Low-latency and Scene-robust Optical Flow Stream and Angular Velocity Estimation

SANGIL LEE¹ AND H. JIN KIM¹
¹Automation and Systems Research Institute, Department of Mechanical and Aerospace Engineering, Seoul National University, Seoul, 08826, Korea, Republic of (e-mail: {sangil07, hjinkim}@snu.ac.kr)
Corresponding author: H. Jin Kim (e-mail: hjinkim@snu.ac.kr)

ABSTRACT Event cameras are bio-inspired sensors that capture intensity changes of pixels individually, and generate asynchronous and independent “events”. Due to the fundamental difference from the conventional cameras, most research on event cameras builds a global event frame by grouping events according to their timestamps or their number to employ traditional computer vision algorithms. However, in order to take advantage of event cameras, it makes sense to generate asynchronous output on an event-by-event basis. In this paper, we propose an optical flow estimation algorithm with low latency and robustness to various scenes to utilize the advantage of the event camera by enhancing the existing optical flow algorithm. Furthermore, we estimate angular velocity with low latency using the proposed optical flow stream. For the validation of algorithms, we evaluate the accuracy and latency of optical flow with publicly available datasets. Moreover, we assess the performance of the proposed angular velocity estimation in comparison to the existing algorithms. Both validations suggest that our asynchronous optical flow shows comparable accuracy to the existing algorithms and the latency is reduced by half compared to the existing block matching algorithm on average. Also, our angular velocity estimation is superior to the existing algorithms in terms of accuracy and robustness while showing low latency within 15 ms consistently.

INDEX TERMS event camera, low-latency, optical flow, angular velocity

I. INTRODUCTION

EVENT camera, also known as a dynamic vision sensor (DVS), is a bio-inspired sensor that behaves differently from conventional cameras. The event camera, first designed in [1] and practically proposed in [2], [3], generates asynchronous events at a pixel whose log-intensity changes, unlike frame-based cameras which capture the whole absolute intensity image. The asynchronous characteristic of event cameras brings some advantages; it generates event steam with lower latency in microseconds and grasps the motion relative to it with higher time resolution than the conventional camera.

However, this new paradigm makes it difficult to apply the existing image-based traditional computer vision techniques, such as feature detection [4], [5], tracking [6], [7], optical flow [8]–[11], and motion estimation [12]–[14], to event cameras directly. The current research on event cameras can be categorized into two approaches. One builds an event frame by stacking events in a specified count or time window. The other utilizes events asynchronously, which is also known as event-by-event basis operation.

A simple way to adopt the asynchronous approach is to use a single incoming event. However, due to sensor noise, an event may occur where an apparent motion does not exist. Even if an algorithm is performed with valid events only, it takes quite a bit of time to build a local intensity map [15] and obtain the desired output for bootstrapping. This is because the individual event has little information and it implies both spatial and temporal information not just either of the two, which makes it difficult to interpret a single event.

In this paper, we are interested in developing computer vision applications for event cameras without deteriorating the low latency characteristic, one of the important strengths of event cameras (see Fig. 1). The latency of the algorithm can be attributed to practical latency and/or theoretical latency. The practical latency includes actual computation time associated with programming setup or hardware performance. Meanwhile, theoretical latency denotes the time to gather
events until estimation of optical flow or angular velocity is completed ignoring computation time. We aim to reduce theoretical latency that occurs independent of programming setup or hardware performance. Particularly, we focus on estimating asynchronous optical flow stream on an event-by-event basis. The reason why we choose optical flow is that it is one of the fundamental elements in many computer vision algorithms, which can be utilized in feature tracking, motion estimation, etc. To measure the latency of the optical flow stream, we estimate 3D angular velocity from a bunch of optical flows. The evaluation suggests that our angular velocity estimation shows higher accuracy and lower latency than the other existing algorithms, which also implies a low latency of the proposed optical flow.

A. RELATED WORK

For dealing with asynchronous event streams, there are roughly two main approaches. The first type of research performs an algorithm on each incoming event without constructing event frames, and makes use of the strength of event cameras. Each event has a small amount of information, thus most research with an event-by-event basis implements filter-based [12], [15] or network-based algorithms [16]. However, since the time for a filter or a network to converge is not negligible, it is an obstacle to the high-speed capability of event cameras.

Another type of research builds an event frame by stacking events according to their timestamp [9], [17] or grouping them with a specific number [13]. The former stacking method is very similar to the frame of the conventional camera, which has a fixed frame rate. However, when the camera moves slowly, the number of stacked events might be insufficient to obtain valid output, because the event frame consists of a small number of events. Conversely, in a fast motion, bleeding edges can appear in event frames [18]. On the other hand, the quality of the latter grouping method depends on the degree of textures. In the scene with sparse textures, the number of events in the group should be reduced to prevent edge bleeding, and the inverse relationship is also established. Particularly, in the case of a scene with non-uniform textures, some areas of the frame suffer from bleeding edge while others suffer from lack of events. Although some research capture the adaptive event frame with different time windows [7] or the number of events [8], [19], their global event frame still loses details of the scene consisting of non-uniform textures because their event frames stack events across the whole frame as mentioned above.

