aggregated within in temporal neighborhood events. The features are detected through fast corner detector [16] and tracked through pyramidal Lucas Kanade (LK) in the motion-compensated event image, and then combined with IMU measurement using graph-based optimization. The authors also extended this method to leverage the complementary advantages of both standard and event cameras in Ultimate-SLAM [7] to fuse events image frame, standard frames, and IMU. Both Ref. [6] and Ref. [7] aggregate the event data into an image frame and then adapt the conventional corner detection algorithms, such as FAST corners [16] or Shi-Tomasi [17] score for the feature detection. Ref. [18] proposed to fuse events and IMU measurement into a continuous-time framework. However, their approach cannot achieve real-time because of the expensive optimization required to update the spline parameters upon receiving every event [6]. IDOL [19] investigates line-feature into VIO framework through directly using asynchronous raw events without using any frame-like accumulation. However, it also doesn't have real-time capabilities even in low-resolution event cameras (240*180). All of these EVIO works are only evaluated on low-resolution DAVIS240C (240*180). Besides, these works lack the loop closure function, which might cause a large amount of drift for long-term motions.

III. Methodology

Fig. 1 gives an overview of our EVIO modules involved and their interactions. Our EVIO framework is composed of two sections: (i) The front-end takes the raw event stream as input and extracts the event-corner features directly using the raw events based on the Surface of Active Events (SAE) [20]. It establishes feature tracking and recovers the inverse depth of each tracked event-corner for generating 3D event-corner features as landmarks. Furthermore, more event-corner features are extracted and passed to the back-end for loop detection. Two kinds of event representations: the TS with polarity and normalized TS without polarity are designed for assisting uniformly distributed event-corner feature detection, feature tracking, and descriptor generation. (ii) The back-end tightly fuses the event-corner landmarks and the IMU measurements to estimate the 6 DoF state of the system while the loop closure is used to eliminate the accumulated drifts.

A. Proposed Two Types of Event Representations

An event is triggered only when the intensity of an individual pixel varies exceeds a specific threshold $T_{threshold}$,

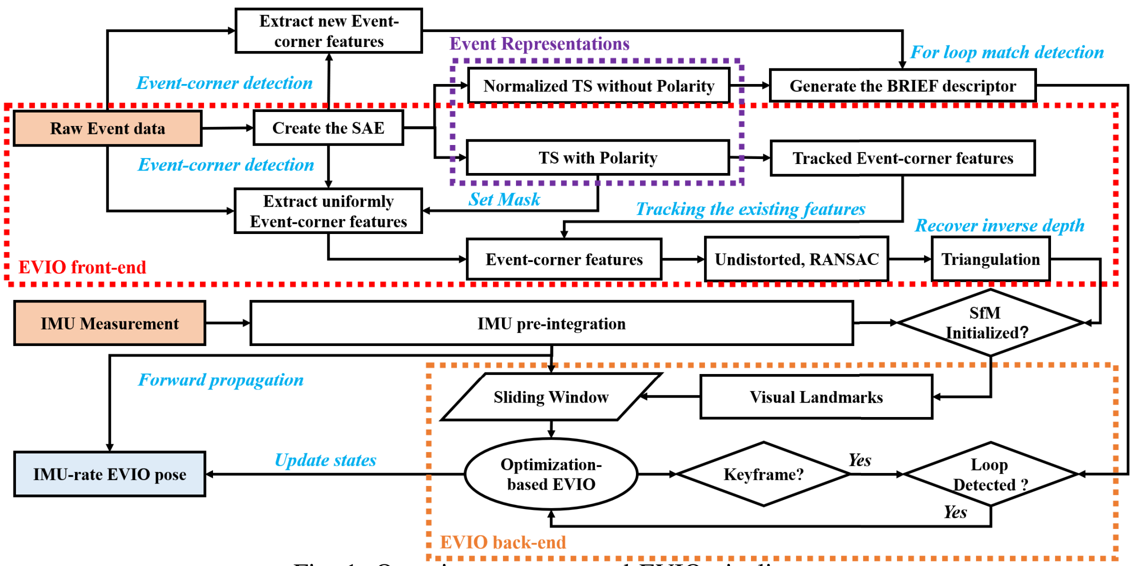


Fig. 1: Overview our proposed EVIO pipeline

which can be represented as the spatio-temporal coordinates of the intensity change and its sign:

$$e = \{t, x, y, p\} \Leftrightarrow I(x,y,t+\triangle t) - I(x,y,t) = p \cdot T_{threshold} \quad (1)$$

