RATE: Real-time Asynchronous Feature Tracking with Event Cameras

Mikihiro Ikura[1], Cedric Le Gentil[2], Marcus G. Müller[3],
Florian Schuler[3], Atsushi Yamashita[1], and Wolfgang Stürzl[3]

***Abstract*— Vision-based self-localization is a crucial technology for enabling autonomous robot navigation in GPS-deprived environments. However, standard frame cameras are subject to motion blur and suffer from a limited dynamic range. This research focuses on efficient feature tracking for self-localization by using event-based cameras. Such cameras do not provide regular snapshots of the environment but asynchronously collect *events* that correspond to a small delta of illumination in each pixel independently, thus addressing the issue of motion blur during fast motion and high dynamic range. Specifically, we propose a continuous real-time asynchronous event-based feature tracking pipeline, named RATE. This pipeline integrates (i) a corner detector node utilizing a time slice of the Surface of Active Events to initialize trackers continuously, along with (ii) a tracker node with a proposed "tracking manager", consisting of a grid-based *distributor* to reduce redundant trackers and to remove feature tracks of poor quality. Evaluations using public datasets reveal that our method maintains a stable number of tracked features, and performs real-time tracking efficiently while maintaining or even improving tracking accuracy compared to state-of-the-art event-only tracking methods. Our ROS implementation is released as open-source: https://github.com/mikihiroikura/RATE**

I. INTRODUCTION

Autonomous robots are emerging as transformative tools for exploring dangerous and uncharted territories, pushing the boundaries of human access. This capability is particularly important for tasks too risky for direct human interaction, as exemplified by quadrupedal robots deployed in challenging environments and equipped with laser sensors and cameras for perception [1], [2]. Similarly, unmanned aerial vehicles (UAVs) have played a crucial role in not only terrestrial but also Martian terrain exploration [3], [4]. Realizing these diverse mission scenarios depends on robust vision-based self-localization systems, such as Visual Odometry (VO) [5] and Simultaneous Localization and Mapping (SLAM) [6]. These systems enable autonomous navigation even when GPS becomes unavailable. However, vision-based approaches are susceptible to errors under challenging conditions including high-dynamic range (HDR) and fast motions (see [7] for an attempt to quantify challenging situations in visual SLAM). Consequently, these limitations can lead to system failures and impede successful mission completion.

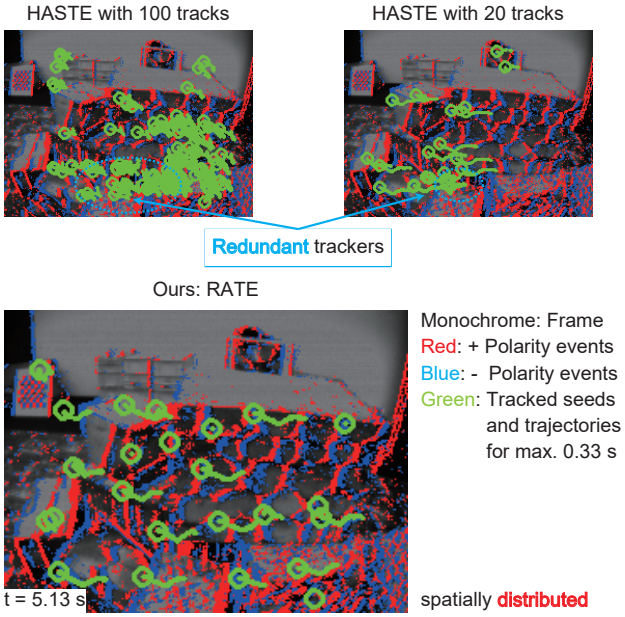


Fig. 1: Comparison of our proposed Real-time Asynchronous Tracking pipeline with Event cameras (RATE) with the baseline of multiple HASTE [8] trackers initialized by Shi-Tomasi corner detector. In the baseline methods (top row), redundant trackers are positioned closely together and persist even after updates. Compared to these baselines, RATE distributes trackers across the entire field of view without overlapping.

As a promising countermeasure for these challenges faced by conventional vision sensors in autonomous robots, event-based cameras, inspired by biological vision systems [9], have recently emerged and gained popularity [10]. Event-based cameras operate asynchronously, recording pixel-level brightness changes as events $e = \{t, x, y, p\}$ in real time. An event e is represented by the pixel location (x, y), a timestamp t with microsecond resolution, and a polarity p indicating the direction of the brightness changes [11]. These characteristics offer numerous advantages in event-based cameras over frame-based cameras, in particular high-dynamic range, robustness to motion blur, and low latency. With a dynamic range of 140 dB, event-based cameras can reduce the impact of lighting conditions such as direct

[1]M. Ikura and A. Yamashita are with the Department of Human and Engineered Environmental Studies, Graduate School of Frontier Sciences, The University of Tokyo, 5-1-5 Kashiwanoha, Kashiwa, Chiba 277–8563, Japan {ikura, yamashita}@robot.t.u-tokyo.ac.jp

[2]C. Le Gentil is with the Robotics Institute at the University of Technology Sydney, 15 Broadway Ultimo, NSW 2007, Australia cedric.legentil@uts.edu.au

[3]M. G. Müller, F. Schuler, and W. Stürzl are with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Münchener Str. 20, 82234 Weßling, Germany {Marcus.Mueller, Florian.Schuler, Wolfgang.Stuerzl}@dlr.de

C. Le Gentil is supported by the Australian Research Council Discovery Project under Grant DP210101336.

sunlight and sudden shadows. Additionally, event-based cameras enable localization in high-speed scenarios thanks to their robustness to motion blur and low latency. Therefore, event-based cameras have the potential to revolutionize autonomous robots by offering powerful solutions to the challenges of vision-based self-localization.

