2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Event-based Real-time Moving Object Detection Based On IMU Ego-motion Compensation

Chunhui Zhao, Yakun Li, and Yang Lyu*.

***Abstract*— Accurate and timely onboard perception is a prerequisite for mobile robots to operate in highly dynamic scenarios. The bio-inspired event camera can capture more motion details than a traditional camera by triggering each pixel asynchronously and therefore is more suitable in such scenarios. Among various perception tasks based on the event camera, ego-motion removal is one fundamental procedure to reduce perception ambiguities. Recent ego-motion removal methods are mainly based on optimization processes and may be computationally expensive for robot applications. In this paper, we consider the challenging perception task of detecting fast-moving objects from an aggressively operated platform equipped with an event camera, achieving computational cost reduction by directly employing IMU motion measurement. First, we design a nonlinear warping function to capture rotation information from an IMU and to compensate for the camera motion during an asynchronous events stream. The proposed nonlinear warping function improves the compensation accuracy by 10%-15%. Afterward, we segmented the moving parts on the warped image through dynamic threshold segmentation and optical flow calculation, and clustering. Finally, we validate the proposed detection pipeline on public datasets and real-world data streams containing challenging light conditions and fast-moving objects.**

I. INTRODUCTION

Event-based cameras are novel, bio-inspired visual sensors [1]–[3]. In contrast to traditional frame-based cameras that produce images at a fixed rate, the pixels in an event camera operate independently and asynchronously, responding to intensity changes by producing 'events'. The sensor offers several advantages: high temporal resolution and low latency (in the order of microseconds), a very high dynamic range (140dB vs. 60dB of standard cameras), and low power and bandwidth requirements, therefore benefiting challenging perception tasks, such as high-speed target segmentation [4]–[6], optical flow estimation [7]–[9], scene reconstruction [10]–[12] and simultaneously localization and mapping (SLAM) [13]–[15]. This paper focuses on real-time fast-moving object detection with an event camera onboard an aggressive moving platform. A simple illustration of the proposed pipeline is provided in Fig. 1.

In most aggressive motion scenarios, the frame-based cameras show unpleasant blur effects and further hampers motion segmentation. In contrast, event cameras generate high-temporal resolution event streams which provide adequate information and motion-sensitive signal for subsequent motion segmentation, therefore can cope with the above scenes well. There are two kinds of motion that trigger event streams. First, the camera ego-motion triggers events of both the static background and the foreground objects. Second, the motion of objects provides overlay events to the first source. To properly detect the moving objects in the scene only, we need to compensate for the camera motion first.

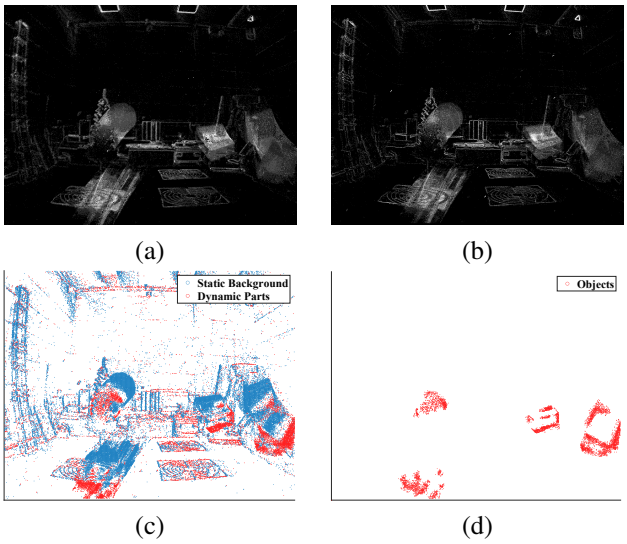


Fig. 1. Example of the proposed moving objects detection pipeline. (a) is the raw events sampled in a time interval. (b) is the wrapped events after nonlinear motion compensation jointed with IMU information (scattered static events are aggregated). (c) performs dynamic threshold segmentation according to (b), the blue points are classified as the background, and the red points are the dynamic candidate events. Dynamic parts are segmented into an independently moving object (d) by filtering and clustering algorithms.

