2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 979-8-3503-7770-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/IROS58592.2024.10802605

Monocular Event-Inertial Odometry with Adaptive decay-based Time Surface and Polarity-aware Tracking

Kai Tang, Xiaolei Lang, Yukai Ma, Yuehao Huang, Laijian Li, Yong Liu*, Jiajun Lv*

***Abstract*— Event cameras have garnered considerable attention due to their advantages over traditional cameras in low power consumption, high dynamic range, and no motion blur. This paper proposes a monocular event-inertial odometry incorporating an adaptive decay kernel-based time surface with polarity-aware tracking. We utilize an adaptive decay-based Time Surface to extract texture information from asynchronous events, which adapts to the dynamic characteristics of the event stream and enhances the representation of environmental textures. However, polarity-weighted time surfaces suffer from event polarity shifts during changes in motion direction. To mitigate its adverse effects on feature tracking, we optimize the feature tracking by incorporating an additional polarity-inverted time surface to enhance the robustness. Comparative analysis with visual-inertial and event-inertial odometry methods shows that our approach outperforms state-of-the-art techniques, with competitive results across various datasets.**

I. Introduction

Accurate environmental perception is essential in robotics. Traditional vision sensors, like conventional cameras, often suffer from motion blur during rapid movement and can lose image details due to their limited dynamic range, thereby undermining perception accuracy and robustness. Event cameras, equipped with Dynamic Vision Sensors (DVS), offer a promising solution to these challenges. They generate events asynchronously whenever a pixel's brightness change surpasses a preset threshold. Consequently, event cameras boast a wide dynamic range, high temporal resolution, low energy consumption, and immunity to motion blur.

It is commonly recognized that high-textured regions trigger events more frequently than low-textured ones, making it possible to extract texture details from the event stream. However, processing these asynchronous events is a challenging task. For this reason, various event representation methods have been proposed, and Gallego et al. [1] classify these event representations into Individual Events, Event Packet, Event Frame/Image, Time Surface, etc. Time Surface [2], a 2D map representation in which each pixel stores the timestamp of the corresponding event, is widely utilized in event-based SLAM or odometry methods [3], [4].

To highlight recent events over past events, time surfaces often employ an exponential decay kernel [5], but this method presents some inherent limitations. Initially, the exponential decay kernel requires a preset parameter—time constant η, which requires manual adjustment for different sequences. It lacks adaptability to the dynamics of the event stream, and this approach is prone to bold edges and noticeable trailing when the event frequency is high. Moreover, it does not filter out events that contribute less to the texture, resulting in superfluous events within time surfaces. Furthermore, the motion direction of the event camera influences the polarities. When sudden changes occur in the direction of motion, the most recent events may exhibit opposite polarities. Consequently, this can lead to fluctuations in the grayscale values of the pixels associated with these events, potentially disrupting feature tracking.

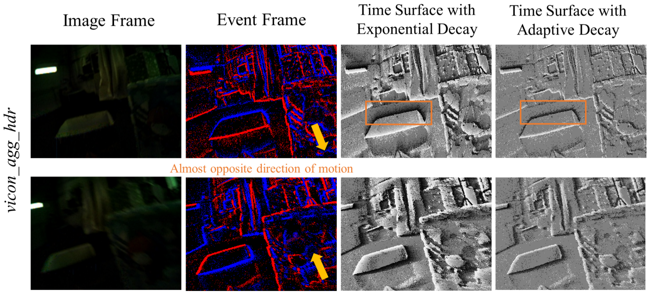
Image Frame
Event Frame
Time Surface with Exponential Decay
Time Surface with Adaptive Decay
vicon_agg_hdr
Almost opposite direction of motion

Fig. 1: Event cameras have potential advantages over traditional cameras in high dynamic range and high-speed motion scenes. Time surfaces with an exponential decay kernel (the third column) are unable to adjust their internal parameter (time constant η) in response to the event stream's dynamic characteristics, resulting in bold edges and redundant events. Conversely, adaptive decay-based time surfaces (the fourth column) offer clear details and less noise.

To tackle the aforementioned challenges, inspired by [6], we propose a monocular event-based visual inertial odometry with the adaptive decay-based time surface representation and polarity-aware tracking. The main contributions of the paper are concluded as follows:

- We propose a real-time monocular event-inertial odometry based on adaptive decay-based time surface and polarity-aware tracking within the MSCKF framework for accurate pose estimation.
- We present an adaptive decay-based time surface to accommodate the dynamic characteristics of the event stream. And we propose a polarity-aware tracking method that improves the stability of feature tracking by utilizing an additional polarity-inverted time surface.
- We evaluate the proposed method using different datasets, comparing its accuracy with visual-inertial odometry and event-inertial odometry methods, and assessing the efficiency of the adaptive decay and the

The authors are with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, China. This work was supported by NSFC 62088101 Autonomous Intelligent Unmanned Systems.

* Yong Liu and Jiajun Lv are the corresponding authors, email: yongliu@iipc.zju.edu.cn, lvjiajun314@zju.edu.cn

979-8-3503-7770-5/24/$31.00 ©2024 IEEE

proposed tracking approach. Experimental results show that our method produces competitive results.

The remainder of this paper is organized as follows: Section II provides a brief summary of recent works. Our system is described in Section III. Section IV presents detailed experimental settings and results in multiple datasets. Finally, Section V offers a succinct overview of our system and outlines future directions.

