Edge Detection for Event Cameras using Intra-pixel-area Events

Sangil Lee
sangil07@snu.ac.kr
Haram Kim
rlgkfka614@gmail.com
H. Jin Kim
hjinkim@snu.ac.kr

Automation and Systems Research Institute, Department of Mechanical and Aerospace Engineering, Seoul National University, Seoul, Korea, Republic of

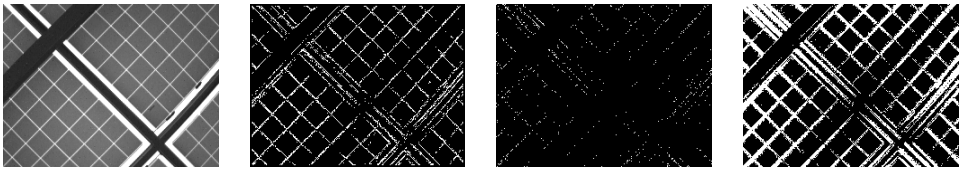
Abstract

In this work, we propose an edge detection algorithm by estimating a lifetime of an event produced from dynamic vision sensor (DVS), also known as event camera. The event camera, unlike traditional CMOS camera, generates sparse event data at a pixel whose log-intensity changes. Due to this characteristic, theoretically, there is only one or no event at the specific time, which makes it difficult to grasp the world captured by the camera at a particular moment. In this work, we present an algorithm that keeps the event *alive* until the corresponding event is generated in a nearby pixel so that the shape of an edge is preserved. Particularly, we consider a pixel area to fit a plane on Surface of Active Events (SAE) and call the point inside the pixel area closest to the plane as a intra-pixel-area event. These intra-pixel-area events help the fitting plane algorithm to estimate *lifetime* robustly and precisely. Our algorithm performs better in terms of sharpness and similarity metric than the accumulation of events over fixed counts or time intervals, when compared with the existing edge detection algorithms, both qualitatively and quantitatively.

1 Introduction

Event camera is a new type of vision sensor, which is motivated by the human eye, unlike standard CMOS cameras[4, 13]. Event camera is composed of an independent circuit for each pixel and these circuits generate *events* asynchronously when the log-intensity of light applied to the pixel changes. These time-stamped event streams have small bit-rate of tens or more kilobytes, whereas the absolute intensity measurement of the standard camera has large bit-rate of tens or more megabytes. By these virtues, the event cameras have several advantages such as low latency, high temporal resolution, whereas the traditional camera captures image frame at a few tens of frames per second, usually.

A wide range of fundamental computer vision applications, essential in autonomous navigation, object detection for avoidance, and augmented/virtual reality (AR/VR), can significantly benefit from the extremely low latency and fast response time of the event camera. However, most algorithms for computer vision cannot be applied directly to the event cameras due the new features mentioned above. So, there has been an increasing interest for de-



(a) Gray image (b) Proposed (c) 1 ms (d) 30 ms

Figure 1: The extracted sharp edge of the proposed algorithm.

veloping algorithms suitable for event cameras in applications such as optical flow[2, 8], visual odometry[7, 15, 21], and SLAM[11, 12]. In this paper, we focus on the time-continuous edge detection algorithm for the event camera, which could be utilised for edge-based visual odometry[9, 10, 20] or SLAM[14].

The simplest way to detect an edge for the event camera is to accumulate events in a specified number or fixed time interval. However, its result is too sensitive to the speed of the camera or image because the value of accumulation volume is heuristically determined as shown in Fig. 1 (c) and (d). If the number of events generated per unit of time is less than the expected accumulation, the edge becomes sparse, otherwise, edge bleeding occurs. Also, such accumulation dilutes the advantages of an event camera (namely, low latency).

In the paper, we aim to detect thin edge pixels regardless of the speed of the camera and to fit local plane robustly against noise using intra-pixel-area pixel approach. Also, the proposed algorithm is designed based on an event-by-event basis, thus updates the edge pixels with low latency.