where t is the timestamp that the intensity of a pixel $I(x,y)$ changes, and p is the polarity which indicates the direction of the intensity change. Our method operates on individual raw events and is aided through two event representations, called TS with polarity and normalized TS without polarity, which are generated from SAE. The SAE S is defined as $S = t_{last}(x,y)$, which is a simple 2D map containing the timestamp $t_{last}(x,y)$ of the last event that occurred at each pixel, for positive and negative polarity, respectively. These two SAE record the triggered events in the stream, which are used for event-corner detection (discussed in subsection III.B). To construct the event-corner feature tracker and descriptor, we use the timestamp value from SAE S to construct two alternative event maps at time t, where $t \geq t_{last}(x,y)$, named as TS with polarity $T_p(x,y,t)$ and normalized TS without polarity $T_{np}(x,y,t)$, which are designed as:

$$T_p(x,y,t) = p \cdot \exp(-\frac{t-S}{\tau}) \quad (2)$$

$$T_{np}(x,y,t) = (\frac{255.0}{max(T^{'}) - min(T^{'})}) \cdot (T^{'} - min(T^{'})) \quad (3)$$

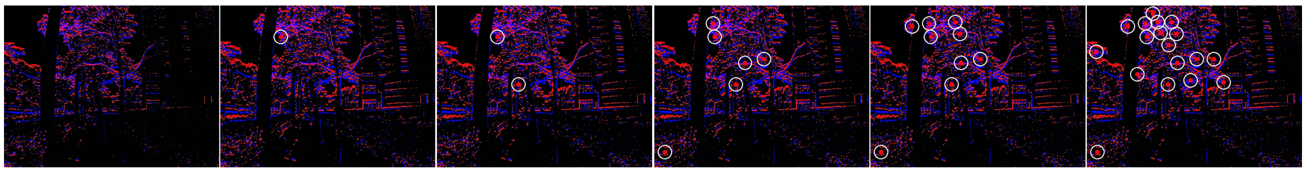
where, $T^{'} = \exp(-\frac{t-S}{\tau})$, and τ presents a constant decay rate (20-30 ms in our experiments). The polarity p is the sign of the brightness change.

The TS with polarity $T_p(x,y,t)$ (shown in Fig.2(b) and the middle of Fig.3(a)) represents the recent history of moving edges with direction. The polarity p is useful for feature tracking, since it records the direction of the event change, which would easily respond to edges in the scene in presence of different relative motions (optical flow). As shown in the middle of Fig. 3(a), the value of $T_p(x,y,t)$ changes from "*white to black*" and "*black to white*" caused by opposite relative motions. However, the polarity might cause ambiguity when the sensor moves in different directions of the same scene, the pixel values of the same edges might change differently according to the polarity. This will influence image matching seriously. Therefore, for the loop detection, we used the second event representation, the TS without polarity, and further refined it to normalized TS without polarity $T_{np}(x,y,t)$ (as shown in the right of Fig. 3(a)), which records the normalized temporal-spatial constraints as scene outline and ensures the spatial consistency. This kind of event representation is representative of the scene structure as it emphasizes the edges, thus containing relevant information for discriminative event-corner descriptors.

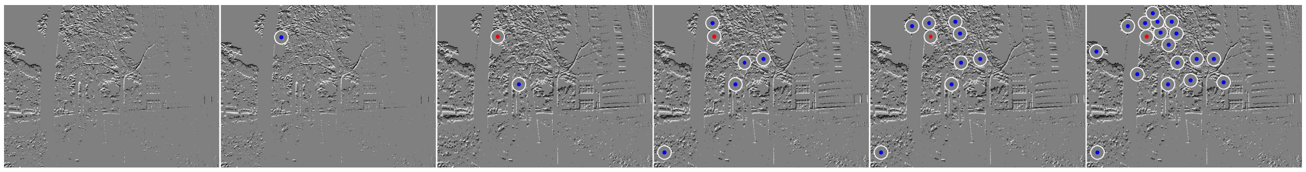
B. Event-corner Feature Detection and Tracking based on the Time Surface with Polarity

For new event-stream coming, firstly ,the existing event-corner features are tracked by the LK optical flow [21] on the TS with polarity $T_p(x,y,t)$. The features that are not successfully tracked in the current timestamp would be discarded. After that, new event-corner would be detected from the latest raw event stream, whenever the number of the tracked features falls below a certain threshold (150-200 in our experiment). Modified from the publicly available implementation of the Arc* algorithm [22] for event-based corner detection, we extract the event-corners on the raw individual event by leveraging the SAE rather than adopting the conventional corner detection algorithms.