II. Related Works

A. Event-based Visual Odometry

Kueng et al. [7] developed a pioneering method for event-based visual odometry that relies on feature extraction from grayscale maps and asynchronous event-based tracking, facilitating 6DoF pose estimation and mapping. However, this method is confined to feature points detected in image frames. Another stride in event-based visual odometry by Kim et al. [8] achieved real-time event-based SLAM by integrating triplet probabilistic filters for pose estimation, scene mapping, and intensity estimation, although this approach demands GPU acceleration due to its computational intensity. EVO [9] advanced the field by presenting an image-to-model alignment-based tracking approach to capture rapid camera movements, recovering semi-dense maps in parallel processing, although a lengthy start-up phase and poor accuracy hinder it. EDS [10] improved upon this with an event generation model to track camera motion, enhancing the accuracy of visual odometry but at the expense of increased computational demand and slower optimization speeds. Lastly, an innovative 6DoF motion compensation mechanism by Huang et al. [11] allowed deblurred event frame generation synchronized to RGB images using an event generation model, addressing the modality disparities between images and event data.

Geometric approaches inadequately harness event data, whereas deep learning-based event odometers introduce an innovative processing paradigm. RAMP-VO [12] is the inaugural end-to-end framework for event- and image-based visual odometry, merging asynchronous event streams with image data through a parallel encoding scheme. Alternatively, DEVO [13] constitutes the premier monocular event-only odometry system, grounded in deep learning, which trains on event voxel grids and inverse depth maps under the guidance of ground truth poses, achieving markedly enhanced accuracy over conventional techniques. Although deep learning odometry signifies a stride in precision, cost-effective accuracy enhancements on computation-constrained platforms, such as drones, may still benefit from integrating event data with IMU measurements.

B. Event-based Visual Inertial Odometry

Zhu et al. [14] introduced the pioneering event-based odometry by integrating an event-driven tracker with IMU data. However, its real-time application is hindered by computationally demanding feature tracking. Vidal et al. [15] advanced the field by establishing a tightly integrated framework combining events, frames, and inertial readings for enhanced state estimation. Complementarily, some research integrates events and IMU within a continuous-time schema. Mueggler et al. [16] devised a continuous-time approach for merging high-frequency event and IMU data for visual inertial odometry. Embracing a sophisticated feature tracker [17], EKLT-VIO [18] demonstrated accurate performance in Mars-like and high-dynamic-range sequences and showed good potential in vision-based exploration on Mars. Dai et al. [19] presented a comprehensive model marrying continuous-time inertial data with events, innovating an exponential decay correlation for events. Guan et al. [20] developed a uniformly distributed event corner detection algorithm for raw events, and designed two event representations to perform feature tracking and loop closure matching for a keyframe-based visual inertial system. Based on this, Guan et al. [21] utilized motion compensation for the event stream based on IMU measurements and introduced line and point feature constraints, improving odometric precision.

Although time surfaces based on exponential decay can be additionally motion compensated to enhance performance, as employed by [11], [21], they still rely on accurate sensor measurements (e.g. IMU) or accurate odometer estimates. To make the exponential decay kernel adaptable to the dynamics of the event stream, we adopted the event activity model proposed by Nunes et al. [6], and introduced an adaptive decay kernel for different event cameras and practical operating conditions. Leveraging such advancements, we propose a monocular event-inertial odometry with the adaptive decay kernel and optimize feature tracking for the problems faced by the polarity-weighted time surface-based approach, resulting in a more robust and accurate odometry.

III. Methodology

A. System Overview

The system overview is depicted in Fig. 2. Raw events are transformed into adaptive decay-based time surfaces (Sec. III-C) using the proposed Time-Priority strategy. Data from a monocular event camera and an IMU are fused in the Multi-State Constraint Kalman Filter (MSCKF) [22], enabling precise and low-latency pose estimation. Given the sparse event outputs from the event camera when the system is in static or slow motion, we utilize a dynamic initialization method [23] to incorporate feature observations across time surfaces and IMU measurements for system initialization. Once initialized, upon receiving a new event packet, the system leverages IMU measurements to propagate the mean and covariance of the state to the timestamp of the new event packet, and augments a cloned IMU pose to the state vector (Sec. III-B). Afterward, sparse feature observations from the proposed tracking method (Sec. III-D) are utilized to update the state (Sec. III-E). Landmarks and old cloned poses are marginalized out of the state vector for computational efficiency.

B. State Vector of Event-Inertial Odometry

The state vector $\mathbf{x}_j$ at timestamp t_j of the system is composed of the inertial state $\mathbf{x}_{I_j}$, inertial pose clones $\mathbf{x}_C$