1.1 Related Work

There are few methods developed for edge detection from the event camera data. Naïve methods are to accumulate events over a fixed time interval or numbers of events. However, in the case of accumulating events over time, the size of the interval depends on the speed of the camera because more events are generated when the camera moves faster. Therefore, the edge bleeding occurs or enough events are not triggered. On the other hands, if an edge is constructed by a fixed number of events accumulation, the quality of edge detection depends on the environment. A sequence captured in a simple environment with little gradient generates fewer events than in a complex environment. Consequently, it is difficult to determine a proper value of fixed time interval or number of events manually.

In order to detect edge robustly against various environmental properties, some algorithms have focused on detecting line edge in a structured environment. ELiSeD[5] computes gradient direction of Surface of Active Events (SAE) with Sobel filters, and clusters similar orientations within neighbour events. However, due to the above accumulation approach, edge bleeding occurs, thus lines fitted on the clustered segments may become incorrect. In [19], their event-by-event basis algorithm is inspired by the Hough transform and spiking neuron model. For each event, Hough transform converts their position to the so-called spikes in the (r,θ) space. The detected spikes stimulate the corresponding neuron in the (r,θ) space. This procedure is repeated while the potential value of the neuron exceeds the threshold, and the algorithm generates the line by using the triggered neuron. However, the performance depends on the resolution of (r,θ) parameter space, and is significantly degraded in areas where multiple lines meet.

Some research aim to detect edges, not just line segments that are frequently found in artifacts. F. Barranco *et al*. [1] detects the contour of foreground objects. They extract features from the accumulated events such as orientation, timestamp, motion, and time texture. Then the boundary is predicted from the learned Structured Random Forest (SRF) given DVS features. However, since this algorithm is developed for object segmentation, it is prioritised to detect the boundary of the foreground, and the performance of the overall edge extraction may be degraded. E. Mueggler *et al*. [16] estimates the lifetime of event from local plane fitting on the SAE based on event-based visual flow[2]. However, naïve RANdom SAmple Consensus (RANSAC) method could not be successfully adapted for the event camera, thus causing imprecise estimation. Therefore, we propose a intra-pixel-area approach for RANSAC in order to estimate a local plane robustly and precisely. Also, we quantitatively evaluate algorithms in terms of similarity metric, which have not been done in most of the previous works.

1.2 Contributions and Outline

Our main contributions can be summarised as follows:

1. We propose a intra-pixel-area event approach for loss function of RANSAC, thus achieving robustness against noise and enhancing the performance of edge detection for event cameras.
2. We evaluate the edge detection algorithms for the event camera by quantitatively measuring similarity metric.

The rest of this paper is organised as follows: In Section 2, we first characterise the event camera and describe the details of the algorithms. Next, we validate the performance of the proposed algorithm qualitatively in Section 3. Finally, Section 4 summarises the extension of this paper.

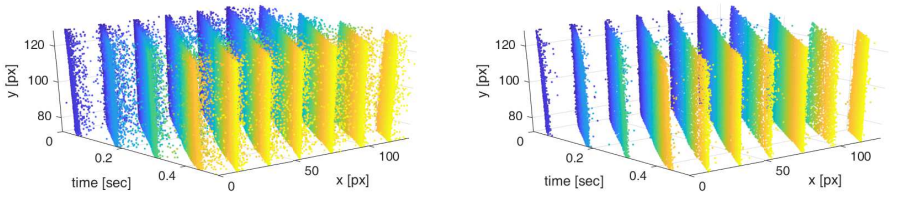
2 Methodology

First of all, we briefly review the definition of event cameras and the surface of active events in Section 2.1. Next, we propose an event buffer pre-processing algorithm to remove noise events. In addition, the existing event-based visual flow and lifetime estimation by fitting a local plane on Surface of Active Events (SAE) is summarised, followed by the introduction of a intra-pixel-area event in Section 2.4.1.

2.1 Event Camera

An event camera has independent pixels that detect light changes[13]. When the log-intensity of light applied to a pixel increases or decreases above/below the factory threshold, $\Delta \log(I) = \pm C$, the corresponding pixel (x_i, y_i) generates an event $\mathbf{e}_i = (x_i, y_i, t_i, p_i)$ asynchronously. The event consists of time-stamp, t_i, uv-coordinated position of the pixel (x_i, y_i), and polarity, p_i, where 1 or positive means log-intensity increments and 0 or negative means decrements.

