Lattice-allocated Real-time Line Segment Feature Detection and Tracking Using Only an Event-based Camera

Mikihiro Ikura
Istituto Italiano di Tecnologia
mikihiro.ikura@iit.it

Arren Glover
Istituto Italiano di Tecnologia
arren.glover@iit.it

Masayoshi Mizuno
Sony Interactive Entertainment Inc.
masayoshi.mizuno@sony.com

Chiara Bartolozzi
Istituto Italiano di Tecnologia
chiara.bartolozzi@iit.it

Abstract

Line segment extraction is effective for capturing geometric features of human-made environments. Event-based cameras, which asynchronously respond to contrast changes along edges, enable efficient extraction by reducing redundant data. However, recent methods often rely on additional frame cameras or struggle with high event rates. This research addresses real-time line segment detection and tracking using only a modern, high-resolution (i.e., high event rate) event-based camera. Our lattice-allocated pipeline consists of (i) velocity-invariant event representation, (ii) line segment detection based on a fitting score, (iii) and line segment tracking by perturbating endpoints. Evaluation using ad-hoc recorded dataset and public datasets demonstrates real-time performance and higher accuracy compared to state-of-the-art event-only and event-frame hybrid baselines, enabling fully stand-alone event camera operation in real-world settings.

1. Introduction

Line segment on images can efficiently encode geometric features of human-made structures, aiding to reduce computational cost for crucial vision tasks such as 3D reconstructions [8], Structure from Motion (SfM) [26], Simultaneous Localization and Mapping (SLAM) [31], and pose estimation [33]. Recently, compared to classic computer vision approaches, such as the Line Segment Detector (LSD) [16], line segment extraction methods with deep learning approaches [17, 30] have become more widespread. These deep networks have been trained with particularly challenging images to ensure highly accurate and robust performance. However, the frame input to the network still suffers from quality degradation due to motion blur for fast moving cameras or targets, and pixel intensity saturation in high dynamic range (HDR) conditions. Moreover, their high computational demands that usually require the use of GPUs limit use on resource-constrained platforms such as embedded systems and mobile robots.

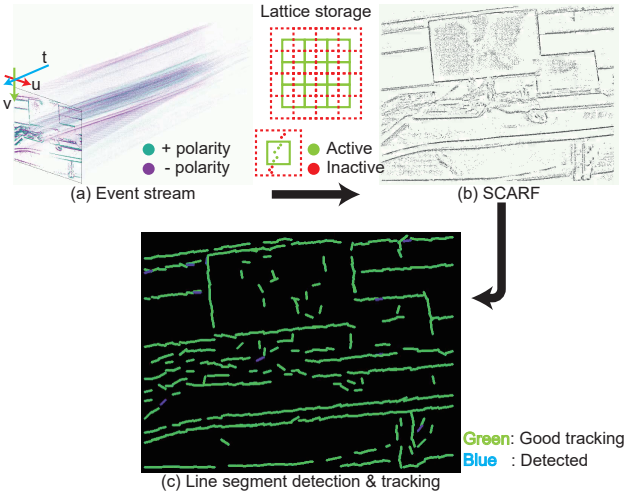

Figure 1. The proposed lattice-allocated real-time line segment detection and tracking. The (a) events are stored into lattice storage by (b) a velocity-invariant method SCARF. Line segments are (c) initialized within a block using only events from the active region of the lattice storage (Detection). Subsequently, their positions are updated both inside and outside the block using events from the active and inactive regions (Tracking).

Event-based cameras, inspired by biological vision, are gaining popularity in robotics and computer vision for low-latency, high dynamic range, and robustness to motion blur [5, 13]. They asynchronously capture pixel-wise brightness changes as events $e = \{t, u, v, p\}$, where t is the timestamp with microsecond resolution, (u, v)

the pixel location, and $p \in \{1, -1\}$ the polarity indicating the direction of the brightness change. This allows edge extraction as event clusters (Fig. 1 (a)) and has motivated line segment detection and tracking methods [9, 11, 16]. Recent event cameras achieve high resolutions (e.g., 640×480, 1280×720), less noise, and event rates exceeding 10 Giga events/s, surpassing older models like DAVIS346 (346×260, 12 Mev/s). However, this brings challenges for real-time, event-by-event processing. In addition, while event-frame hybrid methods [7, 32, 34] have improved performance, their reliance on frame cameras limits the potential of fully event-based processing despite recent hardware advancements of event cameras.

To address the challenges of real-time and low computational cost line segment detection under high event rates, we propose a novel model-based pipeline (Fig. 1) that (i) stores events using a velocity-invariant method, "SCARF" (A **S**et of **C**entre **A**ctive **R**eceptive **F**ields), (ii) detects line segments based on a fitting score from a lattice, and (iii) tracks them by perturbating their endpoints inside and outside the lattice. SCARF efficiently handles large event volumes compared to prior methods [14, 15, 25], enabling real-time operation on high-resolution cameras (e.g., 640×480). By leveraging events stored in the preallocated lattice storage as a batch, our method achieves over 200 Hz for detection and 400 Hz for tracking in a process-driven manner. Once detected, line segments are passed to the tracker, reducing computational cost and increasing efficiency through cooperative processing. Overall, our method enables real-time performance at high event rates and dense short-segment extraction for complex shapes with high accuracy.

In summary, the contributions of this paper are:

- Introduction of lattice-allocated real-time pipeline that, for a recorded dataset (640x480), achieves event-driven storage at over 20 Mev/s, process-driven line segment detection at over 200 Hz, and tracking at over 400 Hz.
- Qualitative and quantitative evaluations with locally recorded dataset (640x480) and a publicly available datasets (240x180, 346x260) for line segment accuracy, real-time performance, and parameter sensitivity
- C++ implementation of the proposed pipeline and recorded dataset are released as open-source: https://github.com/event-driven-robotics/RT-EvLDT.

