IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Event-Frame-Inertial Odometry Using Point and Line Features Based on Coarse-to-Fine Motion Compensation

Byeongpil Choi, Hanyeol Lee, and Chan Gook Park

***Abstract*—An event camera is a vision sensor that captures pixel-level brightness changes and outputs this information as asynchronous events. These events are primarily generated from geometric structures such as edges, which are sensitive to variations in brightness. In this letter, we aim to leverage this line structure information alongside point features to enhance the robustness and accuracy of localization in indoor or human-made environments. To obtain precise line measurements from events, we propose a novel line detection method that incorporates a coarse-to-fine motion compensation scheme, which generates highly sharp event frames. The extracted line features are paired with point features, eliminating the need for traditional line descriptors. Finally, the event features are effectively fused with frame-based point features within a multi-state constraint Kalman filter-based backend, fully exploiting the complementary advantages of both sensors. The performance of the proposed method is verified through an author-constructed experiment and two public datasets, demonstrating improved accuracy in line detection and pose estimation. Open-source implementation is available at: https://github.com/choibottle/C2F-EFIO.**

***Index Terms*—Localization, SLAM, Visual-inertial odometry (VIO), Sensor fusion, Computer vision**

I. Introduction

In ROBOTICS, the perception of the surrounding environment and accurate state estimation are crucial tasks across a wide range of applications. Visual-Inertial Navigation Systems (VINS) have been extensively researched for its advantages, such as low-cost sensors and the integration of Inertial Measurement Units (IMU), which enhance accuracy, robustness, and enable metric scale recovery in monocular setups, thereby improving the performance of Visual-Inertial Odometry (VIO) [1], [2]. However, due to the nature of standard cameras, which output images at a fixed frame rate, fast motion and High Dynamic Range (HDR) are challenging scenarios for the state estimation, and motion blur or low dynamic range can cause the system to degrade or fail.

The Dynamic Vision Sensor (DVS), also called event camera has significant potential to address these issues. Compared to standard cameras, event cameras, which asynchronously output the information of individual pixels when brightness changes are detected, have the following advantages: very low latency, high dynamic range, and high pixel bandwidth [3]. However, event cameras also have notable limitations, such as the lack of signal when there is no motion, which can be a significant drawback in static scenes. These characteristics underscore the need for fusion with standard cameras. Sensors like the Dynamic and Active Pixel Vision Sensor (DAVIS) [4] can generate both frame and event data, enabling the combination of both advantages. Leveraging this fusion enhances the robustness and accuracy of state estimation, especially in challenging scenarios with minimal motion, very fast motion, or HDR conditions.

Regardless of frame- or event-based approaches, the majority of VIO methods rely on point features, which are rich in natural scenes, while research using line features is relatively rare. However, in indoor or human-made environments where geometric structures are prevalent, line features are also abundant. Therefore, incorporating line features with point features can provide additional geometric constraints. Several studies have been proposed to combine point and line features to improve the accuracy and robustness of navigation [5]- [7]. PL-EVIO [8], one of the few event-based VIO (EVIO) that combines point and line features, has demonstrated outstanding performance by effectively applying the VIO with point and line features (PL-VIO) framework, previously studied in traditional visual navigation, to event cameras. [8] utilized the Line Segment Detector (LSD) [9], a popular line detector in traditional computer vision, to extract line features. Although LSD is effective in the image domain, it does not fully leverage the unique characteristics of events, making it a suboptimal approach for event cameras.

In this letter, we propose a novel approach to extract accurate line features from event streams leveraging the event characteristics. The key elements include precise motion compensation via a coarse-to-fine scheme and robust line detection utilizing event intensity—the occurrence frequency of events at each pixel. This consideration of event intensity enhances robustness to noise and outliers, which are fundamental challenges due to the high sensitivity of event cameras. The event-based point and line features are then fused with frame-based point features within a Multi-State Constraint Kalman Filter (MSCKF) framework [10], employing an adaptive synchronization scheme. Our main contributions are summarized as follows:

Manuscript received: September, 19, 2024; Revised December, 16, 2024; Accepted January, 20, 2025.

This paper was recommended for publication by Editor P. Vasseur upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the National Research Foundation of Korea funded by the Ministry of Science and ICT, the Republic of Korea, under Grant NRF-2022R1A2C2012166. *(Corresponding author: Chan Gook Park.)*

All authors are with the Department of Aerospace Engineering, Automation and Systems Research Institute, Seoul National University, Seoul 08826, Republic of Korea (e-mail: bpc1224@snu.ac.kr; han2110@snu.ac.kr; chanpark@snu.ac.kr).

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

- We present a novel line detection method that achieves accurate line measurements in an event frame by utilizing a coarse-to-fine motion compensation scheme.
- The extracted line features are paired with point features, effectively eliminating the need for line descriptors and resolving data association problem.
- The event features are then fused with frame point features using an adaptive synchronization scheme, ensuring consistent feature tracking within an MSCKF-based estimator backend.
- The proposed method is evaluated on a real experiment and two challenging datasets comprising 19 sequences. The performance of line detection and pose estimation is validated both quantitatively and qualitatively.

This work builds on our previous study [11], where we initially explored event line detection with coarse-to-fine motion compensation, demonstrating improved localization using event features alone. In this letter, we present the completed and extended version of that study by integrating frame features into a full hybrid system.

II. Related Work

A. Event-based pose estimation and line detection

Pose estimation using the EVIO framework has been extensively studied in various works [12]- [16]. In [13], events were accumulated in spatio-temporal windows, and motion-compensated event frames were synthesized using IMU measurements. Subsequently, conventional feature tracking, employing methods such as the FAST corner detector [17] and Lukas-Kanade tracker [18], was performed. Furthermore, [14] proposed an extended system that integrates features tracked independently from frames. The recent study in [16] expanded the EVIO framework to stereo setups, unlocking the potential for more robust and accurate pose estimation beyond monocular configurations.

As event-based pose estimation has advanced through the use of point features, research on event-based line detection has also evolved [8], [19]- [23]. For instance, [20] treated events as 3D points and computed their normal vectors to develop a line detection algorithm. In [23], plane fitting on 3D event points was employed to extract edge lines, though this method primarily focused on object pose estimation. In [8], LSD was applied to motion-compensated event streams to detect line features, demonstrating the effective application of traditional techniques in event-based vision.

