Recursive Contrast Maximization for Event-Based High-Frequency Motion Estimation

TAKEHIRO OZAWA\textsuperscript{1}, YUSUKE SEKIKAWA\textsuperscript{2}, AND HIDEO SAIITO\textsuperscript{1}, (Senior Member, IEEE)
\textsuperscript{1}Graduate School of Science and Technology, Keio University, Yokohama 223-8522, Japan
\textsuperscript{2}Denso IT Laboratory, Inc., Tokyo 150-0002, Japan
Corresponding author: Takehiro Ozawa (takehiro.o@keio.jp)

ABSTRACT Achieving high-frequency motion estimation with a fast-moving camera is an important task in the field of computer vision. Contrast Maximization (CMax), a method of motion estimation using an event camera, is the de-facto standard. However, CMax requires the processing of a large number of events at a single time, a computationally expensive task. That makes it difficult to perform high-frequency estimates. Specifically, past events that have already been used once for estimation need to be evaluated again. In this paper, we propose “Recursive Contrast Maximization (R-CMax)” to estimate motions at high frequencies. The proposed method approximates multiple events by two “compressed events” using estimated trajectories of events from the previous time step, which can be updated recursively. By using a small number of “compressed events,” motion estimation can be updated efficiently. Comparing R-CMax with CMax and its extensions, we experimentally show that R-CMax can perform motion estimation with a fraction of the computational complexity while maintaining comparable accuracy.

INDEX TERMS Contrast maximization, event-based camera, motion estimation.

I. INTRODUCTION

A. BACKGROUND
High-frequency motion estimation is necessary for the robust tracking of fast-moving camera poses; for example, in augmented reality and automated driving, the poses of a fast-moving camera must be robustly tracked. Since a fast-moving camera changes its poses in a short period of time, robust pose tracking requires high-frequency motion estimation. However, the limited frame rate of conventional cameras restricts high-frequency motion estimation.

Event-based cameras are suitable sensors for robust camera pose tracking of fast-moving cameras \cite{1}, \cite{2}, \cite{3}. They asynchronously and sparsely detect per-pixel intensity changes, achieving an extremely high temporal resolution equivalent to \(> 10,000\) fps. Taking advantage of these features, which cannot be obtained with conventional cameras, various applications are currently being studied, including optical flow estimation \cite{4}, \cite{5}, \cite{6}, \cite{7}, \cite{8}, depth estimation \cite{9}, \cite{10}, \cite{11}, \cite{12}, and object tracking \cite{13}, \cite{14}, \cite{15}. In particular, motion estimation using event cameras is one of the most active research fields.

One of the de-facto standard methods for motion estimation with event cameras is called Contrast Maximization (CMax), which was proposed by Gallego et al. \cite{16}, \cite{17}. CMax creates an Image of Warped Events (IWE) by warping batches of events according to candidate model parameters and accumulating them on an image plane. Motion can then be estimated by maximizing the contrast of IWE — when events are warped by the correct parameters, they are aligned along the edges, producing a high-contrast IWE.

B. MOTIVATION
The problem is that high-frequency motion estimation by CMax is difficult due to computational complexity. As shown in Fig. 1, many events overlap in the sliding window when CMax is used for high-frequency motion estimation. Despite these many overlapping events, CMax needs to recalculate IWE again. Since creating IWE needs to compute tens of thousands of events in a short time, computational complexity is further increased.
C. CONTRIBUTIONS

To solve this problem, we realized a recursive maximization of contrast called “Recursive Contrast Maximization (R-CMax).” R-CMax realized by the proposed method enables high-frequency motion estimation with reduced computational complexity through recursive updates. To realize a recursive update, we propose summarizing a large number of past events using small numbers of “compressed events.” As shown in Fig. 1, hundreds of spatiotemporal event trajectories are approximated using only two events. This approximation enables recursive maximization of contrast using only the compressed events and newly observed events, thereby reducing computational complexity.