In summary, algorithms on an event-by-event basis suffer from slow convergence of a filter or a network, which diminishes the advantages of event cameras: high-speed capability or low-latency. On the contrary, building an event-stacked image requires heuristic adjustment of the time window or the number of events and also diminishes the advantage of event cameras: high time resolution or low latency. For these reasons, in order to maintain the capability of event cameras, we design an optical flow algorithm that is asynchronous not only temporally but also spatially.

1) Optical flow

The existing optical flow algorithms for event cameras are roughly divided into two categories as above. Adaptive time-slice block-matching optical flow algorithm (ABMOF) [8], [19] computes optical flow at where the event triggers for every incoming event using the two most recent of the previous time slices that are similar to the surface of active events (SAE) [20]. It produces asynchronous optical flow streams, but latency occurs because it uses two of the past time slices, not including the current event. Moreover, in the case of a scene where the texture is not uniform, events are rapidly accumulated in some areas with complex textures, so a new time slice may be generated even though sufficient events are not accumulated in other areas with sparse textures. LocalPlane [21] generates normal flow by using the gradient vectors, which is computed from the local plane fitting on SAE. Although it estimates flow vector with negligible latency, local plane fitting approaches are vulnerable to the aperture problem, thus generating optical flows that are different from true optical flows.

On the other hand, in [9]–[11], [13], [17], they aggregate events to estimate optical flow, thus the latency of their algorithms depend on heuristic parameters such as time window or event counts. Bardow et al. [17] use variational method for solving motion field map and brightness image simultaneously. They discretize the time window into \( K \) intervals each of length \( \delta_t \) ms that is set heuristically depending on the speed of motion in a scene. Contrast maximization framework [13] solves various problems such as optical flow, depth, and motion estimation by maximizing the sharpness of stacked image of warped events according to the model parameter. It uses a set of several tens of thousand events, which is determined heuristically depending on the texture.
of a scene. Also, their simple framework can be adapted to use a set of events within a time window and both versions are validated in this paper. Zhu et al. [9], [10] train convolutional neural network (CNN) to learn to estimate optical flow in self-supervised and un-supervised manners, respectively. However, CNN-based algorithms take image-like data as input, and the input representation consists of tens of thousands of events. In order to handle asynchronous and discrete events over time, there is research on combining event camera and spiking neural network (SNN) [11], [22]. Lee et al. [11] present a deep hybrid architecture by integrating SNNs for encoder and analog neural networks (ANNs) for residual and decoder layers. However, their accumulators in between SNNs and ANNs layers collect output from SNNs until all event images have passed, thus increasing the latency.

2) Angular velocity
Research on angular velocity estimation for event cameras includes an algorithmic method that utilizes an events group [13], [14], [23] and a network-based method with an event-by-event basis [16]. The former algorithm, referred to as contrast maximization, estimates angular velocity using a warping model which transforms an event in \((x, y, \tau)\) space into the same \(xy\)-plane with a rotation parameter. This optimization-based method produces precise angular velocity, but it requires heuristic parameter adjustment depending on the scene as mentioned before. In [16], they propose angular velocity regression with SNN. The input representation of their network is a set of events within a time window of 1 ms. The network successfully predicts the angular velocity, but it takes a settling time of 50 ms due to SNN’s dynamics at the beginning of predictions and has not been validated for a drastic motion.

B. CONTRIBUTIONS AND OUTLINE
We tackle an optical flow and angular velocity estimation problem without any assumptions about the environment, initialization, or additional sensors. Main contributions can be summarized as follows:
- We propose a low-latency algorithm that uses only events and robustly estimates optical flow in various environments.
- We present an accurate 3D angular velocity estimation algorithm, which fetches the proposed asynchronous optical flow stream.
- We compute latency between estimates and ground-truth in an optimization-based method and analyze algorithms thorough in terms of latency and accuracy.

The rest of the paper is organized as follows: In Section II, we explain the proposed asynchronous optical flow and angular velocity estimation algorithms. Next, we evaluate ours and the existing algorithms on publicly available datasets to analyze the accuracy and latency in Section III. Then, Section IV discusses additional validation of performance for various configurations and future work related to our limitations, followed by the summary of the paper in Section V.
OF visual simplicity, we illustrate the case for patch size a pixel, and a new event (magenta cube) is pushed into adjacent bins. For Algorithm 1 use the diamond search method [24] for block matching to construct two time slices. In common with ABMOF, we through 5 of Algorithm 1. For every incoming event, we increases. Consequently, our approach utilizing the current between two previous time slices. Because ABMOF does not of the queue, whereas ABMOF finds the best matching block event to the previous time slice consisting of the front half algorithm compares the current time slice including the latest event to the previous time slice consisting of the front half of the queue, whereas ABMOF finds the best matching block between two previous time slices. Because ABMOF does not use the current event when computing optical flow, its latency increases. Consequently, our approach utilizing the current event immediately estimates optical flow with less latency.