Motivated by the advantages of event-based cameras for VO in autonomous robots [12], [13], this research tackles the challenge of real-time, continuous feature tracking. As a crucial element in vision-based self-localization, feature tracking demands uninterrupted operation with minimal latency. Moreover, the ability to track multiple features in parallel is essential for robust ego-motion estimation. We propose a novel pipeline (named "RATE") specifically designed for continuous, real-time tracking of multiple features using asynchronous event data. As illustrated in Figure 1, RATE focuses on efficiency by tracking solely non-redundant features that are spatially well-distributed throughout the image plane. We describe RATE in detail in section III. To evaluate the proposed pipeline, we employ publicly available event datasets [14] captured in real-world environments. Our results, presented in section IV, demonstrate significant improvements over conventional event-based feature tracking methods, such as EKLT [15], in terms of both the number of simultaneously tracked features and real-time performance.

II. RELATED WORK

Monocular event-based VO and SLAM have emerged as active research topics to overcome the challenges faced by frame-based cameras. The pioneering method, EVO [16], employs a direct approach by generating a 3D depth map from events, enabling accurate estimation of 6 Degree-of-Freedom (DoF) without event-based feature extraction. However, the computational cost associated with 3D depth map generation is impeding real-time processing. In contrast, some indirect approaches in VO and SLAM generate frame-like representations of event data and utilize feature-based methods. For instance, Glover et al. [17] apply Harris corner detection to an edge map updated with each event. On the other hand, Ultimate SLAM [18] incorporates an Inertial Measurement Unit (IMU) for motion compensation and generates accumulated events for FAST corner detection [19] and KLT tracking [20]. These approaches leverage traditional frame-based methods by aggregating batches of events in image-like data structures. However, they partially sacrifice the unique advantages of event-based cameras, particularly asynchronous data processing whereas VIO methods like [21] and [22] show that events can be individually accounted in the state estimation back-end. Unfortunately, the line detection and tracking front-end of [21] suffer from high computational cost, and the tracking strategy of [22] lacks robustness.

Some researchers have also contributed to event-driven feature detection [23] and tracking [24], [25]. These approaches allow for individual and asynchronous updates of features with each event. The event-driven feature detection in [23] achieves real-time performance by exploiting the shape of the Surface of Active Events (SAE) [26], [27]. However, this method does not perform tracking and the detection itself is quite noisy, especially in rotational motion [26]. The event-based tracking proposed in [24] suffers from limitations in accuracy and computational speed, hindering real-time implementation. On the other hand, the event pattern tracking with Gaussian Process [25] has improved accuracy but remains computationally expensive and thus falls short in terms of real-time performance. EKLT [15] combines the advantages of frame-based and event-based cameras, leveraging features that are independent of motion direction in frame-based cameras while benefiting from the high dynamic range and absence of motion blur in event-based cameras. By exploiting these benefits, EKLT achieves asynchronous feature tracking with high temporal resolution. However, the computational cost of this tracking method hinders real-time processing on standard hardware. On the other hand, HASTE [8], a novel event-driven tracking methodology, can track features in patches asynchronously relying solely on events. The method involves generating a template patch and a model updated by the latest event. From a small number of potential movements ("hypotheses"), the one that maximizes the correlation between the template and the model is selected, allowing for continuous updates of the feature state (position and rotation) with low latency. However, HASTE requires manual initialization to initiate tracking and cannot recover tracking once it is lost. Furthermore, efficient handling of multiple tracking instances with HASTE has not been considered so far.

Considering these related works on event-based feature tracking, we propose an approach that achieves (i) continuous, (ii) real-time, and (iii) event-based asynchronous tracking of multiple features.

III. PROPOSED METHOD

Figure 2 shows an overview of our proposed pipeline RATE for continuous real-time event-based asynchronous multiple feature tracking. The input is an event stream where an event $e_i = (t_i, x_i, y_i, p_i)$ consists of timestamps t [μs], pixel coordinates (x, y) [px], and polarities p (-1 or 1). RATE comprises a corner detector node and a tracking node. The corner detector node receives events and publishes track seeds at 30 Hz, which include the initial positions of trackers. The tracker node, with its *distributor*, initializes HASTE trackers [8] in each *sub-image* with published track seeds, updates them for each single event of the input stream, and publishes tracking results asynchronously depending on the timestamp of each single event. Finally, the proposed pipeline RATE outputs tracking ID, updated timestamp, and pixel coordinates as tracking results. The following subsections explain about procedure in the nodes in more detail.

A. Corner detector node with time slice of Surface of Active Events

Figure 3 illustrates the procedure of corner detection for tracking initialization. The Surface of Active Event (SAE) [26], [27] is a frame-like representations of events,

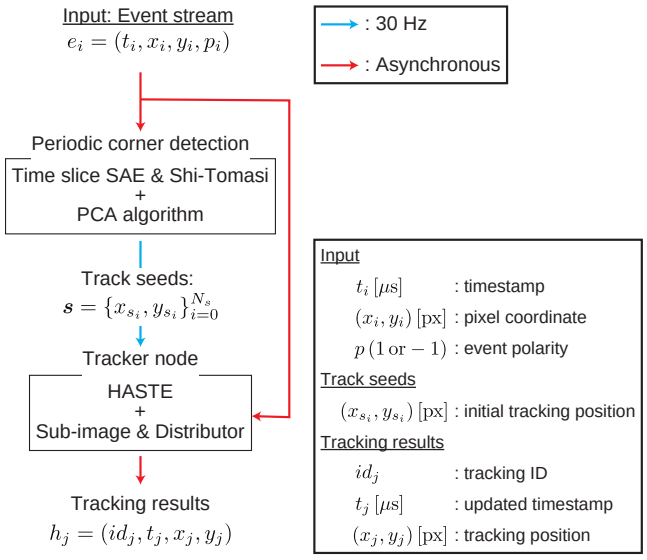


Fig. 2: Overview of proposed real-time pipeline RATE for continuous event-based tracking of multiple features