To demonstrate the effectiveness of the proposed method in terms of accuracy and computational complexity, we compared it to CMax and its extensions on both simulated and real data. We demonstrated that R-CMax could perform motion estimation with only a fraction of the computational complexity of CMax while maintaining comparable accuracy.

In summary, our contributions are:

- Proposal of R-CMax, which enables high-frequency motion estimation with reduced computational complexity through recursive updates (Section III).
- Experiments comparing R-CMax with CMax and its extensions to demonstrate its effectiveness (Section IV).

II. RELATED WORKS

This section presents research on motion estimation using event cameras. The high temporal resolution and low latency of event cameras, which are not available with conventional cameras, make them suitable for motion estimation in a variety of situations that were previously impossible.

Many motion estimation researches have been conducted by combining event cameras with external sensors. Censi and Scaramuzza [18] automatically calibrate the event camera and the conventional camera in spatiotemporal terms, and each time an event is observed, update the relative pose of the event camera with respect to the last frame. Weikerdorfer et al. [19] use a frame-based RGB-D camera attached to an event camera to perform depth estimation and create voxel maps for motion estimation. Kueng et al. [20] use intensity images from conventional cameras to detect feature points and track them asynchronously, using events to perform motion estimation. Gallego et al. [21] perform event-by-event six degrees of freedom (DoF) motion estimation from existing depth maps with intensity values. These methods allow high-frequency motion estimation for each event observed, but they require information from external sensors as well as event data.

Motion estimation methods performed only with event cameras have also been studied. Weikersdorfer et al. [22], [23] use an upward-mounted event camera to capture events from the ceiling and use a particle filter to estimate the position of the camera moving parallel to the ceiling. Kim et al. [24] proposed a method for jointly estimating the camera’s six degrees of freedom, depth to keyframes, and scene intensity gradients using a probabilistic filter. Rebecq et al. [25] perform motion estimation by tracking based on image-model alignment and 3D reconstruction using events in parallel. More recently, there have been successful methods for optical flow, ego-motion, and depth estimation by accumulating observed events into images and using them to train deep learning methods [26], [27], [28], [29]. Although these methods allow motion estimation...
using only event data, they cannot take advantage of the low latency of event cameras, as they cannot perform event-by-event, high-frequency motion estimation for each observed event.

CMax by Gallego et al. [16], which is most related to our method, is the de-facto standard method for motion estimation using only event cameras. One of the characteristics of CMax is that it is a unified method that can estimate various motions using the same framework, such as translation and rotation, given that the motion to be estimated is modeled. In addition, Gallego [17] and Stoffregan [30] have also proposed and validated various loss functions other than contrast. Nunes and Demiris proposed a method based on CMax called Entropy Minimization (EMin) [31]. This method allows motion estimation directly from warped events by introducing an entropy-based measure, whereas CMax is a method that requires events to be converted into an image-based, intermediate representation called IWE. Nunes and Demiris also proposed Dispersion Minimization (DMin) [32] as a further extension of EMin. In addition to the advantage of not having to rely on intermediate representations, this method allows for incremental estimation of model parameters. However, since recalculating past events is inevitable, this method has yet to reduce the computational complexity of high-frequency motion estimation.

III. METHODS
A. PRELIMINARY
1) EVENT DATA REPRESENTATION
The data obtained from the event camera will be described in this section of the paper. An event camera asynchronously outputs the intensity change of each pixel, called an event. Suppose there exists a given set of events \( E = \{ e_k \} \) in the time window \([0, T]\), where \( e_k = (x_k, t_k, p_k) \). Here, \( x_k \) is the coordinate where the event occurs, \( t_k \) is the timestamp where the event occurs with a precision of microseconds, and \( p_k \in \{-1, +1\} \) is the polarity of the event, which indicates brighter or darker changes when the logarithmic intensity changes above a threshold at a certain pixel \( x = (x, y) \).