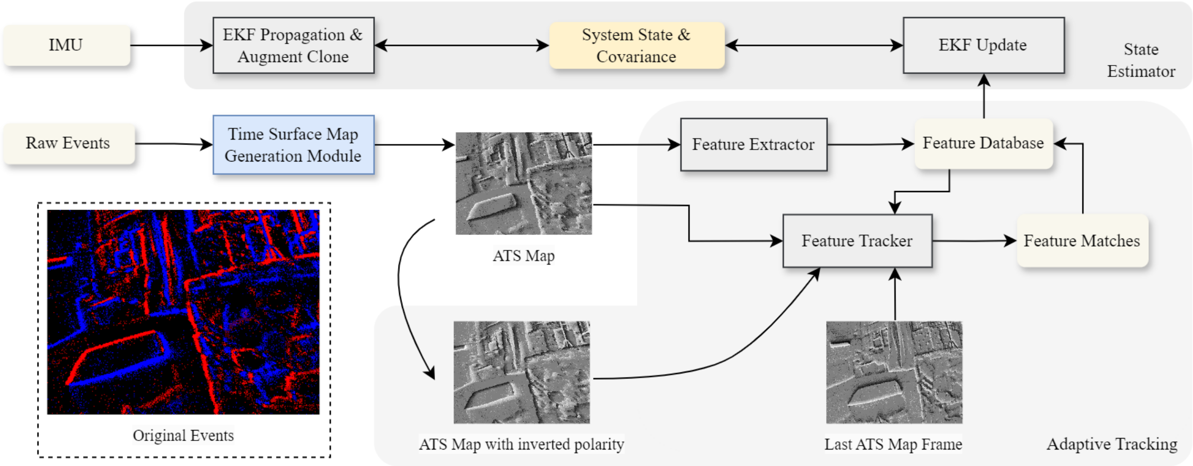

Fig. 2: System overview. The system includes the Time Surface Map Generation Module, the Adaptive Tracking Module and the State Estimator. The Adaptive Tracking Module fuses the tracking results of Polarity-weighted and Polarity-inverted time surfaces to alleviate the tracking loss that occurs when the direction of motion or illumination changes.

and the inverse depths of landmarks $\mathbf{x}_f$, given by

$$\begin{aligned} \mathbf{x}_j &= \begin{bmatrix} \mathbf{x}_{I_j}^T & \mathbf{x}_C^T & \mathbf{x}_f^T \end{bmatrix}^T , \\ \mathbf{x}_{I_j} &= \begin{bmatrix} {}^{I_j}_G\bar{q}^T & {}^G\mathbf{p}_{I_j}^T & {}^G\mathbf{v}_{I_j}^T & \mathbf{b}_{a_j}^T & \mathbf{b}_{g_j}^T \end{bmatrix}^T , \\ \mathbf{x}_C &= \begin{bmatrix} {}^{I_j}_G\bar{q}^T & {}^G\mathbf{p}_{I_j}^T & \cdots & {}^{I_{j-m}}_G\bar{q}^T & {}^G\mathbf{p}_{I_{j-m}}^T \end{bmatrix}^T \\ \mathbf{x}_f &= \begin{bmatrix} \rho_1 & \cdots & \rho_l & \cdots & \rho_s \end{bmatrix}^T , \end{aligned} \tag{1}$$

where ${}^{I_j}_G\bar{q}$ denotes the unit quaternion, and the corresponding rotation matrix is denoted as $\mathbf{R}({}^{I_j}_G\bar{q}) = {}^{I_j}_G\mathbf{R}$. Furthermore, ${}^G\mathbf{p}_{I_j}$ and ${}^G\mathbf{v}_{I_j}$ denote the position and velocity of the IMU in the global frame at time t_j, while $\mathbf{b}_{a_j}$ and $\mathbf{b}_{g_j}$ are the biases of the accelerometer and the gyroscope, respectively.

The sliding window maintains a set of $m + 1$ cloned IMU poses at the timestamps of the event packet for feature triangulation and state update like an RGB-Camera-based VIO. Stable features tracked across the entire sliding window frames are considered as SLAM features and will be augmented to the state vector for extending the time span of active constraints. In this paper, we use the inverse depth parameterization and only include maximum s landmarks in the state for limiting computational complexity. The extrinsic parameters ${}^C_I\mathbf{R}, {}^C\mathbf{p}_I$ between the IMU and the camera are precalibrated and assumed to be known. In our practical experiments, m and s are set to 10 and 50, respectively.

C. Time Surface and Adaptive decay

Event Definition: Events primarily occur in regions with rich environmental textures, such as edges. When the brightness of a pixel changes beyond a certain threshold, an event is triggered. An event at timestamp t_k with pixel coordinate $[u_k, v_k]^T$ is defined as:

$$e_k = \{u_k, v_k, t_k, p_k\} , \tag{2}$$

where the polarity $p_k \in \{-1, 1\}$ indicates whether the brightness increase or decrease.

Time Surface: As illustrated in Fig. 1, event frames (in the second column) generated from spatio-temporal neighborhoods of raw event are the primitive method for representing event positions within an image. This representation method is highly sensitive to event counts and often struggles to accurately capture environmental texture due to the lack of consideration for event timestamps. To address this issue, the Time Surface [2], which retains event timestamps, is proposed. To prioritize recent event information, the time surface is usually used together with an exponential decay kernel [5]:

$$\tau_1(e_p, t) \doteq \exp\left(-\frac{t - t_p}{\eta}\right) , \tag{3}$$

where η represents a predefined time constant and e_p is any previous event and t_p is its timestamp. The time constant η needs to be fine-tuned to mitigate interference from past events. However, this invariant time constant η does not accommodate all camera motion and should be adjusted with different motion situations. A constant η results in bold edges in the time surfaces when the motion is aggressive.

Adaptive Decay Kernel: To reflect the dynamic characteristics of the event stream, [6] proposes the *Event Activity* $\alpha(t)$ as follows:

$$\alpha(t) = \tau_2(e_p, t)\alpha(t_p) + n(t, t_p) , \tag{4}$$

where t_p is the timestamp of any previous event e_p and $n(t, t_p)$ is the count of events within the time interval $[t_p, t]$. Then we can obtain the adaptive decay kernel for the time surface, which is derived from the exponential decay. Given the time derivative of Eq. (3):

$$\frac{\partial \tau_1(e_p, t)}{\partial t} = -\frac{1}{\eta} \exp\left(-\frac{t - t_p}{\eta}\right) = -\lambda(t)\tau_1(e_p, t) , \tag{5}$$

where λ represents the decay rate which we expect to be time-varying and influenced by the event activity. The decay rate $\lambda(t)$ is intended to be proportional to the event activity

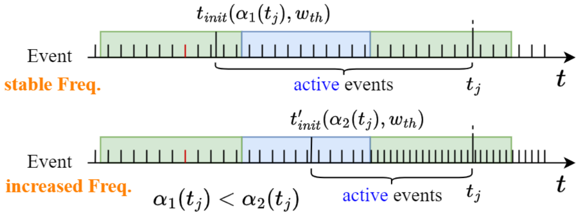
$t_{init}(\alpha_1(t_j), w_{th})$
Event
stable Freq.
active events
t_j
t
$t'_{init}(\alpha_2(t_j), w_{th})$
Event
increased Freq.
$\alpha_1(t_j) < \alpha_2(t_j)$
active events
t_j
t

Fig. 3: The impact of event activity on t_{init}. The increased value of event activity shortens the temporal scope of active events, thereby mitigating interference from past events.