Our proposed approach shares similarities with [13] and [14] in synthesizing motion-compensated event frames from accumulated events in a spatio-temporal window; however, they exclusively rely on point features. On the other hand, our method has similarities with [8] in extracting both point and line features from event frames. However, unlike [8], which simply applies LSD to event streams roughly motion-compensated by IMU measurements, our method aims to outperform [8] by extracting line features from more precisely motion-compensated event frames, considering the characteristics of events.

B. Motion compensation in event cameras

Motion compensation refers to warping events occurring at different timestamps to a reference time, taking into account the motion during that period. A motion can be estimated by an objective function that aligns the events, and the more accurate this motion is, the sharper the event frame produced. In [24], an optimization framework was proposed to estimate motion parameters that maximize the contrast (i.e., variance) of the generated event frame. Additionally, [25] proposed 22 objective functions for motion compensation. In other cases, motion compensation is often implemented within the EVIO framework. In [12], motion-compensated event point sets were generated using the EM algorithm. Similarly, in [13] and [14], sharp event frames were synthesized using IMU measurements and linearly interpolated pixel depth information. In [8] and [16], both rotation and translation were similarly compensated using IMU measurements.

However, since the depths of the events are unknown, it is impossible to accurately compensate for translation using IMU measurements alone. Moreover, applying Contrast Maximization (CM) to an entire event frame is challenging since the motion of events varies locally. To address these issues, we propose a coarse-to-fine motion compensation approach. Instead of estimating the inaccurate depths of events, our method compensates for rotation coarsely using IMU on the entire event frame. Subsequently, fine motion compensation is performed using CM within local windows, resulting in highly sharp and accurate event frames.

III. Line detection from event streams

In this section, we present our coarse-to-fine motion compensation and robust line detection algorithm based on our previous work [11]. The events are accumulated according to an adaptive number, as introduced in Section IV-A, and then motion-compensated precisely in a coarse-to-fine manner.

A. Coarse-to-fine motion compensation

An i-th event, which occurs when the brightness change of a pixel exceeds a threshold, is expressed as

$$e_i = [x_i \quad y_i \quad t_i \quad p_i]^T \tag{1}$$

where $[x_i \quad y_i]^T$ represents the 2D pixel coordinates in the image plane, t_i denotes the timestamp, and p_i is the polarity which takes on a value of either $+1$ or -1, corresponding to the direction of the brightness change.

Coarse motion compensation: Events accumulated over a brief time period are warped to a reference time t_{ref}. Given that the depths of these events are unknown, only the sensor's rotation is coarsely compensated utilizing the angular velocity obtained from IMU. The pixel coordinates of a warped event are calculated as

$$\begin{bmatrix} x_{ref} \\ y_{ref} \\ 1 \end{bmatrix} = \pi \left(R_{C_i}^{C_{ref}} \left(\pi^{-1} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \right) \right) \tag{2}$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

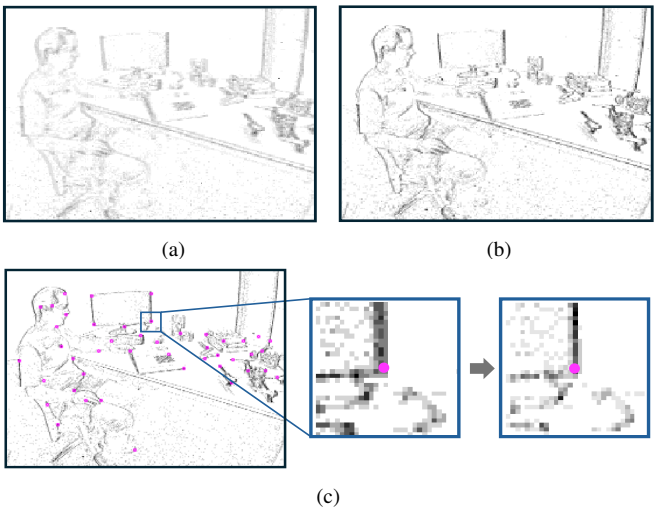

Fig. 1. Coarse-to-fine motion compensation. (a), (b) Event frames before and after rotation compensation. (c) Contrast maximization in a local window around the detected corner (magenta).

where $\pi(\cdot)$ is the camera projection function, $R_{C_i}^{C_{\text{ref}}} = \exp\left(\lfloor \boldsymbol{\omega}_{\text{m}} \cdot \Delta t \rfloor_{\times}\right)$ represents the rotation matrix between the camera frames at time t_i and t_{ref}. During event accumulation, angular velocities are sampled, and the mean angular velocity, $\boldsymbol{\omega}_{\text{m}}$, is computed under the assumption of uniform motion. The time difference Δt for each event is calculated independently to ensure precise synchronization, and $\lfloor \cdot \rfloor_{\times}$ denotes the skew-symmetric matrix form of a vector. Fig. 1(a) and (b) illustrate the entire event frames before and after rotation compensation, respectively.

Fine motion compensation: In the event frame, where rotation has been compensated, point features are extracted using the FAST corner detector. Local windows are then established around each detected corner point. Within these windows, fine motion compensation is carried out by estimating the optical flow $\mathbf{v}$ of events using CM as

$$\mathbf{v}^* = \underset{\mathbf{v}}{\text{argmax}}\ \sigma^2(H(\mathbf{v})) = \underset{\mathbf{v}}{\text{argmax}}\ \frac{1}{N_p}\sum_{x,y}(h_{xy} - \mu_H)^2 \quad (3)$$

where σ^2 represents the variance of the number of accumulated events at each pixel, used as the objective function. $H(\mathbf{v})$ denotes the event frame generated using $\mathbf{v}$, N_p is the total number of pixels within the window, h_{xy} is the count of accumulated events at a single pixel, and μ_H denotes the mean of H [24]. As σ^2 is maximized, the event frame becomes sharper. Fig. 1(c) illustrates the CM process within a window.