2) CONTRAST MAXIMIZATION
Since CMax is the basis for our proposed method, we briefly describe the CMax method below. Let \( E = \{ e_k \}_{k=1}^{N_e} \) be a group of accumulated observed events, where \( N_e \) is the number of events accumulated. Let \( \theta \) be the motion we want to estimate. We can define a function \( W \) that warps the events along their spatiotemporal trajectory as follows:

\[
e_{k} \rightarrow e'_{k} = (x'_{k}, t_{ref}, p_{k}),
\]

\[
x'_{k} = W(x_{k}, t_{k}; \theta).
\]

In CMax, an intermediate representation called IWE is introduced that accumulates the events warped by (2) onto the image plane, defined as follows:

\[
I(x; \theta) = \sum_{k=1}^{N_e} N(x - x'_{k}(\theta)),
\]

where \( N \) is Gaussian. Creating IWE with the optimal parameter \( \theta^{*} \), its contrast is locally maximal. The contrast of IWE can be calculated by taking the variance of IWE, which is:

\[
f(\theta) = \sigma^{2} (I(x; \theta)) = \frac{1}{N_{p}} \sum_{m,n} (i_{mn} - \mu_{I})^{2},
\]

where \( N_{p} \) is the number of pixels in IWE, \( I = (i_{mn}) \) is the coordinate of IWE, and \( \mu_{I} = \frac{1}{N_{p}} \sum_{m,n} i_{mn} \) is the average value of IWE. Therefore, \( \theta^{*} \) is obtained by the following optimization:

\[
\theta^{*} = \arg \max_{\theta} f(\theta).
\]
motion estimation with CMax; despite many overlapping events in the sliding window, CMax needs to recalculate IWE. Therefore, avoiding the recalculation of past events reduces computational complexity.

B. PROPOSED METHOD

To solve this problem, we realized recursive maximization of contrast, called “Recursive Contrast Maximization” (R-CMax). The flow of our method is shown in Fig. 2 and Algorithm 1. To realize a recursive update, we propose “compressed events,” which approximate past events using previous motion estimates (Section III-B1, III-B2). Hundreds of spatiotemporal events can be represented using only two compressed events. With these compressed events, IWE can be created using the compressed and observed events without calculating past events. Additionally, our method updates compressed events by maintaining a two-dimensional event map updated based on the estimation results (Section III-B3).

Note that the motion to be estimated in this paper is a two-dimensional motion that moves parallel to a plane. Therefore, the parameter \( \theta = [v_x, v_y]^T \) is common to all pixels in the event camera, and the motion model \( W \) is expressed as follows:

\[
W(x_k, t_k; \theta) = x_k - \Delta t_k \theta, \tag{6}
\]

where \( \Delta t_k = t_k - t_{\text{ref}} \). The extensibility of this method is discussed in Section V.

1) IWE CREATION

Let \( E_{\text{new}} = \{e_{p,n}\}_{p=1}^{N_{\text{new}}} \) be the newly observed events and \( E_{\text{old}} = \{e_{q,o}\}_{q=1}^{N_{\text{old}}} \) be the past events that have already been used in the calculation of IWE; then, (3) can be decomposed as follows:

\[
I(x; \theta) = \sum_{p=1}^{N_{\text{new}}} N \left( x - x'_{p,n}(\theta) \right) + \sum_{q=1}^{N_{\text{old}}} N \left( x - x'_{q,o}(\theta) \right). \tag{7}
\]

The recalculation for \( E_{\text{old}} \) is the cause of the increase in the number of operations. Therefore, we propose to approximate (7) by using the compressed events \( E_c = \{e_{r,c}\}_{r=1}^{N_c} \), which is created based on the past events and the previous estimation results:

\[
I(x; \theta) \approx \sum_{p=1}^{N_{\text{new}}} \sum_{r=1}^{N_c} w_r N \left( x - x'_{p,n}(\theta) \right) + \sum_{r=1}^{N_c} w_r N \left( x - \hat{x}'_{r,c}(\theta) \right). \tag{8}
\]
where \( w_r \) denotes the weight of each compressed event, \( x_{r,c}' \) and \( \hat{x}_{r,c}' \) denote the coordinates of the compressed events that approximate the coordinates of past events. As shown in (7) and (8), we use two compressed events to approximate past events. Since the number of compressed events is less than past events (\( N_c < N_{old} \)), introducing compressed events can significantly reduce computational complexity. By using (8), IWE can be created using only compressed events and new events. Compressed events can be reused to create the next IWE used for the following estimation, allowing for recursive maximization of contrast.

