Adaptive Time-Slice Block-Matching Optical Flow Algorithm for Dynamic Vision Sensors

Min Liu
minliu@ini.uzh.ch
Tobi Delbruck
tobi@ini.uzh.ch

Institute of Neuroinformatics
ETH Zurich and University of Zurich
Zurich, Switzerland

Abstract

Dynamic Vision Sensors (**DVS**) output asynchronous log intensity change events. They have potential applications in high-speed robotics, autonomous cars and drones. The precise event timing, sparse output, and wide dynamic range of the events are well suited for optical flow, but conventional optical flow (**OF**) algorithms are not well matched to the event stream data. This paper proposes an event-driven OF algorithm called adaptive block-matching optical flow (**ABMOF**). ABMOF uses time slices of accumulated DVS events. The time slices are adaptively rotated based on the input events and OF results. Compared with other methods such as gradient-based OF, ABMOF can efficiently be implemented in compact logic circuits. We developed both ABMOF and Lucas-Kanade (**LK**) algorithms using our adapted slices. Results shows that ABMOF accuracy is comparable with LK accuracy on natural scene data including sparse and dense texture, high dynamic range, and fast motion exceeding 30,000 pixels per second.

Supplementary Material

A video demonstrating ABMOF is available on youtube. For downloading the dataset, refer to the ABMOF dataset readme which is available here. We also published our code on github.

1 Introduction

Computing optical flow (OF) is a fundamental problem of computer vision. There are a variety of algorithms for frame-based cameras. The most popular method is the LK method [16]. It is a sparse gradient method that assumes brightness constancy around detected feature points. Most recent developments have used deep convolutional neural networks to compute dense flow; they achieve high accuracy but are extremely expensive, for example FlowNet2.0 [11] runs HD video at 8 FPS using several hundred watts of PC+GPU power.

Although LK is widely used, it fails when the images are over or underexposed, and when the images are too blurred to extract good features, and when these features have too much displacement between frames. These scenarios arise in high speed vision, for example in fast drone flight, or under low lighting conditions, where the frame exposure increases, or under natural lighting conditions, where extreme lighting variations, glare, and lens flare are common.

The adaptive block matching optical flow (**ABMOF**) method proposes to address these problems. ABMOF is a semi-dense method that computes flow at points where brightness changes. These brightness changes come from a dynamic vision sensor (**DVS**) event camera. The DVS event camera is a relatively new type of camera [6, 8, 12, 13, 15, 20] that provides sparse asynchronous data output, high dynamic range, high time resolution, and low latency. The DVS output is a variable data-rate stream of timestamped pixel brightness change events.

The core of ABMOF is an event-driven computation of block matching optical flow that operates on variable duration time slice images of accumulated DVS events. ABMOF is directly targeted for efficient multiplier-free parallel hardware implementation using simple logic circuits. The original BMOF [14] does not work well on complex scenes, but we here describe two of the main improvements that form ABMOF:

- A new *AreaEventCount* slice rotation method correctly rotates slices that vary in density of features.
- A new feedback control mechanism for adapting slice duration achieves a desired average match distance that is half of the total search distance, increasing dynamic range.

Comparing both block matching and Lucas-Kanade methods on the adaptive slices shows improvement for both methods compared with fixed slice duration used in [14].

1.1 Prior DVS Optical Flow

This section reviews previous DVS OF algorithms.

[10] described an open-source algorithm (called **DS** in this paper) for time-of-flight DVS OF based on oriented edges detected by spatio-temporal coincidence. It works only for sharp edges and suffers from aperture problems since it is edge-based. [4] adapted the frame-based LK algorithm (called **EBLK** in this paper) to DVS. It stores a fixed-queue length window of past events. For each new event, it computes the LK algorithm on a window of fixed time interval of a block of pixels surrounding the current event pixel. The gradient estimation precision is low due to quantization and small 5x5 window size. The small window size was used to limit the computation time in order to keep up with a high rate of events. [3] proposed a contour-based method. In their work, they reconstruct the contrast of the edge to localize the contour but it is not necessary for frames which have the absolute intensity. The optical flow estimation then is obtained from the contour width divided by the time interval. [5] proposed a time-surface method (called **LP** in this paper) that combines the 2D events and timestamps into 3D space. Normal OF is obtained by robust iterative local plane fitting. It works well for sharp edges but fails with dense textures, thin lines, and natural scenes [21, 24] since these both produce complex structures that plane fitting does not model. [21] implemented and compared the DS, EBLK, and LP methods. It concluded that the existing algorithms were both computationally expensive and do not work well natural scenes and noisy sensor data. This paper also proposed an evaluation method and provided a simple benchmark dataset with ground truth. The ground truth OF is obtained by constraining the camera motion to pure rotation and uses the camera's Inertial Measurement Unit (**IMU**) rate gyros to obtain global translational and rotational OF. [2] proposed a frame-based variational algorithm that simultaneously estimates the optical flow, gradient map, and intensity reconstruction from