To perform CM, it is necessary to assume that optical flow within a spatio-temporal window is constant, which is only feasible around local corners, not the entire event frame. Therefore, we opt for a localized approach using small windows. Furthermore, since CM involves nonlinear optimization, it is sensitive to initial values, which can lead to undesired local optima and longer convergence times if inaccurate. To address this, we leverage the pixel displacements of each corner point between two consecutive event frames as the initial values for CM within each window as

$$\mathbf{v}_{\text{init}} = \frac{\mathbf{f}_k - \mathbf{f}_{k-1}}{t_k - t_{k-1}} \quad (4)$$

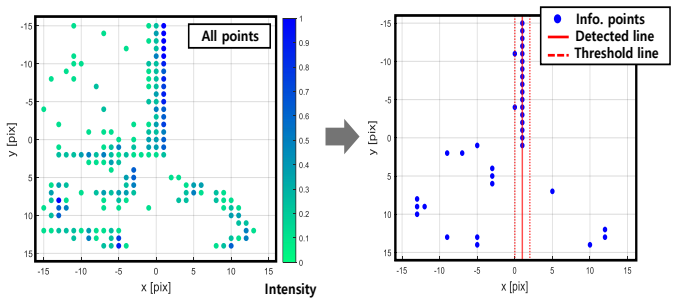

Fig. 2. Line detection in a 2D event point set. A line (solid red) is detected through RANSAC on informative event points (blue) filtered by intensity.

where $\mathbf{f}_k$ and t_k represent the pixel coordinates and timestamps of the corner point in the k-th sequence, respectively. By initializing with $\mathbf{v}_{\text{init}}$, the optimization process starts from a point closer to the global optima, reducing the risk of converging to local optima compared to starting with no initial estimate. This ensures faster and more reliable convergence, ultimately improving the overall stability.

B. Robust line detection in the sharp event frame

After motion compensation, each pixel in the event frame is treated as an 2D event point. The intensity $I(x, y)$ at each pixel is defined as the normalized occurrence frequency of events at that pixel, represented by h_{xy}, divided by the maximum event count $h_{\max}$ across the frame. Informative points, those with intensity values exceeding a threshold ϵ_i, set to 0.4, are then selected to exclude noisy points as

$$\mathcal{P}_{\text{info}} = \left\{ (x, y) \mid I(x, y) = \frac{h_{xy}}{h_{\max}} > \epsilon_i \right\} \quad (5)$$

Subsequently, lines are fitted through RANSAC from $\mathcal{P}_{\text{info}}$. Since RANSAC is effective in filtering outliers, it can robustly detect lines even in the presence of sensor noise or non-significant points in terms of line features. If the inlier ratio of the fitted line exceeds a specified threshold ϵ_r, set to 0.4, the line feature is considered successfully detected. Then, the endpoints of the inlier points, defined as the two points with the smallest and largest pixel coordinates, are projected onto the detected infinite line to determine $\mathbf{x}_s$ and $\mathbf{x}_e$, which represent the start and end points of the line, and will be used in the subsequent line measurement model. Fig. 2 illustrates this robust line detection process.

One of the advantages of our method is that there is no need for a data association process to match the extracted line features in consecutive sequences. Since the line features extracted from the windows correspond to each point feature, the descriptors of the point features can be directly utilized for line association. Therefore, the process of computing line descriptors such as the Line Band Descriptor (LBD) [26] and matching based on them is omitted, and instead, only the following outlier rejection tests are performed:

- The pixel distance between the mid-point of a detected line and another line in a consecutive event frame should be no more than ϵ_{px}, set to 10 pixels.
- The angle difference between the lines in two consecutive event frames should be no more than ϵ_{ang}, set to 5 degrees.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

IV. Estimator Framework

In this section, we present our MSCKF-based estimator framework. Tracked event features are effectively synchronized with frame-based point features and tightly fused with IMU measurements.

A. Event-frame synchronization

To synchronize event features with frame features, it is efficient to accumulate events based on the timestamps of the frame camera to form an event frame. Since the number of events between two timestamps typically varies, the number of events to accumulate should be determined adaptively rather than using a fixed number, as described in [14]. Using a fixed number can result in either the loss of events or their redundant use, leading to failures in feature tracking within the event frame. Therefore, to ensure continuous feature tracking and utilize all information without redundancy, the number of events to accumulate is determined adaptively.

First, similar to [12], we estimate the lifetimes of event point features to move a certain number of pixels and calculate N_{exp}, the expected number of events during this movement. For detailed calculations, please refer to [12]. Meanwhile, the actual number of events M occurring between two timestamps varies according to scene dynamics. To ensure robustness against such variations, we compare N_{exp} and M to adapt the accumulation process as follows:

- If $N_{exp} > M$, all events are accumulated into a single event frame.
- Otherwise, the number of event frames n_f is set to round (M/N_{exp}), and the number of events to accumulate per event frame N_{acc} is calculated as round (M/n_f).

This adaptive accumulation prevents excessively large pixel displacements between consecutive event frames, particularly when scene dynamics are high, by forming multiple event frames. This ensures that feature displacements remain within manageable limits, reducing the risk of feature tracking failure.

When multiple event frames are generated between two timestamps, event point and line features are tracked across all the event frames independently of measurement updates. Subsequently, the tracked event features are fused with frame point features and utilized for MSCKF updates at the frame timestamps. The process is illustrated in Fig. 3.

B. MSCKF using point and line features

The MSCKF backend is comprised of the IMU state and several sliding window states. The IMU error-state can be defined as

$$\widetilde{\mathbf{X}}_I = \begin{bmatrix} \boldsymbol{\delta\theta}_I^{G\,T} & \widetilde{\mathbf{b}}_g{}^T & \widetilde{\mathbf{v}}_I^{G\,T} & \widetilde{\mathbf{b}}_a{}^T & \widetilde{\mathbf{p}}_I^{G\,T} \end{bmatrix}^T \tag{6}$$

where $\boldsymbol{\delta\theta}_I^G$ is the error rotation vector, $\mathbf{v}_I^G$ and $\mathbf{p}_I^G$ are the velocity and position of the IMU in the global frame, and $\mathbf{b}_g$ and $\mathbf{b}_a$ represent the biases of the gyroscope and accelerometer, respectively. The definition of $\boldsymbol{\delta\theta}_I^G$ follows [10], while the other errors are defined as $\tilde{x} = x - \hat{x}$.