In many existing algorithms for DVS[2, 16, 17], the SAE is used to obtain information about when the event occurs in the corresponding pixel. When an event $\mathbf{e}_i$ appears, the timestamp of the event, t_i, is assigned to SAE(y_i,x_i).



(a) Raw events (b) The filtered events

Figure 2: The (x,y,t) accumulation of events for description of the event buffer. For the `stripes` sequence which is captured while the several lines move perpendicular to the camera principal axis. Between each plane, the noise appears as an isolated point.

2.2 Event Buffer

Even in the fixed event camera, the camera generates many events which are regarded as noise. These noise data can be suppressed by lowering the sensitivity of the event camera, but it also reduces the number of meaningful events that occur where the image intensity changes. To deal with noise events, we design an algorithm which precedes local plane fitting. The idea is that an event occurs along the gradient edge of the image, and the edge with adjacent pixels will move simultaneously on the image so that events will occur in a short time at adjacent pixels. In other words, events of the same polarity are generated during a short period of time within milliseconds along the same edge segment.

Fig. 2 illustrates the SAE in the situation where the several straight stripes move perpendicular to the principle axis of the camera from $t = 0$ to $t = 0.5$. In the following evaluation including Fig. 2, we set $\tau_{min} = 0.01$. As shown in Fig. 2(a), the raw event stream has a lot of noise data between planes. By pre-processing raw events with the event buffer procedure, we can reduce noise events successfully.

2.3 Event-based Visual Flow and Lifetime

In order to estimate the lifetime of events, we utilise the existing algorithm[7, 16]. They suggest that the lifetime of an event means the time until the corresponding world point that has generated the current event before triggers a new event, and also this period indicates the maximum amount of time before a new event occurs nearby:

$$\tau(\mathbf{x}) = \max\ \Delta t \ \text{ subject to } \ ||\mathbf{x}|| = 1, \tag{1}$$

where $\Delta t = SAE(\mathbf{x}+\Delta\mathbf{x}) - SAE(\mathbf{x})$ and $\mathbf{x} = (x,y)$ in pixel coordinates.

To find the steepest slope in the SAE, they estimate the normal vector of the plane, $\mathbf{n} = (n_1, n_2, n_3)$, which is fitted on the SAE locally, and visual flow, $(v_x.v_y)$, and lifetime of the event, τ, are calculated below:

$$v_x = -n_3/n_1,\ v_y = -n_3/n_2, \tag{2}$$

$$\tau(\mathbf{x}) = 1/v = 1/\sqrt{v_x^2+v_y^2} = \frac{1}{n_3}\sqrt{n_1^2+n_2^2}. \tag{3}$$

2.4 Local Plane Fitting

The event camera generates a *digital* event when the average brightness of pixel changes, and the position of the generated event is fixed as the center point of the pixel area, thus the true position of an *analog* event is not captured in the event stream. Therefore, we use RANSAC to robustly find a local plane fitted on the SAE. For RANSAC, we use past events in an $N \times N$ window including the current event which is the center of the window. Next, we use k-mean clustering to separate non-triggered or oldest event pixels from the N^2 pixels around the current event by setting k as 2. The initial cluster centroid are set to the minimum and the current timestamp. The set including the current event is selected. Then, we choose the current event and two past events randomly in the selected set to compute the candidate plane.