To unlock the potential of event cameras for low-latency perception in highly dynamic scenes, we should compensate for the ego-motion of the camera at the least computation cost. Some motion segmentation algorithms [5], [6], [10] employ the gradient descent method to restore a clear event image, but the convergence of the cost function consumes massive computational resources. Given that the rotation of the camera dominates background events in a short period of time, we consider compensating for the rotational ego motion to obtain object motion events. Specifically, we associate batch events with IMU data in a certain time window in real-time and derive a nonlinear motion compensation formula based on the principle of rigid body kinematics.

After the motion compensation, the dynamic parts (the foreground motion areas) are separated from the compensated events by a linear dynamic threshold segmentation. Finally, the detection of moving objects is achieved by

This work was supported by the National Natural Science Foundation of China under Grant No. 62203358. *(Corresponding author: Yang Lyu.)*

Chunhui Zhao, Yakun Li, and Yang Lyu are with the School of Automation, Northwestern Polytechnical University, Xi'an, Shaanxi, 710129, PR China. Email: {zhaochunhui, liyakun, lyu.yang}@nwpu.edu.cn

2023 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-2365-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICRA48891.2023.10160472

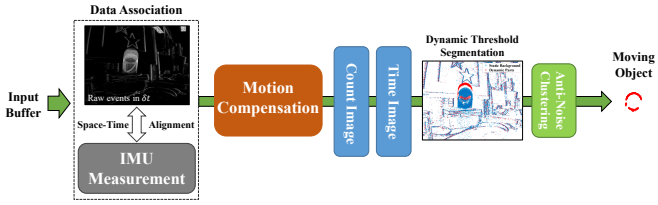


Fig. 2. Overview of the motion compensation and moving objects detection pipeline.

combining the DBSCAN [16] algorithm and the motion attribute (pixel velocity) of the objects. Based on our method described and shown in Fig. 2, the moving object detection can be carried out efficiently on a computation-limited platform, such as a micro-UAV.

The main contributions of our paper are listed as follows:

- First, we propose an 3D rotation compensation algorithm[1] based on a nonlinear formulation which improves the compensation accuracy for aggressive camera rotation and edge pixel events.
- Second, we achieve real-time object detection with an event camera onboard a highly aggressive platform based on our proposed detection pipeline.
- A thorough evaluation based on public datasets and more challenging self-collected data is carried out. Our compensation performance is improved by 10%-15% compared to another sensor-aided method [17].

The rest of the paper is organized as follows: Section II analyzes two types of motion compensation methods. Section III presents our nonlinear motion compensation function and object detection pipeline. The method is evaluated in Section IV and conclusions are drawn in Section V.

II. RELATED WORK

With growing attention on applying event cameras to robot tasks, the key playing role of motion compensation and target segmentation becomes a research common sense. There are mainly two ways to obtain and compensate for its ego-motion, namely the optimization-based methods [5], [6], [10], [18]–[20] and the sensor-aided methods [17], [21]. The optimization-based methods iteratively descend their cost functions to estimate the motion parameters, which costs a lot of computing resources to achieve highly accurate compensation results. On the other hand, the sensor-aided methods obtain motion information directly from additional sensors, therefore can achieve comparable compensation accuracy at a much lower cost.

Gallego *et al.* [19] proposes an optimization-based method to achieve accurate angular rate estimation of an event camera. The algorithm solves the best rotation motion compensation which can achieve the sharpest event image. In their following work [10], the contrast maximization framework is implemented as a general solution in motion, depth, and optical flow estimation applications. Mitrokhin *et al.* [18] proposes an optimization-based motion compensation algorithm that is composed of a rough estimation stage and a refined estimation stage. The two estimation stages based on a four-parameter motion model are carried out through the time image and the count image respectively. Furthermore, Gallego *et al.* [5] uses a one-step optimization method to complete motion compensation and hierarchical target clustering. They achieve event-by-event optimization of different categories of motions which are initially classified by optical flows, so as to achieve image motion compensation and hierarchical clustering simultaneously.

Parameshwara *et al.* [6] combine feature tracking with motion compensation in one optimization framework. They extract features from events and over-classify them and assign a motion model to each class. Compensation and classification are achieved by optimizing model parameters and merging similar models. Zhou *et al.* [20] start with the spatiotemporal graph of events and perform graph cut for clusters. They proposed a simple motion model initialization method to match different segmented graphs. Drawing on [10], they use metrics such as image contrast to optimize motion models and clusters.

