2017 17th International Conference on Control, Automation and Systems (ICCAS 2017)
Oct. 18–21, 2017, Ramada Plaza, Jeju, Korea

Event-based Real-time Optical Flow Estimation

Alex Junho Lee[1] and Ayoung Kim[1*]

[1]Department of Civil and Environmental Engineering, Korea Advanced Institute of Science and Technology, Daejeon, Korea ([alex_jhlee, ayoungk]@kaist.ac.kr)

Abstract: In this paper, we introduce a novel approach defining features and estimating their optical flow with an event-based sensor. Event cameras detect temporal intensity changes on low latency with high dynamic range, arising as a promising solution for improving Visual odometry (VO). However, it produces an event stream rather than a framewise image, requiring additional procedures like rebuilding intensity images to apply frame based algorithms. Hence, some information degenerates to process event data with conventional algorithms. Therefore, there are needs on event-based algorithms to fully utilize the low latency characteristics of event cameras. We propose an algorithm to extract features and estimate its optical flow only with event stream, empowering the event camera on VO.

Keywords: Optical Flow, Visual Odometry, Event-Based

1. INTRODUCTION

Machine vision relies on different methodologies to process various signals from single or multiple sensory inputs. VO, which uses data from imaging sensors, became one of the most popular topics in machine vision, for its broad application including robot navigation, simultaneous localization and mapping (SLAM), and three-dimensional (3D) reconstruction. Its goal is to estimate the current location or the trajectory of a robot using visual inputs from imaging sensors.

In conventional approaches, complementary metal-oxide-semiconductor (CMOS) or charge-coupled device (CCD) imaging sensors are widely used. Frame-based cameras are straightforward, due to its similar output to human vision and a small cost to their high pixel resolution. These were key competencies on earlier VO, thus made CMOS widely used. Therefore, numerous frame based feature extraction algorithm were developed with high-performance, from feature extraction stage to matching and tracking stages. Corner detection methods like Harris [6] enabled us to define simple and accurate local feature and some improvements such as Features from Accelerated Segment Test (FAST) [14] followed.

However, CMOS typically provides around 30 frames-per-second (fps), and upto hundreds of frames per second. Thus information in the blind time interval is lost, and movements during exposure time result blurring. To get higher spatial resolution for moving objects, higher spartial and temporal resolution required. Heavy computation cost, resulted in a trade-off between algorithm speed and resolution. Also, limited bandwidth from using global pixel gains loses intensity information for natural environments with wide dynamic range. Naturally every frame-based algorithms inherently share common issues, making difficulty on lowering error in real environments. Moreover, errors are inevitably enlarged for dynamically changing scenes. To overcome these problems, the event camera was introduced.

Event camera [9] is a time-accurate silicon sensor based on the neuromorphic design on computer vision. The event-based sensor is inspired from retinal ganglion cells (RGC), which have rapid responses to overall intensity changes. Unlike frame-based cameras, event cameras only measures temporal intensity changes (i.e., events) by individual pixel gain. Therefore only pixels with changes transmit signal, removing the latency from waiting for the whole other pixel to integrate photons for the frame-wise intensity images. A single event is demonstrated with location (x,y), polarity (on/off event), and time (ns). In an Event stream, a signal is transferred with low latency (<1 ms) with high dynamic range (>140 dB), making event camera structurally free from the problems in frame-based cameras.

There are some recent trials in VO with event-based vision sensors, to utilize characteristics of event camera. More details on state-of-the-art event-based feature extraction methods will be dealt in §1.2

1.1 Frame-based Feature Extraction

There are numbers of high performing algorithms for frame-based feature extraction. Harris [6] detects corner points by measuring the difference of eigenvalues for a variation in each direction. It succeeded in finding a corner with specified size, but using a fixed size of the window made itself sensitive to scale errors. Thus, Harris-Laplacian [11] was introduced to deal with scale error, finding optimal window size for each point. It requires calculation of every eigenvalue for each pixel with different size of windows, which yields massive computation. Similarly, finding the optimal value of Laplacian on orientation requires information on intensity distribution. It helps to find accurate global corner points, but inappropriate for event-based vision, which only collects partial intensity changes.