2.4.1 Intra-pixel-area Event

By the way, the gradient change occurs at some sub-pixel point inside the pixel theoretically, but actually, the event is triggered by the change of the average brightness of a pixel, and the position of the event indicates the center of the pixel. Therefore, we introduce a *intra-pixel-area event*, $S_{\mathbf{x}}(\delta)$, which exists somewhere inside a pixel area:

$$S_{\mathbf{x}}(\delta) = \{(x,y) | x-\delta < \mathbf{x}_1 < x+\delta, y-\delta < \mathbf{x}_2 < y+\delta\}, \tag{4}$$

where $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ is the pixel of the current event. Thus, we choose the current event and float event of two past events randomly in RANSAC. Then, we compute loss function as the distance between candidate plane and intra-pixel-area events, $\mathbf{z}$, and the distance function is defined as below:

$$d = \min \text{dist}(plane, \mathbf{z}),\ \forall \mathbf{z} \in \mathbf{S}_{\mathbf{x}}(\delta), \tag{5}$$

where

$$\text{dist}(plane, \mathbf{z}) = \frac{|\mathbf{n}^{\mathrm{T}}\mathbf{z} - 1|}{||\mathbf{n}||_2^2} \quad \text{with a plane: } \mathbf{n}^{\mathrm{T}}\mathbf{x} = 1 \tag{6}$$

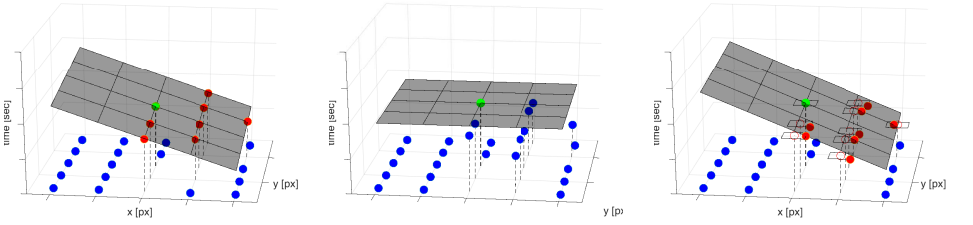
which contains the current event.

We show the result of intra-pixel-area events in Fig. 3. It describes the estimation of local plane fitted on the SAE around the current event (green). RANSAC is executed from events in a 5×5 window, and the value of events are given manually for simulation. In particular, the timestamp value of events in Fig. 3 (b) and (c) are contaminated by severe noise. Comparing the result of Fig. 3 (a), intra-pixel-area event approach makes the candidate plane more likely to be voted on by more events, taking into account the positional variation of past events, thus making more robust against the noise.

Furthermore, we analyse the influence of the intra-pixel-area approach in terms of F-measure and lifetime error as shown in Fig. 4. The event data of a 5×5 window used in the analysis are manually made assuming that a single line passes, as shown in Fig. 3, which is a very common condition in dataset sequence. Moreover, we add Gaussian noise in whole pixels or several randomly-chosen pixels in Fig. 4 (a) and (b), respectively. F-measure is defined below:

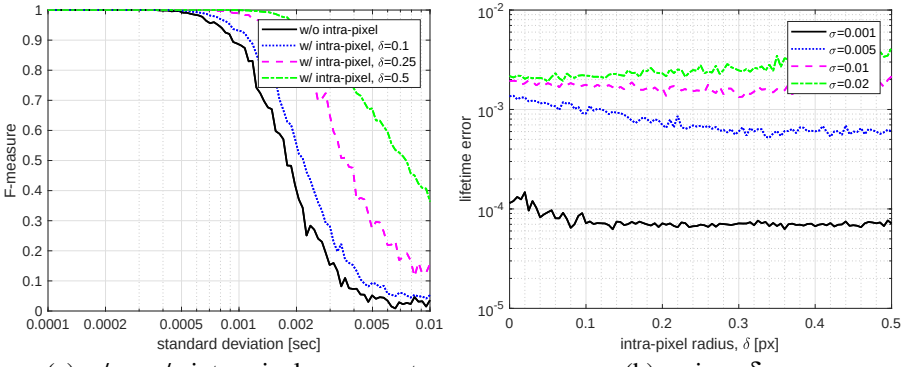
$$F\text{-}measure = 2 \times \frac{Recall \times Precision}{Recall + Precision}, \tag{7}$$

where recall is a fraction of determined true among a total of true, and precision is a fraction of true among a total of determined true. Also, lifetime error is the difference between the



(a) w/o intra-pixel-area under mild noise (b) w/o intra-pixel-area under severe noise (c) w/ intra-pixel-area under severe noise

Figure 3: Description of the intra-pixel-area event. After computing a local plane (gray) with or without intra-pixel-area event by RANSAC, the outliers (blue), inliers (red), and the current event (green) are drawn. In (c), raw inlier events are represented as an *unfilled* red circle, whereas the closest points in each intra-pixel area of inlier event are depicted as a *filled* circle. Also, the intra-pixel areas of the inlier events are depicted as a rectangle.