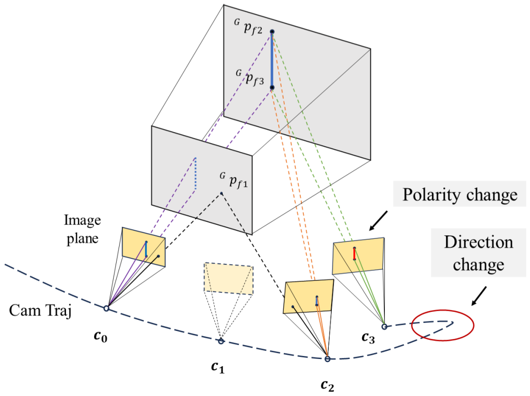

Fig. 4: The direction of motion influences the polarity of events. Abrupt changes in motion direction can cause a reversal in event polarity, which may lead to tracking loss. Lines are **only** used to highlight polarity changes.

$\lambda(t) \propto \alpha(t)$, solving the Eq. (5) yields the *adaptive decay*:

$$\tau_2(e_p, t) = \frac{1}{1 + r \cdot \alpha(t_p)(t - t_p)}, \tag{6}$$

where $\alpha(t_p)$ is the event activity at previous timestamp t_p and r is the coefficient. Eq. (6) introduces the coefficient r, which differs from [6] where the coefficient r is set to a constant of 1. However, this setting ignores the influence of the camera resolution as well as the threshold for triggering events, resulting in too little decay for redundant events or too much decay for active events.

The number of event activities increases with event frequency. As inferred from Eq. (6), when the event frequency increases, events with the later timestamp have less decays. In other words, if the pixel value remains constant, a higher event frequency implies that the event timestamps are closer to t. Event information is retained for a longer period of time when the event frequency is high. Therefore, modeling the dynamics of the event stream by introducing the event activity enables adaptively adjusting the decay for the events and thus provides high-quality imaging.

Adaptive decay-based Time Surface (ATS) : We then discuss how to use the adaptive decay $\tau_2(\cdot)$ to generate an adaptive time surface with timestamp t_j. The adaptive decay kernel $\tau_2(\cdot)$ includes the event activity which could reflect the dynamic characteristics of the event stream. Thus, the adaptive time surface based on the kernel $\tau_2(\cdot)$ is more robust to various event frequencies and naturally provides a novel and effective way to filter out inactive events and reduce noise. Specifically, any previous event e_p with $\nu(e_p, t_j) < w_{th}$ is considered inactive (t_j is the timestamp of the adaptive time surface), with $\nu(e_p, t)$ is defined as:

$$\nu(e_p, t) = \frac{1}{1 + r \cdot \alpha(t)(t - t_p)}, \tag{7}$$

here w_{th} is a flexible threshold and $\nu(\cdot)$ is modified from Eq. (6) for the convenience of calculation. In practical computations, $\alpha(t)$ can be approximated by the event activity associated with the closest event to time t.

When the system receives an event message, as shown in the Fig. 3, the event activity is calculated recursively, i.e., the decay $\tau_2(e_{k-1}, t_k)$ is calculated first from Eq. (6), and then Eq. (4) is used to calculate the event activity $\alpha(t_k)$ at the event timestamp t_k. $n(t_k, t_{k-1})$ is always 1 since there is only one event in adjacent moments. In this recursive manner, calculate all event activity at the moment of the event timestamp and save it with the corresponding event.

And there are two different strategies to determine the initial event timestamp t_{init} and the timestamp t_j of the adaptive decay-based time surface:

- **Data-Priority** [6]: t_{init} starts with the first event timestamp and is typically reset after obtaining a set of active events when $\nu(e_{init}, t_j) < w_{th}$. The method tends to utilize all events without duplication, but the timestamp t_j of the time surface cannot be predicted in advance. In addition, the frequency of the time surfaces generated based on this method is sensitive to the threshold w_{th}.
- **Time-Priority**: we first estimate t_{init} based on the the timestamp t_j of the time surface, event activity $\alpha(t_j)$ and predefined threshold w_{th}, as follows:

  $$\nu(e_{init}, t_j) = w_{th} \implies t_{init} = t_j - \frac{1 - w_{th}}{r \cdot \alpha(t_j) w_{th}}. \tag{8}$$

  This approach allows for arbitrary timestamps for time surfaces and decouples event activity and generation of the adaptive time surfaces for parallel processing.

Events that are considered active have timestamps between t_{init} and t_j. We find the latest active events in each pixel, denoted as $e_l(\mathbf{x})$ and the pixel value is $\tau_2(e_l(\mathbf{x}), t_j)$ or $p_l \cdot \tau_2(e_l(\mathbf{x}), t_j)$ if the time surface is polarity-weighted. If the pixel has no active events, set the value of the pixel to zero. Finally, all pixel values are mapped to $[0, 255]$.