The above works estimate camera ego-motion by defining and optimizing cost functions that are related to some event image qualities, which demonstrate good compensation performance. Nevertheless, the optimization process requires large computation overheads and may not be carried out in real-time on a computation-limited robot platform.

On the other hand, the ego-motion compensation aided by other sensors, such as an IMU, also draws some researchers' attention. In work [21], the authors collect and pre-integrate IMU information to recover the rotation of the camera, which is used to create a warping field to compensate for all events in an online fashion. In [17], an exact formula of the warping function is given, which converts the 3D rotation of the camera to a 2D pixel plane motion. However, its linear approximation compensation introduces non-negligible errors as the motion becomes more aggressive.

In this paper, we collect 3D rotational motion to construct a nonlinear warping function to compensate for camera ego-motion, similar to [17], [21]. Compared with optimization-based methods [5], [6], [10], [18]–[20], our method has a comparable compensation performance while guaranteeing real-time processing. Different from [17], [21], we take into consideration of the linear approximation error of the compensation function and the effect of the pixel initial position in the warping function formulation, therefore can achieve a significant reduction of the compensation error when the camera is moving aggressively.

III. IMU-BASED MOTION COMPENSATION

In this part, the motion compensation is derived from the rigid body rotation motion model. Through the principle of pinhole imaging, the collected camera rotation information is applied to each event on the pixel plane to restore a sharper image. Specifically, our algorithm moves the pixel coordinate of the triggered event at time t along its motion

[1]The C++ implementation of the nonlinear motion compensation algorithm and the corresponding dataset are open-sourced at: https://github.com/Jhonny-Li/Motion-compensation.

trajectory on the image plane back to the position δt ago, which is achieved through a warping field. In this section, we will introduce the necessary data association and the specific construction of the warping function.

A. Preliminaries

1) Event camera: When the cumulative brightness change of an event pixel reaches a certain threshold, it will trigger the data stream called Address-Event Representation (AER). It contains a pixel coordinate $\mathbf{x} = (x, y)^\top$ of the triggered event, an event timestamp t, and an event polarity p (brightened or darkened). Each pixel of an event camera can independently perceive changes in the luminosity of the scene. For a pixel $\mathbf{x}$ at time t, if the surrounding scene brightness changes, it will cause the logarithmic intensity of the pixel to change:

$$\Delta \mathbf{L}(\mathbf{x}, t) \doteq \mathbf{L}(\mathbf{x}, t) - \mathbf{L}(\mathbf{x}, t - \Delta t), \tag{1}$$

where $\mathbf{L}(\mathbf{x}, t) \doteq \log \boldsymbol{I}(\mathbf{x}, t)$ represents the logarithmic intensity of pixel $\mathbf{x}$ at time t, $\boldsymbol{I}$ is the intensity image, and Δt is the time since the last event at the same pixel $\mathbf{x}$. When the change reaches a contrast sensitivity threshold C, which is $\Delta \mathbf{L}(\mathbf{x}, t) = pC,\ p \in \{+1, -1\}$, the pixel will output an event $\boldsymbol{e} = \{\mathbf{x}, t, p\}$.

2) Data association between IMU and Event: The rotation of the camera in 3D space is decomposed into the $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$ axes of the camera frame. And the quantities $\dot{\phi}$, $\dot{\theta}$, and $\dot{\psi}$ donate the tilt, pan, and roll angular rates in the three axes respectively. The IMU which is fixed to the camera body supplies this rotational information. Since there is an installation transformation between the camera and IMU, the angular rates collected by IMU need to be converted to the camera frame through the angular rate rotation matrix between them:

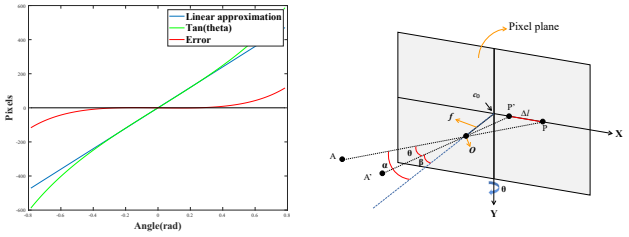
$$\boldsymbol{\omega^c} = \boldsymbol{R^{ci}} \boldsymbol{\omega^i}, \tag{2}$$

where $\boldsymbol{\omega^c} \triangleq \begin{bmatrix} \dot{\phi}^c & \dot{\theta}^c & \dot{\psi}^c \end{bmatrix}^\top, \boldsymbol{\omega^i} \triangleq \begin{bmatrix} \dot{\phi}^i & \dot{\theta}^i & \dot{\psi}^i \end{bmatrix}^\top$ are the three-axis angular rates under camera and IMU frames, and $\boldsymbol{R^{ci}} \in \mathbb{R}^{3\times3}$ is the angular rate rotation matrix between them. The c and i represent the camera frame and IMU frame respectively. We ignore the gyroscope drifts for aggressive motion scenarios in this paper.

B. Motion Compensation with IMU

Motion compensation is based on a rigid body rotation model and a pinhole imaging model. The algorithm restores the picture from motion blur by first integrating the three angular rates during a small interval δt and converting it to the camera frame to get the total rotation angles ϕ, θ, ψ. Then, for an event $\boldsymbol{e}$ at time t, it can be compensated from the pixel coordinates at time t back to its previous position at time $t - \delta t$ through a warping field φ: $\mathbb{R}^3 \to \mathbb{R}^3$, which is $(x', y', t) = \varphi(x, y, t)$. Here we only use the pixel address to simplify the representation of an event at time t: $\mathbf{x}_t = (x, y) \in \mathbb{R}^2$:

$$\mathbf{x}_t{}' = [\mathbf{R}(\mathbf{x}_t - \boldsymbol{c}_0) - \mathbf{T}] + \boldsymbol{c}_0, \tag{3}$$



(a) (b)

Fig. 3. Illustration of motion compensation. (a) shows the linear compensation error. The $\mathbf{x}$-axis is the captured camera rotation angle (which is the change in the incident angle of the objects), and the $\mathbf{y}$-axis is the corresponding compensated pixels. The error curve (red) reflects the difference between the linear method and the tan function increases as the rotation angle increases. (b) gives the $\mathbf{y}$-axis compensation displacement when the initial coordinate of an event is not at $\boldsymbol{c}_0$. P is the event coordinate triggered by object A. After the camera rotates θ around the $\mathbf{y}$-axis, P moves to P'. The incident angles before and after the movement are α, β, respectively. And the compensation displacement should be the red Δl in the pixel plane.

where $\mathbf{x}_t{}'$ and $\mathbf{x}_t$ are the compensated and original pixel coordinates, respectively, and $\boldsymbol{c}_0 \in \mathbb{R}^2$ is the center of the pixel plane. $\mathbf{R} \in \mathbb{R}^{2\times2}$ is a rotation matrix caused by $\mathbf{z}$-axis rotation and $\mathbf{T} \in \mathbb{R}^2$ is a two-dimensional translation vector caused by rotation around the $\mathbf{x}$-axis and $\mathbf{y}$-axis.

A simple plane rotation matrix can compensate the rotation angle ψ of $\mathbf{z}$-axis. For the calculation of the pixel translation vector $\mathbf{T}$, however, it is necessary to consider the triangular tangent relationship formed by the lens focal length f and the event pixel coordinates. In the work of Delbruck *et al.* [17], this tangent relationship is approximated as a linear relationship:

$$\begin{aligned} \mathbf{T} &= K\boldsymbol{\Theta} \\ \boldsymbol{\Theta} = [\theta, \phi]^{\mathrm{T}}, K &= 1/\tan^{-1}(w/f) \end{aligned}, \tag{4}$$

where K is a constant determined by the pixel size w and the camera lens focal length f.

When the camera motion is slow, this linear approximation can output good compensation results. But once the movement becomes aggressive, the intense rotation will cause a large change in the incident angle of the original object point on the imaging plane, which will eventually lead to a non-negligible approximation error in the Equation (4) (see in Fig. 3-a). Additionally, the influence of the initial pixel position of the event needs to be taken into consideration in warping function, unless most of the events are triggered at the center of the image. But more generally, the triggered event coordinates are scattered across the pixel plane, and their compensation displacement for the same rotation angle should be different (Fig. 3-b).