Algorithm 1 demonstrates the pseudocode of the algorithm explained in Section II-B, and Fig. 3 illustrates steps 1 through 5 of Algorithm 1. For every incoming event, we push the event into the corresponding queues which cover the pixel position of the event. Then, if the queue is full, we divide the elements of the queue in half and stack them to construct two time slices. In common with ABMOF, we use the diamond search method [24] for block matching algorithm based on the sum of absolute difference (SAD) between two time slices. For efficient and accurate search, we find the best matching block quickly and iteratively in the current time slice with a large diamond search pattern and make an exhaustive search within a small diamond search pattern. After block matching, asynchronous optical flow stream contains the timestamp \( t_i \) (the latest timestamp of the queue) and pixel location of the event \( p_i \) (starting point of the vector), optical flow vector in pixels \( v_i \), and the average time difference between two time slices \( dt_i \), so that the optical flow can be converted to \( v_i / dt_i \) having a unit of px/s:

\[
OF_i = (t_i, p_i, v_i, dt_i). \tag{2}
\]

In practice, we do not construct time slices from the beginning repeatedly, but update them gradually for each incoming event for efficiency. When a new event is fed into the queue, an event in the middle of the queue begins to construct the previous time slice. At that time, the previous time slice is only affected by the event in the middle and the oldest event which will be popped. Also, the current time slice is modified by only the event in the middle and the new event. In other words, the time slice stacks the timestamp of the event when fetching an event. Next, for an event that will be popped, the value at the corresponding pixel location in the time slice is reset as zero only if the existing value of the time slice is the same as the timestamp of the event.

### C. Angular Velocity Estimation

To deal with a single piece of data from an asynchronous optical flow stream, the use of filters could be considered. Since the equation solving optical flow given an angular velocity and a pixel location is a linear transformation and a linear operator to a Gaussian distribution also results in the same distribution, the Kalman filter can be applied. However, the filter-based method increases latency due to its dynamic characteristics and requires many tuning parameters for the covariance matrices.

Because of the above reasons, we decide to compute angular velocity analytically from a bunch of optical flows. Given the vector \( \omega \) representing 3D angular velocity, optical flow under a pure rotation is computed by \( v_i = KR(\omega dt_i)K^{-1}p_i - p_i \), where \( K \) is the intrinsic matrix of the camera, \( R(\cdot) \) is the rotation matrix of the corresponding \( \omega \) and time passed \( dt_i \), and \( p_i \) is image point in the pixel coordinates. In a short time interval, the equation can be approximated as below:

\[
v_i = \begin{pmatrix} f_x & 0 \\ 0 & f_y \end{pmatrix} \begin{pmatrix} x_i y_i & 1 + y_i^2 \\ -x_i y_i & -x_i \end{pmatrix} \begin{pmatrix} -1 - x_i^2 & y_i \\ -x_i y_i & -x_i \end{pmatrix} dt_i \omega, \tag{3}
\]

where \( f_x, f_y \) are the focal length of the principle axes in pixels, \( x_i, y_i \) are image points in the normalized image coordinates, and \( dt_i \) is the time interval while the image point has moved. Let (3) be \( v_i = A_i dt_i \omega \) for simplicity, and we find the angular velocity by solving the least-squares problem:

\[
\omega = (A_i^T A_i)^{-1} A_i^T \omega, \tag{4}
\]
where $A_{1:n} = \begin{bmatrix} A_1^T, \ldots, A_n^T \end{bmatrix}^T \in \mathbb{R}^{2n \times 3}$, $v_{1:n} = \begin{bmatrix} v_1^T, \ldots, v_n^T \end{bmatrix}^T \in \mathbb{R}^{2n \times 1}$, and $T_{1:n} \in \mathbb{R}^{2n \times 2n}$ is diagonal matrix composed of $\Delta t_i$, i.e., $T_{2i-1,2i-1} = T_{2i,2i} = \Delta t_i, \forall i = 1, \ldots, n$ for $n$ measurements.

In addition, let us assume that each vector of optical flow stream has 2D Gaussian pixel noise, i.e., $v_i \sim N(\bar{v}_i, \Sigma_v)$. Since our asynchronous optical flow stream has integer values $v_i = (v_x, v_y)$ due to the diamond search method, we set $\Sigma_v = 0.5^2 I_2 \in \mathbb{R}^{2 \times 2}$. Then, the covariance of $\omega$ is

$$\operatorname{Var}(\omega) = (A^T A)^{-1} A^T T^{-1} \operatorname{Var}(v) T^{-1} A (A^T A)^{-1} = \sigma^2 (A^T A)^{-1} A^T T^{-2} A (A^T A)^{-1}. \tag{5}$$

To handle asynchronous stream input, our algorithm iteratively computes the above variation using all incoming optical flow, and estimates 3D angular velocity when $\det(\operatorname{Var}(\omega))$ is less than the heuristic threshold value or the number of measurement $n$ reaches the threshold, $n_{\text{max}}$. The determinant of the covariance matrix is computed sequentially:

$$B_{1:n} = B_{1:n-1} + A_n^T A_n / \Delta t_n^2, \tag{6}$$

$$C_{1:n} = C_{1:n-1} + A_n^T A_n, \tag{7}$$

$$\det(\operatorname{Var}(\omega))_n = \sigma^2 \det(B_{1:n}) / \det(C_{1:n})^2, \tag{8}$$

where $B_{1:1} = A_1^T A_1 / \Delta t_1^2$ and $C_{1:1} = A_1^T A_1$. To reduce the influence of optical flow belonging to dynamic object, we utilize random sample consensus (RANSAC) to estimate accurate angular velocity.

