ASL-SLAM: An asynchronous formulation of lines for SLAM with Event Sensors

XIAOQI NONG

CVSSP, University of Surrey, Guildford, UK, x.nong@surrey.ac.uk

SIMON HADFEILD

CVSSP, University of Surrey, Guildford, UK, s.hadfield@surrey.ac.uk

The development of industrial automation is closely related to the evolution of mobile robot positioning and navigation mode. In this paper, we introduce ASL-SLAM, the first line-based SLAM system operating directly on robots using the event sensor only. This approach maximizes the advantages of the event information generated by a bio-inspired sensor. We estimate the local Surface of Active Events (SAE) to get the planes for each incoming event in the event stream. Then the edges and their motion are recovered by our line extraction algorithm. We show how the inclusion of event-based line tracking significantly improves performance compared to state-of-the-art frame-based SLAM systems. The approach is evaluated on publicly available datasets. The results show that our approach is particularly effective with poorly textured frames when the robot faces simple or low texture environments. We also experimented with challenging illumination situations to order to be suitable for various industrial environments, including low-light and high motion blur scenarios. We show that our approach with the event-based camera has natural advantages and provides up to 85% reduction in error when performing SLAM under these conditions compared to the traditional approach.

CCS CONCEPTS • Computer systems organization → Embedded systems; Redundancy; Robotics • Networks → Network reliability

Additional Keywords and Phrases: Event Camera, SLAM, Line Feature, Robot Navigation

ACM Reference Format:
Xiaoqi Nong, CVSSP, University of Surrey, Simon Hadfeild, CVSSP, University of Surrey. 2021. ASL-SLAM: An asynchronous formulation of lines for SLAM with Event Sensors. In Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY. ACM, New York, NY, USA, 12 pages.

1 INTRODUCTION

In recent years applications of robots have transcended the industrial assembly lines, making unmanned factories or package delivery by robots a reality. An essential condition for intelligent robots to conduct industrial tasks is localization and navigation. To be specific SLAM (simultaneous localization and mapping) algorithms attempt to figure out the robot's motion in an unknown environment, and navigation its own path by observing the neighbourhood, while constructing a map of that environment. New sensors have historically changed the way we deal with these types of tasks. The emergence of cheap commercial depth sensors around 2010 led to a massive growth in computer vision research. More recently, new dynamic vision sensors called event cameras are beginning to emerge, which are activity-driven. Techniques have been proposed to use these sensors for feature detection and tracking [11], 3D reconstruction [12], object recognition [13], simultaneous localization

and mapping [14] and more. This paper will attempt to introduce the first formalism for undertaking line-based projective geometry using event streams, and demonstrate its applications to SLAM.

Event cameras output compressed visual data in the form of a visual event stream. The data provides an increased temporal resolution and lower latency compared to conventional images. An event represents a brightness change for one pixel. They form a event stream which record the positions and the polarity of intensity changes across all pixels. The event camera also provides absolute intensity [9] and allows a longer waiting time between two coming events, which means it has better performance in low-light industrial environment than traditional camera. These characters make it possible to combine the benefits of traditional cameras with the unique properties of event-based sensors. This has huge potential for computer vision and high-speed robotics. In essence, visual information from event cameras is asynchronously acquired. Although it sends no information when there is no movement in the scene, it does not have to wait for the next frame before transmitting the signal when motion does occur. This paradigm shift means it has the advantages of high temporal resolution, low latency, high dynamic range and low power consumption.

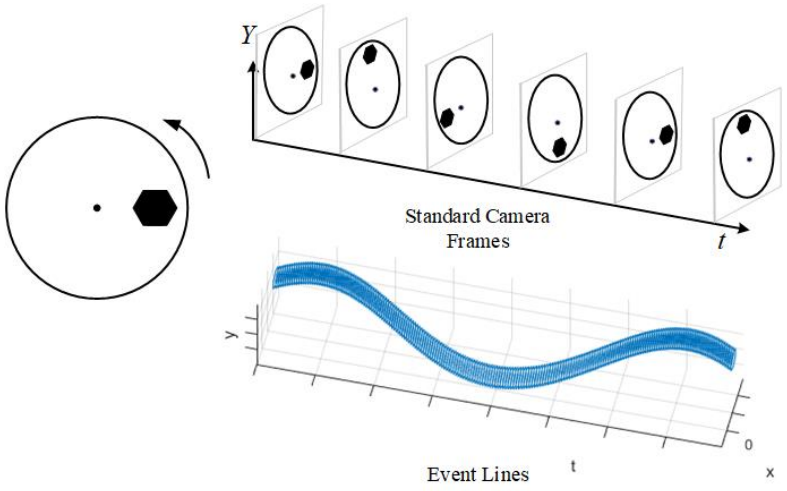

Figure 1: A comparison of a traditional frame-based output (top right) and asynchronous line detections. The scene comprises a black hexagon rotating on a disk. The normal camera outputs frames at a fixed rate, instead the event camera produces the stream of events responding to brightness changes from which asynchronous lines can be extracted.