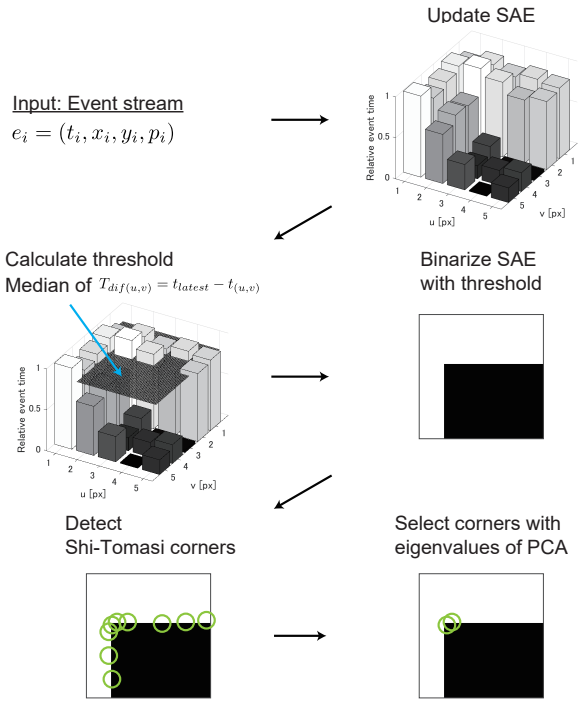


Fig. 3: Procedure of corner detector for tracking initialization by using a time slice of the Surface of Active Events (SAE)

where each pixel stores the timestamp of the latest event occurrence at that pixel. The SAE is updated with each event. An example of a SAE is shown at the top right of Fig. 3, with recent events depicted as light gray bars (independent of polarity). To apply the Shi-Tomasi corner detector [28], the SAE is periodically binarized at 30 Hz with a threshold determined by the median of timestamp differences $T_{dif(u,v)} = t_{latest} - t_{(u,v)}$ between the latest event and the events recorded at each pixel. Consequently, the bright pixels in the binarized SAE correspond to recent events, while the dark pixels correspond to old events. Due to the noisy nature of the event stream and the SAE binarization process, corners from the Shi-Tomasi detector might not be reliable for HASTE's patch-based tracking. Accordingly, after Shi-Tomasi corner detection on the binarized SAE, the PCA algorithm is applied to the neighborhood of each detected corner to evaluate if the surrounding features are on a line or not, based on the eigenvalues of PCA. The features on lines are considered less informative due to the potential linear shift during event-based feature tracking. Therefore, the features identified as being on lines through PCA evaluation are discarded, while the remaining features are published as track seeds to the tracker node.

B. Tracker node with sub-image and distributor

The proposed pipeline RATE utilizes HASTE [8], an asynchronous event-based feature tracker. Each feature in HASTE is initialized with a tracking state $\mathbf{x} = \{x, y, \theta\}$ and a template patch $\mathcal{T}$ configured by a small event window $\mathcal{E}$ of the latest m events. With each event within a certain range, the current event window is updated and the tracking state is calculated according to Eq. (1), which consists of the correlation called "alignment score f" between the current event window $\mathcal{E}$ and the template patch $\mathcal{T}$ for a set of discrete hypothetical states $\mathcal{H}$,

$$\mathbf{x}^{(k+1)} = \underset{\mathbf{x} \in \mathcal{H}(\mathbf{x}^{(k)})}{\arg\max}\, f\left(\mathcal{E}^{(k+1)}, \mathcal{T}^{(k+1)}, \mathbf{x}\right), \qquad (1)$$

where

$$\mathcal{H}(\mathbf{x}) = \{(\mathbf{x} = \{x, y, \theta\}) \cup \{x \pm \Delta x, y \pm \Delta y, \theta \pm \Delta\theta\}\}. \quad (2)$$

Actual values are $\Delta x = \Delta y = 1.0$, and $\Delta\theta = 4.0°$. Moreover, perturbations are performed separately for translation and rotation. This means that when Δx and Δy are 1.0, $\Delta\theta$ is 0. Similarly, when $\Delta\theta$ is $4.0°$, Δx and Δy are 0. Therefore, the number of alignment scores $f(\mathcal{E}, \mathcal{T}, \mathbf{x})$ to be evaluated is 11 based on the hypothetical states $\mathcal{H}(\mathbf{x})$.

To facilitate multiple real-time tracking tasks by distributing and reducing unnecessary multiple HASTE trackers, we propose a "tracking manager" consisting of *sub-images* and a *distributor* as shown in Fig. 4. The sub-image dimensions are determined based on the template patch size used in HASTE. Each tracker is assigned to a specific sub-image for updates. For each event, the distributor dynamically regulates the number of trackers assigned to each sub-image during updates, as shown in Fig. 5. During tracking updates, HASTE trackers can be re-initialized with track seeds from the corner detector node in case no tracker exists in a sub-image. Subsequently, the positions of trackers within the distributor are adjusted based on the updated tracking states of all HASTE trackers triggered by a single event. Finally, two evaluation steps are performed to reduce unnecessary trackers in all sub-images. In the first step, trackers evaluated