To classify a new event as an event-corner, we need to inspect previously triggered events in the stream, especially, the events in the adjacent pixels. Since exploring all the previous events would be impractical, the SAE is used to summarize and update the event stream at any given instant. The Arc* algorithm maintains two circular sets of events around the new arrival event and detects as corners whenever the continuous arc or its complementary arc in these two circulars of SAE is within a certain range. The detected event-corner would be further selected by setting the TS with polarity $T_p(x,y,t)$ as the mask. To enforce the uniform distribution, a minimum distance (10-20 pixels

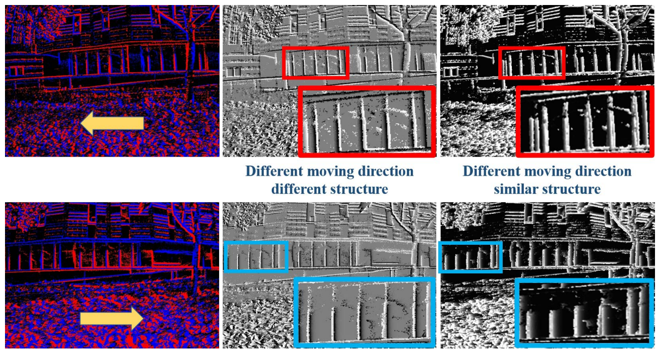


(a)

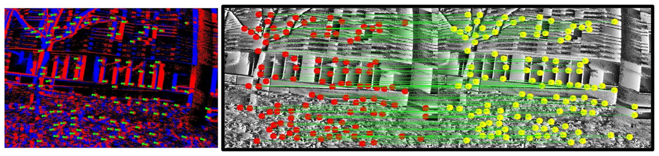


(b)

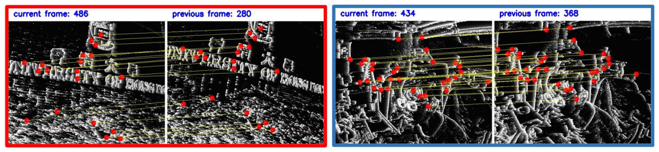
Fig. 2: (a) Detect event-corner from the raw event; (b) Select the features using the Time Surface with polarity as mask



(a)



(b)



(c)

Fig. 3: (a) The raw event stream (left), the Time Surface with polarity (middle), and the normalized Time Surface without polarity (right); (b) Event-corner tracking on Time Surface with polarity; (c) Loop detection using the event-corner features and the normalized Time Surface without polarity

for different resolution event camera) is set between two neighboring event-corner features. Meanwhile, we maintain the event-corners, where the pixel value of the TS with polarity $T_p(x,y,t)$ is not equal to 128.0, to emphasize the detected event-corner features located in the strong edges rather than the too many noisy features in low texture areas. This process is visualized in Fig. 2.

Furthermore, all the event-corner features are first undistorted based on the camera distortion model, and then projected to a normalized camera coordinate system. To remove outliers, we also use the Random Sample Consensus (RANSAC) to further filter the outliers. After that, we recover the inverse depth of the features that are successfully tracked between two consecutive timestamps through triangulation. The landmark whose 3D position has been successfully calculated would be fed to the sliding window for the pose graph optimization. It is worth mentioning that, although we try to maintain the number of event-corners within a certain threshold, the number of event-corners used for pose graph optimization still depends on the relative motion and texture of the scene.

C. Pose Graph Optimization

The full state vector in the sliding window is defined as:

$$\chi = [\chi_b, T_c^b, \lambda_c] \tag{4}$$

where $T_c^b = [R_c^b, t_c^b]$ is the extrinsic transformation from the camera frame c to the body (IMU) frame b; $\lambda_c = [\lambda_0, ..., \lambda_m]$ is the inverse depth of the m^{th} event-corner features in the sliding windows; $\chi_b = [X_1, ..., X_K]$ is the optimization variables in the sliding windows, which comprises the state of the IMU, with K ($K = 10$ in our experiments), the total number of keyframes in the sliding windows. The system state X_k at k^{th} keyframe is given by the position $p_{b_k}^w$, orientation quaternion $q_{b_k}^w$, and the velocity $v_{b_k}^w$ of the IMU in the world frame, and the accelerometer bias b_{a_k} and gyroscope bias b_{g_k} as follows:

$$X_k = [p_{b_k}^w, q_{b_k}^w, v_{b_k}^w, b_{a_k}, b_{g_k}] \tag{5}$$

Joint nonlinear optimization which is solved for the maximum a posteriori estimation of χ, the cost function can be written as:

$$J(\chi) = ||e_p||^2_{W_p} + \sum_{k=0}^{K-1} ||e_i^k||^2_{W_i^k} + \sum_{k=0}^{K-1} \sum_{l \in \zeta} ||e_c^{k,l}||^2_{W_c^k} \tag{6}$$