### III. EVALUATION

We evaluate our optical flow and angular velocity estimation algorithm in terms of accuracy and latency. The accuracy of optical flow is validated on multi vehicle stereo event camera (MVSEC) [9] sequences while the accuracy of angular velocity is on dynamic and active-pixel vision sensor (DAVIS240C) [25], [26] sequences. Each snapshot of sequences is shown in Fig. 4. The latency of optical flow and angular velocity estimation algorithms, which is the main interest of this paper, is computed by the optimization approach. Particularly, we compute the latency of optical flow algorithms indirectly via back-end angular velocity estimation, since it is difficult to compute the latency of optical flow directly. In evaluation, we set block width $w$ as 25 px, the capacity of the queue as 300 for optical flow, i.e., 150 events exist in a single local time slice. Also, the heuristic threshold for the determinant of the covariance of angular velocity is set to $0.001 \approx 3^6 \approx \sigma_\omega^2 \sigma_\omega^2 \sigma_\omega^2 (\text{rad}/s)^2$, and the maximum number of optical flows $n_{\text{max}}$ to 150 for angular velocity. These parameters are the same for all sequences with rich or poor texture of scene, fast or slow motion.

### A. LATENCY COMPUTATION

To estimate the latency of the proposed algorithms, we compare angular velocity estimates and ground-truth motions. Even though DAVIS240C sequences have mainly sinusoidal movements, the frequency spectrum of the angular velocity estimates is corrupted due to the estimation noise, and thus it is difficult to estimate the phase difference between estimates and ground-truth. Instead, we compute latency by minimizing the sum of squared errors between the estimates and ground-truth motions. Also, in order to focus on computing latency and consider overestimation or underestimation and bias, we add scale $A = a I_3 \in \mathbb{R}^{3 \times 3}$ and bias parameters $b \in \mathbb{R}^{3 \times 1}$ for affine transformation and optimize below:

$$\min \sum_i w_i L_\delta \left(A \hat{x}_{\text{est}}(\tau_i + \tau_d) + b - \hat{x}_{\text{gt}}(\tau_i)\right), \tag{9}$$

where $\tau_d$ is an important parameter meaning the latency which we pay attention to. $\tau_i$ is the $i$-th timestamp of estimates $x_{\text{est}}$, and $\hat{x}_{\text{gt}}(\tau_i)$ is an interpolated value of the ground-truth at $\tau_i$. To reduce the sensitivity of outliers, we use the Huber loss function $L_\delta(\cdot)$. Also, to diminish the influence of high-frequency noise, the weight $w_i$ is designed to be proportional to the magnitude of the slope of low-pass-filtered $x_{\text{gt}}$. In evaluation, because the noise level and accuracy are different for each axis, we compute the latency of each angular velocity for each axis and report the maximum latency among them.

### B. OPTICAL FLOW ESTIMATION

We validate the performance of the low-latency optical flow algorithm qualitatively and quantitatively on MVSEC sequences. MVSEC provides multiple sensors’ data including events and ground-truth flowmap of DVS as a form of a frame. Due to the frame-based ground-truth, we scale the magnitude of asynchronous optical flow vector of the proposed algorithm by the ratio between the time interval of flowmap and the time difference of optical flows. For example, the magnitude of optical flow with a small time difference is enlarged under the assumption that the optical flow is constant during the interval between successive flowmaps. Moreover, since the events are generated at the edge of a scene at which the true flowmap has discontinuities,
the starting point of optical flow cannot be matched with the corresponding point of the flowmap exactly under a translational motion. Overall, to reduce faulty end-point errors caused by unsynchronized timestamps and discontinuity, we compensate the magnitude and the starting point of optical flows as below:

\[
\hat{v}_i = v_i \times \frac{t_{gt,k} - t_{gt,k-1}}{dt_i}, \tag{10}
\]

\[
\hat{p}_i = p_i + v_i \times \frac{t_{gt,k} - t_i}{dt_i}, \tag{11}
\]

s.t. \( t_{gt,k-1} < t_i < t_{gt,k}, \forall i, \)

where \( t_{gt,k} \) is the timestamp of the \( k \)-th true flowmap. Since the true flowmap represents optical flows based on the current scene, we validate asynchronous optical flows by projecting them onto the synchronized flowmap. The effect of asynchronous optical flow compensation is shown in Fig. 5. Thanks to the compensation using (10) and (11), the outliers in Fig. 5(a) caused by the faulty comparison disappear, thus making distribution smooth. The compensation not only banishes the outliers but also reduces errors overall.