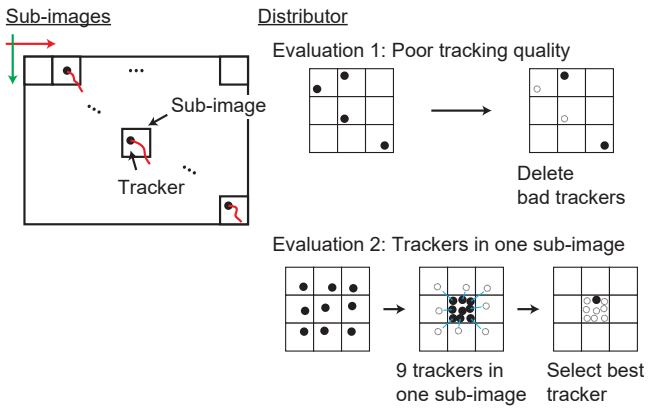


Fig. 4: Grid-based distributor of proposed RATE pipeline. The distributor manages the number of trackers especially in two cases: (i) tracking quality is evaluated as poor, (ii) more than a single tracker is in a sub-image after an update

as poor quality according to Eq. (3) are removed,

$$\frac{\max\limits_{\mathbf{x}\in\mathcal{H}(\mathbf{x})} f\left(\mathcal{E},\mathcal{T},\mathbf{x}\right) - \min\limits_{\mathbf{x}\in\mathcal{H}(\mathbf{x})} f\left(\mathcal{E},\mathcal{T},\mathbf{x}\right)}{\max\limits_{\mathbf{x}\in\mathcal{H}(\mathbf{x})} f\left(\mathcal{E},\mathcal{T},\mathbf{x}\right)} < \text{threshold} \ . \quad (3)$$

This is motivated by the finding that, when tracking quality deteriorates, the range between the best and worst scores narrows. In the second step, in case multiple HASTE trackers occupy the same sub-image S_i, the tracker with the best score $\mathrm{T}_{\mathrm{best}}$ is retained by using one of the evaluation functions described in Eq. (4), Eq. (5), and Eq. (6), while others are removed,

$$\mathrm{T}_{\mathrm{best}} = \underset{\mathrm{T_k}\in S_i}{\arg\max} \left(\max_{\mathbf{x_k}\in\mathcal{H}(\mathbf{x_k})} f\left(\mathcal{E},\mathcal{T},\mathbf{x_k}\right) \right), \quad (4)$$

$$\mathrm{T}_{\mathrm{best}} = \underset{\mathrm{T_k}\in S_i}{\arg\max} \left(ls_k \right), \quad (5)$$

$$\mathrm{T}_{\mathrm{best}} = \underset{\mathrm{T_k}\in S_i}{\arg\max} \left(ls_k \cdot \max_{\mathbf{x_k}\in\mathcal{H}(\mathbf{x_k})} f\left(\mathcal{E},\mathcal{T},\mathbf{x_k}\right) \right). \quad (6)$$

Eq. (4) selects the tracker with the highest alignment score f. On the other hand, Eq. (5) favors the tracker with the longest lifespan ls. Finally, Eq. (6) takes not only the higher alignment score but also longer lifespan into account by simply multiplying them. Future work may lead to more sophisticated combinations of both criteria.

In cases where tracking is terminated or assessed as poor quality by the tracking manager, a new tracker is generated using the latest track seed from the corner detector node, ensuring the continuity of tracking. As the changes in position based on the hypothetical states are sufficiently small compared to the size of sub-images, it can be assumed that the maximum number of trackers within a sub-image following an update can reach a total of 9, as shown in the right side of Fig. 4.

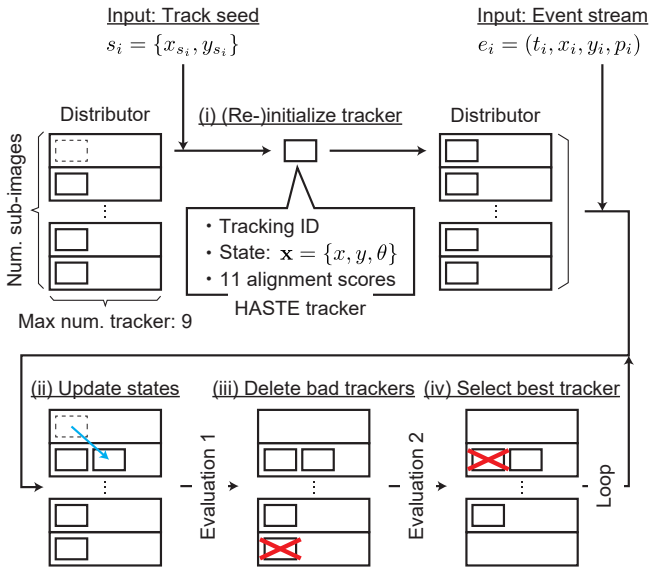


Fig. 5: Real-time tracking update by regulating the number of trackers in each sub-image with distributor. The main steps are: (i) (re-)initialize inactive trackers with track seeds from corner detector node for sub-images without feature tracks (if a seed is available for the considered sub-image), (ii) update state of trackers for each event and determine trackers for each sub-image, (iii) terminate feature tracks of poor quality, (iv) select best tracker for each sub-image. The process from (ii) to (iv) is repeated for each event, while the process (i) intervenes in the loop process based on the distributor.