The Ceres solver is used for solving Eq.(6), which contains the marginalization residuals e_p with weight W_p, the IMU pre-integration residuals e_i^k with weight W_i^k, and the $e_c^{k,l}$ is the event measurement residual from the re-projection function Eq.(7), with weight W_c^k. While ζ is the set of event-corner features that have been observed/tracked at least twice in the current sliding window. Considering the l^{th} feature that is first observed in the i^{th} keyframe, the residual for its observation in the k^{th} keyframe is defined as:

$$e_c^{k,l} = \begin{bmatrix} u_l^k \\ v_l^k \end{bmatrix} - \pi_c \cdot (T_c^b)^{-1} \cdot T_w^{b_k} \cdot T_{b_i}^w \cdot T_c^b \cdot \pi_c^{-1}(\frac{1}{\lambda_l}, \begin{bmatrix} u_l^i \\ v_l^i \end{bmatrix}) \tag{7}$$

where, $\begin{bmatrix} u_l^i \\ v_l^i \end{bmatrix}$ is the first observation of the l^{th} feature in the i^{th} keyframe. $\begin{bmatrix} u_l^k \\ v_l^k \end{bmatrix}$ is the observation of the same feature in the k^{th} keyframe, π_c and π_c^{-1} are the projection and back-projection function of the event camera, respectively, which includes the intrinsic parameters for the transform between the 2D pixel coordinates and normalized camera coordinate. $T_{b_i}^w$ indicates the movement of the body frame related to the world frame in timestamp i, $T_w^{b_k}$ is the transpose of the pose of the body in the world frame in the k^{th} keyframe.

D. Loop Detection based on the Normalized Time Surface without Polarity

To eliminate the accumulated drifts and ensure global consistency for long-term motion, in addition to the event-corner features that are used for EVIO front-end, extra event-corners are detected, and then described by the BRIEF descriptor [23], and further feed to the back-end (as depicted in Fig.1). These additional event-corner features are used to achieve a better recall rate on loop detection. Thanks to our designed normalized TS without polarity, $T_{np}(x,y,t)$, which would be triggered in the scene that has strong edges, it can help for the place recognition and ensure global consistency. The correspondences are found by the BRIEF descriptor matching through identifying by Hamming distance. When the number of the correspondences of the event-descriptors is greater than a certain threshold (16-25 in our experiments), the loop closure is detected (as shown in Fig. 3(c)). We adopt the two-step geometric outlier rejection for wrong BRIEF descriptor matching, DBoW2 for loop recognition, and re-localization scheme from VINS-Mono [2] in our implementation. After detecting the loop, the connection residual of the previous keyframe and the current keyframe are integrated into the pose graph optimization.

E. Additional Implementation Details

Initialization: Adopted from [2] and [24], the initialization procedure of our EVIO starts with a vision-only structure from motion (SfM) to build the up-to-scale structure of camera pose and event-corner feature positions. Through loosely aligning the SfM with the pre-integrated IMU Measurements, it can bootstrap the system from unknown initial states, instead of assuming the sensor remains static during the initialization phase [7] [6] or assuming the local scene is planar to the sensor [8].

Keyframe Selection: A new keyframe is selected by two criteria: (i) When the average parallax of the tracked features, between two consecutive timestamps, is beyond a threshold (10 is set in our experiment). (ii) When the number of successfully tracked features from the last timestamp falls below a certain threshold (30 is set in our experiment).

Still State: Since the event cameras output very little events (only noise) when the sensor is still. We would restart the EVIO estimator whenever the number of event falls below a threshold (1000 is set in our experiment) to avoid divergence.

Low Latency: To achieve low latency, we directly forward propagate (loosely-coupled) the latest EVIO estimation with the IMU measurements to achieve IMU-rate EVIO outputs which can be up to 1000 Hz.

IV. Evaluation

In this section, we assess the accuracy of our EVIO framework both quantitatively and qualitatively on different challenging sequences with different resolution event cameras (Table I). We implemented our EVIO method with C++ in Ubuntu 20.04 and ROS Noetic. All the sequences are evaluated in real-time using a laptop with Intel Core i7-11800H and are recorded in videos (shown on our project website).

In subsection IV.A, we compare the accuracy of our EVIO framework with other current EVIO works in a publicly available Event Camera Dataset [25] which is acquired by the DAVIS240C (240*180, event-sensor, image-sensor, IMU sensor), it contains extremely fast 6-Dof motion and scenes with HDR.

TABLE I: Summary of the data sequences in Section IV

rosbag	Section IV.A Public Dataset [25]	Section IV.B Real-test	Section IV.C Real-test
Sensor	Davis240C	DAVIS346 DVXplorer	DVXplorer
Topic	/optitrack/davis	/dvs_vicon/gt_pose	*No Ground Truth*
	/dvs/events	/davis346/events /dvxplorer/events	/dvs/events
	/dvs/imu	/davis346/imu /dvxplorer/imu	/dvs/imu
	/dvs/image_raw	/davis346/image_raw	*No Image*
Event Stream Rate	30HZ	DAVIS346: 60 HZ DVXplorer: 50 HZ	50HZ
Average Duration	59.8s	156.3 s	171.6s
Data size	7×10^5	DAVIS346: 6×10^5 *DVXplorer: 2×10^6	2×10^6
Resolution	240*180	DAVIS346:346*240 DVXplorer:640*480	640*480
Description	indoor aggressive HDR scenarios under optitrack	indoor aggressive HDR scenarios under vicon	indoor&outdoor HDR scenarios long-term

* *Datasize : The number of events per stream, e.g.* 2×10^6 *indicates the average number of events for the sequences is* $2 \times 10^6 \times 50HZ \times 156.3s = 1.6 \times 10^{10}$.