2) COMPRESSED EVENT
This section describes compressed events used for recursive updates. The concept of compressed events is shown in Fig. 3. Performing CMax can be interpreted as associating events from the same edge. Therefore, performing CMax integrates multiple events from the same edge into a single event. For events occurring from the same edge \( r \), a compressed event is created as follows:

\[
e_{r,c} = (x_{r,c}', t_{ref}, w_r)
\]

where \( x_{r,c}' \) represents the coordinates of the edge on the IWE, \( w_r = \sum_{i=1}^{N_r} p_i \) is the weight that the compressed event has, and \( N_r \) is the number of events on \( r \). In other words, a compressed event is an event with a weight equal to the number of events on the same edge.

The compressed events we introduced allow us to perform recursive optimization, which does not recalculate past events. We assume that motion can be approximated as constant velocity linear motion for a short time and that the estimation results do not change significantly before and after observing a new event. Under this assumption, using compressed events, the coordinates of past events can be approximated as follows:

\[
\hat{x}_{r,c} \approx W^{-1}(x_{r,c}; \hat{\theta}^*)
\]

where \( x_{r,c} \) denotes the coordinates of the compressed event, \( \hat{\theta}^* \) is the \( \theta^* \) from the previous estimation, and \( W^{-1} \) is the inverse transform of modeled motion (Fig. 3).

Creating the IWE is to evaluate the overlap of warped events. To reduce the amount of computation, the formula for IWE shown in (3) practically computes the Gaussian only for the pixels surrounding the warped event, not all of the IWE pixels. However, as shown in Fig. 1, the two compressed events are so far apart that differentiation using a truncated Gaussian does not lead to the true value. On the other hand, increasing the Gaussian size increases computational
FIGURE 6. Graph comparing the motion estimation results of R-CMax with CMax and its extensions for a dataset created with ESIM. GT represents the ground truth of motion estimation obtained from ESIM. R-CMax achieved accuracy comparable to CMax and its extension while reducing computational complexity.

TABLE 1. Experimental results comparing the accuracy of R-CMax with CMax and its extensions, where $e_v$ denotes the error against the true value, $\sigma_{e_v}$ denotes the variance of the error, and RMSE denotes the Root Mean Squared Error. Each value is the average of the estimation results of each dataset.

|          | $e_v$ | $\sigma_{e_v}$ | $N_e$ |
|----------|-------|----------------|-------|
| R-CMax   | 8.90  | 5.59           | 2631.9|
| CMax     | 7.24  | 5.96           | 50000 |
| EMin     | 14.30 | 12.51          | 50000 |
| DMin     | 21.92 | 17.35          | 50000 |

complexity. To mitigate that, we take advantage of the fact that the compressed event has already been associated. We do not need to compute all possible combinations of events, and the overlap of events can be evaluated analytically using the previous estimation results. Therefore, the computational complexity of IWE creation with our method is independent of the kernel size of the Gaussian. Since the number of target events is reduced and the computational complexity of IWE creation is comparable, the overall amount of calculation is significantly reduced.

3) EVENT MAPPING
As the event camera moves, new compressed events must be added when new objects are captured and must be deleted when they are no longer required. To update compressed events, we maintain a two-dimensional event map $\mathcal{M}$ simultaneously with the motion estimation.