2. Event-driven Line Segments Detection and Tracking Methods

Event cameras naturally emphasize edges, therefore, event-based line feature detection and tracking was utilized for many computer vision and robotics tasks, such as 6DoF pose tracking [23, 24], robot control [10, 18], visual odometry [22], and SLAM [6]. However, the focus of these applications does not consider computational cost scaling with camera speed and resolution, leading to non-real-time process. [23, 24] present iterative optimizations by minimizing error to improve accuracy. In addition, robot control methods [10, 18] suffer from heavy computational cost by event-by-event processing. Furthermore, [6] contributes to line SLAM with light-weight line segment extraction based on hough-transform to avoid time-consuming line clustering such as IDOL [22].

According to [9], event-based line detection and tracking methods were typically categorized into: (i) Hough-transform based, (ii) Non-parametric, and (iii) Spatio-temporal approaches. Hough-transform based methods [10, 18] find lines that contain the largest number of events in a line-based parameter space, but are computationally intensive and unsuitable for high event rates. Non-parametric methods [4] cluster pixels based on spatial derivatives from a time surface [3, 28]. While line segments can be extracted even for complex shapes, detected segments are often short and sensitive to pixel-level noise. Spatio-temporal methods assume that events aligned with a line within a sufficiently small time window lie on a plane in u, v, t space, enabling line extraction via plane fitting [10, 11]. This yields longer, more stable segments, though performance on detailed or curvy shapes is limited. Beyond these, deep learning approaches combine events and frames [32, 34], and contrast maximization [12] enables event-image sharpening via motion compensation, leading to accurate line segment extraction [7]. Our method takes a pseudo-spatio-temporal approach: events are buffered in a lattice structure following the FIFO principle without precise timestamps (pseudo-temporal); line fitting is performed independently within each block in u, v space (pseudo-spatial).

3. Lattice-allocated Real-time Line Segment Detection and Tracking

Figure 2 illustrates the parallelized pipeline proposed for lattice-allocated real-time line segment detection and tracking with event-based cameras. It mainly consists of three threads: SCARF, Detection, and Tracking. Each predefined grid block from a lattice consists of a line segment L and a SCARF ring buffer $\boldsymbol{\mathcal{E}}$. The line segment L is defined by: (i) two endpoints $\boldsymbol{q}_0 = (u_{q_0}, v_{q_0})$, $\boldsymbol{q}_1 = (u_{q_1}, v_{q_1})$; (ii) a fitting score f, computed during detection and tracking; (iii) a Line status $s \in \{$NoDetect, Detected, ProhibitDetection, BadTrack, GoodTrack$\}$; (iv) the block coordinate that owns the line segment, referred to as the Admin block (ru_b, rv_b); and (v) a line segment ID l_{id}.

Firstly, in the SCARF thread, input from the event-based camera is processed event-by-event and is stored the SCARF buffers with respect to the event coordinate. By utilizing the stored events, the detection and tracking threads are not required to operate in an event-driven manner, allowing for the construction of a process-driven pipeline.

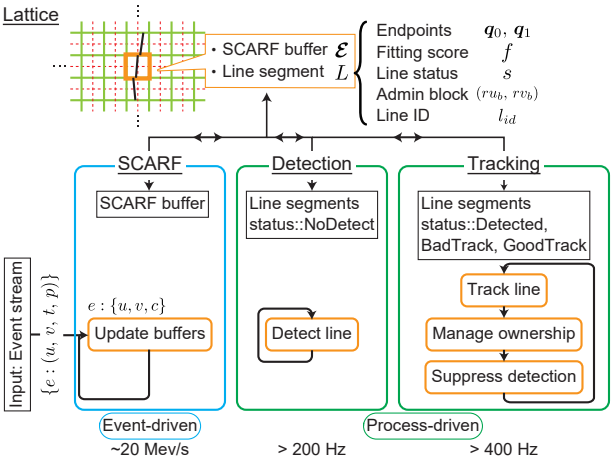

Figure 2. Parallelized pipeline overview of lattice-allocated line segment detection and tracking

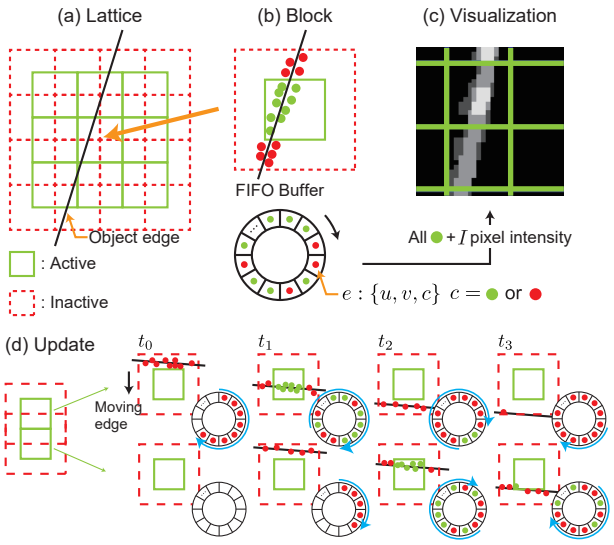

Figure 3. Overview of SCARF to store and visualize events with a lattice structure. (a) The lattice structure is defined with active and inactive regions. (b) Each block maintains a preallocated FIFO buffer for storing events. (c) For visualization, an image-like representation can be generated by assigning a pixel intensity I to all "active" events within the buffer. (d) SCARF buffer update process. At t_0, the upper buffer is filled with inactive events. At t_1, it is "excited" by active events, while the bottom buffer is "inhibited". When the edge leaves the upper active region (t_2), the upper buffer becomes inhibited and the bottom buffer is excited. At t_3, the upper buffer is fully inactive, and inhibition begins in the bottom buffer.

The separation of event- and process-driven threads mitigates the difficulty of real-time processing that arise with increasing input event rates.

In the detection thread, if no line segment currently exists in a block (i.e. LineStatus == NoDetect), a new line segment is extracted from the block using events stored in the SCARF buffer.

In the tracking thread, the state of a line segment is continuously updated by slightly perturbating its two endpoints $\boldsymbol{q}_0, \boldsymbol{q}_1$ from the previous detection or tracking loop (i.e. LineStatus $\in$ {Detected, GoodTrack}). Additionally, this thread manages the suppression of detections in neighboring blocks to avoid conflicts between detection and tracking (i.e. LineStatus == ProhibitDetection) while ensuring that the owner of each line segment is consistently updated. In addition, if the fitting score f falls below a predefined threshold f_{th} or the line segment moves out of camera's field of view, the line status is shifted to BadTrack. The next iteration of the detection thread can then attempt re-detection of a new line segment.