To further explore the performance of our EVIO framework in high-resolution event cameras, in subsection IV.B and IV.C, we use the DAVIS346 (346*240, event-sensor, image-sensor, IMU sensor) and DVXplorer (640*480, event-sensor, IMU sensor) for data collection. It is worth mentioning that, our EVIO only uses the event stream. The image-frame output of the DAVIS is only used for illustration purposes or image-based VIO/VO comparison. The event camera and IMU calibration (including the intrinsic and extrinsic parameters, and the time offset of the camera-imu) are estimated using Kalibr [26] as well as the DV module [1].

[1]https://gitlab.com/inivation/dv/dv-imu-cam-calibration

A motion capture system (VICON) is used to obtain the pose ground truth. Since the active infra-red (IR) emitters on the VICON cameras would influence the event camera greatly, we adopt the IR filter lens to remove the IR influence. For the convenience of the research community, we also release these data sequences with rosbag on our project website.

A. Comparison with Other EVIO Works in Aggressive Motions

The estimated and ground-truth trajectories were aligned with a 6-DOF transformation (in SE3), using 5 seconds [0-5s] of the resulting trajectory. We computed the mean position error (Euclidean distance in meter) as percentages of the total traveled distance of the ground truth, which are calculated by the publicly available RPG Trajectory Evaluation tool [27]. Due to the lack of publicly available EVIO open source code, we directly refer to the raw result from [15] [6] [7], which is also aligned with SE3 using 5 seconds. Table II shows the remarkable accuracy of our method compared to the state-of-the-art EVIO works. It's worth mentioning that, although IDOL [19] also provides their results in this Event Camera Dataset [25], they just run 0-40 s of the dataset to avoid the aggressive motion. Fig.4 presents the estimated trajectories against the ground truth for the sequence *dynamic_translation* and *dynamic_6dof*, which are generated by the publicly available tool EVO [28], and also visualizes the relative translation and yaw error by averaging the drift over different segments of the trajectory. We find that our EVIO achieves fairly good results.

TABLE II: Accuracy of our method compared with the State-of-the-art EVIO Methods

Sequence	Ours	Ref. [15]	Ref. [6]	Ref. [7] (E+I)
boxes_translation	**0.34**	2.69	0.57	0.76
hdr_boxes	**0.40**	1.23	0.92	0.67
boxes_6dof	0.61	3.61	0.69	**0.44**
dynamic_translation	**0.26**	1.90	0.47	0.59
dynamic_6dof	0.43	4.07	0.54	**0.38**
poster_translation	0.40	0.94	0.89	**0.15**
hdr_poster	**0.40**	2.63	0.59	0.49
poster_6dof	**0.26**	3.56	0.82	0.30
Average	**0.39**	2.58	0.69	0.47

Unit:%/m, 0.39 means the average error would be 0.39m for 100m motion.

B. Evaluation with High-resolution Event Cameras in High-Dynamic-Range Scenarios

For further demonstrating the robustness, accuracy, and real-time capability, we also evaluate our EVIO using high-resolution event cameras (DAVIS346 (346*240) and DVXplorer (640*480)) with the ground truth from VICON. The DAVIS346 and DVXplorer are attached together (shown in Fig.5(a)) for facilitating comparison. All the sequences are recorded in HDR scenarios with very low illumination or strong illumination changes through switching the strobe flash on and off, while the 11st sequence (*vicon_aggressive_hdr*) is characterized by aggressive motion. Without loss of generality, we also used the raw image from DAVIS346 to run the VINS-MONO [2] and the VO version of ORB-SLAM3 [1] (the VIO version failed or cannot initialize in all the sequences), as image-based VIO/VO for comparison. The results are shown in Table III. It is worth

TABLE III: Accuracy of our EVIO compared with VINS-MONO and ORB-SLAM3

Sequence	VINS-MONO [2] DAVIS346	ORB-SLAM3 [1] DAVIS346	Our EVIO DAVIS346	Our EVIO DVXplorer
vicon_hdr1	0.96	0.32	0.59	**0.30**
vicon_hdr2	1.60	0.75	0.74	**0.37**
vicon_hdr3	2.28	**0.60**	0.72	0.69
vicon_hdr4	1.40	0.70	0.37	**0.26**
vicon_darktolight1	**0.51**	0.75	0.81	0.80
vicon_darktolight2	0.98	0.76	**0.42**	0.57
vicon_lighttodark1	0.55	0.41	**0.29**	0.81
vicon_lighttodark2	**0.55**	0.58	0.79	0.75
vicon_dark1	0.88	*failed*	1.02	**0.35**
vicon_dark2	0.52	0.60	0.49	**0.41**
vicon_aggressive_hdr	*failed*	*failed*	0.66	**0.65**
Average	1.02	0.61	0.63	**0.54**