Given the previous motion estimation results, the observed events can be warped to the map coordinates as follows:

$$\begin{align*}
M_m(x) &= I_1(x) \\
M_n(x) &= \mu(M_{n-1}(x) + I_n(x - s_n)),
\end{align*}$$

FIGURE 7. Setup of the experiment using an actual event camera. The event camera captured the floor vertically, while the iPhone captured the front.
where $I_n$ is the IWE made by new events at the $n$-th estimation and $s_n = \sum_{i}^{\theta}$ is the camera’s travel distance calculated from previous estimation results. The constant $\mu$ is determined by the following equation using the number of events at initialization $N_0$ and the number of new events $N_c$:

$$
\mu = \frac{N_0 - N_c}{N_0}.
$$

The map is updated so that recently used compressed events have larger weights and compressed events that have not been observed for a while disappear. This is to ensure that compressed events with newer observation times have a larger impact on estimation and that compressed events that have not been observed for a while have a smaller impact on estimation. To achieve this, we update a constant $\mu < 1$ and apply it to the 2-D map for each motion estimation. $M$ can be updated recursively by using (11) and (12).

The compressed events can be updated as follows:

$$
r_n = x, \ w_n = M_n(x) \quad \text{where} \quad M_n(x) > \sigma.
$$

To avoid noise from being included in the compressed events, a threshold $\sigma$ is prepared, and pixels on the map with $M(x) > \sigma$ can be used as compressed events.

IV. EXPERIMENTAL EVALUATION

In this section, we present the experiments and discussions conducted to demonstrate the effectiveness of our method. The novelty of our method is that it allows us to recursively maximize the contrast of IWE and reduce computational complexity by approximating past events.

We conducted the following three experiments on simulated and real data to demonstrate that our method reduces computational complexity and that comparable in accuracy to existing methods. In our first experiment, we applied R-CMax to motion estimation on simulation data created with ESIM [33]. The experimental results were compared by applying the same data to the original CMax, Entropy Minimization (EMin) [31], and Dispersion Minimization (DMin, on-line mode) [32]. In the second experiment, QR codes were captured by an actual event camera. We estimated the event camera’s motion to show that this method can be applied to real event camera data containing noise. Experimental results were compared to DMin and ARKit4 on an iPhone 13 Pro, which uses images, depth sensors, and IMU. In addition, the created map was visualized and compared with the original QR code to show that it can be applied to object restoration. In the third experiment, we performed motion estimation for a fast-moving camera to demonstrate the effectiveness of high-frequency estimation with event cameras. The experimental results of the R-CMax were compared to the original CMax to evaluate its accuracy.

All experiments were performed on an Intel Core i7-6800K CPU and 64 GB of system memory, using Python for the implementation and matplotlib [34] for visualization.

A. EXPERIMENTS WITH SYNTHETIC EVENT DATA

This section describes experiments showing that the proposed method reduces computational complexity, and its estimation accuracy is comparable to existing methods. To compare R-CMax with existing methods, we created a dataset using ESIM, which can simulate an event camera. We created ten event data sets by vertically placing a virtual event camera toward the image shown in Fig. 4 and moving it randomly. We applied our method and comparative methods to the data to estimate the motion of the event camera. For comparison,
the same data was given to CMax, EMin, and DMin (on-line mode) to estimate motion. Each method used a batch size of 50,000 events, and motion estimation was performed every time an event was acquired. The parameters to be estimated were two-dimensional optical flows \((v_x, v_y)\) moving parallel to the plane.

Fig. 5 shows the number of events required for R-CMax and CMax to be estimated at each time. As seen from the figure, the proposed method succeeded in reducing the number of events that need to be computed when performing one estimation compared to CMax. The figure shows that CMax always processed a constant number of events for each motion estimation. This is because CMax accumulates a batch of events of a defined size for each new event acquired, creates an IWE, and calculates the maximum contrast. This results in having to constantly calculate a large number of events in a short period. On the other hand, the proposed method could perform motion estimation with less computation compared to existing methods. This is because the compressed event approximates past events once calculated. As a result, only compressed and newly observed events are needed for a single estimation, reducing the total amount of computation.

The results of the accuracy comparison experiment between R-CMax, CMax, and its extensions are shown in Table 1. The accuracy comparisons are made by taking the average of three metrics for the results of ten data sets, errors in the true value, error variance, and the root mean squared error (RMSE). As seen from the table, the proposed method can estimate motion with an accuracy comparable to the existing methods. As seen in Fig. 5, even though the proposed method performed motion estimation with fewer events than the existing method, it still maintained its accuracy.