For validation, we compare our optical flow algorithm with ABMOF, EV-FlowNet (EV-FN) [9]. We implemented ABMOF in C++ by referring to java tools for address-event representation (jAER) open-source project (http://jaerproject.org). However, ABMOF fails on MVSEC sequences because the sequences have a non-uniformly textured scene. Under such sequences, a slice duration adjustment module of ABMOF does not work properly, thus slowing down update rates and drastically reducing accuracy. Also, its outlier rejection discards a correct optical flow under fast motions in DAVIS240C sequences. Thus, we disable slice duration adjustment and outlier rejection. Instead, to validate the performance of ABMOF in a good condition, we test ABMOF with the same fixed number of events as ours, but other parameters are the same as the original version [8]. Additionally, we also tested LocalPlane proposed in [21], but its performance deteriorates in complicated realistic environments such as MVSEC sequences because its optical flow is computed from the slope of the local SAE plane and is different from the true optical flow. Thus, we did not state the performance of LocalPlane in this paper. The code of EV-FlowNet is available in public, and we use them in evaluations.

![Figure 5](Image 33x637 to 279x722)

**FIGURE 5.** Histogram of optical flow end-point errors w/o or w/ compensation on indoor_flying1 sequence.

---

**TABLE 1.** Evaluations of optical flow on ground-truth flowmap.

| Sequence | Average end-point error (px) |
|----------|-----------------------------|
|          | Ours | ABMOF | Zhu et al. [10] | EV-FN | Spike-FN |
| indoor1  | 1.09 | 1.28  | 0.38           | 1.05  | 0.84    |
| indoor2  | 1.76 | 1.78  | 0.62           | 1.72  | 1.28    |
| indoor3  | 1.54 | 1.74  | 1.42           | 1.25  | 1.11    |
| outdoor1 | 1.94 | 2.38  | 0.14           | 1.92  | 1.09    |

---

**TABLE 2.** Evaluations for latency and accuracy of optical flow. Best results are in bold.

| Sequence  | Average latency (ms) | Average accuracy (rad/s) |
|-----------|----------------------|--------------------------|
|           | Ours | ABMOF | Zhu et al. [10] | EV-FN | Spike-FN |
| shapes    |      |       |                |       |           |
| low       | 14.40 | 31.86 | 26.95          | 6.110 | (0.138)  |
| mid       | 7.15  | 9.16  | 16.56          | 5.238 | (0.195)  |
| high      | 2.93  | 7.09  | 20.74          | 3.588 | (0.195)  |
| boxes     |      |       |                |       |           |
| low       | 7.51  | 11.12 | 19.55          | 2.138 | (0.134)  |
| mid       | 3.36  | 0.87  | 16.97          | 2.138 | (0.134)  |
| high      | 1.83  | 2.97  | 20.91          | 1.287 | (0.134)  |
| posters   |      |       |                |       |           |
| low       | 8.12  | 10.39 | 16.01          | 2.138 | (0.134)  |
| mid       | 4.14  | 8.48  | 20.54          | 2.303 | (0.134)  |
| high      | 2.35  | 1.93  | 19.81          | 2.303 | (0.134)  |
| dynamic   |      |       |                |       |           |
| low       | 8.12  | 8.41  | 20.20          | 2.138 | (0.134)  |
| mid       | 3.80  | 3.96  | 22.94          | 2.060 | (0.134)  |
| high      | 1.30  | 3.91  | 21.83          | 2.060 | (0.134)  |

*The number in parentheses indicates the accuracy of zero-latency angular velocity that is calculated from (9).*

---

We test optical flow algorithms on four sequences which are preferred for validation in existing literature [9]–[11]: indoor_flying1, indoor_flying2, indoor_flying3, and outdoor_day1. Fig. 6 shows qualitative results of optical flow on MVSEC sequences. Grayscale images are shown for visualization purposes only and we do not use them to estimate optical flow. In ground-truth flow images, flow vectors with magnitude are displayed as a color wheel with values, whereas the results of the algorithms are colored by direction only to verify wrong results from a distance. Among them, although EV-FlowNet produces a dense flowmap, we mask its result with a binary image indicating whether there are any events in that pixel, in order to compare with the others clearly.

In the first and second rows, ABMOF produces erroneous optical flows in the upper right quadrant, where events are not triggered enough due to long distance and little apparent movement. This is because ABMOF computes optical flow with the full-size time slices. Likewise, EV-FlowNet fails to estimate accurate optical flow sometimes as shown in the second row. Quantitative results evaluated with image pairs one frame apart are shown in Table 1. Our performance on indoor sequences is comparable with the other algorithms but deteriorates in the outdoor sequence. This is because long straight lines belonging to crosswalks and lane boundaries cause the aperture problem in a local time slice of ours and ABMOF as shown in Fig. 4 (b). Also, whereas Zhu et al. [10], EV-FN, and Spike-FN output flowmap as a form of frame and the true flowmap is also provided as a frame synchronized with gray images for easy comparison, our method and ABMOF generate asynchronous frame-less optical flow. Hence, the compensated output of ours and ABMOF, which are linearly extrapolated to the timestamp of the true flowmap,
may degrade accuracies in this comparison.