DVS. Although the simultaneous constraints results in very regularized output, the results are not quantified, and the method is very expensive compared to others. [24] recently reported the first CNN-based DAVIS OF architecture and published the useful MVSEC dataset. It is trained by minimizing photometric loss from the DAVIS APS frames. It achieves the best published accuracy, but burns 50W to run at 25 Hz frame rate on a laptop gamer GPU.

The above methods are serial algorithms that solve linear or nonlinear constraints; some of them use iteration or exclusion of events to make the solutions more robust. All methods so far have required at least several us/event on a fast PC that consumes many tens of watts.

Hardware implementations: [7] developed a microcontroller-based embedded implementation of a time of flight OF method. It works well for isolated points but not for dense textured scenes; it also has aperture problems with edges. A simplified version of [5] using 3x3 windows was implemented on FPGA in [23]. The small windows restricts it to sharp edges.

This paper improves on the FPGA BMOF algorithm [14]. [14] demonstrated only a basic implementation for a small block size of 9x9 pixels using a single pixel shift in the NSEW compass directions and a fixed time slice duration. It does not work well on most real scenes. But the advantage of [14] is that in hardware, large blocks can be matched with few clock cycles and simple logic. Here we have extended from the [14] implementation to make it suitable for real-world application.

The paper is organized as follows: Sec. 2 explains the idea of block-matching and our improvements. Sec. 3 shows experimental results, and Sec. 4 concludes the paper.

2 ABMOF algorithm

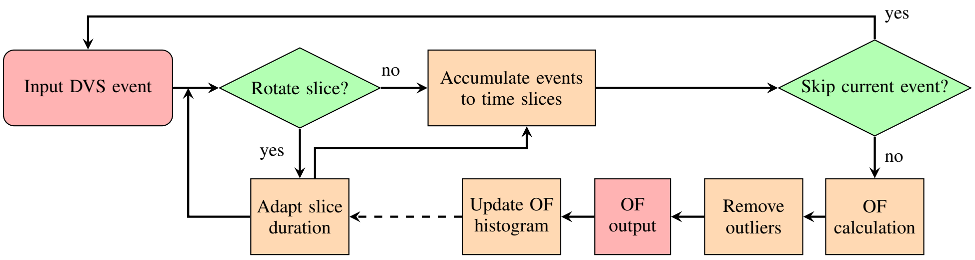


Figure 1: The pipeline of our algorithm.

The pipeline of ABMOF is summarized in Fig 1. When a new Input DVS event arrives, the event's timestamp is used by the rotation logic to determine whether the event slice is to be rotated. If yes, the slices are rotated and the slice duration d or event count parameter K or k is adapted, based on the current slices' OF distribution. The adapted slice duration is sent as an input to the rotation logic. The adaptation takes the OF distribution of the previous slice as the input. We use a dashed connection in the figure to represent their relationship. Details of the rotation logic are introduced in Sec. 2.2. As described in the next section, all the new events will be accumulated to time slices. If the system is busy, the OF calculation for the event is skipped. Otherwise it triggers the OF calculation. After removing outliers, the OF histogram is updated.

All the parameters that are used in this paper are summarized in Table 1.

symbol	description	typical values (default)
$w \times h$	width $\times$ height of pixel array	346x260
d	slice duration	1–100 ms (50)
K	global event number	1k–50k events (10k)
k	area event number	100–1k events (1k)
a	area dimension subsampling	5 bits
b	block dimension	11–21 pixels (21)
r	search radius	4–12 pixels (4)
D	average match distance	ideally $r/2$
(v_x, v_y)	OF result	pixels/sec (pps)

Table 1: Symbols used in this paper.