IV. EXPERIMENTS

A. Experimental configuration

We compare our proposed tracking pipeline RATE with four methods: EKLT [15], Zhu'17 [24], HASTE-100, and HASTE-20. HASTE-100 and HASTE-20 are straightforward extensions of HASTE [8] to multi-feature tracking that we implemented. Using the respective open-source codes[1,2,3,4], the methods were integrated in a ROS environment, except Zhu'17[†]. EKLT leverages both frames and events for enhanced tracking accuracy, while Zhu'17 and HASTE focus solely on event data. Table I provides information about detector, tracker, and tracker initialization frequency used for each method to evaluate continuous real-time tracking. EKLT utilized Harris corner detector on frames for initialization of tracking. In all other methods, tracking is initialized using events only. Zhu'17 uses accumulated events, while both HASTE and RATE rely on the SAE as event-based image representations, and apply Harris or Shi-Tomasi corner detector to these images. In RATE, as described in section III, PCA is used to remove features on lines and a tracking

[1]https://github.com/uzh-rpg/rpg_feature_tracking_analysis
[2]https://github.com/uzh-rpg/rpg_eklt
[3]https://github.com/daniilidis-group/event_feature_tracking
[4]https://github.com/ialzugaray/haste
[†]The published code is implemented in MATLAB and has no optimization in terms of computational cost for real-time process.

TABLE I: Detectors and Trackers in all methods for evaluation of continuous real-time tracking

Method	Data	Detector	Tracker	(Potential) Tracker initialization frequency
EKLT	Frames & Events	Frame & Harris	EKLT	Only first frame
Zhu'17	Events	Accumulated events & Harris	Zhu'17	Whenever number of trackers reaches 0
HASTE-100	Events	Time slice SAE & Shi-Tomasi	HASTE + Maximum number: 100	Every event stream (30 Hz)
HASTE-20	Events	Time slice SAE & Shi-Tomasi	HASTE + Maximum number: 20	Every event stream (30 Hz)
Ours: RATE	Events	Time slice SAE & Shi-Tomasi + PCA	HASTE + Sub-image & Distributor	Every event stream (30 Hz)

manager is deployed. The default function for evaluation 2 (see Fig. 4) is Eq. (4), and the threshold in Eq. (3) is set to 0.1. We analyze the sensitivity of RATE to these parameters with respect to tracking accuracy and lifespan in Section IV-C. As baselines for our RATE, HASTE-100 and HASTE-20 have the number of trackers limited to 100 and 20 respectively. These methodologies were evaluated on the Event Camera Dataset [14], focusing on the"boxes_6dof" and "bicycles" datasets. The event data stream in these datasets is published as variable-length arrays of events at around 30Hz. The resolution of the event-based camera in these datasets is 180×240; HASTE patch size and sub-image dimensions are set to 30×30. Consequently, RATE can accommodate a maximum of 48 HASTE trackers. To analyze the tracking results more comprehensively, we modified the code of the "rpg_feature_tracking_analysis" repository [29] to evaluate the number of continuous trackers, real-time performance, feature distribution, tracking error, and lifespan across various tracking runs starting at different timestamps of the data sets. The evaluation setup included an AMD Ryzen 9 5900X Processor, 32 GB RAM running at 2667 MT/s memory speed, and each node was executed on a single CPU core.

B. Qualitative evaluation

We qualitatively evaluated the continuous real-time tracking behavior in three methods - HASTE-100, HASTE-20, and RATE. Figure 1 displays the tracking result of three methods for the dataset "boxes_6dof" at a specific time ($t = 5.13$ s). The timeline of these images was managed by the ROS master, hence any delays of published data relative to the ROS master are reflected in these figures. The two sub-figures in the top row show spatially clustered and overlapping features for HASTE-100 and HASTE-20, while RATE distributes trackers throughout the entire field of view, ensuring the absence of redundant feature tracks and a better spatial coverage, which is known to be beneficial for reliable ego-motion estimation.

Figure 6 shows another tracking example for the same dataset as Fig. 1. The orange arrow indicates the ground truth movement of objects towards the bottom left in frames. The directions of tracked seeds in HASTE-20 and RATE, illustrated by green lines, are closely aligned with the ground truth (orange arrow). However, the direction shown by the blue arrow, representing the movement of a cluster of tracked seeds in HASTE-100, differs significantly from ground truth. This behavior was qualitatively attributed to a tracking delay in HASTE-100, as 100 trackers can not be handled in real-time in the setup of HASTE-100.

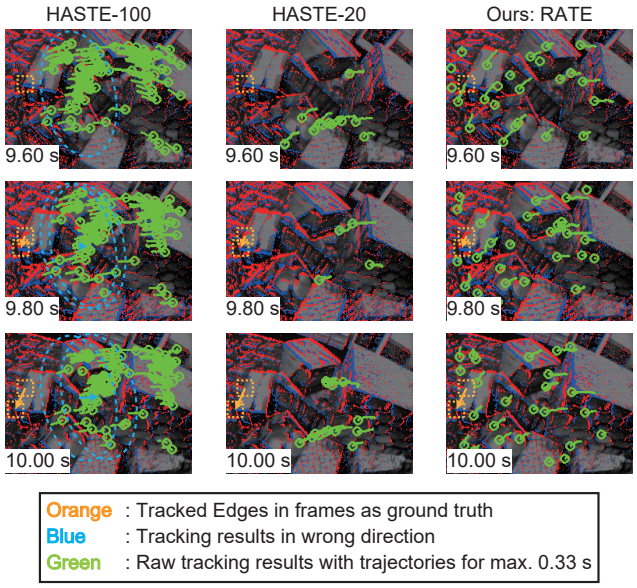


Fig. 6: Comparison of feature tracks for HASTE-100, HASTE-20 and RATE ($t = 9.60, 9.80, 10.00$ s)