Compared to the line detection process, the tracking process with perturbation is computationally more efficient. Therefore, the cooperation between detection and tracking reduces computational costs and enhances the process frequency in process-driven threads. The following subsections explains each thread in more detail.

3.1. SCARF

SCARF generates a consistent frame-like representation for scenes containing multiple objects moving at different speed. SCARF adopts an approach where, instead of managing events with a buffer across the entire field of view (FOV), each predefined small block maintains its own buffer to store events. Such an approach assumes the motion within a single block is consistent over the block (e.g. not two objects moving with different speeds within a block) which becomes more valid as the block size decreases. Multiple objects, and camera motion, with different speeds are instead invariantly represented across different blocks.

Figure 3 shows that each SCARF block consists of an active region that covers the entire block and an inactive region, which is half the size of the block on each side and overlaps with adjacent blocks. Each event $e = \{t, u, v, p\}$ is therefore stored as an active event in the single buffer, and as an inactive event in up to three neighboring buffers. Each buffer has a finite size $N = \alpha \times b^2$, where α is a tunable parameter and b is the block size [px], and is managed by FIFO (First-In, First-Out) principle, whereby the newest event replaces the oldest event in the buffer. Events are removed after N subsequent (active/inactive) events have been added to the buffer. Block overlap and inactive events are very important and are used to "clear" blocks as edges move out of the block region. They do so by overwriting active events, and are not considered themselves by any downstream visualization or processing, as shown in Fig. 3. In this study, the raw events stored in the buffer are used for line segment detection and tracking. In addition, all active events stored in

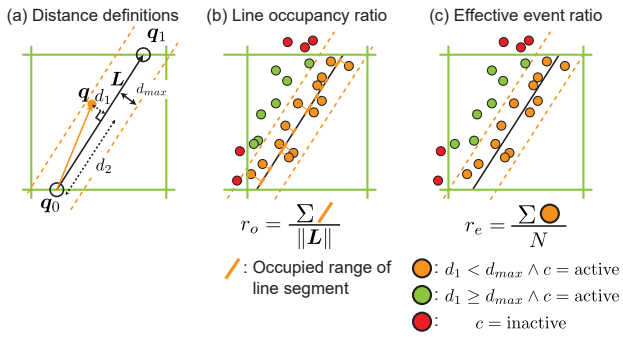

Figure 4. Definition of fitting score

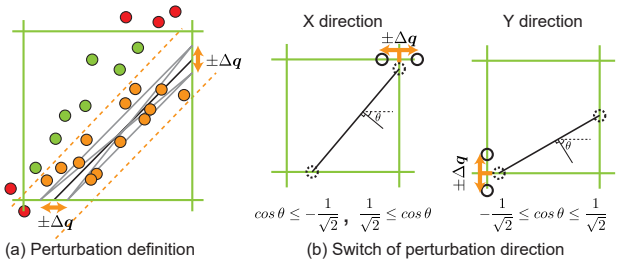

Figure 5. Perturbation of endpoints for tracking

all blocks can be visualized on the image plane by accumulating a pixel intensity I for each event at its pixel location $\{u, v\}$.

3.2. Detection

This study unifies SCARF active blocks and the blocks for line segment management. The main objective of detection is to initialize line segments consisting of endpoints $\boldsymbol{q}_0, \boldsymbol{q}_1$ and fitting score f inside blocks. An optimal line is fitted to the 2D "active" event point cloud extracted from the buffer in the block, with its intersections with the block boundaries defining the segment's endpoints. This fitting process, which minimizes the total distance between the line and events, is computationally expensive, highlighting the importance of the lower-cost tracking process.

The fitting score f is initialized with two ratios, (i) line occupancy ratio r_o and (ii) effective event ratio r_e shown in Fig. 4 by following this equation:

$$f = r_o \times r_e. \tag{1}$$

To compute these two ratios, the distances d_1 and d_2 shown in Fig. 4 are defined as follows:

$$d_1 = \frac{\|(\boldsymbol{q} - \boldsymbol{q}_0) \times \boldsymbol{L}\|}{\|\boldsymbol{L}\|}, d_2 = \frac{(\boldsymbol{q} - \boldsymbol{q}_0) \cdot \boldsymbol{L}}{\|\boldsymbol{L}\|} \tag{2}$$

where $\boldsymbol{q}$ is the 2D position of an event (u, v) from a buffer $\boldsymbol{\mathcal{E}}$, $\boldsymbol{q}_0$ is one of the endpoints of a line segment, and $\boldsymbol{L}$ is a vector on the line segment $\boldsymbol{q}_0 - \boldsymbol{q}_1$. d_1 represents the perpendicular distance from the point to the line, while d_2 denotes the distance of the projected point along the line segment. The line occupancy ratio quantifies how evenly events are distributed along the entire line segment. To calculate this, the occupancy token $T(i)$ is determined with:

$$T(i) = \begin{cases} 1.0, & \text{if } i = \lfloor d_2 \rfloor \text{ and } d_1 < d_{max} \text{ for } \boldsymbol{q} \in \boldsymbol{\mathcal{E}} \\ 0, & \text{else} \quad (0 \leq i \leq \lfloor \sqrt{2}b \rfloor), \end{cases} \tag{3}$$

where $\lfloor \cdot \rfloor$ is a floor function, d_{max} is a predetermined threshold parameter, and b is block size. Finally, the line occupancy ratio r_o is defined as:

$$r_o = \frac{1}{\|\boldsymbol{L}\|} \sum_{i=0}^{\lfloor \sqrt{2}b \rfloor} T(i), \ (0 \leq r_0 \leq 1). \tag{4}$$

Effective event ratio r_e can be calculated by counting "active" events closer to the line segment than d_{max} from:

$$r_e = \frac{1}{N} \sum_{p \in \mathcal{E}} \mathbb{1}(d_1 < d_{max} \wedge c = \text{active}), \ (0 \leq r_e \leq 1) \tag{5}$$

where c is a SCARF indicator, N is the buffer size, and $\mathbb{1}(\cdot)$ is an indicator function. Finally, the fitting score of the line segment f is calculated with Eq. (1). The detection succeeds if this score is above the predetermined threshold f_{th} and the detected line segment is switched to tracking process for continuous status updates.