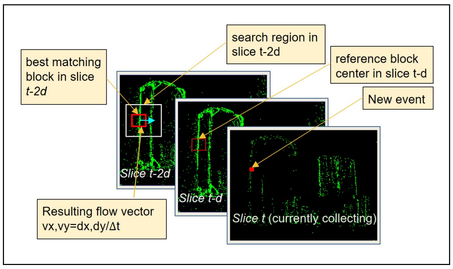

best matching block in slice t-2d
search region in slice t-2d
reference block center in slice t-d
New event
Slice t-2d
Slice t-d
Slice t (currently collecting)
Resulting flow vector vx,vy=dx,dy/Δt

Figure 2: BMOF block matching, on `boxes`, from [21].

2.1 Block-Matching Optical Flow from DVS Time Slices (BMOF)

Figure 2 shows the main principle of BMOF: Three time-slice memories store the events as 2D event histograms: Slice t accumulates the current events. Slices *t-d* and *t-2d* hold the previous two slices. d is the slice duration. When a new event arrives, it is accumulated to slice t by incrementing the pixel value, ignoring the event polarity in this paper. After accumulating the event, then the other two slices are then used to compute the OF based on the current event's location. A reference block ($b * b$ pixels) is centered on the incoming event's location on the *t-d* slice map (red box in slice *t-d*). The best matching block on the *t-2d* slice is found based a diamond search [25] on Sum of Absolute Difference (**SAD**) inside a $(2r+1)^2$ search region, shown as a white rectangle in the *t-2d* slice. Thus the OF result is obtained by using these two blocks' offset (d_x, d_y), divided by the time interval Δt between these two slices. The time of each slice is taken as the average of the first and last timestamp of events accumulated to each slice.

The slices are rotated according to slice rotation logic (Sec. 2.2). The rotation discards the *t-2d* slice and uses its memory for the new slice t; similarly slice t becomes slice *t-d* and slice *t-d* becomes slice *t-2d*. In [14], the slice duration d was set by user manually. For general application a fixed d limits the speed range. In this paper, we propose several methods to adjust it adaptively.

2.2 Slice Rotation Methods

Slice rotation is the core of our algorithm. It calculates when to rotate the slices to ensure good slice quality. Good slices should have sharp features, not too much displacement, and not be too sparse. This goal is achieved by feed-forward and feedback control. We show the details of these two algorithms in the following subsections.

2.2.1 Feed-forward Slice Rotation

The new events are accumulated into the latest slice, slice t, which is only used for accumulation. After that, it will be rotated to be as a past slice and used for OF calculation.

The original BMOF work [14] implemented the method *ConstantDuration*, where each slice has the fixed duration d. It has the disadvantage that the movement between slices may be too large to be matched for fast scene motion, or too small for slow scene motion.

Another obvious method is to rotate slices after a constant number K of events have been accumulated, called *ConstantEventNumber*, which we first used for generating DVS frames

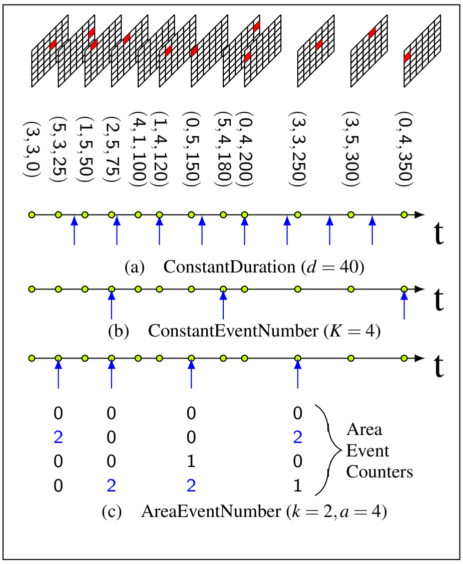


Figure 3: Three feed-forward slice rotation methods.