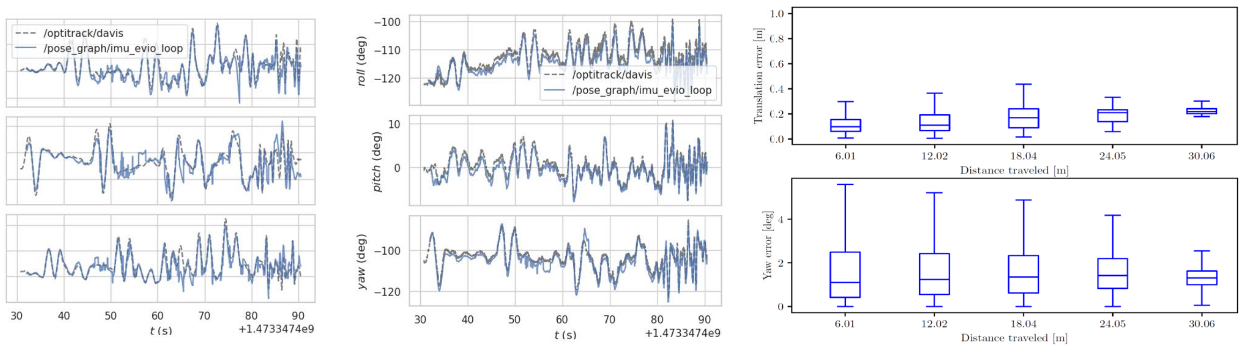
Unit:%/m, 0.54 means the average error would be 0.54m for 100m motion.

mentioning that, for the results of *vicon_aggressive_hdr*, our EVIO produces reliable and accurate pose tracking even when the image-based VIO and VO fails. Although the VO version of ORB-SLAM3 performs comparably to our EVIO in DAVIS346, it would track failures and lose tracking frames during the aggressive motion or too dark scenarios which would affect the generation of the descriptor seriously. Thanks to the re-localization scheme, the ORB-SLAM3-VO can handle the tracking failed issue after re-detecting the ORB descriptor in good illumination condition, but this would cause interruption during the estimation. While our EVIO can provide continuous and smooth state estimation. In addition, one of the main challenging aspects of this dataset is that most of the scenarios are dark (please see the supplemental video), this would introduce many noise events. However, our EVIO still can obtain satisfactory results.

Another challenging aspect is the heavy event load of this dataset compared with the ones in subsection IV.A. We found that the amount of event data from the high-resolution event camera (such as DVXplorer) is as large as the order of 10^6 (for each event stream, shown in Table I). The public event camera dataset [25], which only works on 30 HZ event stream inputs. However, to further evaluate the performance of our EVIO in the large throughput event load, we modified the driver code for DAVIS346 and DVXplorer with a higher stream rate and not limited maximum events for each stream, which can also ensure a steady frequency of the event stream. The large throughput of our well-designed feature management ensures our system can handle 60 HZ data input from DAVIS346 with 30HZ front-end output, and 50 HZ data input from DVXplorer with 25HZ front-end output. It can ensure the real-time ability for heavy event load in the high-resolution event camera rather than slowing down the rosbag reproduction just like [13] [14]. The running time of each module in our EVIO can be seen in Table IV

C. Real-test for Outdoor Scenarios

The workspace of the sequences in subsection IV.A and IV.B is relatively small, it is difficult to distinguish between drift and failure from error value alone. Therefore, in this


(a) dynamic_translation

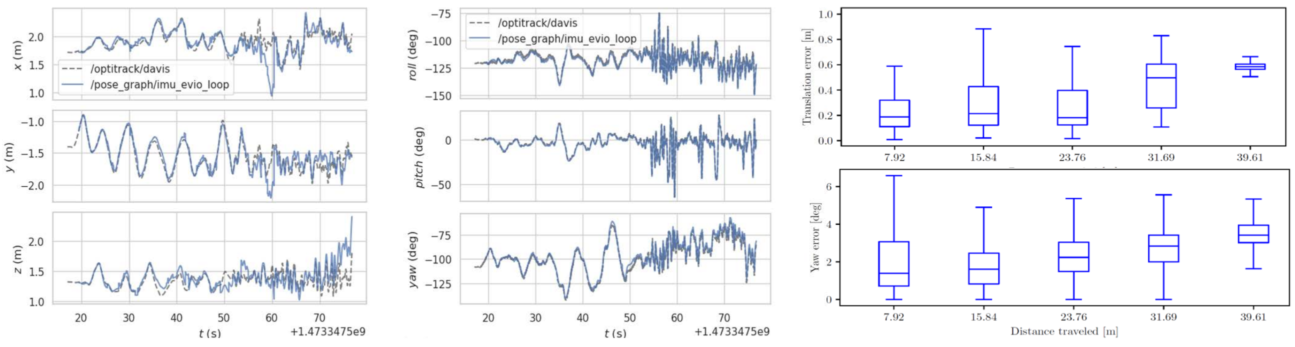

(b) dynamic_6dof

Fig. 4: Comparison of translation and rotation estimates of our proposed EVIO against ground truth; The relative errors of the translation and yaw angle