Traditional visual sensors like monocular, stereo and RGBD cameras have been extensively applied to visual odometry [1], [2], and SLAM [4] for many years. Efforts have been made to improve the efficiency of these approaches. However, the frame-based data acquisition places a fundamental limit on the computation cost. Many algorithms struggle to operate in complex and noisy scenes, on embedded robotics hardware. Standard cameras also can be affected by motion blur during fast movement. In contrast event cameras do not suffer from motion blur and have greatly reduced data bandwidth [10], which means event cameras are promising for visual odometry or SLAM tasks. In this paper we tackle the problem of line-based SLAM with event cameras in natural scenes and arbitrary 6-DoF motion. We integrate these alongside a low-speed visual point tracking in a framework inspired by PL-SLAM [17]. In contrast to most previous work using event cameras, we make full use of asynchronous data to produce line feature tracks with high temporal resolution (in Figure.1), by avoiding the

need to accumulate events into frames. To achieve this, we design a line feature extraction approach that works directly on the event streams. **Contributions**: In summary, our contributions are:

- A novel asynchronous line detection method based directly on the event stream.
- The first publicly available SLAM system built on event cameras, and the first event-only SLAM system built on asynchronous lines.
- An extensive experimental evaluation is conducted on asynchronous features compared with other SLAM methods on publicly available datasets, demonstrating that the system is computationally efficient, running in real-time on a standard CPU.

2 RELATED WORK

The advent of event sensors has the potential to revolutionize our approach to visual perception for mobile robots. This process is still in its early stages. Initial forays have been undertaken in many areas using Event cameras. However, there has been little opportunity for the field to iterate these ideas, and a lot of prior work still falls back on the frame-based aggregation approach.

2.1 Feature-based SLAM/VO Using Traditional Cameras

ORB-SLAM [18] is one of the most widely used traditional feature-based approaches to SLAM. Under normal operating conditions it can provide robust camera tracking and mapping. Subsequently ORB-SLAM2 [16] was proposed to improve performance by using bundle adjustment. Both approaches rely on fast and continuous tracking of ORB features. However, these traditional feature methods fail when the mobile platform comes to a poorly texture environment or suffers from motion blur. These issues led researchers to explore a more robust representation such as Edge SLAM [20]. This method detects edge points in frames and tracks them using optical flow for data association. This line-based approach is shown to work well in both low-textured and highly-textured environments. In [21], direct lines are used to guide feature selection rather than key points which increase the efficiency. Both [22] [23] take feature points to get straight lines and perform position estimation in monocular VO. However, this idea is extended by using stereo camera in [19]. All of these line-based approaches are shown to have high computational efficiency.

2.2 Event-based feature detection and tracking

As shown above corner detection and tracking using frame-based cameras is fundamental to most visual odometry. As such, it is unsurprising that event cameras have primarily been applied to asynchronous feature point tracking, providing the benefits of high dynamic range and microsecond latency [24]. How to detect features using event data becomes the fundamental question. Significant research effort has been dedicated to reconstructing pseudo-frame-based images from event data, and then applying traditional frame-based features such as Harris on these event frames [25] [26]. This unfortunately introduces a synchronization delay that removes the primary benefit of the event camera. In contrast to this, [27] [28] extend the normal features in order to detect them directly on event streams. This was taken a step further by [29] [30] which also performed feature tracking on the event stream. [31] went beyond flow-based point tracking, and proposed a novel local region descriptor for corner-events with a corresponding tracker. [32] proposed the use of multiple pools of trackers, to track different visual features in real-time, while handling position variations, orientation and scale.