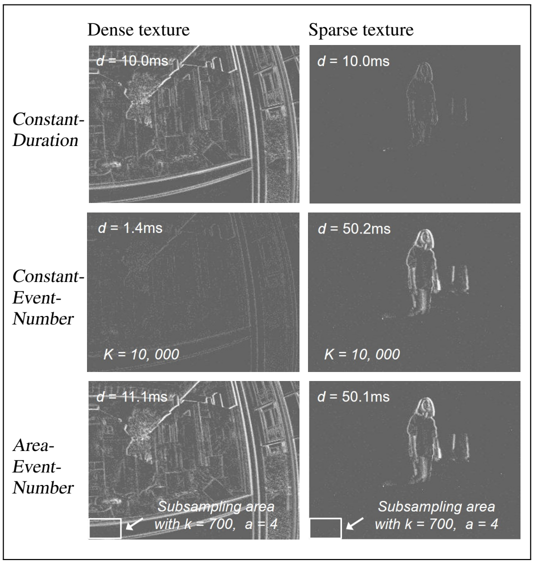


Figure 4: Comparison between event slices generated by three methods.

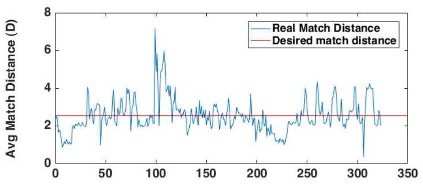
for CNN inference [17] A drawback of the *ConstantEventNumber* method is its global nature. For scenes that vary between having many or few features, it is impossible to set a suitable global K. In order to address this problem, we propose a new rotation method called *AreaEventNumber*. Instead of rotating the slices based on the sum of the whole slice event number, *AreaEventNumber* triggers the slice rotation once any one of the area's event number (Area Event Counters) exceeds the threshold value k. Each area is $(w \times h)/2^a$ pixels, i.e. 10×8 pixels.

Fig 3 compares these three methods. The event stream is at the top of the figure and the information including event address (x,y) and timestamp is shown under the event stream. The blue arrows pointing to the three time axes represent these three rotation method results. Either the time interval or the global event number interval are fixed for *ConstantDuration* and *ConstantEventNumber*. However, both of them vary in the *AreaEventNumber* method which makes it more adaptive to the dynamic scene.

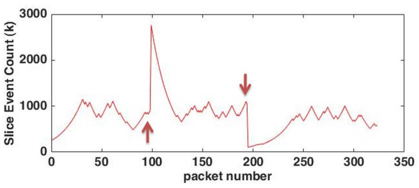
In Fig 4, we compare these three methods on two different scenes, one with sparse texture and the other with dense texture. Both the dense and sparse scenes use the same parameter for every method which is $d = 10ms$ for *ConstantDuration*, $K = 10000$ for *ConstantEventNumber* and $k = 700$ for *AreaEventNumber*. The resulting slice durations are shown overlaid on each scene. Neither *ConstantDuration* nor *ConstantEventNumber* work well on both of these two scenes with fixed values of d or K. For example, *ConstantDuration* fails in the sparse scene because d was set for the faster motion and the duration was too short for the slower motion. *ConstantEventNumber* makes the slice duration too short in the dense scene, because K was set to make a good slice for sparse scene. However, *AreaEventNumber* with fixed parameter k functions well on both of scenes, because it correctly adjusts a good slice duration for the sparse scene after being set for the dense scene. It shows that *AreaEventNumber* is more robust to dynamic scene content.

2.2.2 Feedback Control of Slice Duration

Another method to automatically adjust the slice duration is possible via feedback control. A Blank OF distribution histogram is created after each slice rotation, and then it will be used to collect the OF results until the next slice rotation. The histogram's average match distance D is calculated. If $D > r/2$ where r is the search radius, it means that the slice duration is too long, and so the slice duration or event number is decreased. Otherwise, if $D < r/2$, then it indicates the slices are too brief in duration, and the slice duration or event number is increased. It is possible that slice durations that are too brief or lengthy result in OF results of very small matching distance that are the result of a bias in the search algorithm towards zero motion (small match distance). Stability is improved by limiting the slice duration range within application-specific limits. Here we show results of using bang-bang control, which is the simplest control policy we implemented. A fixed factor of $\pm 5\%$ adjusts the slice duration, where the sign of the relative change of duration is the sign of $r/2-D$. More sophisticated control policies are clearly possible, since the value of the error is directly predictive of the necessary change in the duration.



(a) OF result's match distance



(b) Feedback on slice event count

Figure 5: Feedback on slice event number. (a) shows the OF result's real match distance and its desired match distance. (b) represents the slice event count number.