To compute the latency of asynchronous optical flow stream, we utilize the result of back-end angular velocity estimation. The back-end process results in an additional latency of 0.02 (poster) or 0.2 (shapes) ms on average, but this value is added equally to the compared algorithms, so it does not affect the performance comparison and is not subtracted in Table 2. For the analysis of latency, we validate algorithms on DAVIS240C sequences, and Table 2 outlines the average latency and accuracy of the angular velocity estimation algorithm mentioned in Section II-C, which fetches the output of the compared optical flow algorithms. Average accuracy is calculated as the root mean square error. Since the latency of the event camera is affected by the speed of motion, we manually divide each sequence into three sections according to the degree of movement. The fourth row of each sequence represents an execution over the whole sequence. Our algorithm is superior to the other algorithms in terms of latency and accuracy. ABMOF shows slightly worse accuracy than ours and has an additional latency of up to 15 ms. Meanwhile, EV-FlowNet loses accuracy when the camera moves fast as mentioned in their paper. Summarizing Tables 1 and 2, the accuracy of our algorithm is comparable with the existing algorithms while significantly reducing the latency.

In comparing ours and EV-FlowNet, since they estimate angular velocity from optical flow in the same way, the accuracy of angular velocity is affected by the quality of optical flow and latency. In order to eliminate the influence of latency and compare them in terms of the accuracy of optical flow, we calculate the zero-latency angular velocity \( \hat{\omega}_{est}(\tau + \tau_d) \) from (9) by time-shifting with the latency value, \( \tau_d \), and display its accuracy within parentheses in Table 2. Because the zero-latency estimates of EV-FlowNet are still less accurate than ours, this analysis suggests that our optical flow is more accurate than EV-FlowNet on DAVIS240C sequences. As supported by the fact that the ground truth angular velocity with high temporal resolution evaluates our optical flow well, the error of our algorithm may have been lower than in Table 1 if the ground truth flowmap was provided with a high temporal resolution.

C. ANGULAR VELOCITY ESTIMATION

For performance validation of angular velocity estimation, we utilize DAVIS240C sequences that are captured under rotational motion, since MVSEC does not provide rotation sequences. Then, we analyze average latency and accuracy as explained in Section III-B. We compare ours to the event-based spiking neural network for angular velocity regression (eSNN) [16] and the implemented version of contrast maxi-
TABLE 3. Evaluations for latency and accuracy of angular velocity. Best results are in bold.

| Sequence | Average latency (ms) | Average accuracy (rad/s) |
|----------|----------------------|--------------------------|
|          | Ours | eSNN | CM | Ours | eSNN | CM |
| shapes   |      |      |    |      |      |    |
| low      | 4.10 | 4.22 | 4.24 | 0.76 | 0.76 | 0.76 |
| mid      | 5.35 | 5.46 | 5.46 | 1.03 | 1.03 | 1.03 |
| high     | 6.93 | 7.03 | 7.03 | 1.36 | 1.36 | 1.36 |
| whole    | 5.88 | 5.88 | 5.88 | 1.36 | 1.36 | 1.36 |
| boxes    |      |      |    |      |      |    |
| low      | 3.51 | 3.51 | 3.51 | 0.76 | 0.76 | 0.76 |
| mid      | 4.71 | 4.71 | 4.71 | 1.03 | 1.03 | 1.03 |
| high     | 6.26 | 6.26 | 6.26 | 1.36 | 1.36 | 1.36 |
| whole    | 5.28 | 5.28 | 5.28 | 1.36 | 1.36 | 1.36 |
| poster   |      |      |    |      |      |    |
| low      | 4.11 | 4.11 | 4.11 | 1.03 | 1.03 | 1.03 |
| mid      | 6.30 | 6.30 | 6.30 | 1.36 | 1.36 | 1.36 |
| high     | 7.71 | 7.71 | 7.71 | 1.36 | 1.36 | 1.36 |
| whole    | 5.49 | 5.49 | 5.49 | 1.36 | 1.36 | 1.36 |
| dynamic  |      |      |    |      |      |    |
| poster   |      |      |    |      |      |    |

Fig. 7 shows the angular velocity estimates of the compared algorithms. As can be seen from the time gap between dashed and solid lines, EV-FlowNet and eSNN show large latency than the others. For reference, we omit estimates during the settling time in the plot of eSNN. Since poster sequence has huge amounts of textures, CM also estimates angular velocity with a negligible latency like ours and ABMOF. In Table 3, our algorithm estimates angular velocity with consistently low latency and high accuracy for all sequences. Although CM shows better performance in terms of latency on boxes and poster sequences that are captured in front of rich textures, its latency is large on shapes and dynamic sequences: shapes and dynamic sequences were collected in front of texture-less simple shapes and natural office environment, respectively. In the zoomed-in plot (Fig. 1), our algorithm shows much lower latency and higher accuracy than the other algorithms in a low texture environment.