2.3 Event-based VO/SLAM

As discussed above, there has been a great deal of work on feature point detection and tracking using event cameras. It is unsurprising that some of these techniques have been adapted for VO/SLAM. Similar to the traditional camera, these techniques cover both monocular event-based VO and stereo event-based VO. A ground breaking work on event-based visual odometry with the monocular DAVIS sensor was introduced in [37]. This work takes the approach of detecting features on the standard grayscale frames and tracking them asynchronously with event streams to complete the 6-DOF motion estimation of the sensor. Some researchers were able to reconstruct intensity images from the event stream through a combination of event-based feature tracking techniques and event-based 3D reconstruction techniques [38]. The work in [39] performs VO with stereo event-based cameras instead of one monocular event camera. There are fused event-based tracking algorithms with an inertial measurement unit (IMU) in [40], providing metric tracking of 6-DOF pose. All of these prior techniques focus on point-based features, which are sparse and hard to extract from event camera streams. Additionally, [43] proposed a novel event line-feature but achieved VO by fusing IMU sensor and event camera. However, there has been no work focusing on event-only line-based VO/SLAM to the date. Thus in this paper we demonstrate the first use of asynchronous line features within a SLAM pipeline shown in Figure 2. Unlike PL-SLAM, we used event lines to track and save the computation time of traditional line-feature.

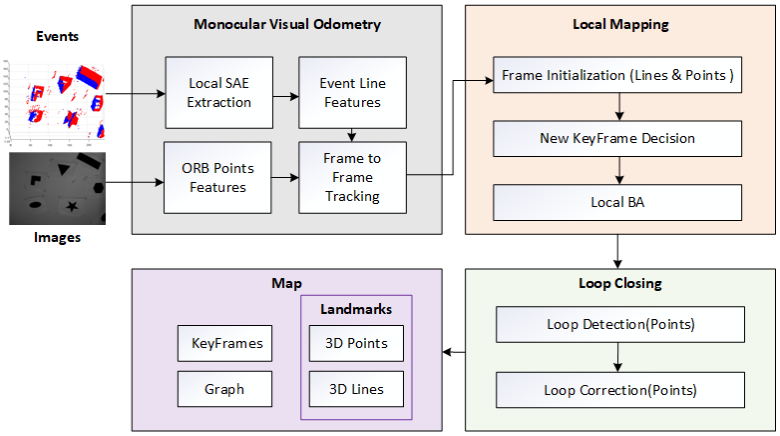

Figure 2: The scheme of ASL-SLAM is composed of three main threads: Data Extraction & Tracking, Local Mapping and Loop Closing.

3 METHORD OVERVIEW

The event stream is equivalent to a high-speed camera taking pictures at a rate of thousands of frames per second, with the additional benefit of far less redundant data [8]. The event camera has a higher temporal resolution because it responds to intensity changes in the environment independently and asynchronously for each pixel. The flow chart shows the working process of the event camera in Figure 3. The output of the event camera is a discrete representation of intensity change (log intensity) and the position of the corresponding pixel. The events have ON/OFF polarity which correspond to the increase and decrease of brightness. The threshold explained the details of events' trigger, the evolution of pixel's voltage is V_p which can be roughly interpreted as the intensity of light received by the pixel over time. It shows the corresponding generation of ON

(voltage increases above change threshold) and OFF (voltage decreases) events, from which the change of V_p can be reconstructed. The event camera captures the events with a pixel array before formulating the event stream through the peripheral circuit and outputting the stream using a shared digital output bus. This route will take advantages from a kind of address event representation (AER) readout, which grantees a faster read speed.

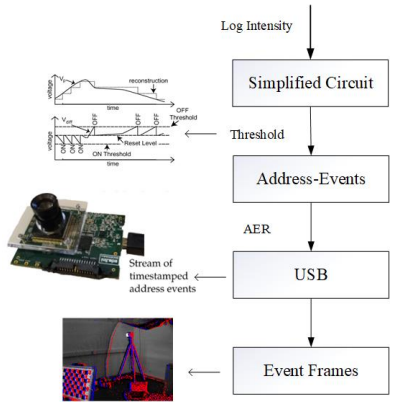

Figure 3: Working process of event camera.

Our method illustrated in Figure 3 aims to estimate the camera motion by extracting line information from event streams. The pipeline of our system is inspired by the integrated point and line tracking of PL-SLAM, but using our asynchronously extracted event lines. The main idea behind the algorithm is to identify the coherent space-time event surfaces using the local SAE. These event surfaces are then used to define both the line features in the image and their motion. Local Bundle Adjustment is applied to optimize the pose of all lines after obtaining the initial line feature set, finally further correspondences can be established by projecting the local map onto the image.

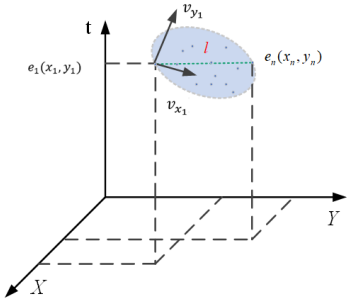

Figure 4: An event line is extracted from the SAE with line orientation and amplitude of motion defined by the orientation of the SAE. Line extremities are determined by the inlier points within the local SAE neighbourhood.