D. Polarity-aware Feature Tracking

To verify the performance of adaptive decay-based time surfaces in the odometry, we utilize the Fast corners [24] and track them with the LK optical flow [25] method. This approach functions properly within datasets containing event cameras. However, we observe that abrupt changes in motion direction may lead to tracking failures as they violate the assumption of brightness constancy. Diverse motion directions induce events with opposite polarities at identical scene positions, leading to fluctuations in pixel values, as illustrated in Fig. 1 and Fig. 4. Such fluctuations notably impact the accuracy of odometry estimation.

Actively inverting event polarities generates a polarity-inverted time surface map, and this map can be regarded as an approximation under the condition of maintaining event polarity, which to some extent mitigates the issue of the violated brightness constancy assumption. By simultaneously tracking both the original and polarity-inverted time surfaces and adaptively merging the tracking results, we ensure continuous and stable feature tracking. Estimating the velocity based on IMU measurements to make this determination may suffer from misjudgments or delays. We track both images separately and then compare the number of successfully tracked features. When the number of successfully tracked features on the polarity-weighted time surface is less than that on the polarity-inverted map, we merge the both tracking results. Finally, we apply the RANSAC method to eliminate outliers and obtain the final matches by estimating the fundamental matrix from the matched features.

E. Update with Tracked Features

When tracking results are available, we select features within the sliding window for triangulation and system update. The system update follows two principles. Firstly, we aim to triangulate using as much data as possible to ensure accuracy, thereby ensuring positive gains from system updates. Secondly, stable tracked points are selectively included in the state vector for continuous estimation, aiming to improve system accuracy while maintaining control over increased computation time. Based on these principles, features are categorized into SLAM and MSCKF types.

Specifically, for the current time surface feature tracking, if tracking fails or reaches the maximum length (equal to the sliding window), suggesting that landmark observations are maximized, we attempt to triangulate. For a landmark ${}^{G}\mathbf{p}_f$ associating with a successfully triangulated feature, we have

$$ {}^{G}\mathbf{p}_f = {}^{I_{j-m}}_{G}\mathbf{R}^T \left({}^{C}_{I}\mathbf{R}^T \frac{1}{\rho_l} \pi^{-1}\left(\begin{bmatrix} u_i \\ v_i \end{bmatrix}\right) + {}^{C}\mathbf{p}_I \right) + {}^{C}\mathbf{p}_{I_{j-m}} , \tag{9} $$

where $\pi(\cdot)$ represents the back projection function that transforms a pixel to the normalized image plane. The inverse depth ρ_l is defined in the anchor frame. For instance, the SLAM feature always selects the oldest frame $\{C_{j-m}\}$ in the sliding window as the anchor frame. Consequently, the position ${}^{G}\mathbf{p}_f$ could be computed using the observation e_k in the anchor frame. It's worth noting that for the MSCKF feature, the anchor frame is determined as the first observed frame. Given the new observation (e.g. e_i) in the latest frame $\{C_j\}$, the nonlinear measurement model is given by:

$$ \mathbf{z}_{E_i} = h_e\left(\mathbf{x}_{C_j}, {}^{G}\mathbf{p}_f\right) + \mathbf{n}_{E_i} = \pi({}^{C_j}\mathbf{p}_f) + \mathbf{n}_{E_i} \tag{10} $$

$$ {}^{C_j}\mathbf{p}_f = {}^{C}_{I}\mathbf{R}\, {}^{I_j}_{G}\mathbf{R} \left({}^{G}\mathbf{p}_f - {}^{G}\mathbf{p}_{I_j}\right) + {}^{C}\mathbf{p}_I \tag{11} $$

where the measurement noise is associated with the event pixel noise and follows a white Gaussian distribution $\mathcal{N}\left(\mathbf{0}, \mathbf{Q}_e\right)$. Features tracked throughout the entire sliding window are augmented into the state vector, continuously updating the system with subsequent landmark observations—referred to as SLAM features. Other successfully triangulated points, termed MSCKF features, are updated using the efficient MSCKF nullspace projection [22], avoiding the inclusion of landmarks in the state vector and thus reducing system complexity.

IV. Experiments

In this section, we perform extensive experiments to evaluate the efficacy of the proposed method. Initially, we conduct comparisons with classical visual-inertial odometry approaches, specifically two optimization-based methods: ORB-SLAM3 [26] and VINS-Mono [27], along with a filtering-based method, OpenVINS [28], on the HKU dataset [29]. Subsequently, we compare our method with event-inertial odometry [14], [15], [19], [20], [30], [31] on the DAVIS 240C dataset [32], which is collected using the event camera (with a resolution of 240×180) and an internal IMU. Additionally, we conducted ablation experiments on different decay kernels and feature tracking methods, on the dataset [20]. Both datasets [20], [29] contain the DAVIS 346 event camera with a resolution of 346×260. Finally, we assess the time consumption of the system to evaluate its real-time performance. The experiments are conducted with the left camera if the stereo event camera is available.

The experiments are carried out on a desktop computer equipped with an Intel Core i7-8700 CPU running Ubuntu 20.04 and ROS Noetic. Accuracy metrics for odometry evaluation consist of Mean Position Error (MPE, %, per 100 meters), Mean Yaw Error (MYE, deg/m) and Absolute Trajectory Error (ATE, m), while the trajectories are aligned with the ground truth using SE(3) Umeyama alignment [33] before evaluation. Taking into account the differences between event cameras and to obtain better environmental textures, r in Eq. (6) is set to 0.2 in the DAVIS 240C dataset and 0.1 for another two datasets.