The data in Figure 5 shows an example of event number control using the *AreaEvent-Number* rotation policy with feedback control of k. Figure 5(a) shows the average OF match distance D. The feedback control of event number holds D at its $r/2$ value of about 2.5 pixels. Figure 5(b) shows the event number k. It has a steady state value of about 1000. Around packet 100 (1st arrow), k was manually perturbed to a large value, resulting in a increase in D, but it rapidly returns to the steady state value. At around packet 200 (2nd arrow), k was manually reduced to a small value, resulting in a small D of about 1 pixel. Again D returns to the steady state value. This data shows the stability of the event number control with this feedback mechanism.

3 Results

ABMOF was implemented in jAER [9]. In this section, we show several experiments to validate the ABMOF algorithm. They can be classified into two types. The first type are the experiments with ground truth OF. We show both the qualitative and quantitative results in these experiments; see Sec. 3.1. In the second type of experiment, we test our algorithm on several complex scenes. They consist of camera rotation over gravel (`gravel`), flow through an indoor office environment (`office`), and uniform flow created by a variety of shapes (`shapes`, from [18]). Details and download link of the dataset are provided[1] to support the tests for other future algorithms. For the new data we gathered for this work, we used an advanced event camera with 346x260 pixel resolution. It is a DAVIS [6] that supplies concurrent conventional image output with the DVS data stream, but these images are not used by ABMOF. The dataset web page will be included in accepted paper, to support future algorithm research. Due to unknown ground truth in these files, we show only a comparison between the ABMOF and Lukas Kanade results using the generated slices for these data; see Sec. 3.2.

3.1 Type I experiment result

We show the results of type I experiments in this subsection. We measured four metrics to evaluate the algorithms. They are event density (**ED**), translational global flow (**GF**), Average Endpoint Error (**AEE**) and Average Angular Error (**AAE**). ED is the fraction of DVS events that result in OF results. OF results are skipped when block matching fails to pass outlier rejection tests. ED relates to the density of the flow computation. LK has very low density because it relies on features. An ED of 100% means that all pixel brightness changes result in OF events. AEE and AAR are defined for DVS OF in [21].

Besides the ABMOF, we also implemented the Lucas Kanade (**LK**) OF calculation based on our generated adaptive event slices using OpenCV, which we here name **ABMOF_LK**. ABMOF_LK uses our algorithm to set the time slice duration, and these generated slices are treated as conventional gray scale image frames. In ABMOF_LK, corners are first extracted by Shi-Tomasi corner detector [22] and then they are are passed to the LK tracking algorithm implemented in OpenCV [1]. LK estimates the OF result based on these features.

We also compared the AMBOF OF methods with previously published implementations from [21]: *DirectionSelectiveFlow* (DS) [10], the event-based *LucasKanadeKFlow* (EBLK) [4], and *LocalPlanesFlow* (LP) [5].

3.1.1 `slider` scene

Figure 6 shows the qualitative results of ABMOF and ABMOF_LK on the `slider_hdr_far` data [18]. It is a high dynamic range scene of a flat poster with uniform flow about about 90 pixels/sec (pps). Because of the lighting contrast, the APS images are sometime extremely over- or underexposed, but the DVS events respond to the local brightness changes.

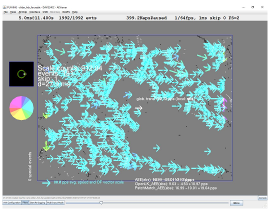
This experiment validates that the slice rotation methods result in quantitative flow magnitude that is the same as from Frame based LK . and all ABMOF methods are all much more accurate and less noisy than the prior DS, EBLK, and LP methods.

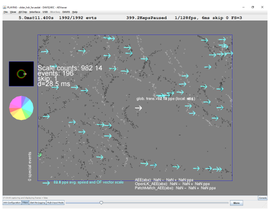
Table 2 reports the quantitative comparison. Because a dataset consists of several data packets, we measure the average and variance of all the packets' global flow. Table 2 shows that ABMOF_LK's GF's average on the `slider_hdr_far` is less than [1 pps, 0.44 pps] and ABMOF's GF's average error is less than [4 pps, 0.2 pps]. ABMOF_LK is more accurate than ABMOF, but has much lower ED.