C. Quantitative evaluation

We quantitatively evaluated the number of trackers, real-time performance, feature distribution, tracking errors, and lifespans in five methodologies - EKLT, Zhu'17, HASTE-100, HASTE-20, and RATE - on the dataset "bicycles" spanning about 24 seconds. We analyzed the correlation between the timestamp and the number of trackers per update in each methodology, as illustrated in Fig. 7. The behavior observed varies due to the different tracker initialization frequency as described in Table I. In the original implementation, EKLT initializes tracking by employing Harris corner detection only on the first frame with predefined maximum number of detected corners of 100. This allows for an initial configuration of up to 100 trackers at the beginning. However, the number of trackers in EKLT drops to 0 within 5 seconds and does not recover due to the absence of feature

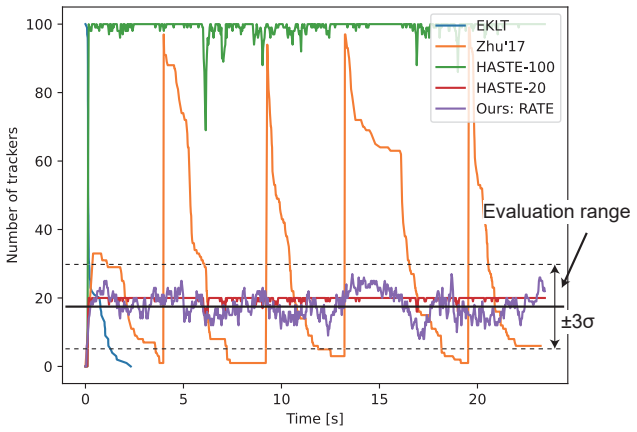


Fig. 7: Evolution of number of trackers with time in each method. The range, which was utilized to evaluate real-time performance, covers a 3σ deviation of the number of trackers in RATE.

TABLE II: Real-time performance evaluation based on RT-ratio, the ratio between total actual and processing time (limited to intervals where tracker count is within the evaluation range indicated by dashed lines in Fig. 7 for fair comparison)

Method	Total actual time [s]	Total processing time [s]	RT-ratio	Real-time
EKLT	0.93	5.58	5.99	No
Zhu'17	9.47	511.46	53.99	No
Ours: RATE	23.39	3.87	0.17	**Yes**

re-detection in the EKLT code. For Zhu'17, trackers were initialized by creating a frame of accumulated events from the first 0.001 seconds and applying Harris corner detector to it. The frame of accumulated events is generated by counting the event occurrence in each pixel during the update interval. Whenever the tracker count of Zhu'17 dropped to 0, new trackers were re-initialized to a maximum of 100 by the same process. However, similar to EKLT, the number of trackers in Zhu'17 also experienced significant reductions within about 5 seconds shortly after each re-initialization phase. For HASTE-100 and HASTE-20, new trackers are re-initialized using the most recently detected track seeds, whenever the number of trackers decreased from the maximum value, i.e. 100 and 20 respectively, which can happen due to features leaving the FOV of the camera. Similarly, RATE initializes HASTE trackers by employing the Shi-Tomasi corner detector at a fixed frequency of 30 Hz on a time slice of the SAE. However, when multiple trackers occupy the same sub-image or the tracking quality is evaluated as poor, redundant trackers are removed and new trackers re-initialized with track seeds from the corner detector node. This process is repeated continuously for each event in the event stream at a high frequency. Consequently, the tracker count increases gradually at the beginning and stabilizes at approximately 20 until the end of the dataset.

With the results obtained from the evaluation of the number of trackers, we analyzed the real-time performance

TABLE III: Real-time performance evaluation based on RT-ratio over the whole sequence

Method	Total actual time [s]	Total processing time [s]	RT-ratio	Real-time
HASTE-100	23.45	31.78	1.36	No
HASTE-20	23.45	5.96	0.25	Yes
Ours: RATE	23.45	3.87	0.16	**Yes**

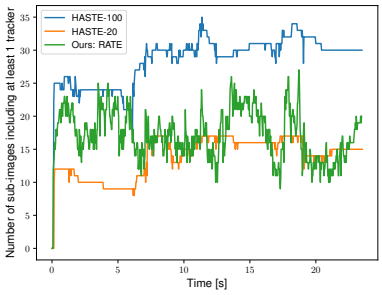
of each method by calculating the "RT-ratio", defined as the ratio of total processing time to actual time as shown in the following Eq. (7),

$$\text{RT-ratio} = \frac{\text{Total processing time [s]}}{\text{Total actual time [s]}} . \tag{7}$$

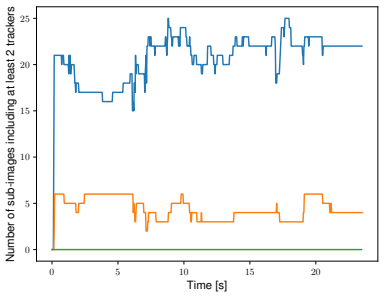
RT-ratios less than 1.0 indicate real-time processing. In particular, as indicated by standard deviations of RT-ratios greater than 100 (EKLT: 134.4, Zhu'17: 108.1, respectively), both EKLT and Zhu'17 exhibited significant variations in RT-ratios depending on the number of trackers (0 - 100) because of the difference in tracker initialization frequency. To ensure a fair comparison of processing times among EKLT, Zhu'17, and RATE, we constrained both total actual and processing times within the 3σ range of the number of trackers in RATE, defined as "evaluation range" ([5.81, 29.59], mean 17.70), as shown in Fig. 7. As detailed in Table II, the RT-ratios for EKLT and Zhu'17 exceed real-time constraints (values > 1.0) with values of 5.99 and 53.99, respectively. In contrast, the RT-ratio of RATE was 0.17, demonstrating the achievement of real-time performance during operation. In the case of HASTE-100, HASTE-20, and RATE, as shown in Fig. 7, the number of trackers remains stable throughout the tracking process. Consequently, we evaluated the impact of the number of trackers on real-time performance by comparing RT-ratios across these methods calculated without any range limitations. Table III indicates that HASTE-20 and RATE can operate in real time due to sufficiently low numbers of trackers, in contrast to HASTE-100. In summary, the results concerning the number of trackers and real-time performance confirmed that RATE successfully achieved continuous, real-time event-based asynchronous tracking of multiple features.