A. Odometry Accuracy Evaluation and Comparison

Comparison with VIO: We first test the accuracy of three visual odometry methods with the proposed odometry to validate the superiority of event cameras in challenging environments. The results are presented in Tab. I. VINS-Mono [27] and OpenVINS [23] failed in all sequences in this dataset, while ORB-SLAM3 [26] (using the monocular visual-inertial method) demonstrates a larger MPE in all sequences, with an average MPE of 1.023. Our method exhibits consistent performance across all sequences with an average MPE of 0.453. From the above results, our event-based odometry is more accurate than conventional camera-based methods in environments with high dynamic ranges and intense motion.

Comparison with EIO: Next, we compare the accuracy of this method with other event-based approaches. The estimated and ground-truth trajectories were aligned with the subset [5-10]s. The results are presented in Tab. II. The proposed method achieves an average MPE of 0.35, indicating an error of 0.35m for 100m motion. The test

TABLE I: Comparison of MPE (Unit: %) results between the proposed method and visual-inertial odometry. Notably, both VINS-Mono [27] and OpenVINS [23] failed across all sequences. Our method demonstrates better adaptability in high-speed and HDR scenes.

Sequence	hku_agg_rota	hku_agg_small_flip	hku_agg_tran	hku_agg_walk	hku_dark_normal	hku_hdr_agg	hku_hdr_circle	hku_hdr_slow	hku_hdr_tran_rota
ORB-SLAM3 [26]	0.711	1.895	0.841	1.389	0.695	0.623	1.451	0.635	0.971
Ours	0.276	0.807	0.211	0.350	0.524	0.271	0.714	0.430	0.496

TABLE II: Comparison of Odometry MPE (Unit, %) on the DAVIS 240C dataset [32]. The methods tested below are all EIO methods. The MPE results are obtained from respective articles, with Alzugaray's results specifically cited from [34]. Because of the differences in the calculation of rotational errors for these methods used for comparison, only MYEs of the proposed method are included.

Sequence	Length (m)	Zhu's [14]	Rebecq's [30]	Vidal's [15]	Alzugaray's [31]	Guan's [20]	Dai's [19]	Ours
boxes_6dof	69.852	3.61	0.69	0.44	2.03	0.61	1.5	**0.32** (0.02)
boxes_translation	65.237	2.69	0.57	0.76	2.55	**0.34**	1.0	0.36 (0.01)
dynamic_6dof	39.615	4.07	0.54	**0.38**	0.52	0.43	1.5	0.49 (0.05)
dynamic_translation	30.068	1.90	0.47	0.59	1.32	**0.26**	0.9	0.59 (0.05)
hdr_boxes	50.088	1.23	0.92	0.67	1.75	0.40	1.8	**0.31** (0.02)
hdr_poster	55.437	2.63	0.59	0.49	0.57	0.40	2.8	**0.18** (0.02)
poster_6dof	61.143	3.56	0.82	0.30	1.50	**0.26**	1.2	0.31 (0.03)
poster_translation	49.265	0.94	0.89	**0.15**	1.34	0.40	1.9	0.23 (0.04)
Average	52.588	2.58	0.69	0.47	1.45	0.39	1.58	**0.35** (0.03)

results demonstrate that our method outperforms others in three sequences and shows comparable performance in the remaining sequences. The estimated trajectories of sequence *hdr_boxes* and *hdr_poster* are shown in Fig. 5. The figure shows that our odometry can provide good pose estimates, but can still find a partial difference between the estimated trajectory and the ground truth. When the event camera moves slowly, the output events are not sufficient to reflect the texture of the environment, which inevitably leads to inadequate features and inaccurate tracking. Compared to other event-inertial odometry methods, our approach estimates poses more accurately. The adaptive decay-based method enables high-quality texture for feature extraction and tracking, enhancing odometry accuracy with the proposed tracking method.

B. Decay Comparison and Parameter Setting

In this section, we analyze how different decay functions (exponential and adaptive decay) for time surface representations affect odometry accuracy. Specifically, we evaluate various time constants η for the exponential decay kernel and compare them with the adaptive decay kernel (as shown in Tab. III). The experiments utilize a dataset publicly available in [20], including a DAVIS 346 event camera.

Comparison of different decays: Tab. III reveals that different time constants η affect the accuracy of the odometry. Usually, a larger time constant will keep more information from older events but also introduce unwanted disturbances. Setting η to 90ms may result in coarser and overlapping edges, leading to lower precision. However, in sequences such as *vicon_hdr1* and *vicon_hdr2*, slightly larger time constants can retain more information, with η set to 60ms exhibiting slightly higher accuracy compared to η set to 30ms. The odometry with exponential decay-based time surface achieves the relatively best average accuracy when the time constant η is 30ms. But it is inevitably prone to bold edges and trailing when the event camera motion suddenly becomes faster. The experimental results show that the odometry with the adaptive decay-based time surface achieves the lowest average ATE of 0.182m. The adaptive decay, which adjusts the decay rate according to the dynamics of the event stream, facilitates the creation of high-quality time surface maps, providing better odometry accuracy than exponential decay.