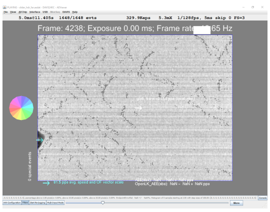
3.1.2 `pavement_fast` scene

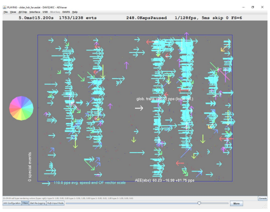
Figure 7 shows the results of a very high speed experiment on `pavement_fast`. The global flow is an extremely fast 32k pps, which means that a pixel crosses the 346-pixel

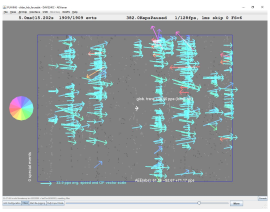
[1] ABMOF dataset README link


(a) ABMOF


(b) ABMOF_LK


(c) APS image 2 and LK result


(d) DS


(e) EBLK

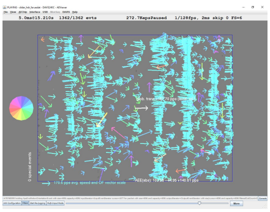
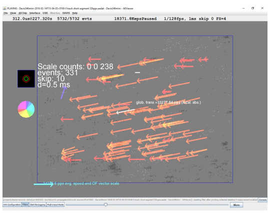
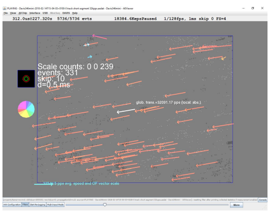

(f) LP

Figure 6: Result of ABMOF, ABMOF_LK and standard LK on image frames on `slider_hdr_far`. For 6(a) and 6(b), we use *AreaEventNumber* with feedback enabled, $r = 4$ and used standard LK on successive APS images in 6(c).

array in about 10 ms. Figs. 7(a) - 7(b) compares ABMOF and ABMOF_LK on DVS time slices, and Figure 7(c) shows conventional LK on successive DAVIS APS images (the 2nd image is shown under the flow result). Both ABMOF and ABMOF_LK correctly measure the true flow using a slice duration of only 450 us, equivalent to a frame rate of 22 kHz and a 14 pixel displacement between slices. The consecutive APS image frames were collected at the maximum frame rate of 50 Hz, but because the motion is so fast, even the short DAVIS global shutter exposure of 0.7 ms resulted in visible image blur of several pixels. And since the consecutive frames are separated by 20ms, the images are completely uncorrelated and the resulting flow is meaningless, as seen in Figure 7(c).


(a) ABMOF


(b) ABMOF_LK

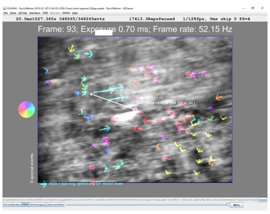

(c) APS image 2 and LK result

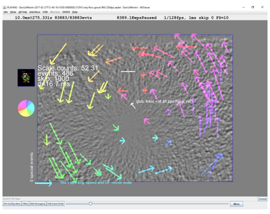
Figure 7: Result of ABMOF, ABMOF_LK and standard LK on image frames on `pavement_fast`. For 7(a) and 7(b), we fixed $d = 450$ us, $r = 12$ and used standard LK on successive APS images in 7(c).

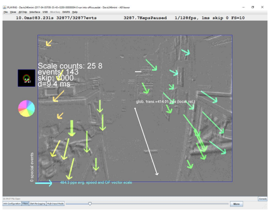
method	**event density**	**global flow (pps)**	**AEE (pps)**	**AAE (°)**
Groundtruth	-	$[90.50, 0] \pm [0.43, 0]$	-	-
ABMOF_LK	0.39%	$[89.75, 0.44] \pm [6.30, 3.56]$	8.75±27.51	2.95±3.41
ABMOF	37.96%	$[86.85, 0.17] \pm [8.46, 1.25]$	12.68±16.28	3.66±8.31
Frame_based_LK	-	$[89.51, 0.20] \pm [3.203.48]$	5.47±42.07	1.30±4.72
DS ([10, 21])	49.86%	$[74.97, 2.98] \pm [17.42, 4.79]$	57.71±53.31	21.46±39.13
EBLK ([4, 21])	17.53%	$[28.06, -0.11] \pm [4.09, 1.32]$	60.32±15.92	13.52±25.51
LP ([5, 21])	83.88%	$[161.14, 11.69] \pm [8.67, 12.13]$	99.00±75.86	16.99±24.41

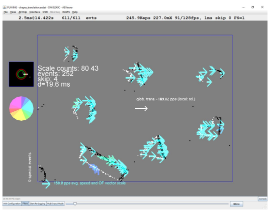
Table 2: Comparison of algorithm's overall accuracy on `slider_hdr_far`.