TABLE IV: Running Time of our EVIO in different resolution event cameras (ms)

Modules	DAVIS240c	DAVIS346	DVXplorer
Creation of Event Representations	0.37	0.98	3.65
Event-corner Feature Detection	0.49	0.40	1.60
Event-corner Feature Tracking	0.86	0.71	1.16
The Whole Front-end Process	3.98	3.81	12.38
Event-corner Loop Matching	27.73	19.56	67.95

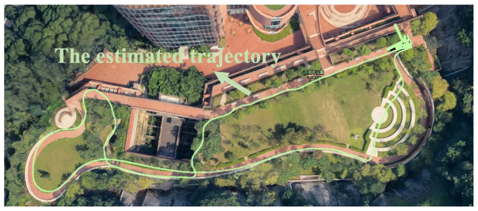
section, several sequences were further recorded outdoors, in HKU campus, features aggressive motion, long-term movement, strong sunlight, or indoor-outdoor conversion. Additionally, there are several pedestrians in the scene generating outlier events. Since the motion capture system is not available outdoors, we just evaluate the qualitative performance, and we also returned to the same location after a large loop to evaluate the loop closure. As can be seen from Fig. 5(b), the estimated trajectory is aligned and almost coincide with the Google map. It is worth mentioning that our EVIO can effectively detect feature points at a distance of up to 20 meters (shown in Fig.3(b)). More evaluations for the outdoor environment, robust feature tracking, and the loop closure can be seen in our demonstrations[2].

V. Quadrotor Experiment using DVXplorer-Mini

In this section, we demonstrate the quadrotor flighting based on our proposed EVIO using DVXplorer-Mini (640*480). We build our quadrotor (Fig. 6(a)) from selected off-the-shelf components and custom 3D printed items. Our quadrotor relies on a QAV380 frame with T-MOTOR F60 KV2550. The electronic parts of our quadrotor comprise a Pixracer (FMUv4) autopilot with Up Xtreme i7 8665ue

[2]https://b23.tv/0oxnv9U

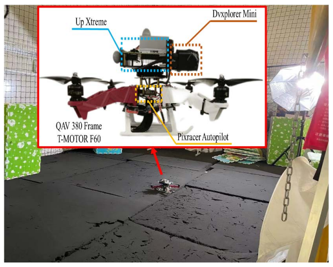
(a)



(b)

Fig. 5: (a)The setup for section IV.B; (b) Estimated trajectory in outdoor environment aligned with Google map

computer, which runs Ubuntu 20.04 and ROS Noetic. The DVXplorer-Mini is mounted on the front of the quadrotor, looking forwards, which is connected to the Up Xtreme via USB 3.1 A to C cable and transmits events and inertial measurements for our EVIO state estimation. The quadrotor is commanded to hover with slight jitter, our EVIO estimated result against with the VICON ground truth can be seen in Fig. 6(b).



(a)

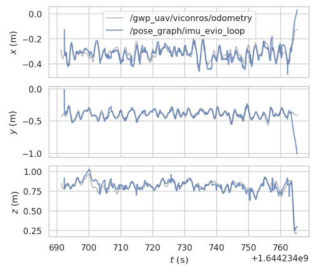

(b)

Fig. 6: (a) Our quadrotor platform and VICON room; (b) The estimated trajectory and its comparison against VICON

VI. CONCLUSIONS

In this paper, we have developed a low-latency, real-time, monocular event-based visual-inertial odometry framework to provide accurate metric tracking of the 6 DoF pose. The method is based on our designed steady and uniformly distributed event-corner feature detector, which is done with raw individual events but consecutively tracked in TS with polarity, and spatially matched in normalized TS without polarity. Our EVIO can estimate up to 1000 HZ poses while recovering a sparse 3D map of the environment. The performance of the proposed is quantitatively and qualitatively evaluated in different resolution event cameras: DAVIS240C (240*180, publicly dataset), DAVIS346 (346*240, real-test), DVXplorer (640*480, real-test), and DVXplorer-Mini (640*480, quadrotor flighting). Our method achieve fairly good performance compared with state-of-the-art EVIO works, VINS-MONO, and ORB-SLAM3. However, the limitations of our work might be that the event cameras tend to only trigger events over edge-like features, low texture areas generate very few events. Therefore, our EVIO might suffer some problems in less low texture scenarios. In our future work, we might explore the line-based features for EVIO. Besides, multi-sensors, including LiDAR, visible light positioning [29], or GPS, might be fused together to achieve more robust state estimation and exploit the complementary advantage of different sensors with event cameras.