To evaluate the spatial distribution of trackers across the entire field of view, we counted the number of sub-images containing trackers as shown in Fig. 8. A comparison of three graphs in Fig. 8 revealed that multiple trackers in HASTE-100 and HASTE-20 were located within the same sub-image, resulting in tracking of closely spaced features. In contrast, all trackers in RATE were consistently positioned within their respective sub-images due to the proposed tracking manager. Therefore, these results indicate that RATE efficiently distributes around 20 trackers spatially, a similar number of trackers as in HASTE-20.

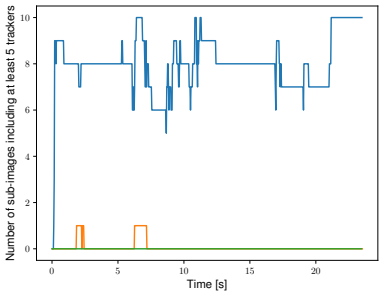
We used the first 30 trackers in each methodology to fairly compare average errors and lifespans because (i) the timings of the first tracker initialization were the same, and (ii) the number of trackers in Zhu'17 was initialized to about 30 and did not recover until it reached 0. The "rpg_feature_tracking_analysis" [29] was selected as a KLT-based tracking method on intensity frames to compute aver-



(a) Number of sub-images including at least 1 tracker



(b) Number of sub-images including at least 2 tracker



(c) Number of sub-images including at least 5 tracker

Fig. 8: Number of sub-images with (a) at least 1 tracker, (b) with 2 or more, and (c) with at least 5 trackers using HASTE-100 (blue), HASTE-20 (orange) and RATE (green). In contrast to HASTE-20 and HASTE-100, which can have 2 or more tracker per sub-image, RATE has at most 1 tracker per sub-image, leading to a more evenly spread distribution of trackers.

age errors and lifespans because of the absence of the ground truth for feature tracking in the dataset [14]. Similar to [25], good performance of the KLT tracker on intensity frames was ensured by utilizing the dataset "bicycles" spanning about 24 seconds and the first 25 seconds of the dataset "boxes_6dof" as they have less fast motion and motion blur compared to other datasets. The results in Table IV reveal that event-only methods, including RATE, exhibit significantly lower performance in lifespan compared to EKLT, which uses both frames and events for tracking. However, at least for the dataset "boxes_6dof", RATE achieves similar performance in tracking accuracy as EKLT, indicated by the p-value of

TABLE IV: Average error and lifespan of tracked features evaluated using the first 30 trackers, and p-values of t-Tests between RATE and the others with alpha level of 5 %

Dataset "bicycles" (24 s)

Method	Error		Lifespan	
	Average [px]	p-value [%]	Average [s]	p-value [%]
EKLT	0.67	5.20×10^{12}	1.50	1.84×10^{8}
Zhu'17	4.24	1.24×10^{4}	1.42	3.81×10^{8}
HASTE-100	4.23	3.44×10^{6}	0.60	1.52×10^{11}
HASTE-20	3.98	1.38×10^{4}	0.45	4.26×10^{3}
Ours: RATE	2.90	–	0.32	–

Dataset "boxes_6dof" (first 25 s [25])

Method	Error		Lifespan	
	Average [px]	p-value [%]	Average [s]	p-value [%]
EKLT	0.83	83.4	1.38	4.49×10^{1}
Zhu'17	3.34	2.32×10^{14}	1.94	1.14×10^{3}
HASTE-100	1.98	1.91×10^{4}	2.76	2.29×10^{8}
HASTE-20	3.39	1.73×10^{13}	2.11	2.57×10^{4}
Ours: RATE	0.79	–	0.76	–

TABLE V: Sensitivity evaluation in RATE with different thresholds and selection functions by using average error and lifespan of tracked features of the first 30 trackers, and p-values of t-Tests with alpha level of 5 %

Dataset "boxes_6dof" (first 25 s [25])

Threshold	Eval.	Error		Lifespan	
	Eq.	Average [px]	p-value [%]	Average [s]	p-value [%]
0.1	(4)	0.79	–	0.76	–
0.5	(4)	0.68	56.9	0.59	30.1
0.02	(4)	0.61	30.0	0.76	100
0.1	(5)	1.70	4.27×10^{2}	1.22	5.01
0.1	(6)	1.62	7.94×10^{2}	1.17	7.13

more than 80 %. In addition, given that the p-values in errors between RATE and other event-only baselines are much smaller than 5 %, it is evident that RATE enhances the tracking accuracy compared to the event-only baselines. This improvement can be attributed to two main factors: (i) the improvement of corner accuracy through PCA algorithm during initialization, and (ii) the removal of poorly performing trackers as illustrated in Fig. 4. On the other hand, considering p-values in lifespans, the feature lifespan for RATE was statistically significantly the shortest among all methods. This can be explained by the fact that RATE deletes bad trackers using Eq. (3) and selects only the tracker with the best alignment score f for each sub-image in Eq. (4) to enhance tracking accuracy and spatial distribution, as depicted in Fig. 5.