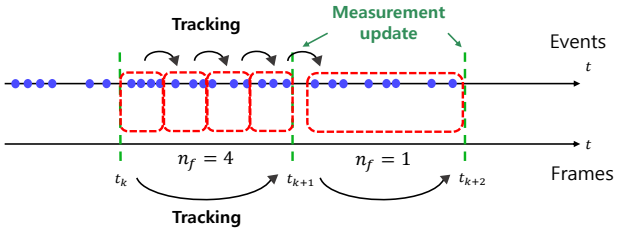

Fig. 3. Event features are tracked across all event frames (red) generated from events (blue). These features are then fused with frame features and utilized for measurement updates at the timestamps of the frame (green).

At time k, the full state vector, comprising the IMU error-state $\widetilde{\mathbf{X}}_{I_k}$ and the last N sliding window pose states, is described as

$$\widetilde{\mathbf{X}}_k = \begin{bmatrix} \widetilde{\mathbf{X}}_{I_k}{}^T & \boldsymbol{\delta\theta}_{I_1}^{G\,T} & \widetilde{\mathbf{p}}_{I_1}^{G\,T} & \cdots & \boldsymbol{\delta\theta}_{I_N}^{G\,T} & \widetilde{\mathbf{p}}_{I_N}^{G\,T} \end{bmatrix}^T \tag{7}$$

where $\boldsymbol{\delta\theta}_{I_i}^G$ and $\widetilde{\mathbf{p}}_{I_i}^G$ denote cloned IMU pose error at time i, for $i = 1, \cdots, N$. The propagation of state and covariance is performed using IMU. For more details, please refer to [10].

In the measurement update step, both point and line features are incorporated. The tracked features are triangulated via Gauss-Newton optimization, utilizing the estimated camera poses. The resulting 3D features are then projected onto the image planes to calculate residuals. The residual of a point feature, $\mathbf{z}_p$, which represents the discrepancy between the observed $\mathbf{x}$ and estimated $\hat{\mathbf{x}}$ 2D pixel coordinates, is defined as

$$\mathbf{z}_p = \mathbf{x} - \hat{\mathbf{x}} = [u \quad v]^T - [\hat{u} \quad \hat{v}]^T \,. \tag{8}$$

We describe line features using the Plücker representation [27], [28], and transform it into a four-parameter orthonormal form for minimal parametrization and optimization convergence [29]. The residual of a line feature, $\mathbf{z}_l$, which represents the distance between the observed endpoints of the line and the projected 2D line on the image plane, is defined as

$$\mathbf{z}_l = \begin{bmatrix} \frac{\bar{\mathbf{x}}_s^T \hat{\mathbf{l}}}{\sqrt{l_1^2 + l_2^2}} & \frac{\bar{\mathbf{x}}_e^T \hat{\mathbf{l}}}{\sqrt{l_1^2 + l_2^2}} \end{bmatrix}^T \tag{9}$$

where $\bar{\mathbf{x}} = [u \quad v \quad 1]^T$ is a 2D endpoint in homogeneous coordinates, and $\hat{\mathbf{l}} = [l_1 \quad l_2 \quad l_3]^T$ is the estimated 2D line.

After projecting onto the left null space of feature Jacobians and vertically stacking all the residuals, we obtain the final residual vector, which is described as

$$\begin{bmatrix} \mathbf{r}_p^{(1)} \\ \vdots \\ \mathbf{r}_p^{(N_p)} \\ \hline \mathbf{r}_l^{(1)} \\ \vdots \\ \mathbf{r}_l^{(N_l)} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{f_p}^{(1)\,T} \mathbf{H}_{s_p}^{(1)} \\ \vdots \\ \mathbf{A}_{f_p}^{(N_p)\,T} \mathbf{H}_{s_p}^{(N_p)} \\ \hline \mathbf{A}_{f_l}^{(1)\,T} \mathbf{H}_{s_l}^{(1)} \\ \vdots \\ \mathbf{A}_{f_l}^{(N_l)\,T} \mathbf{H}_{s_l}^{(N_l)} \end{bmatrix} \widetilde{\mathbf{X}}_k + \begin{bmatrix} \mathbf{A}_{f_p}^{(1)\,T} \mathbf{n}_p^{(1)} \\ \vdots \\ \mathbf{A}_{f_p}^{(N_p)\,T} \mathbf{n}_p^{(N_p)} \\ \hline \mathbf{A}_{f_l}^{(1)\,T} \mathbf{n}_l^{(1)} \\ \vdots \\ \mathbf{A}_{f_l}^{(N_l)\,T} \mathbf{n}_l^{(N_l)} \end{bmatrix} . \tag{10}$$

$\mathbf{r}$ represents the final residual, $\mathbf{H}_s$ is sliding window state Jacobian, $\mathbf{A}$ is a unitary matrix whose columns form the basis of the left null space of feature Jacobian $\mathbf{H}_f$, $\mathbf{n}$ is

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

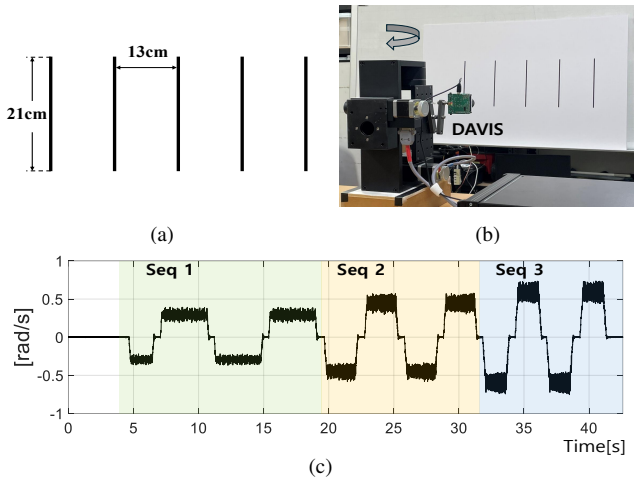

Fig. 4. Line detection experiment setup. (a) 5 vertical lines. (b) The DAVIS 240C is rigidly mounted on a rate table, which rotates along a single axis. (c) 3 different rotation rates are represented using distinct colors for each interval.