3.3. Tracking

Considering the assumption in SCARF that each line segment is approximately the size of the predefined block, endpoints of line segments can be fixed on lattice. This constraint for perturbation simplifies the tracking algorithm. The tracking algorithm in this research is inspired by [2, 21], whereby hypothetical states $\mathcal{H}(\mathbf{x})$ are generated by perturbating tracking state $\mathbf{x}^{(k)}$ and the hypothetical state that produces the highest score is selected as the next tracking state $\mathbf{x}^{(k+1)}$. In this research, the tracking state is defined with the endpoints of the line segment as $\mathbf{x}^{(k)} = \{\boldsymbol{q}_0, \boldsymbol{q}_1\}$, and the tracking state is updated according to Eq. (6), which consists of the fitting score f of the line segment L and SCARF buffer(s) $\mathcal{E}$:

$$\mathbf{x}^{(k+1)} = \begin{cases} \underset{\boldsymbol{h} \in \mathcal{H}(\mathbf{x}^{(k)})}{\arg\max} \ f\left(\bigcup_{(i,j) \in \mathcal{I}(L)} \mathcal{E}_a^{(k+1)}, \boldsymbol{h} \right), & \text{if } |\mathcal{I}(L)| > 1 \\ \underset{\boldsymbol{h} \in \mathcal{H}(\mathbf{x}^{(k)})}{\arg\max} \ f\left(\mathcal{E}_{a \cup i}^{(k+1)}, \boldsymbol{h} \right), & \text{else} \end{cases} \tag{6}$$

where

$$\mathcal{H}(\mathbf{x}) = \{(\mathbf{x} = \{\boldsymbol{q}_0, \boldsymbol{q}_1\}) \cup \{\boldsymbol{q}_0 \pm \Delta\boldsymbol{q}, \boldsymbol{q}_1 \pm \Delta\boldsymbol{q}\}\}, \tag{7}$$

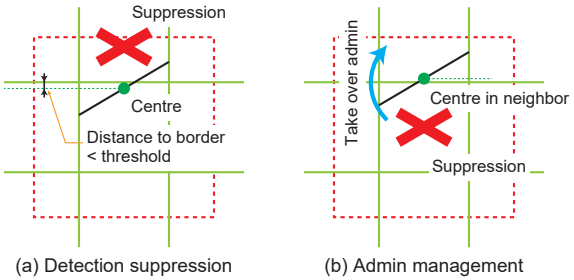

Figure 6. Detection suppression and admin management to avoid conflict between detection and tracking

and $\mathcal{I}(L)$ represents coordinate of all blocks through which the line segment L passes. Eq. (6) considers two key factors: (i) A line segment spanning multiple blocks uses all active events $\bigcup_{(i,j)\in\mathcal{I}(L)} \mathcal{E}_a$ from their buffers, avoiding any redundancy due to overlap in the inactive region. (ii) A line segment within a single block considers both active and inactive events $\mathcal{E}_{a\cup i}$ for perturbations both within and beyond the block. The norm of the perturbation vector $\Delta \boldsymbol{q}$ is a tunable parameter, but the direction of $\Delta \boldsymbol{q}$ is determined by which direction (X/Y) each endpoint currently lies on in the lattice as shown in Fig. 5 (a). Therefore, the total number of fitting scores $f\left(\mathcal{E}, \boldsymbol{h}\right)$ to be evaluated is 5 based on the hypothetical state Eq. (7). When an endpoint gets sufficiently close to a block corner, the perturbation direction is adjusted to align more closely with the normal vector of the current line segment, as illustrated in Fig. 5 (b). This process prevents the line segment from extending infinitely while allowing perturbation both within and beyond the block, resulting in longer-lived line segments. After updating the tracking status, the best fitting score f is compared with the same threshold f_{th} used in detection to categorize the line segment as either GoodTrack or BadTrack.

When a tracked line segment gets close to the border of the block, as shown in Fig. 6, the detection and tracking are in conflict between 2 adjacent blocks. To avoid this conflict, the neighbouring block is suppressed to detect a new line segment when a centre of line segment is close enough to the block edge (Fig. 6 (a)). Additionally, as the perturbation progresses and the centre of the line segment moves into a neighbouring block, the ownership (admin) of the line segment is transferred to that block and then the previous block is suppressed for detection (Fig. 6 (b)). In the following processing step, that block will be responsible for tracking.

4. Evaluation

4.1. Experimental Configuration

We compared the proposed line segment detection and tracking pipeline with four baselines: ELiSeD [4], Powerline [9], C2F-EFIO [7], and FE-LSD [34] as shown in Table 1. We modified and reimplemented the open source code[1,2,3,4] in C++ or Python environment for fair comparison. All methods were executed on a Intel(R) Core(TM) i7-9750H CPU @ 2.60 GHz x 12 Cores, 16 GB RAM at 2667 MT/s, and a GPU NVIDIA GeForce GTX 1650 for only deep-learning method FE-LSD. Further details about hyperparameter settings in each baseline for evaluations are in the supplementary.

Baselines	Input				Line segments	
	Event	Frame	IMU	GPU	Detection	Tracking
ELiSeD [4]	✓				✓	✓
Powerline [9]	✓				✓	✓
C2F-EFIO [7]	✓	✓	✓		✓	✓
FE-LSD [34]	✓	✓		✓	✓	
Ours	✓				✓	✓

Table 1. Comparison of input requirements, GPU dependency, and line segment outputs for ours and baseline methods

The five pipelines were benchmarked using public datasets called Event Camera Dataset [29] (240x180) and MVSEC dataset [36] (346x260) which also include frames and arbitrary 6DoF motion from IMU. Additionally, to test the limits of real-time processing for higher event rates, we evaluated them on a novel dataset recorded from Prophesee's EVK3 (640x480). The quantitative evaluation on line segment accuracy requires the ground truth line segments. For recorded data with low complexity, ground truth line segments were created through hand annotation. For public datasets where pixel-to-pixel matching between RGB images and event data is ensured, LSD [16] was applied to the RGB images to imitate the ground truth. Table 2 describes the details of dataset for line segment evaluations.