Furthermore, we evaluated the sensitivity of parameters in RATE by changing the selection function for evaluation 2 (see Fig. 4) and the threshold in Eq. (3) as shown in Table V. The sensitivity of the threshold in Eq. (3) was considered not to be strong because tracking errors and lifespans are not significantly different according to the p-values in this table. However, average lifespan was improved but tracking accuracy significantly reduced with evaluation functions that include the lifespan value, Eq. (5) and Eq. (6). These findings indicate that RATE has the potential to optimize tracking accuracy and lifespan by designing appropriate evaluation functions.

Overall, the results in this section show that the proposed RATE pipeline can stabilize the number of trackers

effectively, and thereby achieves real-time tracking, while distributing features evenly over the FoV of the camera. In addition, a modest improvement in tracking accuracy has been observed compared to conventional event-only baselines, despite the inherent trade-off between real-time performance and accuracy.

V. CONCLUSIONS

We have introduced RATE, a continuous real-time capable pipeline for asynchronous event-based tracking of multiple features. Within this pipeline, the corner detector node, incorporating the SAE, the Shi-Tomasi corner detector, and a PCA-based selector, provides seeds for the initialization of multiple HASTE trackers. The tracking manager node, consisting of sub-images and a distributor minimizes redundant HASTE trackers after asynchronous updates. Throughout all experiments, the proposed pipeline maintained a consistent number of trackers in real-time, while exhibiting similar or slightly enhanced tracking accuracy compared to state-of-the-art event-only tracking methods.

Future work includes implementing the advancements proposed in this research within an event-driven VO architecture. This application aims at vision-based self-localization in real-world environments, particularly under challenging conditions such as drastic lighting changes and fast motions. In addition, hybrid approaches complementing events with frames that can convey rich texture and color information independent of motion, are very promising. For instance, they can provide means to tackle one of the major challenges for event-only methods, namely the re-detection of previously used features, for which tracking was lost or that temporarily moved out of the camera's field of view.

REFERENCES

[1] D. Wooden, M. Malchano, K. Blankespoor, A. Howardy, A. A. Rizzi, and M. Raibert, "Autonomous navigation for BigDog," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 4736–4741.

[2] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "ANYmal - a highly mobile and dynamic quadrupedal robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 38–44.

[3] B. Pipenberg, M. Keennon, J. D. Tyler, B. D. Hibbs, S. A. Langberg, J. Balaram, H. F. Grip, and J. Pempejian, "Design and fabrication of the mars helicopter rotor, airframe, and landing gear systems," in *AIAA Scitech Forum*, 2019.

[4] P. Lutz, M. G. Müller, M. Maier, S. Stoneman, T. Tomić, I. von Bargen, M. J. Schuster, F. Steidle, A. Wedler, W. Stürzl, and R. Triebel, "ARDEA — an MAV with skills for future planetary missions," *Journal of Field Robotics*, vol. 37, no. 4, pp. 515–551, 2020.

[5] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robotics and Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[6] S. Sumikura, M. Shibuya, and K. Sakurada, "OpenVSLAM: A versatile visual SLAM framework," in *27th ACM International Conference on Multimedia*, 2019, p. 2292–2295.

[7] X. Zhao, Z. Gao, H. Li, H. Ji, H. Yang, C. Li, H. Fang, and B. M. Chen, "How challenging is a challenge? CEMS: a challenge evaluation module for SLAM visual perception," *Journal of Intelligent & Robotic Systems*, vol. 110, no. 1, p. 42, March 9 2024.

[8] I. Alzugaray and M. Chli, "HASTE: multi-hypothesis asynchronous speeded-up tracking of events," in *31st British Machine Vision Virtual Conference (BMVC)*, 2020.

[9] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128× 128 120 db 15 μs latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[10] G. Gallego, T. Delbruck, 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. 01, pp. 154–180, 2022.

[11] H. Kim, "Real-time visual SLAM with an event camera," Ph.D. dissertation, Imperial College London, 2017.

[12] 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.

[13] R. Pellerito, M. Cannici, D. Gehrig, J. Belhadj, O. Dubois-Matra, M. Casasco, and D. Scaramuzza, "End-to-end learned event- and image-based visual odometry," 2023. [Online]. Available: https://arxiv.org/abs/2309.09947

[14] 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, p. 142–149, 2017.

[15] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza, "EKLT: Asynchronous, photometric feature tracking using events and frames," *International Journal of Computer Vision*, 2019.

[16] 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.

[17] A. Glover, A. Dinale, L. D. S. Rosa, S. Bamford, and C. Bartolozzi, "luvHarris: A practical corner detector for event-cameras," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 12, pp. 10 087–10 098, 2022.

[18] 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," in *IEEE Robotics and Automation Letters*, 2018.

[19] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision (ECCV)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443.

[20] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *International Joint Conference on Artificial Intelligence*, 1981, p. 674–679.

[21] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5863–5870.

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

[23] 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.

[24] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based feature tracking with probabilistic data association," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 4465–4470.

[25] C. L. Gentil, I. Alzugaray, and T. Vidal-Calleja, "Continuous-time gaussian process motion-compensation for event-vision pattern tracking with distance fields," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 804–812.

[26] Ö. Yılmaz, C. Simon-Chane, and A. Histace, "Evaluation of event-based corner detectors," *Journal of Imaging*, vol. 7, no. 2, 2021.

[27] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi, "Event-based visual flow," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 2, pp. 407–417, 2014.

[28] J. Shi and Tomasi, "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1994, pp. 593–600.

[29] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza, "Feature tracking analysis," 2019. [Online]. Available: https://github.com/uzh-rpg/rpg_feature_tracking_analysis