For the above two problems, we reconstruct the compensation function for rotations around the $\mathbf{x}, \mathbf{y}$ axis. First, the $\tan^{-1}$ function is used to replace the linear approximation constant term K. Then the initial coordinates of the events are added to the warping function to further reduce the compensating error, especially for the events away from the imaging center. For easier understanding, we assume that the camera only rotates around the $\mathbf{y}$-axis, and the initial position of the event is on the $\mathbf{x}$-axis, and the process is as follows:

1) Calculate the angels of incidence for $\mathbf{x}_t$ and $\mathbf{x}_t{}'$: α, β.

$$\begin{cases} \alpha = \tan^{-1}(x * w/f) \\ \beta \approx \alpha - \theta \end{cases}. \tag{5}$$

2) Nonlinear displacement compensation.

$$\begin{cases} \Delta l = x - \rho \tan(\beta) \\ \rho = f/w \end{cases}. \tag{6}$$

Here, x is the $\mathbf{x}$-axis component of $\mathbf{x}_t - \boldsymbol{c}_0$, Δl is the $\mathbf{x}$-axis displacement and θ is the rotation angle around the $\mathbf{y}$-axis. It should be noted that the value of β depends on the change of rotation, and its value belongs to $(-90°, 90°)$, that is, the ray cannot be incident from an angle perpendicular to the optical axis. For more general rotation motions, the improved translation vector and rotation matrix are as follows:

$$\mathbf{T} = (\mathbf{x}_t - \boldsymbol{c}_0) - \rho \tan(\boldsymbol{\beta}),\ \mathbf{R} = \begin{bmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{bmatrix}, \tag{7}$$

where $\boldsymbol{\beta} = [\beta_x, \beta_y]^{\mathrm{T}} = [\alpha_x - \theta, \alpha_y - \phi]^{\mathrm{T}}$ represents the incident angels of $\mathbf{x}_t{}'$ in $\mathbf{x}$ and $\mathbf{y}$-axis, and α_x, α_y are the incident angels of $\mathbf{x}_t$. Eventually, the compensated events set $\boldsymbol{C}' \in \mathbb{R}^3$ is generated by the initial set $\boldsymbol{C} \in \mathbb{R}^3$ through the warping function:

$$\begin{aligned} \boldsymbol{C}' &= \prod \{\varphi(\boldsymbol{C})\} \\ &= \prod \{\varphi(x, y, t)\} = \{x', y', t\}, \forall \{x, y, t\} \in \boldsymbol{C}. \end{aligned} \tag{8}$$

IV. MOVING OBJECTS SEGMENTATION AND DETECTION

Since the timestamps of the warped events are not changed, the average timestamps of events that triggered by an independently moving object is different from those that triggered by the background motion. And this enables segmentation and detection of moving objects. Specifically, the event clusters belong to the independent moving objects are distributed relatively discretely on the compensated image, while the background are more concentrated. This will be reflected on the average timestamp of the pixels, so as to detect the moving parts [18], [21].

A. Objects Segmentation

With Equation (8), we gather the static background according to the trajectory of the camera, and the edges of it become clear and sharp on the image plane. In contrast, the events caused by moving parts are still scattered over the image as the motions of them are random and independent of the camera motion. Therefore, with the immutability of event timestamp, the average timestamp of events on a pixel can be an indicator to distinguish dynamic parts from the static in the picture.

To obtain the average timestamp of the image $\mathcal{T}$, we first calculate the count-image $\mathcal{I}$. It is a matrix consisting of the number of events contained in each pixel:

$$\xi_{ij} = \{\{x', y', t\} : \{x', y', t - \delta t\} \in C', i = x', j = y'\}, \tag{9}$$

where ξ_{ij} is the set of events contained in one count-image pixel (i, j), and $\mathcal{I}_{ij} = |\xi_{ij}|$ records the number of it, and here, the $\mathcal{I}_{ij} \in \mathcal{I}$. Next, the time image $\mathcal{T}$ can be formed by the average event timestamp of each pixel:

$$\mathcal{T}_{ij} = \frac{1}{\mathcal{I}_{ij}} \sum t; t \in \xi_{ij}. \tag{10}$$