The Scale Invariant Feature Transform (SIFT) [10] exploited Difference of Gaussian (DoG) method to compensate the scale problems in Harris corner. Calculating DoG often avoid a full computation by approximating Laplacian, and is faster than Harris-Laplacian and scale invariant. However, an intensity distribution is required.

The Speeded Up Robust Features (SURF) [1] first generate an integrated image with the variant sized box filter and find scale-invariant features. SURF has high performance as SIFT and is much faster in computation. However, the procedure to get an integrated image, preventing a direct application to event-based vision. Certain algorithms on feature extraction, are inappropriate for event-based vision, because of their requirement for global intensity information.

The FAST [14] is one of the widely used methods for corner detection. FAST does not calculates eigenvalues or generates an integrated image, instead searching for feature candidates with high-intensity differences around neighbor. Since it finds local feature with few data, it also can be applied to event-based vision. Moreover, FAST descriptor provides where dark and bright corners are. This acts as hints to event-based vision when making expectations on what would happen if some corner point moves on directions.

Our algorithm was inspired from characteristics of FAST to estimate the flow direction of the corner, to make expectations on polarity of potential events based on previous information.

1.2 Event-based Feature Extraction

Some recent trials were on dealing event cameras for image reconstruction [7] [13], event-based corner detection [5] [12] and feature tracking [15]. Event-based feature extraction can be classified as whether it utilizes information from other imaging sensor or not. Generally, image from CMOS sensor is often used for initialization of event-based algorithm. In dynamically changing environment, however, algorithms using only event stream show better performances.

Low-latency VO using event-based feature tracks [8] proposed a tracking method with incoming events and frame image. They first obtained an intensity image from Active Pixel Sensor (APS) sensor, then extracted Harris feature to track them with incoming event signals with Canny edge detector [4]. Next, they built edge map, to estimate camera pose drift from the calculation of weighted Iterated Closest Point (ICP) [2] on events for edge map. This algorithm works well on determining camera pose from a static scene, yet only valid when possible to get sufficiently sharp APS image, to extract Harris corner. Accordingly, we could expect expanded error in rapid scenes and high dynamic ranges.

Surfaces of time [16] proposed a spatiotemporal time surface, from the characteristics of exponentially decaying pixel intensity after an edge pass. It uses three kinds of time surfaces, that is linear, exponential, and accumulated exponent. By these time of surfaces, they successfully constructed an intensity image with incoming event signals, further allowing an application of traditional frame-based ones. It made possible to use existing algorithms but still computes from a fixed image, degenerating the characteristics of high time resolution in the event-based sensor.

Surface of active events [5] first introduced a method to process event stream without frame images, estimating optical flow by line fitting in xyt coordinate. However it needs large computation on handling large amount of events to inliers and outliers, not retrieving real-time.

Then [12] applied FAST corner detector on Surface of Active Events, successfully reducing computing cost. However, is still not fully exploiting the benefits of low latency and high temporal resolution of event camera since it is based on surface of active events.

To fully exploit the low latency characteristics of event camera, we propose a method on event-based feature extraction and its optical flow estimation using only event stream, without any preprocessing.

2. METHOD

Our algorithm is to extract features by building an expected map of polarization for each pixel based on their event polarity, with only event stream. Inspired from FAST [14] as mentioned in §1, we put to use some information consisting a FAST feature points. The basic idea of the proposed method is at adding temporal information in the feature descriptor. Polarity obtained from event camera is used when estimating this temporal data.

FAST defines itself by the length of consequent brighter or darker segments around the center point. In Fig. 1, meshed point (center) represents an extracted feature point, and its corresponding brighter or darker segments are illustrated as dot patterned. Since segment information is calculated during extraction of FAST feature points, every feature points contain its information on relative brightness around the corner. Reminding the characteristics of event camera that responds to the changes in intensity, we handled this information as a hint of changes in brightness of the corner pixel.

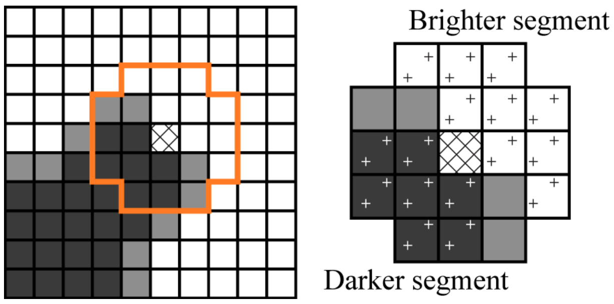


Fig. 1: FAST feature point and its segment information. (left) is an intensity image acquired from imaging sensor. (right) describes corner segments around a feature point, with the location of brighter and darker segments.