3.1 Asynchronous Line Extraction

We define the event stream as the set of all events $\mathbf{E} = \{ e_0, e_1 \ldots e_N \}$, where $e_i \in \mathbf{R}^3$. Each event is constructed as $e_i = [x, y, t]$, where $[x, y]$ is the position of the event in the event frame coordinates at time t. For every

incoming event, we define an event neighbourhood which includes all events with a similar time and position. This SAE neighbourhood is used to provide an estimate of edge of orientation and motion as shown in Figure 4.

More formally, a spatio-temporal neighbourhood of events Ω_e is defined as the set of all events falling within a window of size $2L \times 2L \times 2\Delta T$ centered on event e:

$$\begin{aligned} \Omega_{e_i} = \{e_j,\ where\ x_j \in (x_i - L, x_i + L), \\ \mathrm{y}_j \in (y_i - L, y_i + L),\ t_j \in (t_i - \Delta T, t_i + \Delta T)\} \end{aligned} \quad (1)$$

We parametrize a space-time plane as $\beta = \begin{pmatrix} a & b & c & d \end{pmatrix}^T$. We note that for any event e_i which lies on this plane, the following equality must satisfy: $\beta\,[x, y, t, 1] = 0$.

For each incoming event, we optimize the plane $\beta to extract the local Surface of Active Events. The optimization process is defined as:

$$\beta^* = \arg\min_{\beta} \sum_{e_j \in \Omega_{e_i}} \left| \beta\,[x_i, y_i, t_i, 1]^T \right|^2 \quad (2)$$

Once the initial plane candidate has been computed, we extract the set of inlier events from the neighbourhood as:

$$\Omega_{e_i}^* = \{e_j,\ where\ \left| \beta\,[x_i, y_i, t_i, 1]^T \right| < \lambda,\ \forall e_j \in \Omega_{e_i}\} \quad (3)$$

We then repeat equation 2 to refine the plane parameters on the inlier events set. We define the update size as $\Delta\beta = \left\| \beta - \beta^* \right\|$, and iterate equations 2 and 3 until $\Delta\beta < \lambda_2$, at which point the local SAE has converged.

3.2 Synchronous Data Association with Asynchronous Lines

Although the asynchronous event information is discrete, the space-time planes computed above provide a continuous representation. This makes it possible to extract the corresponding line at any intermediate timestamp. As such it becomes possible to synchronize the asynchronous event lines with the visual feature points extracted from the frame-based camera. Intuitively we simply "slice" the SAE with a horizontal plane at the desired timestamp, and extract the line along the intersection of these planes. Thus we use a homogeneous definition of line $l = \left[a/d,\ b/d,\ ct_f/d \right]$ where a,b,c,d are the parameters from the corresponding SAE plane and t_f is the timestamp of the frame which the line is being synchronized too. Any 2D point which lies on the 2D visual line at frame t_f must then satisfy the equality: $l_i^T\ \left[x,\ y,\ 1\right]\ =\ 0$.

Moreover, it can be inferred that all the events on the line should have the same velocity, and the velocity remains constant during this time interval. This provides the tracking for the lines. The velocity $\left[v_x, v_y \right]$ can be obtained from the local SAE by computing the slope between the temporal axis and the spatial axes. After obtaining the parameters of the synchronized 2D lines, and their velocities, we compute the data association with the 3D line segments currently contained in the map. The 3D end points of these line segments are

projected into the image space. $M, N \in \mathbf{R}^3$ are the extremities of the line segment. As in [15] we define, the algebraic point-line error E_{pl} for a 3D point as:

$$E_{pl}(\theta, M_{t_f}, l_i) = (l_i)^T \pi(\theta, M_{t_f}) \quad (4)$$

where the pose parameters $\theta = \{R, t\}$ include the rotation and translation parameters which align the camera and the world coordinate systems, and π gives out $\mathbf{R}^3$ which is the homogeneous projected 2D point. Furthermore the algebraic line segment error E_l is defined as the sum of squares of the two event-line errors for the 3D line segment endpoints:

$$E_l(\theta, M_{t_f}, l_i) = E_{pl}(\theta, M_{t_f}, l_i)^2 + E_{pl}(\theta, N_{t_f}, l_i)^2 \quad (5)$$