A plot of one of the results of the accuracy comparison experiment is shown in Fig. 6. As seen from the figure, even for data involving changes in velocity, estimation by the proposed method was comparable to the existing method.

Our method assumes that the velocity changes little before and after the observation of an event. Still, even data regarding such a change in velocity can be regarded as constant velocity for a very short time, such as before and after the observation of an event. Thus, the estimation can be performed without failure.

B. EXPERIMENTS WITH REAL EVENT DATA

This section describes the quantitative experiments that were conducted to demonstrate the effectiveness of this method, even with an actual event camera, thus showcasing its applicability to real-world applications. In Section IV-A, we experimented with simulated event data, where the event data occurred only at pixels with intensity changes. Therefore, by conducting experiments with actual event cameras,
we show that the motion estimation of this method is possible even in more practical situations that feature noise. As shown in Fig. 7, the event camera was placed 20 cm from the floor, facing directly down to the ground, and motion estimation was performed when the camera moved in parallel over the QR code. The event camera used for this experiment was a Prophesee Gen3 VGA (and its supporting evaluation kit, EVK). As a comparison, the same data set was given to DMin (on-line mode) for motion estimation. Each method was used with a batch size of 50,000 events, and motion estimation was performed for each event acquisition. As a reference value, we performed motion estimation using ARKit4 (iOS) with an iPhone 13 Pro Max mounted on a dolly with the front facing forward.

Fig. 8 shows this method’s estimation results along with the results of DMin and ARKit4. As can be seen from the figure, the proposed method could estimate motion even for actual event data. DMin could also estimate motion, but the proposed method was more effective because it reduced the number of operations. It could also be confirmed that the proposed method was capable of similar motion estimation compared to ARKit4, which used an RGB camera and LIDAR. Fig. 9 shows the accumulated events before and after applying R-CMax. The alignment of the events qualitatively indicates that contrast was correctly maximized.

Fig. 10 compares the estimated trajectory of the dolly from the estimated motion with DMin and ARKit4. R-CMax was able to estimate the trajectory with the same accuracy as ARKit4, which used cameras and depth sensors, indicating that this method can also be applied to visual odometry.

A visualization of the map $\mathcal{M}$ that was stored during the estimation process is shown in Fig. 11. The result shows that the edges of the QR code had been restored and that the noise generated from the event camera had been eliminated from the map. This result also shows that the motion estimation by the proposed method is correct for actual event data.

Although the actual event cameras used in these experiments contain noise in the event data that are not present in the simulation data, these experimental results show that motion estimation can be performed even in the presence of noise.
of noise. This is because the method of maintaining the map reduces the weight of randomly occurring events such as noise. Reducing unnecessary compressed events caused by noise makes motion estimation more robust. The experimental results demonstrate the potential for applications with motion estimation moving parallel to a plane using an event camera.

C. EXPERIMENTS WITH FAST-MOVING EVENT CAMERA

This chapter describes experiments conducted to demonstrate the effectiveness of R-CMax in tracking fast-moving cameras. High-frequency motion estimation is effective in tracking fast-moving cameras. Therefore, we compared the motion estimation results of fast-moving event cameras between R-CMax and CMax. In the experiment, we acquired data by moving the event camera at high speed by hand at 20 cm from the plane, and we performed motion estimation with R-CMax and CMax. The event camera used for this experiment was a Prophesyee Gen3 VGA (and its supporting evaluation kit, EVK). Each method performed estimation for each event acquisition.

Fig. 12 shows the proposed method’s estimation results and CMax’s results. The figure shows that the proposed method could estimate motion with the same accuracy as CMax, even for a fast-moving event camera. Also, as the figure shows, the maximum speed of the event camera in this experiment was about 6,000 pix/sec. If we assume that the frame rate of a conventional camera is 30 fps, the camera moved 200 pixels per frame, which would be difficult to track with a conventional camera. Thus, high-frequency motion estimation with event cameras using R-CMax is effective for tracking fast-moving cameras.