ACKNOWLEDGMENT

This research is supported by General Research Fund under Grant No. 17204222, the Seed Funding for Strategic Interdisciplinary Research Scheme and Platform Technology Fund.

We also thank Cai Yixi for the advice on code, Huang Yilun, Lu Minghao for data collection, Peng Rui for quadrotor setting, and Zhang Tiening for 3D printer.

REFERENCES

[1] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, “Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam,” *IEEE Transactions on Robotics*, 2021.

[2] T. Qin, P. Li, and S. Shen, “Vins-mono: A robust and versatile monocular visual-inertial state estimator,” *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[3] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, *et al.*, “Event-based vision: A survey,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 1, pp. 154–180, 2020.

[4] H. Kim, S. Leutenegger, and A. J. Davison, “Real-time 3d reconstruction and 6-dof tracking with an event camera,” in *European Conference on Computer Vision*. Springer, 2016, pp. 349–364.

[5] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza, “High speed and high dynamic range video with an event camera,” *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[6] H. Rebecq, T. Horstschaefer, and D. Scaramuzza, “Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization,” 2017.

[7] 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 Robotics and Automation Letters*, vol. 3, no. 2, pp. 994–1001, 2018.

[8] 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 Robotics and Automation Letters*, vol. 2, no. 2, pp. 593–600, 2016.

[9] D. Weikersdorfer, D. B. Adrian, D. Cremers, and J. Conradt, “Event-based 3d slam with a depth-augmented dynamic vision sensor,” in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 359–364.

[10] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza, “Low-latency visual odometry using event-based feature tracks,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 16–23.

[11] D. Weikersdorfer, R. Hoffmann, and J. Conradt, “Simultaneous localization and mapping for event-based vision systems,” in *International Conference on Computer Vision Systems*. Springer, 2013, pp. 133–142.

[12] H. Rebecq, G. Gallego, E. Mueggler, and D. Scaramuzza, “Emvs: Event-based multi-view stereo—3d reconstruction with an event camera in real-time,” *International Journal of Computer Vision*, vol. 126, no. 12, pp. 1394–1414, 2018.

[13] Y. Zhou, G. Gallego, and S. Shen, “Event-based stereo visual odometry,” *IEEE Transactions on Robotics*, 2021.

[14] A. Hadviger, I. Cvišić, I. Marković, S. Vražić, and I. Petrović, “Feature-based event stereo visual odometry,” in *2021 European Conference on Mobile Robots (ECMR)*. IEEE, 2021, pp. 1–6.

[15] A. Zihao Zhu, N. Atanasov, and K. Daniilidis, “Event-based visual inertial odometry,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5391–5399.

[16] E. Rosten, R. Porter, and T. Drummond, “Faster and better: A machine learning approach to corner detection,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 1, pp. 105–119, 2008.

[17] J. Shi *et al.*, “Good features to track,” in *1994 Proceedings of IEEE conference on computer vision and pattern recognition*. IEEE, 1994, pp. 593–600.

[18] E. Mueggler, G. Gallego, H. Rebecq, and D. Scaramuzza, “Continuous-time visual-inertial odometry for event cameras,” *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1425–1440, 2018.

[19] C. Le Gentil, F. Tschopp, I. Alzugaray, T. Vidal-Calleja, R. Siegwart, and J. Nieto, “Idol: A framework for imu-dvs odometry using lines,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5863–5870.

[20] E. Mueggler, C. Bartolozzi, and D. Scaramuzza, “Fast event-based corner detection,” 2017.

[21] B. D. Lucas, T. Kanade, *et al.*, “An iterative image registration technique with an application to stereo vision.” Vancouver, British Columbia, 1981.

[22] I. Alzugaray and M. Chli, “Asynchronous corner detection and tracking for event cameras in real time,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3177–3184, 2018.

[23] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “Brief: Binary robust independent elementary features,” in *European conference on computer vision*. Springer, 2010, pp. 778–792.

[24] T. Qin and S. Shen, “Robust initialization of monocular visual-inertial estimation on aerial robots,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4225–4232.

[25] 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,” *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 142–149, 2017.

[26] P. Furgale, J. Rehder, and R. Siegwart, “Unified temporal and spatial calibration for multi-sensor systems,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1280–1286.

[27] Z. Zhang and D. Scaramuzza, “A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7244–7251.

[28] M. Grupp, “evo: Python package for the evaluation of odometry and slam,” *Note: https://github. com/MichaelGrupp/evo Cited by: Table*, vol. 7, 2017.

[29] Z. Yan, W. Guan, S. Wen, L. Huang, and H. Song, “Multi-robot cooperative localization based on visible light positioning and odometer,” *IEEE Transactions on Instrumentation and Measurement*, 2021.