4.2. Qualitative Evaluation

We conducted a qualitative evaluation by assessing tracking accuracy with the recorded data "monitor_6dof" (640x480), ECD "dynamic_6dof", and MVSEC "indoor_flying1". Figure 7 displays the comparison between temporal window representation [20] and SCARF for event representation, and line segments from ELiSeD, Powerline, C2F-EFIO, FE-LSD, and ours. ELiSeD, PowerLine, C2F-EFIO, and FE-LSD required high computational cost and were not capable of real-time processing, while our method performed real-time processing. In ELiSeD, the extracted line segments were generally shorter than those from other methods, leading to visible discontinuities in the representation of object boundaries, particularly under low number of events. In contrast, Powerline produced longer line segments; however, these often failed to capture fine details

[1]https://github.com/SensorsINI/jaer
[2]https://github.com/uzh-rpg/line_tracking_with_event_cameras
[3]https://github.com/choibottle/C2F-EFIO
[4]https://github.com/lh9171338/FE-LSD

	Dataset	Resolution	Complexity	Event	Frame	IMU	Line segment GT
Recorded	three_vertical_lines_fast	640x480	Low	✓			Annotation
	circle_board	640x480	Low	✓			Annotation
	monitor_6dof	640x480	High	✓			None
Public	ECD [29]: dynamic_6dof	240x180	High	✓	✓	✓	LSD
	MVSEC [36]: indoor_flying1	346x260	High	✓	✓	✓	LSD

Table 2. Datasets used for the evaluation of line segment detection and tracking. Each dataset has a duration of 10 seconds. For the MVSEC dataset in particular, the evaluation is performed over a time range from 15 to 25 seconds after the drone begins flying.

or the shapes of small and curvilinear objects, as observed in the MVSEC dataset. The line segments extracted by C2F-EFIO and FE-LSD were of intermediate length, longer than those from ELiSeD and shorter than those from Powerline, providing a reasonable balance for geometric encoding. Thanks to the lattice-based detection and tracking framework, our method extracted a larger number of line segments compared to both C2F-EFIO and FE-LSD. Furthermore, in comparison to FE-LSD, the temporal consistency of line segments between successive frames was notably improved in our method, owing to the tracking mechanism and the velocity-invariant event representation SCARF as shown in squares of Fig. 7.

4.3. Quantitative Evaluation

We quantitatively evaluate our method and baselines in terms of line segment accuracy (precision and recall), real-time performance, and parameter sensitivity (block size b, SCARF ratio α, threshold f_{th}). Line heat map accuracy [19, 35] was computed on two recorded datasets (640×480), "three_vertical_lines_fast" and "circle_board", and two public datasets, ECD [29] "dynamic_6dof" (240×180) and MVSEC [36] "indoor_flying1" (346×260). Tolerances were set to 1% (recorded) and 2% (public) of image diagonal [19]. We computed precision and recall using the cumulative sum at each second over 10 seconds to evaluate temporal variations of accuracy. Figure 8 shows the distributions for Powerline, C2F-EFIO, and ours (30 trials) and deterministic ELiSeD and FE-LSD (1 trial). In ELiSeD, some line segments have cracks due to the lack of events. In particular, in the dataset "circle_board", horizontal line segments were often omitted due to limited vertical motion. Powerline showed similar omissions. In the dataset "three_vertical_lines_fast", faster horizontal motion caused tracking failures, reducing recall in Powerline. In addition, Powerline was not able to detect and track line segments accurately from complicated scene such as the dataset "dynamic_6dof" and "indoor_flying1". C2F-EFIO was sensitive to variations in event rate caused by changes in motion speed, which resulted in incomplete extraction of line segments across the entire field of view. This limitation contributed to its lower recall. FE-LSD could obtain line segments more accurately than ours. However, FE-LSD achieved accurate detection but required fine-tuning for curved shapes. Our method outperformed all baselines in F-score across datasets. Moreover, the velocity-invariant SCARF representation enabled uniform line segment extraction even in regions with insufficient events due to motion pattern ("circle_board") and excessive events due to fast motion ("three_vertical_lines_fast"). In more complex scenes ("dynamic_6dof" and "indoor_flying1"), the lattice-based management efficiently facilitated line segment extraction.

dynamic_6dof (240x180, 0.49 Mev/s)	***-driven	Mean event rate	Mean process frequency
ELiSeD		Event	$3.6e^{-3}$ Mev/s ↓	-
Powerline		Event	0.14 Mev/s ↓	-
C2F-EFIO		Process	-	32.3 Hz
FE-LSD w/ GPU		Process	-	1.3 Hz
Ours w/ tracking	SCARF	Event	26.0 Mev/s ↑	-
	Detection	Process	-	1088 Hz
	Tracking	Process	-	2246 Hz

monitor_6dof (640x480, 1.81 Mev/s)	***-driven	Mean event rate	Mean process frequency
ELiSeD		Event	$1.8e^{-3}$ Mev/s ↓	-
Powerline		Event	0.16 Mev/s ↓	-
Ours w/ tracking	SCARF	Event	24.7 Mev/s ↑	-
	Detection	Process	-	209.2 Hz
	Tracking	Process	-	427.2 Hz
Ours w/o tracking	SCARF	Event	20.13 Mev/s ↑	-
	Detection	Process	-	43.2 Hz

Table 3. Real-time performance evaluation in three event-only and two event-frame hybrid methods with the dataset "dynamic_6dof" and "monitor_6dof". The upward arrow denotes real-time processing, while the downward arrow indicates a lag.