Note that when the event camera moves at high speed, events can be generated from pixels that should not be generated. In the case of the above experiment, some estimations failed if there was too much noise of that kind. We believe this problem can be solved by appropriately designing noise filters and removing those noises before processing.

V. CONCLUSION

In this paper, we proposed R-CMax for high-frequency, two-dimensional motion estimation by recursively maximizing contrast for each event acquisition. CMax, used for motion estimation, has a problem: the number of operations required increases when trying to perform high-frequency motion estimation. This is because high-frequency motion estimation via CMax requires the recalculation of past events already used for estimation. The proposed method approximates multiple events by “compressed events” using estimated trajectories of events from the previous time step. By creating an IWE with only compressed events and new events, we can maximize contrast while preventing event recalculation. This allows motion estimation to be performed each time an event is observed, allowing for the outputs of estimation results at a high frequency with reduced computational complexity. To demonstrate the effectiveness of the proposed method, we showed that in multiple sequences created using ESIM, the proposed method performed motion estimation without loss of accuracy and reduced the amount of computation required for a single estimation compared to existing methods. Motion estimation experiments using an actual event camera were also conducted. It was shown that the proposed method could robustly estimate motion, even with an actual event camera that generates random events such as noise. We also demonstrated that the high-frequency motion estimation by the proposed method is effective even when the event camera moves at high speed.

One limitation of our method is that the motion to be estimated is limited to two-dimensional motion moving parallel to a plane. One possible future direction would be extending motion to more complicated, such as rotation, homography, etc. However, if we extend this method to rotations and homography, the event trajectories are no longer spatiotemporally linear, even though the motion is linear in parameter space. In the present method, the approximation is based on the assumption that the event trajectories are spatiotemporally linear, so the current approximation method cannot be applied directly. Therefore, to realize modeling higher degree motion, we need to formulate spatiotemporal trajectories of events as a non-linear function of time.

REFERENCES

[1] P. Lichtsteiner, C. Posch, and T. Delbruck, “A 128 × 128 120 dB 15 µs latency asynchronous temporal contrast vision sensor,” *IEEE J. Solid State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008.
[2] 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 J. Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, Oct. 2014.
[3] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, “Event-based vision: A survey,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 154–180, Jan. 2022.
[4] R. Benosman, S.-H. Ieng, C. Clercq, C. Bartolozzi, and M. Srinivasan, “Asynchronous frameless event-based optical flow,” *Neural Netw.*, vol. 27, pp. 32–37, Mar. 2012.
[5] T. Brosch, S. Tschechne, and H. Neumann, “On event-based optical flow detection,” *Frontiers Neurosci.*, vol. 9, p. 137, Apr. 2015.
[6] B. Rueckauer and T. Delbruck, “Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor,” *Frontiers Neurosci.*, vol. 10, p. 176, Sep. 2016.
[7] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi, “Event-based visual flow,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 2, pp. 407–417, Feb. 2014.
[8] M. M. Almatrafi, R. Baldwin, K. Aizawa, and K. Hirakawa, “Distance surface for event-based optical flow,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 7, pp. 1547–1556, Jul. 2020.
[9] D. Gehrig, M. Ruegg, M. Gehrig, J. Hidalgo-Carrió, and D. Scaramuzza, “Combining events and frames using recurrent asynchronous multimodal networks for monocular depth prediction,” *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 2822–2829, Apr. 2021.
[10] A. Z. Zhu, Y. Chen, and K. Daniilidis, “Realtime time synchronized event-based stereo,” in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 433–447.
[11] Y. Zhou, G. Gallego, H. Rebecq, L. Kneip, H. Li, and D. Scaramuzza, “Semi-dense 3D reconstruction with a stereo event camera,” in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 235–251.
[12] G. Haessig, X. Berthelon, S.-H. Ieng, and R. Benosman, “A spiking neural network model of depth from defocus for event-based neuromorphic vision,” *Sci. Rep.*, vol. 9, no. 1, p. 3744, Mar. 2019.
[13] A. Mitrokhin, C. Fermüller, C. Parameshwara, and Y. Aloimonos, “Event-based moving object detection and tracking,” in Proc. IEEE/RSJ Int. Conf. Intel. Robots Syst. (IROS), Oct. 2018, pp. 1–9.