(a) w/ or w/o intra-pixel-area event (b) various δ

Figure 4: F-measurement evaluation graph with a standard deviation of data noise.

truth and the estimated values. For each evaluation, we take the average value by repeating a total of 1,000 times.

In Fig. 4 (a), a higher value of the intra-pixel radius, δ, makes the RANSAC obtain more true inliers robustly under globally-generated noise. However, as shown in Fig. 4 (b), it also tends to cause lifetime estimation error under scattered noise, because RANSAC may choose undesired noise data due to large intra-pixel radius. From the above analyses, δ is manually set to 0.25 for the evaluation.

2.4.2 Constant Velocity Assumption

For the local plane fitting, it is assumed that an edge moves at a constant velocity locally. But in practice, edge segments do not move at the same velocity, and they experience acceleration and deceleration. If an edge undergoes acceleration, a lifetime of the edge is overestimated, thus edge bleeding appears. In view of the accelerating situation, when an event is generated earlier than expected before, edge bleeding can disappear by lowering the lifetime of the pix-

els present in the opposite direction of the local SAE gradient heuristically as the following, rather than deleting the only opposite pixel[16]:

$$\tau(\mathbf{z}) = \max(\tau(\mathbf{z})/\gamma, 0), \quad \text{if } \gamma > 1, \tag{8}$$

where

$$\gamma = -2 \cdot \langle \nabla SAE(\mathbf{x}), (\mathbf{z}-\mathbf{x})/||\mathbf{z}-\mathbf{x}|| \rangle, \tag{9}$$

with $\mathbf{z} \in S_{\mathbf{x}}(1)$ around the current event pixel, $\mathbf{x}$, as mentioned in Eq. (4) and $\langle \cdot, \cdot \rangle$ is inner product in $\mathbb{R}^2$. On the other hands, the lifetime is underestimated when an edge is decelerating, and this case could be a difficult problem to handle by the lifetime approaches. When the camera stops completely in the moment, it is impossible to predict the lifetime of the latest pixel.

3 Experimental Evaluation

We qualitatively evaluate the comparison algorithms using a sequence provided by DVS (128×128 pixels) in [16]. The other sequences are gray images and events captured using DVS240C (240×180 pixels) for quantitative comparison of edge detection performance.

3.1 Qualitative Evaluation

For qualitative evaluation, we first show the accumulation and histogram of lifetime estimates. These analyses are provided for only a `stripe` sequence because the sequence is captured at a constant velocity making it easy to verify consistent lifetimes. Through the analysis of lifetime accumulation in Fig. 5, it is possible to see how uniform the lifetime is in areas where the stripes pass at a constant speed. Also, the analysis of lifetime histogram performed in [16] shows how precisely the lifetime is estimated through two main peaks as shown in Fig. 6.

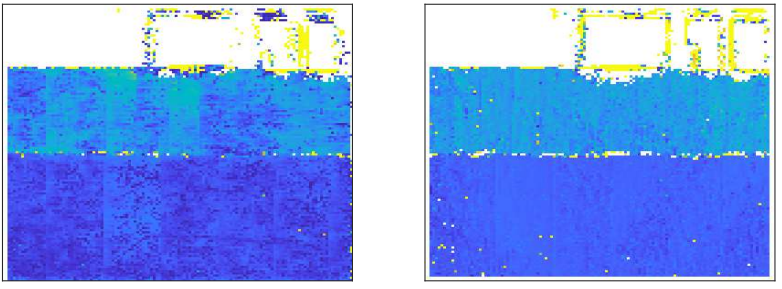
Fig. 5 shows the accumulation of lifetime estimated during whole `stripes` sequence. Comparing (a) and (b), E. Mueggler *et al.*'s algorithm shows small lifetime denoted as deep blue sometimes, while our algorithm shows uniform values across each area, in top and bottom. Further, as in the left part of the lifetime histogram in Fig. 6, our result represents a sharper histogram, which means it estimates lifetime precisely.