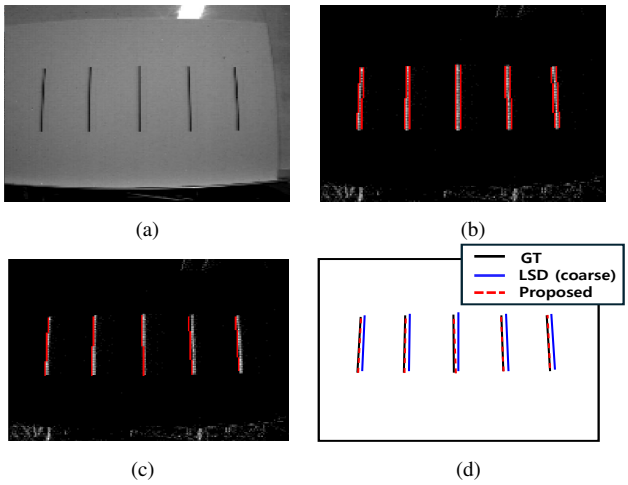

Fig. 5. (a) Ground truth lines from standard frame. (b), (c) Lines detected using LSD and our method on the corresponding event frame. (d) Comparison of the detected lines with the ground truth.

the noise vector, and N_p and N_l are the numbers of point and line features, respectively. The subscripts p and l refer to point and line, respectively, while the superscripts indicate the corresponding feature indices.

V. Experiments

In this section, the proposed method is validated using an author-constructed experiment and two public datasets. The evaluation includes line detection accuracy, improvements in feature tracking performance from the proposed synchronization scheme, and pose estimation accuracy along with its real-time performance. Additional implementation details are as follows:

- Line detection: The hyperparameters ϵ_i, ϵ_r, ϵ_{px}, and ϵ_{ang} were uniformly applied across all sequences.
- VIO: Point features in standard frames were extracted using the FAST corner detector, consistent with event frames. To mitigate drift in static scenes (due to the monocular VIO setup), a closed-form Zero Velocity Update (ZUPT), as proposed in [30], was applied.

The overall framework was implemented in C++ on Ubuntu 16.04 with ROS Kinetic, and tested on an Intel Core i7-14700KF processor.

A. Line detection accuracy

We numerically evaluate the accuracy of our line detection algorithm through a specific experiment. Given that the objective is to evaluate the accuracy of the detected lines, the experiment is designed to be straightforward. As illustrated in Fig. 4(a), five vertical lines are detected using the DAVIS 240C [4]. To validate the effectiveness of our coarse-to-fine motion compensation approach, we simulate a scenario where both rotation and translation are applied to the sensor, as depicted in Fig. 4(b), using a rate table. Rotations are applied at three different rates along a single axis parallel to the lines, as shown in Fig. 4(c). Due to the distance between the sensor and the rotation axis, known as the lever arm, translation is also induced to the sensor.

TABLE I
Line Detection Error (px) in Author-Constructed Experiment

	LSD (coarse)	LSD (fine)	Proposed
Seq 1	3.83	3.33	**0.88**
Seq 2	4.02	3.56	**1.00**
Seq 3	4.11	3.42	**0.96**

We compare our method with LSD, a widely used line detector in PL-VIO algorithms, implemented via OpenCV. For ground truth, we use lines detected by LSD from standard frames captured at the same timestamps as the event frames. Fig. 5 illustrates the results of line detection. When applying LSD on an event frame after rotation compensation, it is observed that lines are extracted in pairs from less sharp edge structures, as shown in Fig. 5(b). This occurs because the events caused by translation are not compensated for, leading to a line structure that is thicker than the actual line. In contrast, Fig. 5(c) shows that our method detects only one line from a single edge structure. The fine motion compensation, utilizing the temporal information of events around the corner for CM, results in more sharp edge structures. This allows us to more accurately determine the position of the line at the reference time.

To evaluate quantitatively, we calculate the residuals in pixel units between the ground truth line and the detected two endpoints, similar to Eq. (9), and use their norm as the error metric. In three sequences with varying rotation rates, the errors for five tracked lines are computed for all event frames. The average error for these five lines is presented in Table I. When using LSD, two lines are detected within a thick line structure, and the error is calculated for the line farther from the ground truth, as the error for the nearer line was comparable to that of our method. For more rigorous validation, we also compare the LSD results after applying the proposed fine motion compensation. As shown in Table I, the errors slightly decrease after fine motion compensation, but they still remain larger than those observed with our method. This highlights the effectiveness of our line detection, which identifies the thinnest line structure by considering event intensity.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