[14] J. P. Rodríguez-Gómez, A. G. Eguíluz, J. R. Martínez-de Dios, and A. Ollero, “Asynchronous event-based clustering and tracking for intrusion monitoring in UAVs,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2020, pp. 8518–8524.

[15] T. Delbrück and M. Lang, “Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor,” Front. Neurosci., vol. 7, p. 223, Nov. 2013.

[16] 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 Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 3867–3876.

[17] G. Gallego, M. Gehrig, and D. Scaramuzza, “Focus is all you need: Loss functions for event-based vision,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 12280–12289.

[18] A. Censi and D. Scaramuzza, “Low-latency event-based visual odometry,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2014, pp. 703–710.

[19] D. Weikersdorfer, D. B. Adrian, D. Cremers, and J. Conradt, “Event-based 3D SLAM with a depth-augmented dynamic vision sensor,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2014, pp. 359–364.

[20] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza, “Low-latency visual odometry using event-based feature tracks,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Oct. 2016, pp. 16–23.

[21] G. Gallego, J. E. A. Lund, E. Mueggler, H. Rebecq, T. Delbrück, and D. Scaramuzza, “Event-based, 6-DOF camera tracking from photometric depth maps,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 10, pp. 2402–2412, Oct. 2018.

[22] D. Weikersdorfer and J. Conradt, “Event-based particle filtering for robot self-localization,” in Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO), Dec. 2012, pp. 866–870.

[23] D. Weikersdorfer, R. Hoffmann, and J. Conradt, “Simultaneous localization and mapping for event-based vision systems,” in Proc. Int. Conf. Comput. Vis. Syst. Berlin, Germany: Springer, 2013, pp. 133–142.

[24] H. Kim, S. Leutenegger, and A. J. Davison, “Real-time 3D reconstruction and 6-DOF tracking with an event camera,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2016, pp. 349–364.

[25] H. Rebecq, T. Horstschäfer, G. Gallego, and D. Scaramuzza, “EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real time,” IEEE Robot. Autom. Lett., vol. 2, no. 2, pp. 593–600, Apr. 2017.

[26] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, “Unsupervised event-based learning of optical flow, depth, and egomotion,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 989–997.

[27] A. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, “EV-FlowNet: Self-supervised optical flow estimation for event-based cameras,” in Proc. 14th Robot., Sci. Syst., Jun. 2018, pp. 1–9.

[28] 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 Proc. Eur. Conf. Comput. Vis. (ECCV), 2020, pp. 366–382.

[29] F. Paredes-Vallés, K. Y. W. Scheper, and G. C. H. E. D. Croon, “Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: From events to global motion perception,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 8, pp. 2051–2064, Aug. 2020.

[30] T. Stoffregen and L. Kleeman, “Event cameras, contrast maximization and reward functions: An analysis,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 12300–12308.

[31] U. M. Nunes and Y. Demiris, “Entropy minimisation framework for event-based vision model estimation,” in Proc. Eur. Conf. Comput. Vis. (ECCV), Cham, Switzerland: Springer, 2020, pp. 161–176.

[32] U. M. Nunes and Y. Demiris, “Robust event-based vision model estimation by dispersion minimisation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 12, pp. 9561–9573, Dec. 2022.

[33] H. Rebecq, D. Gehrig, and D. Scaramuzza, “ESIM: An open event camera simulator,” in Proc. Conf. Robot. Learn. (CoRL), vol. 87, 2018, pp. 969–982.

[34] J. D. Hunter, “Matplotlib: A 2D graphics environment,” Comput. Sci. Eng., vol. 9, no. 3, pp. 90–95, May 2007.