Then, we normalizing $\mathcal{T}$ to get $\boldsymbol{T}$. A threshold λ can be used to distinguish the dynamic and static parts in $\boldsymbol{T}$:

$$T_{ij} = \frac{\mathcal{T}_{ij} - \frac{1}{n} \sum \mathcal{T}_{ij}}{\delta t}, \lambda = a \left\| \boldsymbol{\omega} \right\| + b. \tag{11}$$

If $T_{ij} > \lambda$, then the pixel (i, j) belongs to a dynamic target, otherwise it belongs to the static background. Our nonlinear compensation algorithm improves the ability to filter out static background events. And to enhance the robustness of the process, the threshold parameter λ is dynamically adjustable. It is a function linearly related to the angular rate $\boldsymbol{\omega}$ of camera motion, and the parameters a and b are determined by experiments in real scenarios.

B. Objects Clustering

In order to detect objects with different motions, a joint clustering method is employed to gather events belonging to the same motion attributes. The DBSCAN algorithm [16] is a density-based clustering method, which obtains the noise resistance and low average time complexity. It is of great help to event data with low signal-to-noise ratio and can maintain the real-time performance of the algorithm. However, the classification principle based on point density is not effective in distinguishing two adjacent targets (*e.g.*, when the projections of two moving objects are close on the image plane). Events are triggered sequentially on the time axis, so we decided to use the timestamp value of the event to calculate its velocity, i.e. 2D optical flow, to provide a prior to distinguish two adjacent objects. But, robust object detection results are still not available for event streams with poor signal-to-noise ratios. We found that the spatio-temporal properties of events can be exploited to achieve good noise suppression: Trajectories triggered by moving or stationary objects are continuous in space and time, while the noise is random. Eventually, the clustering metric is given by the combination of event 3D position and 2D optical flow:

$$C_{i,j}(\boldsymbol{p}, \boldsymbol{v}) = w_p \left\| \boldsymbol{p}_i - \boldsymbol{p}_j \right\| + w_v \left\| \boldsymbol{v}_i - \boldsymbol{v}_j \right\|, \tag{12}$$

where $\boldsymbol{p} \in \mathbb{R}^3$ is the spatio-temporal coordinates of the three-dimensional event point composed of x, y and t. $\boldsymbol{v}$ is the optical flow velocity of the event point. The indices i, j represent two different event points, and $\mathbf{w} = [w_p, w_v]^{\mathrm{T}}$ is the weight vector similar to that in the work [21]. In addition, we also perform median filtering on the compensated image to filter out salt and pepper noise.

V. EXPERIMENTS & EVALUATION

Overview. We run our algorithm on public datasets [18], [22] and real-world scenes to evaluate the motion compensation module and target detection capabilities under challenging conditions. Qualitative and quantitative experiments and analysis are performed to compare our method with

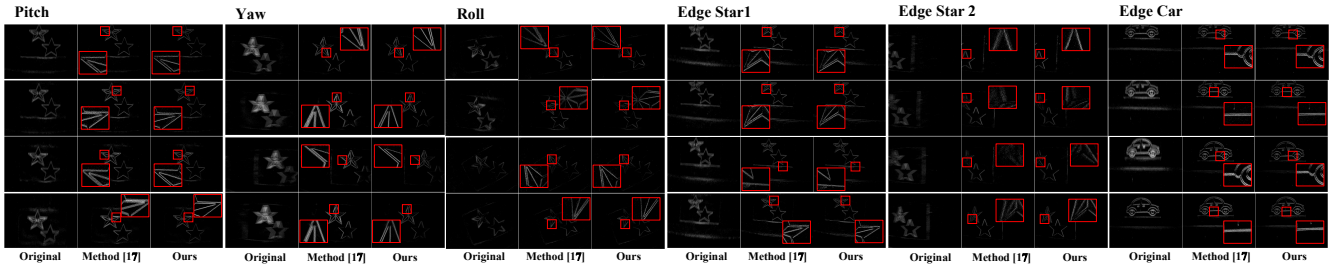


Fig. 4. Motion compensation comparison with the sensor-aided method. From left to right are the experiments of the three-axis fast rotational motions and edge motions, respectively. The red rectangular area is the local zoom-in to see the details.