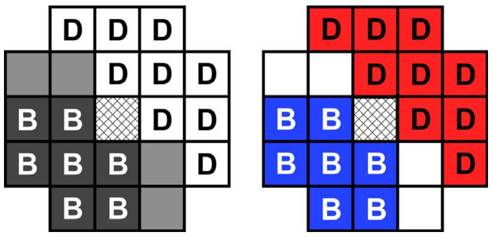
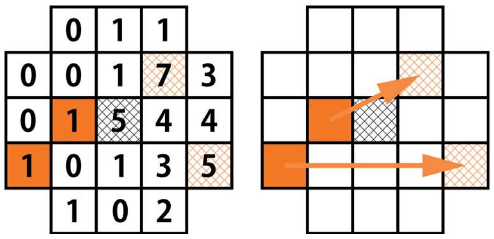


Fig. 2: Expected intensity changes on motion. Based on this map of expected polarity, we can detect a motion of the feature by counting the number of incoming events with expected polarity for each pixel. (left) describes corner pixels around a FAST feature point from an intensity image as illustrated in Fig. 1. (right) describes corner pixels around a feature point extracted by our algorithm. The image is for illustrative use only, since event camera does not produce a framewise image. In here, pixels are colored by the major polarity of events coming into each pixel. ON event dominating pixels are colored with red, and OFF event dominating pixels are marked as blue.

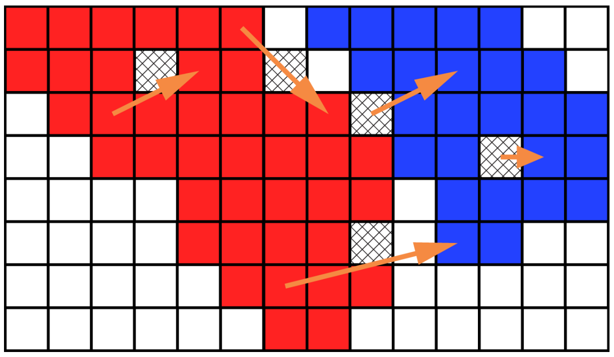
Next, based on this information, we may expect changes in pixel intensity on motion occurrence follow as in Fig. 2. Particularly, if a FAST feature point slides into a brighter segment, the pixel around the corner will turn darker as it was brighter than the center point. Contrarily an opposite will happen when the feature point moves in the direction of the darker segment. Considering this in an event-based vision, we first divided neighbor points as ON (1) event dominating pixel and OFF (0) event dominating pixel. From the same principle of Fig. 2, we may expect a brighter change (ON-event) if an object in center point slides into OFF-event dominant (blue marked), and darker change (OFF-event) if it slides into ON-event dominant segment (red). Therefore, a map of expected polarization guides incoming events to estimate the optical flow of feature points. Our algorithm consequently updates expected polarization with event stream, to score pixel points to define feature and define its future path.

Some event–based cameras like DAVIS [3] provides event stream together with images from APS sensor. For a static scene when a camera is fixed with nearly no moving object in the view, only a small number event is generated. For that sequences, we first extracted FAST [14] feature points and its segment information (as in Fig. 1) for building a map of expected polarization. By contrary in a dynamically changing scene when either camera is moving fast, or the object is translating in the view, We used only event stream to build expected polarization.

For the next step, we recorded numbers and dominating polarity of consequently arriving events. Then using the map of expected polarization, pixels are classified by the number of corner segments with the corresponding polarity. Next, points are categorized into four states (i.e; initialized, event, potential, and feature). Since expected polarization would not be accurate if there are few events, we only build expected polarization on potential and feature points. In other words, potential points are not features, but an intermediate step for computing expected polarization. Accordingly, introducing a stepwise state description first serves as a filter to neglect random noise from the sensor, and boosts more point to be processed on calculating expected polarization.