3.2 Quantitative Evaluation

In this section, we evaluate our algorithm by measuring similarity to the result of the traditional edge detector from a gray image. For the similarity metric with an edge image, we use the implementation of Closest Distance Metric (CDM) in the work of [3, 18]:

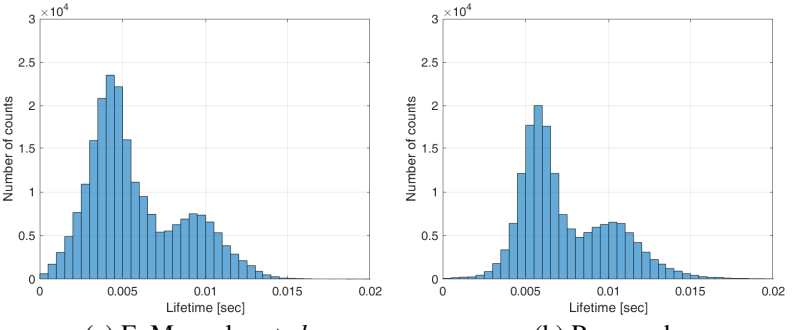
$$CDM_{\eta}(f,g) = 100\left(1 - \frac{\mathcal{C}(\mathcal{M}_{cd}(f,g))}{|f \cup g|}\right), \tag{10}$$

where η is the neighbourhood radius to find matching edge pixels between two images, f and g, $\mathcal{C}(\mathcal{M}_{cd}(f,g))$ is the cost of a pair matched by closest-distance criteria, and $|f \cup g|$ is the number of edge pixels belonging to f or g. In evaluation, η is set to 3, and we use 8-connected chessboard distance metric as mentioned in [18]. In addition, for comparison, the results of the Canny Edge Detector[6] are considered as groundtruth and the performance



(a) E. Mueggler *et al*. (b) Proposed

Figure 5: The accumulation of lifetime estimates.



(a) E. Mueggler *et al*. (b) Proposed

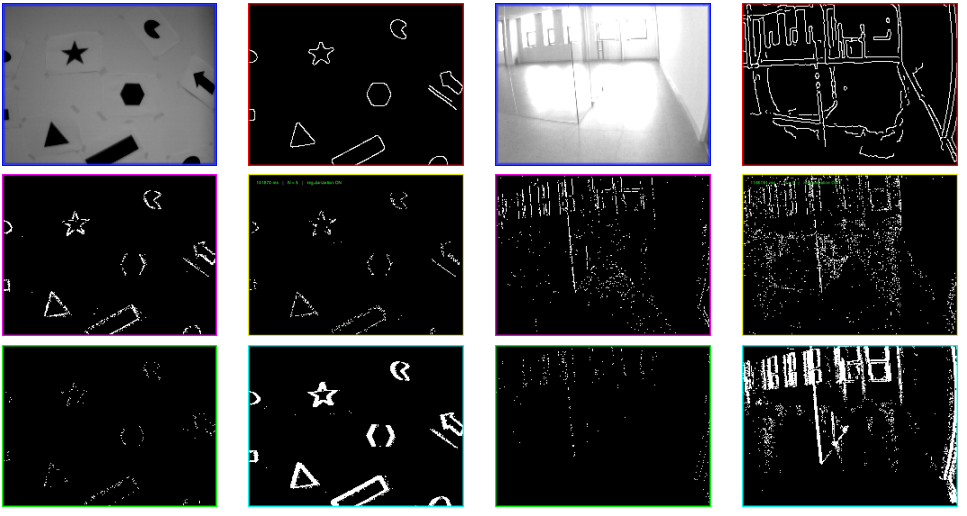
Figure 6: The histogram of lifetime estimates.

of algorithms is confirmed by various Canny Edge Detector parameters because Canny's algorithm can produce fine and well-connected edges with 1-pixel width.