For the real-time performance evaluation, we measured the average event rate in event-driven pipeline and the process frequency in process-driven pipeline. The recorded dataset "monitor_6dof" was utilized as an input into three event-only methods, while the public dataset "dynamic_6dof" which includes both events and frames was used for all baselines. The average event rate and processing frequency were computed by taking the mean over the processing performed between 5 and 10 seconds, after the system stabilized following data input. Table 3 shows the results of the real-time performance. Compared to the average event rate of the dataset, the event-driven pipeline ELiSeD and Powerline exhibited lower event processing rates, indicating that these event-driven baselines do not operate in real-time. On the other hand, SCARF demonstrated a higher event processing rate than the input event rate, resulting in real-time performance. Furthermore, even with lower resolution in event data and GPU acceleration, pro-

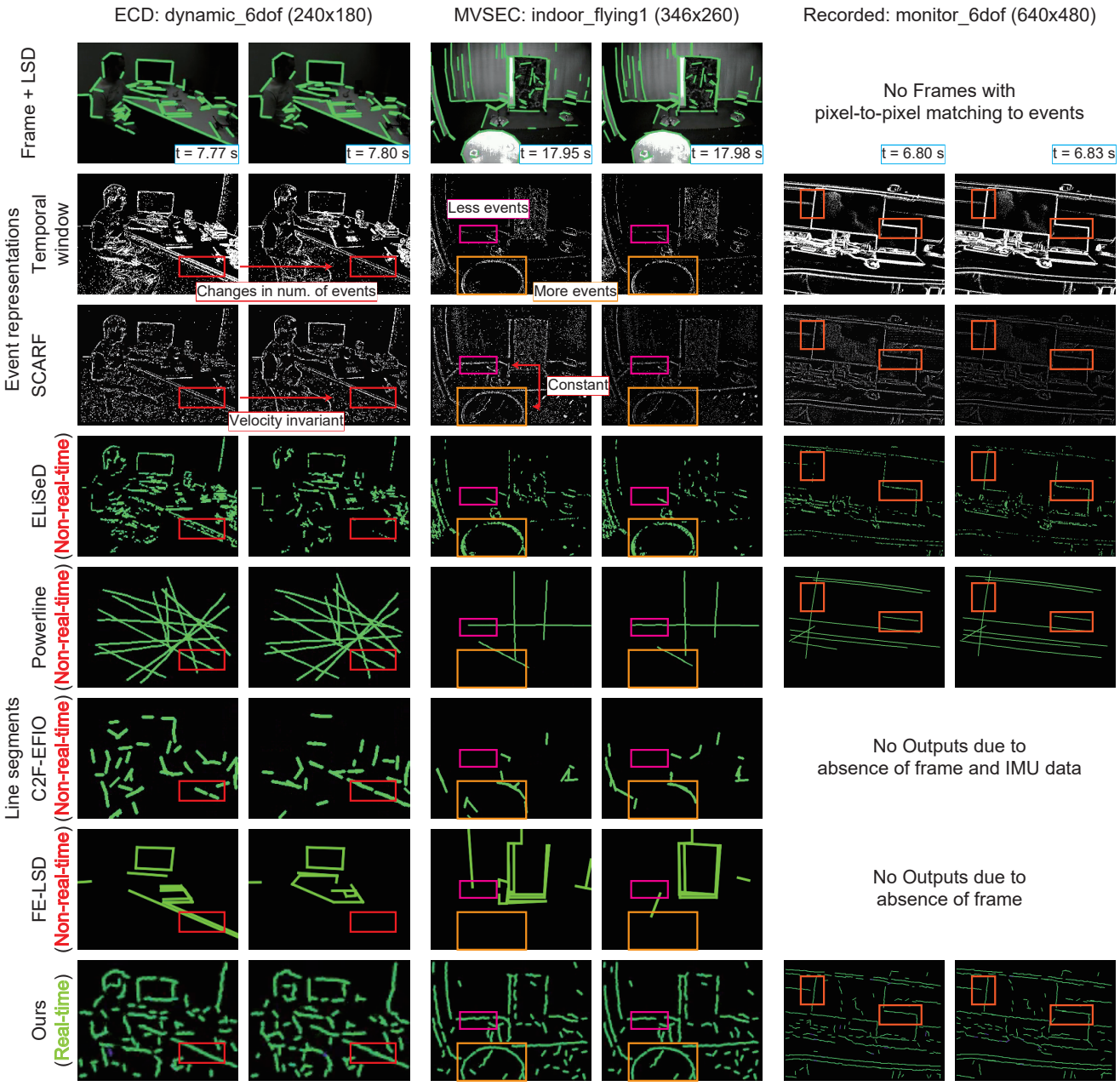

Figure 7. Qualitative comparison of line segment detection and tracking results for ELiSeD, Powerline, C2F-EFIO, FE-LSD, and our method. In the ECD dataset (dynamic_6dof), motion changes between consecutive frames cause variations in relative velocity, which in turn lead to differences in the number of generated events. These variations are visualized as brightness differences in the temporal window, highlighted in the red square. In the MVSEC dataset (indoor_flying1), the presence of objects at varying depths also results in differences in relative velocity and event density, as indicated by the light orange and pink squares.

cess frequency in FE-LSD and C2F-EFIO is much lower than our method. In our method, adding tracking to the line segment detection resulted in significantly higher processing frequency in detection and tracking through lightweight processing based on endpoint perturbation, ensuring greater efficiency.

Finally, we evaluated our method sensitivity to parameters, such as threshold f_{th}, block size b, and α for the SCARF buffer size N with respect to accuracy and real-time performance. The input dataset is the first 10 seconds of "dynamic_6dof" (240x180) whose average event rate is 0.49 Mev/s. As shown in Fig. 10 and Table 4, we