(a)Estimating direction from a feature point (meshed black) with predetermined past (filled orange) and calculated future paths (orange mesh). Future paths are determined by the level of corresponding events of the corner.



(b)Estimation of optical flow based on past / future path. Estimated flow direction (orange arrow) is a path from current point (or past path if any) to future point. Circle radius is chosen by parameter.

Fig. 3: Estimating optical flow of a feature. Pixel velocity is filtered, by taking an average with a value from past point. Thus connected features have similar velocity.

Finally, we make estimations on the speed and direction of the feature point. Speed is calculated from the elapsed time between an extract of past feature point (if any) to definition of current feature. Otherwise if there's no past path, it is not calculated and life duration of feature point is set as default. So next, direction is calculated from defining a new future path of the feature point. We have already calculated expected polarization from the last procedures. From that, the number of the events with expected polarization is recorded to determine future path. Then if the point with the highest number has equal or higher level than center point, it determined as a future path. As in Fig. 3, every feature points may or may not have the past or future path. A direction is described as a Euclidean angle from x axis to a line between past to future path.

Algorithm 1 Feature extraction and Flow estimation

```
1:  for every pixels do
2:      procedure FIND EXPECTED POLARIZATION
3:          Do this procedure for every event bus
4:          if $pixel.status = events$ then
5:              return polarization map = expect both
6:          end if
7:          past ← if pixel.past exist
8:          future ← if Find pixel.future exist
9:          return polarization map = func(past, future)
10:     end procedure
11:     procedure DETERMINE STATUS
12:         scan around corner : find neighbor.status
13:         pixel.status = threshold(# of neighbor.status)
14:         if $pixel.status = potential/feature$ then
15:             Find pixel.futurepath
16:             futurepath.status = potential / feature
17:         end if
18:     end procedure
19:     procedure ESTIMATE OPTICAL FLOW
20:         pixel.speed = feature define time interval
21:         pixel.direction = func(past,future)
22:     end procedure
23: end for
```

3. EVALUATION

For evaluation, we recorded a fast rotating object with Dynamic and Active-pixel Vision Sensor (DAVIS) camera as in Fig. 4. For next, we extracted features and estimated its flow direction.

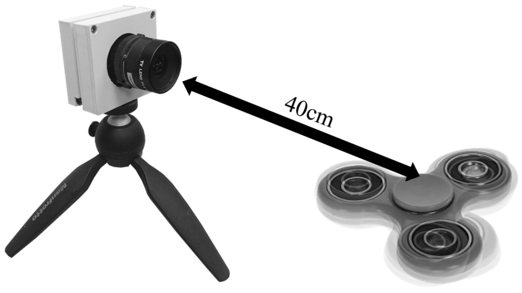


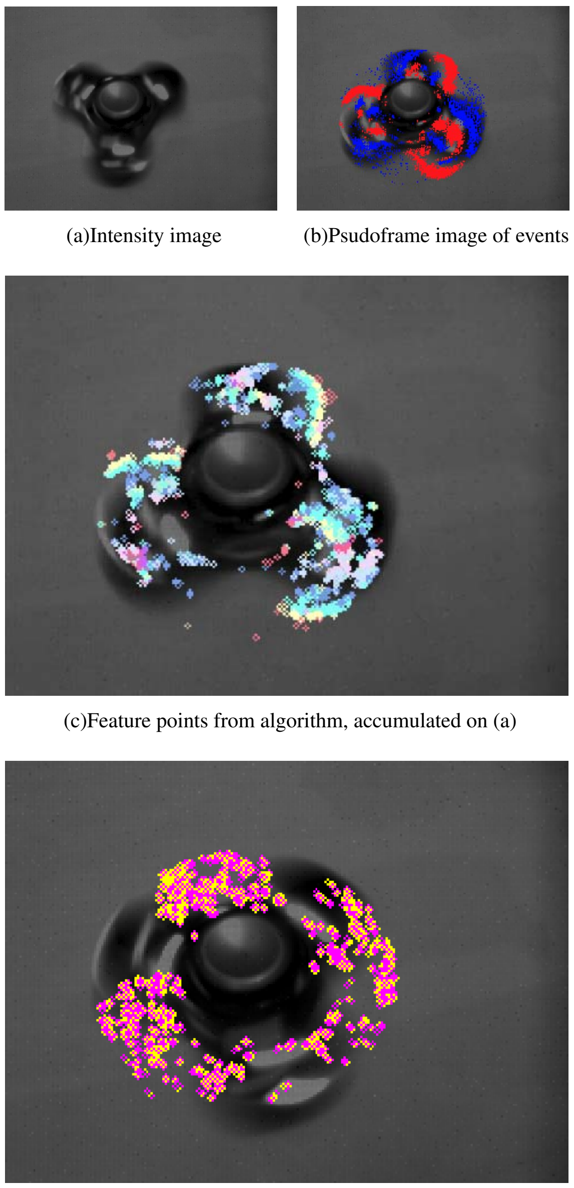
Fig. 4: Experiment setting on filming fast rotating object. (left) DAVIS camera and the targeted object in FOV.