We associate the newly synchronized event lines with the closest line segment in the current map $\mathbf{M}$, assuming the algebraic error is less than λ_3:

$$l \longleftrightarrow \begin{cases} \underset{[M_{t_f}, N_{t_f}] \in \mathbf{M}}{\arg\min}\ E_l(\theta, M_{t_f}, N_{t_f}, l_i), \text{ if } E_l(\theta, M_{t_f}, N_{t_f}, l_i) < \lambda_3 \\ 0, \text{ otherwise} \end{cases} \quad (6)$$

3.3 Local Bundle Adjustment with Points and Synchronized Lines

Once the synchronized event lines are associated with the line segments in the map, we combine them with frame-based point feature correspondences. To get the optimized results of the camera pose, a bundle adjustment (BA) of the local map is used. This constrains θ to an $SE(3)$ pose for each keyframe(KF). After we associate the data with the i^{th} KF, let $X_{ij} \in \mathrm{R}^3$ be the generic j^{th} point of the map. The projection error e_{ij} represents the 2D distance between the observation x_{ij} of the i^{th} KF:

$$\varepsilon_p(i, j) = x_{ij} - \pi(\theta_j, X_{ij}) \quad (7)$$

For the event lines defined in the previous section the BA uses the same error function as that used for data association. Namely, we define the error function by the distances between the projected endpoints of the 3D line and its corresponding infinite line in the 2D image plane:

$$\varepsilon_l(i, k) = \begin{bmatrix} l_{ik}{}^T \pi(\theta_j, M_j) \\ l_{ik}{}^T \pi(\theta_j, N_j) \end{bmatrix} \quad (8)$$

where M_j and N_j refer to the 3D endpoints of the line segments in the world coordinate and l_{ik} is the equation of the 2D line in the corresponding i^{th} KF. Because of the unknown camera pose and the noise of observation, the problem is defined as a joint optimization over the camera pose θ, map points $\mathbf{X}$ and map lines $\mathbf{M}$:

$$\theta^*, \mathbf{X}^*, \mathbf{M}^* = \underset{\theta, \mathbf{X}, \mathbf{M}}{\arg\min} \sum_{k=0}^{K} [\sum_{j=0}^{P} \varepsilon_p(i, j)^T u_{\varepsilon_p(i,j)}{}^{-1} \varepsilon_p(i, j) + \sum_{i=0}^{l} \varepsilon_l(i, k)^T u_{\varepsilon_l(i,k)}{}^{-1} \varepsilon_l(i, k)] \quad (9)$$

where K, P, L represent the number of KF, points and lines respectively. $u_{e_{ij}}^{-1}$ and $u_{e_{ik}}^{-1}$ are the covariance matrices for the keypoints and line endpoints. Given this definition the problem can be iteratively solved by a Damped Newton method which is more robust than standard Guassian Newton methods.

3.4 Mapping With Points and Asynchronous Lines

It is known that current frame-based SLAM methods sometimes fail when running in low texture environments which lack keypoints. In contrast to using points, event lines are more robust for map initialization between two frames. To initialize the frame relationship and further pose recovery, we make an assumption that the camera's angular acceleration is small between two times steps. This means for three consecutive poses, the rotational transition between each camera view is equal. After accounting for the change in camera orientation, the 3 observations of the line should be colinear with each other. We can quantify this by checking that the cross product of any 2 of the lines must also be perpendicular to the third line. This constraint can be written as:

$$l_t^T((R_{t-1\to t}l_{t-1})\times(R_{t+1\to t}l_{t+1}))=0 \quad (10)$$

Moreover, for this small rotation R can be approximated as a skew symmetric matrix. For this parametrization, a polynomial solver which produces up to eight solutions is applied to solve three quadratic with three unknowns r_1, r_2 and r_3 after we have three matched lines. For each rotation candidate we can estimate the corresponding t using the trifocal tensor equations [42]. Then one out of the eight possible solutions is selected to minimize R.

4 EXPERIMENTS

In this section we give the performance evaluation of the ASL-SLAM in several scenes from two different datasets. We compare our system with the current state-of-art frame-based SLAM methods, such as ORB-SLAM and PL-SLAM by employing their open resource implementations. There are no publicly available monocular SLAM algorithms using event cameras which we can compare against. We assess the computation time of the detection and tracking for the asynchronous lines with other novel event feature algorithms. All experiments were conducted with an Intel Core i7-8700K (12$\times @ 3.70$GHz), Ubuntu 18.04, ROS melodic.