IV. DISCUSSION

A. ROBUSTNESS

For validating robustness against texture and motion speed which are observed in DAVIS240C sequences, we test four more algorithms in Table 4: filter-based angular velocity estimation using our asynchronous optical flow stream (KF), our angular velocity estimation using ABMOF with the number of events, 100, that is fewer than ABMOF and ours in Table 2 (ABMOF₂), and another two versions of CM framework (CM₅₅₀ms). CM₅₅₀ms fetches a less number of events, 5000, that is heuristically chosen to have similar latency to ours on shapes sequence. On the other hand, CM₅₅₀ms uses time windows of 50 ms to collect events. In the table, latency and errors lower than ours are shown in bold, and (−) denotes failure whose error increases monotonically.

For KF, we implement a robust random-walk-model-based Kalman filter with a 6-dimensional state including angular acceleration based on [27]. As mentioned in Section II-C, latency is larger than our least-squares-based algorithm, but showing lower accuracy due to our accurate optical flow. A Kalman filter with a 3-dimensional state including rotation
only has also similar results. In the case of ABMOF∗, it constructs time slices with a fewer number of events than ABMOF and ours, to decrease the latency of ABMOF tested previously. Consequently, its latency is reduced, but its error becomes large overall. Like between ABMOF and ABMOF∗, CM∗k loses accuracy on dynamic sequence, even though its latency is decreased. It is noteworthy that CM∗k fails on boxes and poster sequences that require more events due to the rich texture. Conversely, CM50ms shows consistent latency but its accuracy degrades on the fast section of each sequence. When we test CM5ms to further reduce latency, the accuracy becomes much worse. CM with a larger time window fails to minimize contrast due to edge bleeding. Also, as the number of events varies, the accuracy of CM depends highly on the texture level of a scene. On the contrary to the above algorithms that utilize the image of stacked events, our approach shows reliable performance in various environments without fine-tuning parameters depending on the scene.

B. FUTURE WORK

Our algorithm computes optical flows through the diamond search for efficiency, so the \( v_x \), \( v_y \) values of the optical flow vector \( v \) are integers in pixels. Because of the nature of this search method, the direction of an optical flow vector is discretized depending on the magnitude of the vector. For example, if the best matching block in the current local time slice is one pixel away from the previous local time slice, the angle of the vector has eight directions: (-1,-1), (0,-1), (1,-1), (-1,0), (1,0), (-1,1), (0,1), (1,1). Nevertheless, we can precisely derive the magnitude of optical flow in px/s units, thanks to the high temporal resolution, and obtain accurate angular velocity from the bunch of optical flow vectors. Further, an algorithm that computes an optimum optical flow vector with a sub-pixel resolution using quadratic interpolation between the adjacent matching scores is able to output optical flow vectors in a precise direction even in a small movement and improve the performance of the angular velocity estimation.

Further, to reduce the theoretical latency of optical flow estimation, we construct local time slices for all pixels while increasing memory requirement and computational cost. Compared to ABMOF, our approach incurs average 25 times of computation, 6.5Kev/s (kilo events per second) on an i7 laptop without GPU computing. However, since adjacent queues fetch the same event at the same time for each incoming event as shown in Fig. 3, parallel programming can enhance computation efficiency.

V. CONCLUSION

In the paper, we aim to decrease the theoretical latency which is one of the important characteristics of event cameras, leading to short response time and high accuracy. In particular, we estimate asynchronous optical flow stream, and 3D angular velocity with low-latency to compute the latency of optical flow quantitatively. Contrary to the previous works, our algorithm builds and maintains local time slices for every pixel in the form of a queue, thus generating optical flow that is independent of one another, like the event stream. Moreover, these highly informative optical flows can provide the exact analytic solution of angular velocity, thus satisfying low-latency. The overall evaluations suggest that our algorithm shows higher accuracy than the previous works while reducing latency significantly. Besides, the accuracy and latency of our algorithm are more consistent than other existing algorithms, regardless of the degree of texture and the speed of the camera. In particular, the latency has been reduced significantly in an environment with low texture. In summary, our algorithm produces an asynchronous data stream like a DVS camera, but the significance of the paper is that the container of the output is an optical flow having high-level information than an event. The proposed asynchronous optical flow stream can also perform the same role as optical flow in traditional computer vision problems such as object tracking, motion segmentation, and motion estimation. Thus, our event-based optical flow stream can be utilized to handle visual perception problems for agile robotic systems and estimate the motion of the system with very low latency, especially even in texture-less environments such as indoor corridors.

REFERENCES

[1] M. Mahowald, “Vlsi analogs of neuronal visual processing: a synthesis of form and function,” Ph.D. dissertation, California Institute of Technology Pasadena, 1992.