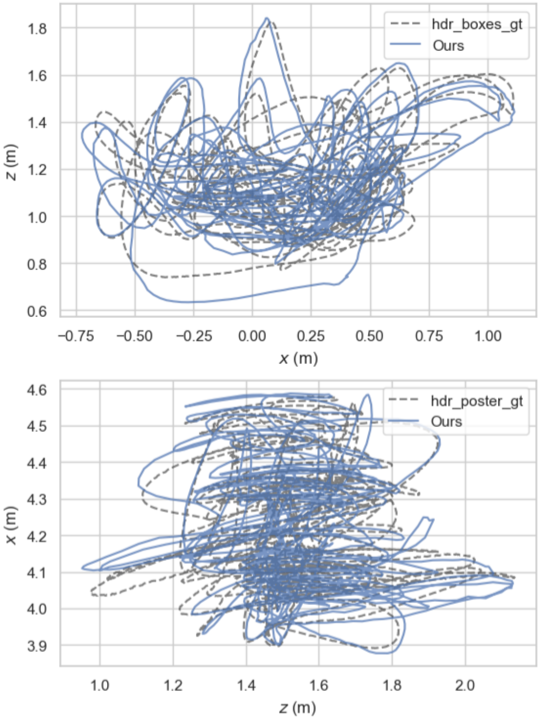

Fig. 5: Trajectory Estimates Comparison for hdr_boxes (top) and hdr_poster (down) in the DAVIS 240C Dataset [32]

Comparison of Different Thresholds w_{th}: The threshold w_{th} provides a novel solution for filtering out events that contribute little to the texture and saving the time surface generation time. From Eq. (8), t_{init} is first estimated based on the event activity to bound the time range of active events. If the event activity remains constant, a larger threshold w_{th}

TABLE III: Comparison of odometry ATE (m) for different time surface representations. TS stands for exponential decay, while ATS stands for adaptive decay. The parameter η denotes the time constant in the exponential decay, tested at values of 30, 60, and 90 ms. The threshold w_{th} for the adaptive decay is consistently set to 0.01, and (T) denotes the improved tracking algorithm.

Sequence	TS $\eta = 30$	TS $\eta = 60$	TS $\eta = 90$	ATS $w_{th} = 0.01$	ATS (T) $w_{th} = 0.01$
vicon_aggressive_hdr	0.377	0.397	0.450	**0.155**	0.163
vicon_dark1	0.248	0.294	0.251	0.124	**0.111**
vicon_dark2	**0.120**	0.163	0.191	0.253	0.187
vicon_darktolight1	0.273	0.309	0.359	**0.177**	0.219
vicon_darktolight2	0.235	0.353	0.249	0.209	**0.187**
vicon_hdr1	0.277	0.263	0.426	0.267	0.180
vicon_hdr2	0.365	0.282	0.866	0.232	**0.220**
vicon_hdr3	0.187	0.280	0.269	0.123	**0.122**
vicon_hdr4	0.229	0.272	0.520	0.192	**0.171**
vicon_lighttodark1	0.393	0.362	0.405	0.228	**0.226**
vicon_lighttodark2	0.306	0.501	0.596	**0.227**	0.211
Average	0.274	0.316	0.417	0.198	**0.182**

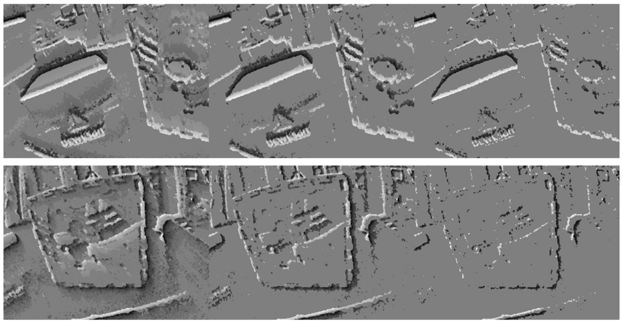

Fig. 6: Comparison of threshold w_{th} settings for adaptive decay-based time surfaces: 0.01, 0.05, and 0.1. A median blur with the kernel size of 1 is used to emphasize the difference. The threshold w_{th} needs to be set reasonably to ensure a sufficient number of events while reducing interference.

tends to utilize events within a smaller time interval, occasionally hindering the formation of desired environmental textures. Conversely, a smaller threshold tends to use events within a larger time interval, enriching image information but introducing interference from old events. Fig. 6 depicts adaptive time surface maps with different threshold settings. A large threshold of 0.1 results in insufficient texture, while a small threshold of 0.01 incorporates older events that cause trailing effects. It requires a reasonable setting of the threshold w_{th} according to practical requirements to achieve optimal texture performance. And a good threshold w_{th} setting usually accommodates all sequences in the dataset without having to set it individually for each sequence. It is important to note that the pixel values for pixels containing active events are determined by the decay kernel and the active event, rather than the threshold w_{th}.

C. Polarity-aware Feature Tracking Evaluation

Subsequently, we investigate the effectiveness of the polarity weighting and the polarity-aware feature tracking process, as presented in Tab. III. Experimental results indicate some improvement when using polarity weighting, achieving average ATE of 0.210m. Polarity weighting enhances the distinction between brighter and darker pixels in the time surface maps, particularly accentuating nearby edges. While offering the above benefits, it also introduces challenges for feature tracking, mainly when the event camera's motion direction changes abruptly, causing shifts in event polarity and undermining the assumption of intensity constancy. Our method incorporates the tracking results of time surfaces with inverted polarity, which somewhat mitigates the problem of tracking failures due to abrupt changes in the direction of motion, and thus achieves an improvement in the accuracy.

TABLE IV: Average Execution Time of Each Step in milliseconds

Operation	ATS Generation	Feature Detection & Tracking	MSCKF Feature Update	SLAM Feature Update	Others	Total
Average Time (ms)	5.4	6.7	1.3	0.2	2.4	16.0

D. Computational Efficiency

Finally, we conduct a detailed analysis of the time consumption for each operation in the proposed method, as outlined in Tab. IV. The analysis is conducted on a typical sequence *vicon_aggressive_hdr*, which has aggressive motion speed and includes high dynamic range scenes. The reported results represent the average values obtained from multiple tests. The statistical results indicate that the average time required for generating an adaptive time surface map is 5.4ms, while the adaptive tracking module is 6.7ms. The total time required for the system to complete one update is 16ms, indicating its capability for real-time performance.