4.1 Dataset and Parameter Settings

In order to test the validity and efficiency of the asynchronous line extraction separately from the SLAM framework we use the event camera datasets proposed by [6]. It was generated using a DAVIS240C and provides the images, events, IMU estimations, and calibration with different sensors as well as ground truth obtaining by a motion-capture system. We use the shape ROS bag which has the rotation and translation of some typical shapes in order to show the detection result more clearly. The approaches work directly on event streams under two different scenes. In this part, we set $\lambda_1 = 0.001$, $\lambda_2 = 0.05$, $\Delta T = 0.01s$.

To test the robustness of the full ASL-SLAM pipeline under varying illuminations, we use the VIVID (Vision for Visibility Dataset) [41]. This dataset captured unconventional visual data obtained from various lighting and motion conditions. The dataset provides normal and poor illumination sequences captured by RGB-D camera, and event data using DAVIS240C. They also provide moving sequence under robust motion as well as unstable

motion making it possible to detect the effects of motion blur. These four scenarios allow us to provide a broad comparison of our SLAM performance. The resolution of the RGBD image is 640×480 .

4.2 Asynchronous Feature Extraction Benchmark

There are four existing event features mentioned in section 2.2. For the first experiment we use these event corner detectors and our method to extract asynchronous event features from the event streams. Table \ref{table1} illustrates the average processing time in µs of one single event and the maximal event bandwidth in millions of events per second. It also shows the reduction rate, which explains the number of detected features over the number of input events. The results show that our ASL algorithm can detect a similar number of features to event corner detectors with a similar level of efficiency. According to the results, our method shows consistent performance under the different types of motion, and it can deal with the highest incoming event rate among the 5 methods. In fact, the neighbourhood of the incoming events are being processed jointly in our method. This is likely the reason why the max event rate of our method stays the highest. In addition, the number of lines is lower than feature points the efficiency of the following SLAM system is also improved.

Table 1: Performance of different event corner detector

Scene	Algorithm	eHarris	Fa-harris	eFast	Arc*	ASL
Shape rotation	Reduction rate (%)	85.66	93.30	78.16	77.90	93.14
	Time per event (us/ev)	5.92	0.90	1.88	0.19	2.32
	Max. event rate(Mev/s)	0.17	1.11	0.53	5.29	10.71
Shape translation	Reduction rate (%)	94.24	98.3	94.19	91.60	95.52
	Time per event(us/ev)	4.23	0.46	1.50	0.12	1.78
	Max. event rate(Mev/s)	0.24	2.19	2.00	8.05	13.36

4.3 SLAM performance

In this part we explore the robustness of our ASL-SLAM algorithm in VIVID dataset. Example frames from the four scenarios of the VIVID dataset are shown in Figure 5. We can see that the two global light images have brighter illumination than the local light do. At the same time, the frames from the dataset of unstable motion are more blurred than in robust motion which means tracking may lost when using traditional keypoints. Since there are no publicly available pipe line for event-based SLAM algorithm, we choose a common baseline method ORB-SLAM2 which is more stable and robust than ORB-SLAM3, and a state-of-art line-based method PL-SLAM.

The running time of each method is calculated, and tracking performance is shown in Table 2. It is obvious that our ASL method significantly outperforms the frame-based baseline ORB-SLAM2 and the line-based PL-SLAM. The tracking time has decreased by more than 20% in comparison to the other two methods, despite the high rate of incoming events. It can be seen that under the global light condition, all three methods spent more time tracking which means there are more features when it is brighter in the room. Regardless of the light condition, our method only cares about the number of events. The Absolute Trajectory Error (ATE) is also utilized to compare the accuracies of the different techniques.

Figure 5: Frames of four different experimental conditions. Far left is local light and robust motion, centre left is global light and robust motion, centre right is local light and unstable motion, far right is global light and unstable motion.

Before computing the error, all trajectories are aligned using a similarity warp and scaled by calculating the Euclidean distance between the estimated trajectory and the ground truth. We then evaluated the ATE by calculating its RMS (Root mean square) under all four scenes which is demonstrated in Table 3.

Table 2: Comparing of tracking time for each method(ms)

Scene	Tracking Time	ORB-SLAM2	PL-SLAM	ASL-SLAM
Local Robust	Medium	61.6	62.9	**45.6**
	Mean	62.7	64.2	**49.6**
Global Robust	Medium	66.2	70.6	**48.1**
	Mean	66.1	70.2	**52.1**
Local Unstable	Medium	55.3	48.8	**38.1**
	Mean	56.8	50.3	**39.0**
Global Unstable	Medium	50.1	56.4	**43.7**
	Mean	49.9	57.2	**44.5**

It can be seen from the results that our method offers extremely robust performance under adverse light and motion. Under the easiest condition with low motion blur and strong lighting (global robust) ORB-SLAM2 is able to slightly outperform ASL-SLAM. Under any combination of adverse conditions ASL-SLAM is able to outperform competitors by 15\%-85\%. It can be inferred that the efficiency of our method results in less error than PL-SLAM does.