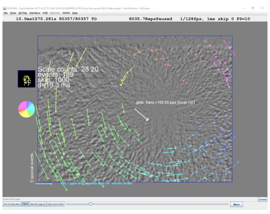
3.2 Type II experiment result

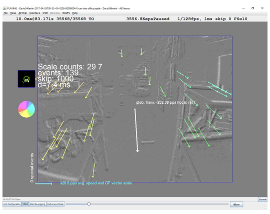
The final experiments are from natural scenes that contain a range of directions and speeds. Since we lack ground truth OF for these natural scenes, we only show the qualitative comparison of ABMOF and ABMOF_LK. These results are shown in Fig 8.


(a) ABMOF for `gravel`


(b) ABMOF for `office`


(c) ABMOF for `shapes`


(d) *ABMOF_LK* for `gravel`


(e) ABMOF_LK for `office`

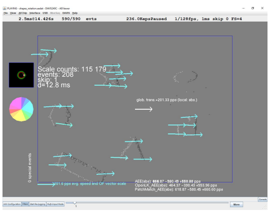

(f) ABMOF_LK for `shapes`

Figure 8: Result of the algorithms in different scenes. All scenes were captured using block size $b = 21$ pixels, using diamond search, with search distance $r = 4$ pixels and using feedback control of *AreaEventNumber* k. OF color and angle represents direction according to the color wheel and vectors' length means OF speed relative to the scale shown at bottom left of each frame.

Vectors represent the OF result and color shows the direction. The ABMOF and ABMOF_LK produce very similar OF for these natural scenes. For `gravel` and `office`, both methods correctly extract the structure of the flow. For `shapes`, ABMOF flow is quite dense along object edges and the large block size of 21 pixels results in true OF rather than normal flow. For this same scene, ABMOF_LK attaches OF only to object corners; ABMOF_LK misses the OF on the upper right ellipse, but the overall flow is more uniform.

4 Conclusion

We proposed improvements for the basic BMOF algorithm for DVS. A new pipeline adjusts the slice duration based on the local movement rather than the global motion. A feedback mechanism for slice duration makes the average displacement between two slices close to the half of the search distance. ABMOF achieves good quality of OF estimation with low algorithm hardware complexity. With the adaptive slice duration, fast motion can result in slice durations that are fractions of a millisecond, as in the `pavement_fast` example of Sec 3.1.2, allowing measurement of speeds $>$ 10k pps. This result was previously only the domain of high-end gaming mouse sensors such as [19].

By extrapolating the FPGA hardware implementation costs from [14], we estimate that ABMOF can be implemented on a medium sized FPGA fabric, such as Xilinx Zynq-7000 family chip XC7Z100. The total/percent utilization will require about 35k/6% Flip-Flops, 100k/36% Look-Up Tables and 400 kb/1% block RAM. The resulting IP block could later be integrated together with the sensor in a custom digital core. The result would provide economical and fast OF that works well in natural scenes, and that takes advantage of the basic DVS abilities to sparsely and asynchronously signal brightness changes in high dynamic range and fast moving scenes.

Acknowledgments

This work is funded by Swiss National Center of Competence in Research Robotics (NCCR Robotics).

References

[1] OpenCV: Optical Flow. URL https://docs.opencv.org/3.3.1/d7/d8b/tutorial_py_lucas_kanade.html.

[2] Patrick Bardow, Andrew J Davison, and Stefan Leutenegger. Simultaneous optical flow and intensity estimation from an event camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 884–892, 2016.

[3] Francisco Barranco, Cornelia Fermüller, and Yiannis Aloimonos. Contour motion estimation for asynchronous event-driven cameras. *Proceedings of the IEEE*, 102(10): 1537–1556, 2014.