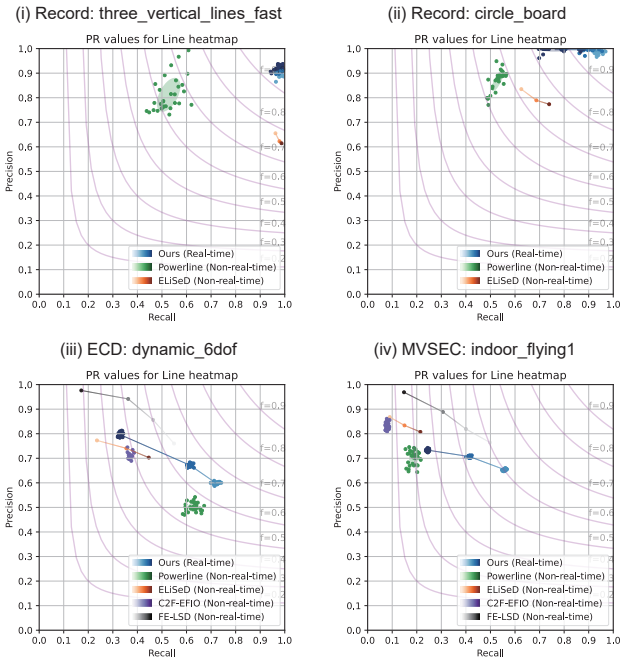

Figure 8. Precision and Recall distributions of line heat maps in four dataset and with five methods. The results of our method, ELiSeD, and FE-LSD for different thresholds are represented by using a color gradient. The purple curves show F scores as harmonic mean of the precision and recall.

prepared 7 parameter sets and 30 trials in each parameter set for the sensitivity evaluation. These results show that a larger threshold f_{th} results in higher precision and lower recall. In addition, it leads to a lower detection process frequency and a higher tracking process frequency. This result is because a higher threshold reduced the number of successful detections, decreasing the number of line segments available for tracking. As a result, the accuracy of the detected line segments improved. Larger block size b results in higher precision, lower recall, and higher process frequency for both detection and tracking. A larger block size extends line segments and reduces their total number. This improved the accuracy of each line segment and increased the process frequency. Regarding the sensitivity of α, precision increased and recall decrease regardless of whether α was large or small. However, a small α leads to higher processing frequency for both detection and tracking. Reducing α decreases the number of events in the buffer, meaning that a fitting score f above the threshold f_{th} indicates more accurate fitting even with fewer events. On the other hand, increasing α amplifies the effect of the $\frac{1}{N}$ term in Eq. (5), also leading to a smaller effective event ratio r_e. As a result, similar to the larger α, the line fitting becomes more precise. Besides, reducing α decreases the number of processed events, lowering the computational cost and significantly increasing the processing frequency for both detection and tracking.

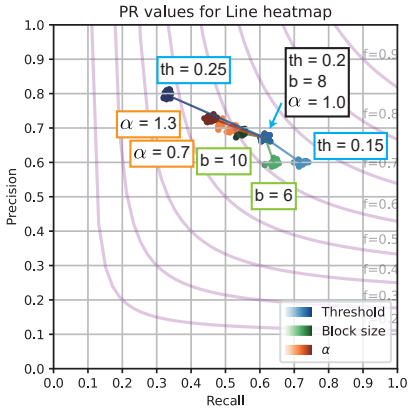

Figure 9. Sensitivity evaluation in threshold f_{th}, the block size b, and α in our method with respect to accuracy

threshold f_{th}	block size b	α	Subprocess	***-driven	Average process speed
0.2	8	1.0	SCARF	Event	25.96 Mev/s
			Detection	Process	1087.8 Hz
			Tracking	Process	2246.2 Hz
0.15	8	1.0	SCARF	Event	28.43 Mev/s
			Detection	Process	1581.2 Hz
			Tracking	Process	2116.8 Hz
0.25	8	1.0	SCARF	Event	28.18 Mev/s
			Detection	Process	765.9 Hz
			Tracking	Process	2610.8 Hz
0.2	10	1.0	SCARF	Event	27.18 Mev/s
			Detection	Process	1376.1 Hz
			Tracking	Process	2304.4 Hz
0.2	6	1.0	SCARF	Event	23.69 Mev/s
			Detection	Process	961.8 Hz
			Tracking	Process	1956.2 Hz
0.2	8	1.3	SCARF	Event	26.88 Mev/s
			Detection	Process	836.1 Hz
			Tracking	Process	1716.1 Hz
0.2	8	0.7	SCARF	Event	23.83 Mev/s
			Detection	Process	1025.4 Hz
			Tracking	Process	3123.5 Hz

Table 4. Sensitivity evaluation in threshold f_{th}, the block size b, and α in our method with respect to real-time performance

5. Conclusion

We have introduced a novel lattice-allocated real-time line segment detection and tracking pipeline that relies only on an event-based camera. This consists of parallelized an event-driven subprocess to store events efficiently with velocity-invariant manner in SCARF, and two process-driven subprocesses to initialize line segments by calculating endpoints and fitting scores in detection and keep updating line states by endpoints perturbation. Throughout all experiments, the proposed pipeline maintained line segments in real-time and improved the line segment accuracy.

Future work includes clustering multiple line segments from blocks into one "strong" line segment to improve temporal consistency. This makes line segments more stable, enabling potential robotics applications such as SLAM.

Acknowledgments

The authors thank Sony Interactive Entertainment for their research support and funding.

References

[1] Laure Acin, Pierre Jacob, Camille Simon-Chane, and Aymeric Histace. VK-SITS: a robust time-surface for fast event-based recognition. In *2023 Twelfth International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–6, 2023. 1

[2] I. Alzugaray and M. Chli. HASTE: multi-hypothesis asynchronous speeded-up tracking of events. In *31st British Machine Vision Virtual Conference (BMVC)*, 2020. 4

[3] Ryad Benosman, Sio-Hoi Ieng, Charles Clercq, Chiara Bartolozzi, and Mandyam Srinivasan. Asynchronous frameless event-based optical flow. *Neural Networks*, 27:32–37, 2012. 2

[4] Christian Brändli, Jonas Strubel, Susanne Keller, Davide Scaramuzza, and Tobi Delbruck. ELiSeD — an event-based line segment detector. In *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, pages 1–7, 2016. 2, 5, 1

[5] Bharatesh Chakravarthi, Aayush Atul Verma, Kostas Daniilidis, Cornelia Fermuller, and Yezhou Yang. Recent event camera innovations: A survey. Presented at NeVi 2024, Workshop on Neuromorphic Vision: Advantages and Applications of Event Cameras, ECCV2024, 2024. 1

[6] William Chamorro, Joan Solà, and Juan Andrade-Cetto. Event-based line slam in real-time. *IEEE Robotics and Automation Letters*, 7(3):8146–8153, 2022. 2