In particular scenarios with motion blur are particularly challenging for previous approaches. Meanwhile ASL-SLAM is able to make use of the increased event density resulting from unstable motions to continue extracting reliable line features. However, events can be obtained as long as the event camera moves, this also means the event lines can be detected. In this case, our ASL-SLAM can provide a more stable and robust localization.

	ORB	PL	ASL
Local Robust	53.54	54.73	**45.38**
Global Robust	**24.11**	34.69	27.96
Local Unstable	526.53	498.72	**321.3**
Global Unstable	324.50	306.97	**46.80**

Table 3: Absolute trajectory error (RMS)[t:mm]

5 CONCLUSION

In this work, we propose ASL-SLAM, an event-based visual SLAM approach that covers both feature points and event lines simultaneously. Our SLAM architecture is able to operate with low latency line features at a high accuracy and data efficiency that can be suitable for industrial environments. It consistently outperforms the traditional feature point-based approaches in the presence of motion blur or low textured scenes. We also demonstrate our approach performs well in low-light situations, where traditional approaches are prone to failure. On the VIVID dataset this led to consistent improvement of up to 85% reduction in error compared to the current competing methods. For future work, further exploration of asynchronous line-based SLAM and incorporation of higher dimensional geometric primitives like planes could be fruitful.

ACKNOWLEDGMENTS

Xiaoqi Nong is financially supported by China Scholarship Council and the University of Surrey Scholarship for her PhD study.

REFERENCES

[1] Nister D, Naroditsky O, Bergen J. Visual odometry. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR),1: I-I,2004.

[2] Kitt B, Geiger A, Lategahn H. Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme. 2010 ieee intelligent vehicles symposium. IEEE, 2010: 486-492.

[3] Forster C, Pizzoli M, Scaramuzza D. SVO: Fast semi-direct monocular visual odometry. In 2014 IEEE international conference on robotics and automation (ICRA), 15-22, 2014.

[4] Bailey T, Durrant-Whyte H. Simultaneous localization and mapping (SLAM): Part II. IEEE robotics \& automation magazine, 2006, 13(3): 108-117.

[5] Steffen L, Reichard D, Weinland J, et al. Neuromorphic stereo vision: A survey of bio-inspired sensors and algorithms. Frontiers in neurorobotics, 2019, 13: 28.

[6] E Mueggler, Rebecq H, Gallego G, et al. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. The International Journal of Robotics Research, 2017, 36(2): 142-149.

[7] T Delbruck, Linares-Barranco B, Culurciello E, et al. Activity-driven, event-based vision sensors. In 2010 IEEE International Symposium on Circuits and Systems. IEEE, 2010: 2426-2429.

[8] Brandli C , Berner R , Yang M , et al. A 240x180 130dB 3us Latency Global Shutter Spatiotemporal Vision Sensor. IEEE Journal of Solid-State Circuits, 2014, 49(10):2333-2341.

[9] Simon Chane C, Ieng S H, Posch C, et al. Event-based tone mapping for asynchronous time-based image sensor. Frontiers in neuroscience, 2016, 10: 391.

[10] Gallego G, Delbruck T, Orchard G, et al. Event-based vision: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020.

[11] Lagorce X, Meyer C, Ieng S H, et al. Asynchronous event-based multikernel algorithm for high-speed visual features tracking. IEEE transactions on neural networks and learning systems, 2014, 26(8): 1710-1720.

[12] Baudron A, Wang W, Cossairt O, et al. E3D: Event-Based 3D Shape Reconstruction. arXiv preprint arXiv:2012.05214, 2020.

[13] Bi Y, Chadha A, Abbas A, et al. Graph-based object classification for neuromorphic vision sensing. In the IEEE/CVF International Conference on Computer Vision. 2019: 491-501.

[14] Rebecq H, Horstschafer T, Gallego G, et al. Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real time. IEEE Robotics and Automation Letters, 2016, 2(2): 593-600.

[15] A. Vakhitov, J. Funke, and F. Moreno-Noguer. Accurate and linear time pose estimation from points and lines. In ECCV, 2016.

[16] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," in IEEE Transactions on Robotics, vol. 33, no. 5, 1255-1262, 2017.