[2] P. Lichtsteiner, C. Posch, and T. Delbruck, “A 128x128 120dB 30mW asynchronous vision sensor that responds to relative intensity change,” 2006 IEEE International Solid State Circuits Conference-Digest of Technical Papers, pp. 2060–2069, 2006.

[3] P. Lichtsteiner, C. Posch, and T. Delbruck, “A 128x128 120dB 15μs latency asynchronous temporal contrast vision sensor,” IEEE journal of solid-state circuits, vol. 43, no. 2, pp. 566–576, 2008.

[4] E. Mueggler, C. Bartolozzi, and D. Scaramuzza, “Fast event-based corner detection,” BMVC, 2017.

[5] R. Li, D. Shi, Y. Zhang, K. Li, and R. Li, “Fa-harris: A fast and asynchronous corner detector for event cameras,” in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019, pp. 6223–6229.

[6] I. Alzugaray and M. Chli, “Asynchronous corner detection and tracking for event cameras in real time,” IEEE Robotics and Automation Letters, vol. 3, no. 4, pp. 3177–3184, 2018.

[7] A. Z. Zhu, N. Atanasov, and K. Daniilidis, “Event-based feature tracking with probabilistic data association,” IEEE International Conference on Robotics and Automation (ICRA), pp. 4465–4470, 2017.

[8] M. Liu and T. Delbruck, “Adaptive time-slice block-matching optical flow algorithm for dynamic vision sensors,” BMVC, 2018.

[9] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, “Ev-flownet: Self-supervised optical flow estimation for event-based cameras,” arXiv preprint arXiv:1802.06898, 2018.

[10] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, “Unsupervised event-based learning of optical flow, depth, and egomotion,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 989–997, 2019.

[11] C. Lee, A. K. Kosta, A. Z. Zhu, K. Chaney, K. Daniilidis, and K. Roy, “Spike-flownet: event-based optical flow estimation with energy-efficient hybrid neural networks,” in European Conference on Computer Vision. Springer, 2020, pp. 366–382.

[12] H. Kim, S. Leutenegger, and A. J. Davison, “Real-time 3d reconstruction and 6-dof tracking with an event camera,” European Conference on Computer Vision, pp. 349–364, 2016.
[13] G. Gallego, H. Rebecq, and D. Scaramuzza, “A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3867–3876.

[14] H. Kim and H. J. Kim, “Real-time rotational motion estimation with contrast maximization over globally aligned events,” IEEE Robotics and Automation Letters, 2021.

[15] H. Kim, A. Handa, R. Benosman, S.-H. Ieng, and A. J. Davison, “Simultaneous mosaicing and tracking with an event camera,” J. Solid State Circ, vol. 43, pp. 566–576, 2008.

[16] M. Gehrig, S. B. Shrestha, D. Mouritzen, and D. Scaramuzza, “Event-based angular velocity regression with spiking networks,” IEEE International Conference on Robotics and Automation (ICRA), 2020.

[17] P. Bardow, A. J. Davison, and S. Leutenegger, “Simultaneous optical flow and intensity estimation from an event camera,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 884–892, 2016.

[18] E. Mueggler, C. Forster, N. Baumli, G. Gallego, and D. Scaramuzza, “Lifetime estimation of events from dynamic vision sensors,” 2015 IEEE international conference on Robotics and Automation (ICRA), pp. 4874–4881, 2015.

[19] J. Li, F. Shi, W.-H. Liu, D. Zou, Q. Wang, P. K. Park, and H. Ryu, “Adaptive temporal pooling for object detection using dynamic vision sensor.” BMVC, 2017.

[20] E. H. Adelson and J. R. Bergen, “Spatiotemporal energy models for the perception of motion,” Josa a, vol. 2, no. 2, pp. 284–299, 1985.

[21] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi, “Event-based visual flow,” IEEE transactions on neural networks and learning systems, vol. 25, no. 2, pp. 407–417, 2013.

[22] F. Paredes-Vallés, K. Y. Scheper, and G. C. de Croon, “Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: From events to global motion perception,” IEEE transactions on pattern analysis and machine intelligence, vol. 42, no. 8, pp. 2051–2064, 2019.

[23] G. Gallego and D. Scaramuzza, “Accurate angular velocity estimation with an event camera,” IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 632–639, 2017.

[24] S. Zhu and K.-K. Ma, “A new diamond search algorithm for fast block-matching motion estimation,” IEEE transactions on Image Processing, vol. 9, no. 2, pp. 287–290, 2000.

[25] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, “A 240 × 180 130 db 3 μs latency global shutter spatiotemporal vision sensor,” IEEE Journal of Solid-State Circuits, vol. 49, no. 10, pp. 2333–2341, 2014.

[26] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, “The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam,” The International Journal of Robotics Research, vol. 36, no. 2, pp. 142–149, 2017.

[27] G. Agamennoni, J. I. Nieto, and E. M. Nebot, “An outlier-robust kalman filter,” in 2011 IEEE International Conference on Robotics and Automation. IEEE, pp. 1551–1558.