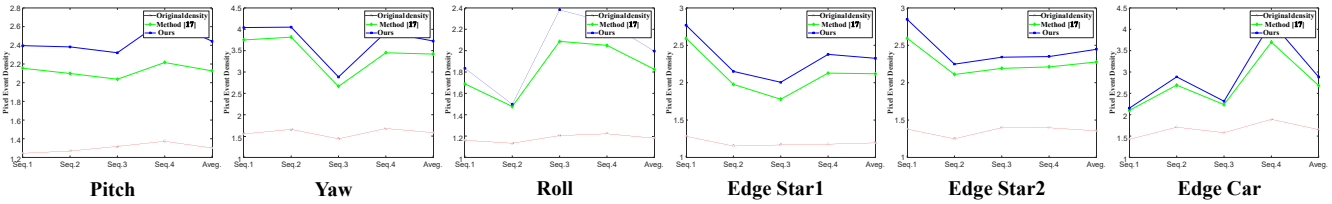


Fig. 5. Pixel-event density curves. In the figures, the $\mathbf{x}$-axis represents the different experimental sequences and the final averaged results, and the $\mathbf{y}$-axis is the density value.

other sensor-aided [17] and optimization-based [10], [18] methods. At the same time, we give an evaluation metric, the pixel event density D, to quantify the performance of motion compensation. It is calculated from the total number of triggered pixels and the count-image $\mathcal{I}$:

$$D_{\delta t} = \frac{\sum \mathcal{I}_{ij}}{\sum p_{ij}}, \tag{13}$$

where $D_{\delta t}$ represents the density of compensated events in time window δt and p_{ij} is the pixel that is triggered by the warped events. Equation (13) shows that if most of the events are gathered on fewer pixels, the greater the density of events and the sharper image. And therefore, a larger value of D represents a better compensation.

In this section, we will show the good performance and low latency of the improved motion compensation algorithm for intense motions and edge events conditions. Besides that, the robust ability of the algorithm to detect moving targets (single or multiple objects) in public datasets and self-collected data streams is presented. Through the experiments, the ability of our pipeline to perceive moving objects in highly dynamic scenes is demonstrated, which is mainly reflected in : (i) The sensor-aided nonlinear compensation algorithm improves the compensation performance under both intense camera motion and image edge compensation situations, and its accuracy is comparable to that of state-of-the-art optimization methods. (ii) The low latency of motion compensation enables the robustness of the detection pipeline for real-time object detection in high-speed scenes.

Datasets and Hardware. The experiments tested diverse scenarios and public datasets. First, the EED dataset [18] was recorded under extreme lighting conditions, including high-speed single and multi-target motions in a variety of scenarios, which is challenging for traditional target detection algorithms. Second, the *"shapes rotations"* scene in dataset [22] contains high-speed rotation motion. Third, the self-collected data stream was recorded in real scenes by hand-held, containing single or multiple moving objects. The movement of the camera is mainly composed of rotation around three axes, and it is more intense, with the maximum angular rate reaching more than 500 deg/s. Datasets [18], [22] were collected by DAVIS240 series sensors which have a resolution of 240×180 with a pixel size of 18.5 μm and support RGB frame output. While the hand-collected data was collected by a higher-resolution Dvxplorer sensor, reaching 640×480 with a pixel size of 9 μm, but no frame output. The pipeline Synchronizes events and IMU data within a time window of 10 ms, and runs the object detection task in real time on the Intel NUC11PAHi7 (i7-11700) platform.

TABLE I
AVERAGE PIXEL EVENT DENSITY FROM FIG. 4.