V. Conclusions and Future Work

Cameras frequently experience motion blur during rapid movement and are constrained by a limited dynamic range. Event cameras, with novel dynamic vision sensors, offer potential solutions to these problems by tracking changes in pixel brightness. However, the asynchronous output of event cameras poses challenges in leveraging event information. We propose a monocular event-inertial odometry with an adaptive decay kernel-based time surface and an MSCKF in the back-end for state propagation and update. The adaptive decay highlights recent events according to the dynamic characteristics of the event stream and can also filter out inactive events. Polarity-weighted time surfaces suffer from polarity shifts when motion direction changes abruptly, and we refine feature tracking by incorporating an additional polarity-inverted time surface map to improve robustness. Extensive experiments demonstrate the competitive accuracy of our proposed method. The number of events may decrease during slow motion, resulting in inadequate textures that pose challenges to stable feature tracking. We will continue to work on improving its robustness and accuracy.

References

[1] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154–180, 2022.

[2] T. Delbruck *et al.*, "Frame-free dynamic digital vision," in *Proceedings of Intl. Symp. on Secure-Life Electronics, Advanced Electronics for Quality Life and Society*, Citeseer, vol. 1, 2008, pp. 21–26.

[3] Y. Zhou, G. Gallego, and S. Shen, "Event-based stereo visual odometry," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1433–1450, 2021.

[4] Y. Zuo, J. Yang, J. Chen, X. Wang, Y. Wang, and L. Kneip, "Devo: Depth-event camera visual odometry in challenging conditions," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 2179–2185.

[5] X. Lagorce, G. Orchard, F. Galluppi, B. E. Shi, and R. B. Benosman, "Hots: A hierarchy of event-based time-surfaces for pattern recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 7, pp. 1346–1359, 2017.

[6] U. M. Nunes, R. Benosman, and S.-H. Ieng, "Adaptive global decay process for event cameras," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 9771–9780.

[7] 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)*, 2016, pp. 16–23.

[8] H. Kim, S. Leutenegger, and A. J. Davison, "Real-time 3d reconstruction and 6-dof tracking with an event camera," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI 14*, Springer, 2016, pp. 349–364.

[9] H. Rebecq, T. Horstschaefer, 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, 2017.

[10] J. Hidalgo-Carrió, G. Gallego, and D. Scaramuzza, "Event-aided direct sparse odometry," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 5771–5780.

[11] J. Huang, S. Zhao, T. Zhang, and L. Zhang, "Mc-veo: A visual-event odometry with accurate 6-dof motion compensation," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 1756–1767, 2024.

[12] R. Pellerito, M. Cannici, D. Gehrig, J. Belhadj, O. Dubois-Matra, M. Casasco, and D. Scaramuzza, *End-to-end learned visual odometry with events and frames*, 2024. arXiv: 2309.09947 [cs.CV].

[13] S. Klenk, M. Motzet, L. Koestler, and D. Cremers, "Deep event visual odometry," in *2024 International Conference on 3D Vision (3DV)*, 2024, pp. 739–749.

[14] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based visual inertial odometry," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5816–5824.

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

[16] 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.

[17] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza, "Eklt: Asynchronous photometric feature tracking using events and frames," *International Journal of Computer Vision*, vol. 128, no. 3, pp. 601–618, 2020.

[18] F. Mahlknecht, D. Gehrig, J. Nash, F. M. Rockenbauer, B. Morrell, J. Delaune, and D. Scaramuzza, "Exploring event camera-based odometry for planetary robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 8651–8658, 2022.

[19] B. Dai, C. L. Gentil, and T. Vidal-Calleja, "A tightly-coupled event-inertial odometry using exponential decay and linear preintegrated measurements," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 9475–9482.

[20] W. Guan and P. Lu, "Monocular event visual inertial odometry based on event-corner using sliding windows graph-based optimization," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 2438–2445.

[21] W. Guan, P. Chen, Y. Xie, and P. Lu, "Pl-evio: Robust monocular event-based visual inertial odometry with point and line features," *IEEE Transactions on Automation Science and Engineering*, pp. 1–17, 2023.

[22] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 3565–3572.

[23] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "Openvins: A research platform for visual-inertial estimation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4666–4672.

[24] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*, Springer, 2006, pp. 430–443.

[25] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI'81: 7th international joint conference on Artificial intelligence*, vol. 2, 1981, pp. 674–679.

[26] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[27] 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.

[28] P. Geneva and G. Huang, "Openvins state initialization: Details and derivations," Tech. Rep. RPNG-2022-INIT, University of Delaware, 2022. Available: https . . ., Tech. Rep.

[29] P. Chen, W. Guan, and P. Lu, "Esvio: Event-based stereo visual inertial odometry," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3661–3668, 2023.

[30] H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization," in *British Machine Vision Conference (BMVC)*, 2017.

[31] I. Alzugaray and M. Chli, "Asynchronous multi-hypothesis tracking of features with event cameras," in *2019 International Conference on 3D Vision (3DV)*, 2019, pp. 269–278.

[32] 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.

[33] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[34] W. Guan, P. Chen, H. Zhao, Y. Wang, and P. Lu, "Evi-sam: Robust, real-time, tightly-coupled event–visual–inertial state estimation and 3d dense mapping," *Advanced Intelligent Systems*, p. 2 400 243, 2024.