Fig. 7 shows the result of: gray image, Canny edge, proposed algorithm, 1ms, 30ms accumulation, and E. Mueggler[16]. Although the lifetime estimation including ours and E. Mueggler's algorithm tend to leave tracks on the edge image, they show a thinner edge than the result of accumulation. In particular, the performance of edge detection over time for a whole sequence is shown in Fig. 8. Our algorithm (magenta) shows relatively consistent performance in terms of the CDM rather the other methods regardless of the camera motion as shown in Fig. 8 (b) and (c). Sometimes, our algorithm shows bad performance denoted as red-cross outliers when the camera motion is almost stationary. The reason is that a local SAE gradient cannot be estimated precisely when the camera moves slowly. Besides, when the camera stops, the lifetime estimation is theoretically impossible due to the lack of events. Note that the performances of the accumulation methods depend on the speed of the camera in the bottom figure.

4 Conclusion

In the work, we proposed an algorithm to detect an edge from events of a dynamic vision sensor by virtue of the lifetime estimation with an event buffer and intra-pixel-area approach.



(a) `shape_translation` sequence (b) `corridor` sequence (self-collected)

Figure 7: The result of : gray image (blue), groundtruth computed by Canny Edge Detector (red), E. Mueggler *et al.*'s algorithm (yellow), 30ms (cyan), 1ms (green) accumulation, and proposed algorithm (magenta) for each sequence in clockwise from top-left.

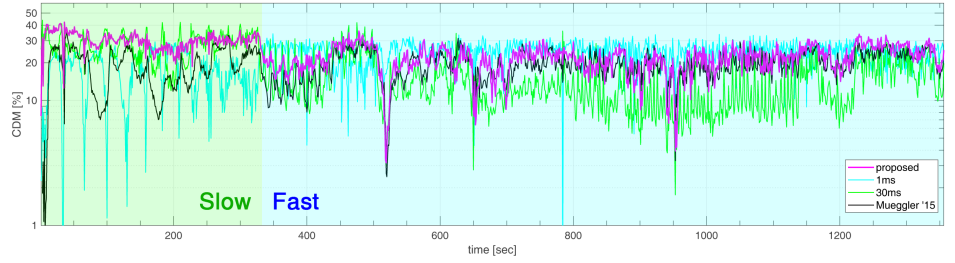
The designed event buffer regards isolated events as noise and reduces them effectively. Also, the proposed intra-pixel-area approach makes the algorithm find local plane fitted on SAE robustly so that the lifetime is estimated precisely. Moreover, we analysed the effectiveness of the intra-pixel-area approach by the F-measure versus the standard deviation of timestamp noise and the estimation error versus the intra-pixel radius. For evaluation, we qualitatively compare our algorithm with the existing lifetime estimation or naïve event accumulation approach. In addition, with the well-known edge similarity metric, we measure the performance of the algorithm quantitatively. Then, we confirm that our algorithm performs better in terms of sharpness and similarity to the Canny edge than the accumulation of events over fixed counts or time intervals, and the existing lifetime estimation algorithm. Further, by utilising the detected edge, the proposed algorithm can be employed as a pre-processing part of edge-based visual odometry or SLAM for autonomous robots.

5 Acknowledgement

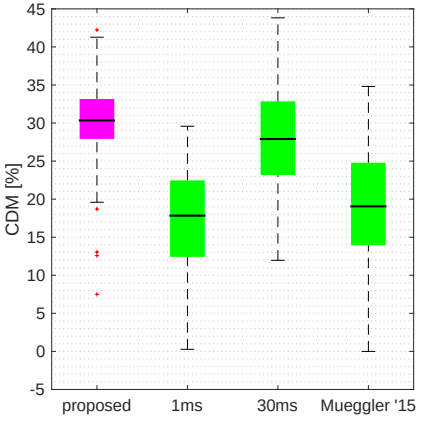
The SNU-Samsung smart campus research center (0115-20190023) at Seoul National University provides research facilities for this study. This research was supported by Samsung Electronics.