[7] Byeongpil Choi, Hanyeol Lee, and Chan Gook Park. Event-frame-inertial odometry using point and line features based on coarse-to-fine motion compensation. *IEEE Robotics and Automation Letters*, 10(3):2622–2629, 2025. 2, 5, 1

[8] Patrick Denis, James H. Elder, and Francisco J. Estrada. Efficient edge-based methods for estimating manhattan frames in urban imagery. In *Computer Vision – ECCV 2008*, pages 197–210, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. 1

[9] Alexander Dietsche, Giovanni Cioffi, Javier Hidalgo-Carrio, and Davide Scaramuzza. Powerline tracking with event cameras. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2021. 2, 5, 1

[10] Rika Sugimoto Dimitrova, Mathias Gehrig, Dario Brescianini, and Davide Scaramuzza. Towards low-latency high-bandwidth control of quadrotors using event cameras. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4294–4300, 2020. 2

[11] Lukas Everding and Jörg Conradt. Low-latency line tracking using event-based dynamic vision sensors. *Frontiers in Neurorobotics*, 12, 2018. 2

[12] Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3867–3876, 2018. 2

[13] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J. Davison, Jörg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(1):154–180, 2022. 1

[14] Arren Glover, Aiko Dinale, Leandro De Souza Rosa, Simeon Bamford, and Chiara Bartolozzi. luvharris: A practical corner detector for event-cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):10087–10098, 2021. 2, 1

[15] Arren Glover, Luna Gava, Zhichao Li, and Chiara Bartolozzi. EDOPT: Event-camera 6-dof dynamic object pose tracking. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 18200–18206. IEEE, 2024. 2, 1

[16] Rafael Grompone von Gioi, Jérémie Jakubowicz, Jean-Michel Morel, and Gregory Randall. LSD: a Line Segment Detector. *Image Processing On Line*, 2:35–55, 2012. 1, 2, 5

[17] Geonmo Gu, Byungsoo Ko, SeoungHyun Go, Sung-Hyun Lee, Jingeun Lee, and Minchul Shin. Towards light-weight and real-time line segment detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(1):726–734, 2022. 1

[18] A. Gómez Eguíluz, J.P. Rodríguez-Gómez, J.R. Martínez-de Dios, and A. Ollero. Asynchronous event-based line tracking for time-to-contact maneuvers in uas. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5978–5985, 2020. 2

[19] Kun Huang, Yifan Wang, Zihan Zhou, Tianjiao Ding, Shenghua Gao, and Yi Ma. Learning to parse wireframes in images of man-made environments. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 626–635, 2018. 6

[20] Massimiliano Iacono, Stefan Weber, Arren Glover, and Chiara Bartolozzi. Towards event-driven object detection with off-the-shelf deep learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, 2018. 5, 1

[21] Mikihiro Ikura, Cedric Le Gentil, Marcus G. Müller, Florian Schuler, Atsushi Yamashita, and Wolfgang Stürzl. RATE: Real-time asynchronous feature tracking with event cameras. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11662–11669, 2024. 4

[22] Cedric Le Gentil, Florian Tschopp, Ignacio Alzugaray, Teresa Vidal-Calleja, Roland Siegwart, and Juan Nieto. IDOL: A framework for imu-dvs odometry using lines. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5863–5870, 2020. 2

[23] Zibin Liu, Banglei Guan, Yang Shang, Qifeng Yu, and Laurent Kneip. Line-based 6-dof object pose estimation and tracking with an event camera. *IEEE Transactions on Image Processing*, 33:4765–4780, 2024. 2

[24] Zibin Liu, Banglei Guan, Yang Shang, Yifei Bian, Pengju Sun, and Qifeng Yu. Stereo event-based, 6-dof pose tracking

for uncooperative spacecraft. *IEEE Transactions on Geoscience and Remote Sensing*, 63:1–13, 2025. 2

[25] Jacques Manderscheid, Amos Sironi, Nicolas Bourdis, Davide Migliore, and Vincent Lepetit. Speed invariant time surface for learning to detect corner points with event-based cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10245–10254, 2019. 2

[26] Branislav Micusik and Horst Wildenauer. Structure from motion with line segments under relaxed endpoint constraints. In *2014 2nd International Conference on 3D Vision*, pages 13–19, 2014. 1

[27] Anton Mitrokhin, Chengxi Ye, Cornelia Fermüller, Yiannis Aloimonos, and Tobi Delbruck. EV-IMO: Motion segmentation dataset and learning pipeline for event cameras. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6105–6112, 2019. 1

[28] Elias Mueggler, Chiara Bartolozzi, and Davide Scaramuzza. Fast event-based corner detection. In *British Machine Vision Conference (BMVC)*, 2017. 2

[29] Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam. *The International Journal of Robotics Research*, 36(2):142–149, 2017. 5, 6

[30] Rémi Pautrat, Daniel Barath, Viktor Larsson, Martin R. Oswald, and Marc Pollefeys. DeepLSD: Line segment detection and refinement with deep image gradients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17327–17336, 2023. 1

[31] Chengyu Qiao, Tingming Bai, Zhiyu Xiang, Qi Qian, and Yunfeng Bi. Superline: A robust line segment feature for visual slam. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5664–5670, 2021. 1

[32] Xinya Wang, Haitian Zhang, Huai Yu, and Xianrong Wan. EvLSD-IED: Event-based line segment detection with image-to-event distillation. *IEEE Transactions on Instrumentation and Measurement*, 73:1–12, 2024. 2

[33] Chi Xu, Lilian Zhang, Li Cheng, and Reinhard Koch. Pose estimation from line correspondences: A complete analysis and a series of solutions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1209–1222, 2017. 1

[34] Huai Yu, Hao Li, Wen Yang, Lei Yu, and Gui-Song Xia. Detecting line segments in motion-blurred images with events. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–16, 2023. 2, 5, 1

[35] Yichao Zhou, Haozhi Qi, and Yi Ma. End-to-end wireframe parsing. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 962–971, 2019. 6

[36] Alex Zihao Zhu, Dinesh Thakur, Tolga Özaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. *IEEE Robotics and Automation Letters*, 3 (3):2032–2039, 2018. 5, 6