[4] Ryad Benosman, Sio-Hoi Ieng, Charles Clercq, Chiara Bartolozzi, and Mandyam Srinivasan. Asynchronous frameless event-based optical flow. *Neural Networks*, 27(30):32–37, 2012. doi: 10.1016/j.neunet.2011.11.001. URL http://www.sciencedirect.com/science/article/pii/S0893608011002930?via%3Dihub.

[5] 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, 2014.

[6] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240x180 130 dB 3 us latency global shutter spatiotemporal vision sensor. 49(10): 2333–2341. ISSN 0018-9200. doi: 10.1109/JSSC.2014.2342715.

[7] Jörg Conradt. On-board real-time optic-flow for miniature event-based vision sensors. In *Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on*, pages 1858–1863. IEEE, 2015.

[8] T. Delbruck, B. Linares-Barranco, E. Culurciello, and C. Posch. Activity-driven, event-based vision sensors. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2426–2429. doi: 10.1109/ISCAS.2010.5537149.

[9] Tobi Delbruck. Java AER framework, 2007. URL https://jaerproject.org.

[10] Tobi Delbruck. Frame-free dynamic digital vision. In *Proceedings of Intl. Symp. on Secure-Life Electronics, Advanced Electronics for Quality Life and Society*, pages 21–26, 2008.

[11] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. *arXiv:1612.01925 [cs]*, December 2016. URL http://arxiv.org/abs/1612.01925. arXiv: 1612.01925.

[12] Chenghan Li, Christian Brandli, Raphael Berner, Hongjie Liu, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. An RGBW color VGA rolling and global shutter dynamic and active-pixel vision sensor. In *2015 International Image Sensor Workshop (IISW 2015)*, pages 393–396. imagesensors.org.

[13] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128 x 128 120 dB 15 Âţs latency asynchronous temporal contrast vision sensor. 43(2):566–576. ISSN 0018-9200. doi: 10.1109/JSSC.2007.914337. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4444573.

[14] M. Liu and T. Delbruck. Block-matching optical flow for dynamic vision sensors: Algorithm and FPGA implementation. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, May 2017. doi: 10.1109/ISCAS.2017.8050295.

[15] Shih-Chii Liu, Tobi Delbruck, Giacomo Indiveri, Adrian Whatley, and Rodney Douglas. *Event-Based Neuromorphic Systems*. John Wiley and Sons Ltd., UK. URL 10.1002/9781118927601.

[16] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[17] D. P. Moeys et al. Steering a predator robot using a mixed frame/event-driven convolutional neural network. In *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, pages 1–8, June 2016. doi: 10.1109/EBCCSP.2016.7605233. URL https://drive.google.com/open?id=0BzvXOhBHjRheVWVoRlBLUGViUlk.

[18] 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.

[19] PixArt Imaging Inc. PixArt PMW3360 Optical Gaming Navigation Sensor, 2014. URL https://docs.google.com/viewer?url=http%3A%2F%2Fwww.pixart.com.tw%2Fupload%2FPMS0058-PMW3360DM-T2QU-NNDS-R1.30-06042016_20160902201411.pdf.

[20] C. Posch, D. Matolin, and R. Wohlgenannt. An asynchronous time-based image sensor. In *IEEE International Symposium on Circuits and Systems, 2008. ISCAS 2008*, pages 2130–2133. doi: 10.1109/ISCAS.2008.4541871.

[21] Bodo Rueckauer and Tobi Delbruck. Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor. *Frontiers in Neuroscience*, 10:176, 2016. ISSN 1662-453X. doi: 10.3389/fnins.2016.00176. URL http://journal.frontiersin.org/article/10.3389/fnins.2016.00176.

[22] Jianbo Shi et al. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[23] M. Tun Aung, R. Teo, and Garrick Orchard. Event-based Plane-fitting Optical Flow for Dynamic Vision Sensors in FPGA. In *IEEE Int. Symp. Circuits Syst. (ISCAS)*, Florence, Italy, 2018.

[24] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. EV-FlowNet: Self-supervised optical flow estimation for event-based cameras. URL http://arxiv.org/abs/1802.06898.

[25] Shan Zhu and Kai-Kuang Ma. A new diamond search algorithm for fast block matching motion estimation. In *Information, Communications and Signal Processing, 1997. ICICS., Proceedings of 1997 International Conference on*, volume 1, pages 292–296. IEEE, 1997.