Fig. 5(a) is an image from APS. Since the object is rotating even faster than the frame rate (30 fps), motion blur occurs. Moreover, we cannot measure the exact rotation speed because of under–sampling.

Fig. 5(b) is a psudoframe image, generated by accumulating event for a short time. Different from Fig. 5(a), fast rotating object is detected without under–sampling. With this psudoframe image, we can apply conventional corner detection algorithm. However, computation costs for extracting and matching features are enlarged due higher frame rate. Moreover, the benefits from high time resolution are depleted.

Fig. 5(c) displays features from our algorithm. We randomly assigned the colors by the definition time of each features, and made them to have the same color if connected to other features. Some connections are made in Fig. 5(c), perpendicular to the flow direction.

Fig. 5(d) illustrates connections between the extracted features. Most of features have velocity information. However, since ground truth of pixel velocity cannot be obtained, method on verifying estimation should be considered in future works.



(a)Intensity image (b)Psudoframe image of events

(c)Feature points from algorithm, accumulated on (a)

(d)Connected points for each feature points. Past paths of each points are marked as yellow, and Future paths as magenta.

Fig. 5: Output image from APS, psudoframe generated from DVS, extracted feature points, and estimated flow.

4. CONCLUSION

We have described defining an event-based feature points and their optical flow, only with event camera. Potential applications are on rapidly performing robot (drone and autonomous car) navigation, and for accurate VO in too dark or bright condition to make robots robustly find and track feature points for every environments.

REFERENCES

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *Computer vision–ECCV 2006*, pages 404–417, 2006.

[2] Paul J Besl, Neil D McKay, et al. A method for registration of 3-d shapes. *IEEE Transactions on pattern analysis and machine intelligence*, 14(2):239–256, 1992.

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

[4] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

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

[6] Christopher G Harris and JM Pike. 3d positional integration from image sequences. *Image and Vision Computing*, 6(2):87–90, 1988.

[7] Hanme Kim, Ankur Handa, Ryad Benosman, Sio-Hoi Ieng, and Andrew J Davison. Simultaneous mosaicing and tracking with an event camera. In *BMVC*, 2014.

[8] Beat Kueng, Elias Mueggler, Guillermo Gallego, and Davide Scaramuzza. Low-latency visual odometry using event-based feature tracks. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 16–23. IEEE, 2016.

[9] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128 128 120 db 15 s latency asynchronous temporal contrast vision sensor. *IEEE JOURNAL OF SOLID-STATE CIRCUITS*, 43(2), 2008.

[10] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[11] Krystian Mikolajczyk and Cordelia Schmid. Indexing based on scale invariant interest points. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 525–531. IEEE, 2001.

[12] Elias Mueggler, Chiara Bartolozzi, and Davide Scaramuzza. Fast event-based corner detection.

[13] Christian Reinbacher, Gottfried Graber, and Thomas Pock. Real-time intensity-image reconstruction for event cameras using manifold regularisation. *arXiv preprint arXiv:1607.06283*, 2016.

[14] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. *Computer Vision–ECCV 2006*, pages 430–443, 2006.

[15] David Tedaldi, Guillermo Gallego, Elias Mueggler, and Davide Scaramuzza. Feature detection and tracking with the dynamic and active-pixel vision sensor (davis). In *Event-based Control, Communication, and Signal Processing (EBCCSP), 2016 Second International Conference on*, pages 1–7. IEEE, 2016.

[16] Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, 2008.