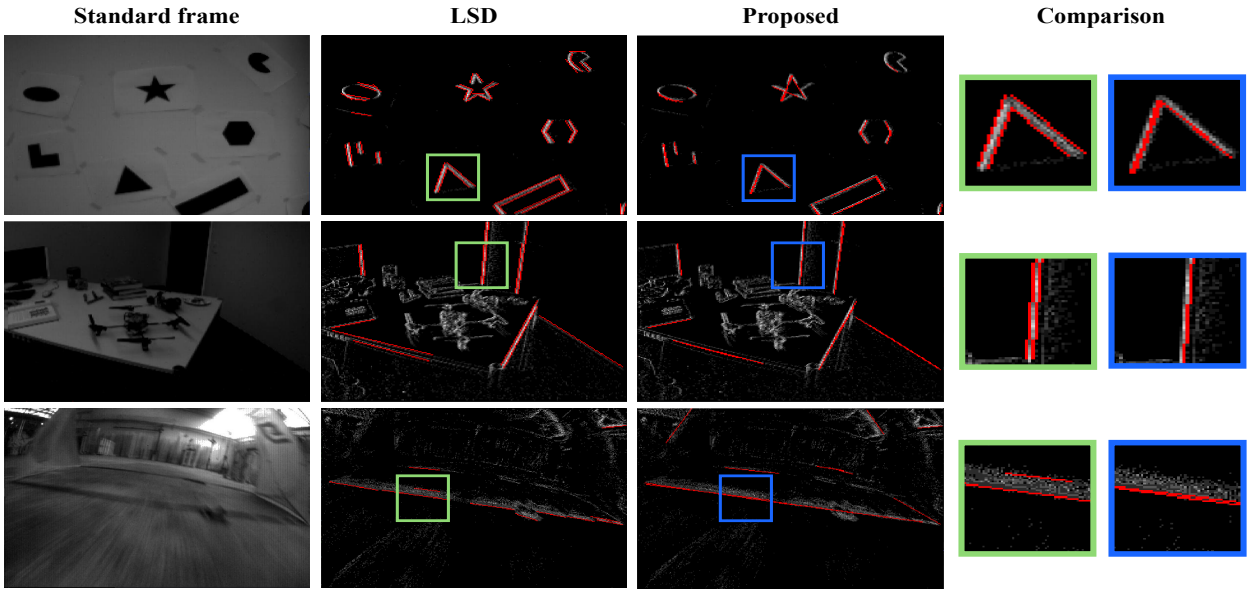

Fig. 6. The columns are as follows: 1st: Standard frame; 2nd: Lines detected using LSD; 3rd: Lines detected using our method; 4th: Enlarged comparison of specific regions. The rows represent different sequences: 1st: 'shapes_translation'; 2nd: 'dynamic_translation'; 3rd: 'indoor_forward_6'.

TABLE II
AVERAGE POSITION ERROR (%) DEPENDING ON THE WINDOW SIZE

r	0.15	0.2	0.25	0.3
ECDS [31]	0.274	**0.242**	0.248	0.251
UZH-FPV [32]	0.335	**0.315**	0.323	0.342

Next, we qualitatively present the line detection results for specific sequences from public datasets capturing real scenes: the Event-Camera Dataset [31] and the UZH-FPV Dataset [32], as shown in Fig. 6. As demonstrated by the the green and blue boxes in the rightmost column of Fig. 6, our method (blue box), with fine motion compensation, achieves precise line detection that is not attainable with LSD.

Impact of line segment length on localization: We extract lines within local windows and represent these lines as infinite lines using Plücker representation. The accuracy of a line feature is thus affected by the size of the local window. A scheme for determining the appropriate window size and its experimental evaluation are presented below.

We assume that beyond a certain length, the error between the true line and the partial line becomes negligible. While increasing the window size enables the extraction of longer lines, excessively large windows violate the constant optical flow assumption, necessitating an optimal window size. The window size S [px] $\times$ S [px] is defined as

$$S = r \times \min(W, H) \tag{11}$$

where r is the ratio factor, and W and H represent the width and height of the camera resolution, respectively. Table II shows the average position error calculated as % of the total trajectory length, for each dataset ([31], [32]) across all sequences, for different values of r. The smallest error is observed at $r = 0.2$, with further increases in r resulting in either no change or slight increases in error, likely due to the weakening of the constant optical flow assumption. Therefore, $r = 0.2$ was chosen as the optimal window size.

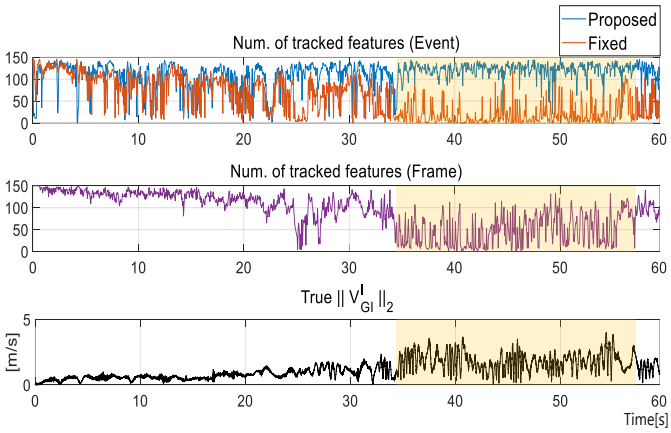

Fig. 7. Top: Number of tracked event features. Middle: Number of tracked frame features. Bottom: True velocity norm. Yellow-colored region indicates period of fast motion.

B. Consistency of feature tracking

We validate the effectiveness of the proposed adaptive event accumulation and synchronization scheme, introduced in Section IV-A, through the consistency of feature tracking. We compare the number of successfully tracked event point features in event frames using our method against the fixed event accumulation approach from [14]. Fig. 7 presents the results from the 'boxes_6dof' sequence in [31], displaying the number of tracked event features, frame features, and the true velocity norm of the IMU frame. As shown in the true velocity graph, the yellow-colored region indicates fast motion period. During this period, a large number of events occur in a very short time. In the fixed case, the number of event features decreases significantly, similar to the frame features. In contrast, the proposed method adapts to this high event activity by accumulating events dynamically, forming multiple event frames. This enables the stable tracking of features across these frames. Without this adaptive accumulation, as in the fixed case, the pixel movements between two event

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

TABLE III
COMPARISON OF POSE ESTIMATION ACCURACY ON THE EVENT-CAMERA DATASET [31]