[17] Pumarola A, Vakhitov A, Agudo A, et al. PL-SLAM: Real-time monocular visual SLAM with points and lines. 2017 IEEE international conference on robotics and automation (ICRA). IEEE, 2017: 4503-4508.

[18] Mur-Artal R, Montiel J M M, Tardos J D. ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE transactions on robotics, 2015, 31(5): 1147-1163.

[19] R. Gomez-Ojeda, D. Zuniga-Noel, F.-A. Moreno, D. Scaramuzza, and J. Gonzalez-Jimenez. PL-SLAM: a Stereo SLAM System through the Combination of Points and Line Segments.

[20] Soumyadip Maity, Arindam Saha, and Brojeshwar Bhowmick. Edge slam: Edge points based monocular visual slam. In Proceedings of the IEEE International Conference on Computer Vision Workshops, pages 2408-2417, 2017.

[21] Shi-Jie Li, Bo Ren, Yun Liu, Ming-Ming Cheng, Duncan Frost, and Victor Adrian Prisacariu. Direct line guidance odometry. In 2018 IEEE international conference on Robotics and automation (ICRA), pages 1-7. IEEE, 2018.

[22] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer. Pl-slam: Real-time monocular visual slam with points and lines. In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 4503-4508, 2017.

[23] Paul Smith, ID Reid, and Andrew J Davison. Real-time monocular slam with straight lines. 2006.

[24] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Joerg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based Vision: A Survey. pages 1?25, 2019.

[25] Valentina Vasco, Arren Glover, and Chiara Bartolozzi. Fast event-based harris corner detection exploiting the advantages of event-driven cameras. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016.

[26] Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-based feature tracking with probabilistic data association. In 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017.

[27] Xavier Clady, Sio Hoi Ieng, and Ryad Benosman. Asynchronous event-based corner detection and matching. Neural Networks, 2015, 66: 91-106.

[28] Ruoxiang Li, Dianxi Shi, Yongjun Zhang, Kaiyue Li, and Ruihao Li. Fa-harris: A fast and asynchronous corner detector for event cameras. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pages 6223?6229, 09, 2019.

[29] I. Alzugaray and M. Chli. Asynchronous corner detection and tracking for event cameras in real time. IEEE Robotics and Automation Letters, 3(4):3177-3184, 2018.

[30] Elias Mueggler, Chiara Bartolozzi, and Davide Scaramuzza. Fast Event-based Corner Detection. In British Machine Vision Conference (BMVC), 1-8, 2017.

[31] I. Alzugaray and M. Chli. Ace: An effcient asynchronous corner tracker for event cameras. In 2018 International Conference on 3D Vision (3DV), pages 653-661, 2018.

[32] Xavier Lagorce, Cedric Meyer, Sio Hoi Ieng, David Filliat, and Ryad Benosman. Asynchronous event-based multikernel algorithm for high-speed visual features tracking. IEEE Transactions on Neural Networks and Learning Systems, 26(8):1710-1720, 2015.

[33] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio-Hoi Ieng, and Chiara Bartolozzi. Event-based visual flow. IEEE transactions on neural networks and learning systems, 25(2):407?417, 2013.

[34] Patrick Bardow, Andrew J Davison, and Stefan Leutenegger. Simultaneous optical flow and intensity estimation from an event camera. IEEE Conference on Computer Vision and Pattern Recognition, pages 884-892, 2016.

[35] Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. IEEE Conference on 32 Computer Vision and Pattern Recognition, pages 3867-3876, 2018.

[36] E. Pitkowska, A. N. Belbachir, S. Schraml, and M. Gelautz. Spatiotemporal multiple persons tracking using dynamic vision sensor. In 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pages 35-40, 2012.

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

[38] 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, April 2017.

[39] Zhou Y, Gallego G, Shen S. Event-based Stereo Visual Odometry. IEEE Transactions on Robotics, 2020.

[40] A. Z. Zhu, N. Atanasov and K. Daniilidis. Event-Based Visual Inertial Odometry. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5816-5824, 2017.

[41] A. J. Lee, Y. Cho, S. Yoon, Y. Shin, and A. Kim. ViViD : Vision for Visibility Dataset. ICRA Workshop on Dataset Generation and Benchmarking of SLAM Algorithms for Robotics and VR/AR, 2019.

[42] Hartley R I. Lines and points in three views and the trifocal tensor. International Journal of Computer Vision, 1997, 22(2): 125-140.

[43] Le Gentil C, Tschopp F, Alzugaray I, et al. IDOL: A framework for IMU-DVS odometry using lines. 2020 IROS, 5963-5870.