References

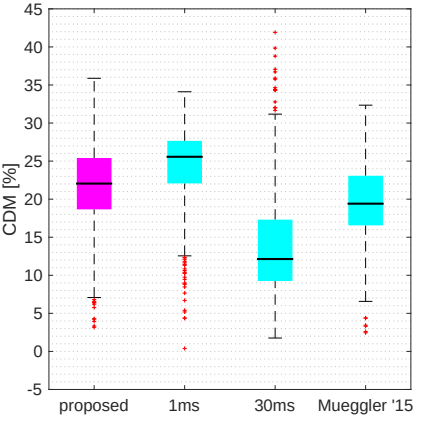
[1] Francisco Barranco, Ching L Teo, Cornelia Fermuller, and Yiannis Aloimonos. Contour detection and characterization for asynchronous event sensors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 486–494, 2015.



(a) Similarity over time



(b) Boxplot on the slow section



(c) Boxplot on the fast section

Figure 8: Performance analysis of the whole sequence on the `shapes_translation` sequence. The speed of camera is slow in the front and fast in the back. Thus, 1ms and 30ms accumulation of event shows different performance depending on the camera's speed.

[2] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio-Hoi Ieng, and Chiara Bartolozzi. Event-based visual flow. *IEEE Trans. Neural Netw. Learning Syst.*, 25(2):407–417, 2014.

[3] K Bowyer, C Kranenburg, and S Dougherty. Edge detector evaluation using empirical roc curves. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 1, pages 354–359. IEEE, 1999.

[4] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240× 180 130 db 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014.

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

[6] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[7] Andrea Censi and Davide Scaramuzza. Low-latency event-based visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 703–710. IEEE, 2014.

[8] 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 *IEEE Int. Conf. Comput. Vis. Pattern Recog.(CVPR)*, volume 1, 2018.

[9] Juan Jose Tarrio and Sol Pedre. Realtime edge-based visual odometry for a monocular camera. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 702–710, 2015.

[10] Changhyeon Kim, Pyojin Kim, Sangil Lee, and H Jin Kim. Edge-based robust rgb-d visual odometry using 2-d edge divergence minimization. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

[11] Hanme Kim, Ankur Handa, Ryad Benosman, Sio-Hoi Ieng, and Andrew J Davison. Simultaneous mosaicing and tracking with an event camera. *J. Solid State Circ*, 43: 566–576, 2008.

[12] Hanme Kim, Stefan Leutenegger, and Andrew J Davison. Real-time 3d reconstruction and 6-dof tracking with an event camera. In *European Conference on Computer Vision*, pages 349–364. Springer, 2016.

[13] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128×128 120 db 15μ s latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits*, 43(2):566–576, 2008.

[14] 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*, pages 2408–2417, 2017.

[15] Elias Mueggler, Basil Huber, and Davide Scaramuzza. Event-based, 6-dof pose tracking for high-speed maneuvers. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2761–2768. IEEE, 2014.

[16] Elias Mueggler, Christian Forster, Nathan Baumli, Guillermo Gallego, and Davide Scaramuzza. Lifetime estimation of events from dynamic vision sensors. In *2015 IEEE international conference on Robotics and Automation (ICRA)*, pages 4874–4881. IEEE, 2015.

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

[18] Miguel Segui Prieto and Alastair R Allen. A similarity metric for edge images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1265–1273, 2003.

[19] Sajjad Seifozzakerini, Wei-Yun Yau, Bo Zhao, and Kezhi Mao. Event-based hough transform in a spiking neural network for multiple line detection and tracking using a dynamic vision sensor. In *BMVC*, 2016.

[20] Yi Zhou, Hongdong Li, and Laurent Kneip. Canny-vo: Visual odometry with rgb-d cameras based on geometric 3-d–2-d edge alignment. *IEEE Transactions on Robotics*, 35(1):184–199, 2019.

[21] Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-based visual inertial odometry. In *CVPR*, pages 5816–5824, 2017.