Sequence	F + I		E + I								E + F + I					
	VINS-MONO [1]		EVIO [12]		EVIO-O [13]		Ours (P)		Ours (P + L)		USLAM [14]		EKLT-VIO [15]		Ours (P + L)	
	MPE	MYE	MPE	MYE	MPE	MYE	MPE	MYE	MPE	MYE	MPE	MYE	MPE	MYE	MPE	MYE
boxes_6dof	1.11	0.07	4.13	0.92	**0.69**	**0.09**	0.79	0.16	0.70	0.13	0.68	0.03	0.84	0.09	**0.17**	**0.02**
boxes_translation	0.57	0.06	3.18	0.67	**0.57**	0.04	0.95	0.04	0.91	**0.02**	1.12	2.62	0.48	0.25	**0.33**	**0.04**
dynamic_6dof	0.35	0.04	3.38	1.20	0.54	0.26	0.56	0.12	**0.45**	**0.10**	0.76	0.09	0.79	**0.06**	**0.32**	0.08
dynamic_translation	*failed*		1.06	0.25	0.47	0.11	0.58	0.16	**0.37**	**0.08**	0.63	0.22	0.40	**0.04**	**0.19**	0.08
hdr_boxes	0.31	0.10	3.22	0.15	0.92	**0.01**	**0.47**	0.09	0.53	0.06	1.01	0.31	0.46	**0.06**	**0.37**	0.09
hdr_poster	0.77	0.19	1.41	0.13	0.59	**0.09**	0.62	0.16	**0.44**	0.27	1.48	0.09	0.65	**0.04**	**0.23**	**0.04**
poster_6dof	1.42	0.18	5.79	1.84	0.82	0.11	0.66	0.04	**0.52**	**0.02**	0.59	0.03	0.35	**0.02**	**0.19**	0.12
poster_translation	0.22	0.04	1.59	0.38	0.89	**0.03**	**0.11**	0.04	0.22	0.11	0.24	**0.02**	0.35	0.03	**0.11**	0.06
shapes_6dof	1.33	0.05	2.52	0.61	1.15	**0.08**	0.61	0.14	**0.58**	0.11	1.07	**0.03**	0.60	**0.03**	**0.29**	0.04
shapes_translation	1.06	0.07	4.56	2.60	1.28	0.41	0.51	0.18	**0.42**	**0.03**	1.36	**0.01**	0.51	0.03	**0.21**	0.10
Average	0.79	0.09	3.08	0.88	0.79	0.12	0.59	0.11	**0.51**	**0.09**	0.89	0.35	0.54	**0.07**	**0.24**	**0.07**

TABLE IV
COMPARISON OF POSE ESTIMATION ACCURACY ON THE UZH-FPV DATASET [32]

Sequence	F + I	E + F + I		
	VINS-MONO [1]	USLAM [14]	PL-EVIO [8]	Ours (P + L)
	MPE	MPE	MPE	MPE
indoor_forward_3	0.65	*failed*	0.38	**0.19**
indoor_forward_5	1.07	*failed*	**0.90**	1.05
indoor_forward_6	0.25	*failed*	0.30	**0.14**
indoor_forward_7	0.37	*failed*	0.55	**0.21**
indoor_forward_9	0.51	*failed*	0.44	**0.32**
indoor_forward_10	0.92	*failed*	1.06	**0.33**
indoor_45_2	0.53	*failed*	0.55	**0.14**
indoor_45_4	1.72	9.79	1.30	**0.23**
indoor_45_9	1.25	4.74	0.76	**0.23**
Average	0.81	7.27	0.70	**0.32**

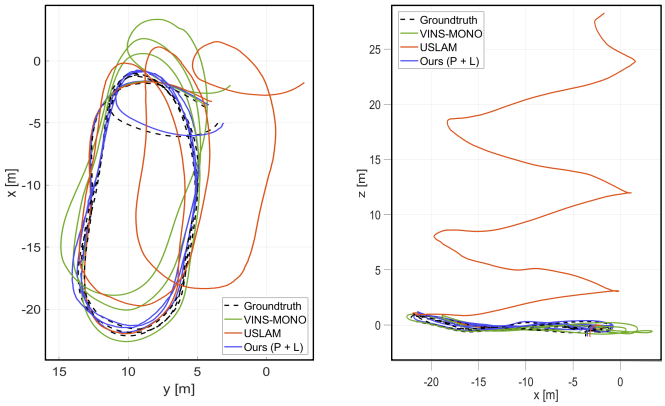

Fig. 8. Trajectories of the comparative methods on 'indoor_45_4' [32]. Left: Top view; Right: Side view.

frames are too significant, leading to feature tracking failure. To quantify this improvement, we set the maximum number of trackable features to 150 and calculate the average number of tracked event features for both the fixed case and the proposed method. Our method shows an average increase of 56.35% across 10 sequences in [31] and 54.13% across 9 sequences in [32] compared to the fixed case.

C. Pose estimation accuracy

We present a quantitative comparison of pose estimation accuracy between our method and state-of-the-art EVIO algorithms using [31], with results shown in Table III. The baselines include [1] (a VIO using standard frames) and [12]-[15], as detailed in Section II. EVIO-O denotes optimization-based EVIO, and the notations E, F, and I represent the use of event, frame, and IMU, respectively. As in [14] and [15], the estimated and ground-truth trajectories are aligned with a 6-DOF transformation in SE(3), using 5 seconds of the trajectory (from second 3 to second 8), and Mean Position Error (MPE) and Mean Yaw Error (MYE) are calculated as % of the total trajectory length and deg/m, respectively. Note that [8] cannot be directly compared due to incomplete information for all sequences and the absence of orientation error.

In the E + I section, we validate the effectiveness of line features by implementing two versions of our method: one using only point features, denoted by Ours (P), and another using both point and line features, denoted by Ours (P + L). The results show that incorporating line features consistently achieves lower MPE and MYE in nearly all sequences, demonstrating the benefits of additional visual measurements from line features. This improvement is particularly evident in the 'dynamic' and 'shapes' sequences, where the presence of abundant edges leads to significant performance gains. Our point + line (P + L) version achieves the lowest position error in 6 out of 10 sequences and the best yaw error in 5 out of 10 sequences, resulting in the smallest average error in both.

In the E + F + I section, where frame features are also utilized, our method outperforms the baselines by achieving the lowest position error across all 10 sequences and demonstrating comparable yaw error to [15], which exhibits the best yaw error. Overall, our method achieves the smallest average errors, confirming its superior performance.

To further validate our method in more aggressive and challenging scenarios, we evaluate it on [32] using 9 indoor drone racing sequences. We compare it with [1], [14], and [8]. As shown in Table IV, our method achieves the smallest position errors in 8 out of 9 sequences, with the lowest average error. Although yaw error was not directly compared, our method demonstrated an average yaw error of 0.02 deg/m. Fig. 8 illustrates the trajectories of our method and the baselines on the 'indoor_45_4' sequence.

TABLE V
RUNNING TIME (MS) OF OUR MODULES IN DIFFERENT RESOLUTIONS

Modules	240 × 180 [31]	346 × 260 [32]
Coarse M.C.	0.98	0.75
Fine M.C. and event-line detection	6.46	6.59
Event-line tracking	2.03	2.84
Event-point detection and tracking	1.02	2.25
Frame-point detection and tracking	5.89	5.54
Total process of frontend	22.32	24.98
Total process of backend	4.82	4.42

D. Real-time analysis

Finally, we present the running time analysis of our key modules in Table V. Using datasets [31], [32] with different event camera resolutions, the average processing time across all sequences was calculated. The entire process completes within 30 milliseconds for both resolutions, confirming the real-time capability of our approach. Note that M.C. in Table V refers to motion compensation.

VI. CONCLUSIONS

In this letter, we present an event-frame-inertial odometry that integrates point and line features to exploit additional visual information. To obtain more accurate line features from events, we employ IMU and CM for precise motion compensation in a coarse-to-fine manner, leading to highly sharp event frames. For effective fusion of events and frames, we adaptively accumulate events and synchronize them with frames. The proposed approach enhances both line detection and pose estimation accuracy on challenging real-world datasets. Notably, it enables accurate line detection, which LSD cannot achieve, while delivering superior pose estimation accuracy compared to state-of-the-art EVIO algorithms.

REFERENCES

[1] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.,* vol. 34, no. 4, pp. 1004-1020, Aug. 2018.

[2] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "OpenVINS: A research platform for visual-inertial estimation," in *Proc. IEEE Int. Conf. Robot. Autom.,* 2020, pp. 4666-4672.

[3] G. Gallego et al., "Event-based vision: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 44, no. 1, pp. 154-180, Jan. 2022.

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

[5] X. Zuo, X. Xie, Y. Liu, and G. Huang, "Robust visual SLAM with point and line features," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.,* 2017, pp. 1775-1782.

[6] S. Heo, J. H. Jung and C. G. Park, "Consistent EKF-based visual-inertial navigation using points and lines," *IEEE Sens. J.,* vol. 18, no. 18, pp. 7638-7649, Sep. 2018.

[7] Y. Yang, P. Geneva, K. Eckenhoff, and G. Huang, "Visual-inertial odometry with point and line features," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.,* 2019, pp. 2447-2454.

[8] W. Guan, P. Chen, Y. Xie, and P. Lu, "PL-EVIO: Robust monocular event-based visual inertial odometry with point and line features," *IEEE Trans. Autom. Sci. Eng.,* vol. 21, no. 4, pp. 6277-6293, Oct. 2024.

[9] R. G. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: A fast line segment detector with a false detection control," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 32, no. 4, pp. 722-732, Apr. 2008.

[10] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proc. IEEE Int. Conf. Robot. Autom.,* 2007, pp. 3565-3572.

[11] B. Choi, H. Lee, and C. G. Park, "Event-based visual-inertial odometry using point and line features with a coarse-to-fine motion compensation scheme," in *Proc. 40th Anniv. IEEE Int. Conf. Robot. Autom.,* 2024, pp. 1092-1094.

[12] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based visual inertial odometry," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.,* 2017, pp. 5391-5399.

[13] H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization," in *Proc. Brit. Mach. Vis. Conf.,* 2017, pp. 1-8.

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

[15] F. Mahlknecht et al., "Exploring event camera-based odometry for planetary robots," *IEEE Robot. Autom. Lett.,* vol. 7, no. 4, pp. 8651-8658, Oct. 2022.

[16] P. Chen, W. Guan, and P. Lu, "ESVIO: Event-based stereo visual inertial odometry," *IEEE Robot. Autom. Lett.,* vol. 8, no. 6, pp. 3661-3668, June. 2023.

[17] M. Trajković and M. Hedley, "Fast corner detection," *Image Vis. Comput.,* vol. 16, no. 2, pp. 75-87, Feb. 1998.

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

[19] C. Brändli, J. Strubel, S. Keller, D. Scaramuzza, and T. Delbrück, "ELiSeD – An event-based line segment detector," in *Proc. Int. Conf. Event-based Control, Commun., Signal Process.* 2016, pp. 1-7.

[20] 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 *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.,* 2020, pp. 5863-5870.

[21] W. Chamorro, J. Solà, and J. Andrade-Cetto, "Event-based line SLAM in real-time," *IEEE Robot. Autom. Lett.,* vol. 7, no. 3, pp. 8146-8153, Jul. 2022.

[22] L. Gao, D. Gehrig, H. Su, D. Scaramuzza, and L. Kneip, "An n-point linear solver for line and motion estimation with event cameras," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.,* 2024, pp. 14596-14605.

[23] Z. Liu, B. Guan, Y. Shang, Q. Yu, and L. Kneip, "Line-based 6-DoF object pose estimation and tracking with an event camera," *IEEE Trans. Image Process.,* vol. 33, pp. 4765-4780, Aug. 2024.

[24] G. Gallego, H. Rebecq, and D. Scaramuzza, "A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.,* 2018, pp. 3867-3876.

[25] G. Gallego, M. Gehrig, and D. Scaramuzza, "Focus is all you need: Loss functions for event-based vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.,* 2019, pp. 12280-12289.

[26] L. Zhang and R. Koch, "An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency," *J. Vis. Commun. Image Represent.,* vol. 24, no. 7, pp. 794-805, Oct. 2013.

[27] A. Bartoli and P. Sturm, "Structure-from-motion using lines: Representation, triangulation, and bundle adjustment," *Comput. Vis. Image Understanding,* vol. 100, no. 3, pp. 416-441, Dec. 2005.

[28] J. Solà, T. Vidal-Calleja, J. Civera, and J. M. M. Montiel, "Impact of landmark parametrization of monocular EKF-SLAM with points and lines," *Int. J. Comput. Vis.,* vol. 97, no. 3, pp. 339-368, 2012.

[29] G. Zhang, J. H. Lee, J. Lim, and I. H. Suh, "Building a 3-D line-based map using stereo SLAM," *IEEE Trans. Robot.,* vol. 31, no. 6, pp. 1364-1377, Dec. 2015.

[30] Q. Xiaochen, Z. Hai, and F. Wenxing, "Lightweight hybrid visual-inertial odometry with closed-form zero velocity update," *Chin. J. Aeronaut.,* vol. 33, no. 12, pp. 3344-3359, 2020.

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

[32] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, "Are we ready for autonomous drone racing? The UZH-FPV drone racing dataset," in *Proc. IEEE Int. Conf. Robot. Autom.* 2019, pp. 6713-6719.