Motions / Sources	Original Events	Method [17]	Ours
Pitch Rotation	1.3015	2.1277	**2.4440**
Yaw Rotation	1.5812	3.4232	**3.7279**
Roll Rotation	1.1807	1.8254	**1.9960**
Edge Motion (star1)	1.1968	2.1210	**2.3258**
Edge Motion (star2)	1.3570	2.2779	**2.4484**
Edge Motion (car)	1.6531	2.6832	**2.8842**

Motion Compensation Evaluation. We conduct comparative experiments with the same sensor-aided [17] and the state-of-the-art optimization-based [10], [18] methods, using real-sense data stream and public dataset [22]. In the comparative experiment with [17], we use the hand-drawn patterns of stars and cars (Fig. 4) with rich texture information as the capturing object to test the ability of algorithms to compensate for detailed textures. And the curves in Fig. 5 present the quantitative results of the two sensor-aided approaches. Our method results in sharper images and higher pixel event densities. Furthermore, we reproduce two optimization-based works [10] and [18], which build optimization problems based on local optical flow and 4-DOF

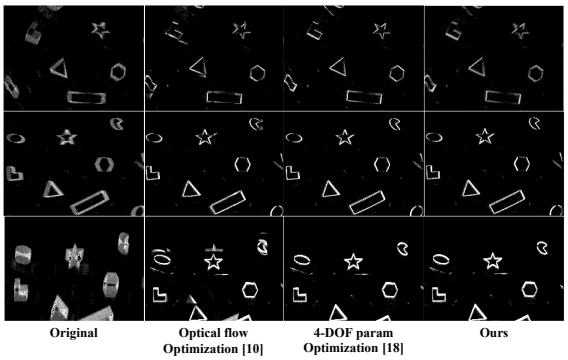


Fig. 6. Motion compensation comparison with optimization-based approaches. The first column is the three original events from dataset [22], the second column is the compensation results from the local optical flow optimization in [10], the third column is the 4-DOF motion model optimization results in [18], and finally, our results are in the fourth column.

TABLE II

PIXEL EVENT DENSITY FROM FIG. 6.

Motions /Sources	Original Events	Method [10]	Method [18]	Ours
Roll Rotation	1.8505	2.6805	**3.3484**	2.6300
Yaw Rotation	2.7175	4.0836	**4.3002**	4.2454
Pitch Rotation	3.3143	4.6879	5.0446	**5.1115**

motion models, respectively (Fig. 6). The local optical flow in [10] lacks global consistency, resulting in partial overlap of images, which can be improved by increasing the size of the local block, but will also invalidate the local optical flow consistency. The performance of our method is close to [18], which demonstrates the accuracy of the improved compensation method. A more quantitative description can be found in Table I,II, they list the comparison of average pixel event density values for several sequences under different scenes. **Algorithm latency.** The time consumption of sensor-aided methods is much lower than that of optimization-based methods. Our method has the same low latency as another sensor-aided method [17] (within $10ms$). Conversely, one single iteration of the two optimization frameworks at a resolution of 240×180 takes at least 15 ms, not to mention a higher resolution, so it is difficult to process in real time.

Results of Object Detection. Experiments on multiple real throws and datasets [18] verified the capability of the algorithm to detect fast-moving objects. We choose basketball and tennis as throwing objects. The balls are thrown into the field of view from an arbitrary angle while the camera moves randomly. The preprocessing of the motion compensation module brings a clear background image and highlights the properties of dynamic objects (the layered color part in Fig. 8), which makes object detection more accurate and robust. Several scenarios in [18] were also tested: *"Fast drone"*, *"Multiple objects"*, *"What is a Background?"*, *"Strobe"*, etc. In some scenes with slow-moving objects, due to the low resolution of davis240, the dynamic object information is limited under small targets and extreme illumination, resulting in an extremely low signal-to-noise ratio. At this time, motion compensation does little help to segmentation. To ensure the success of detection, the anti-

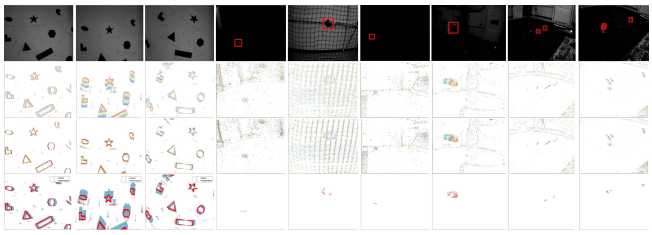


Fig. 7. Experimental results of our pipeline on public datasets. The first three columns are the motion compensation results on dataset [22]. From top to bottom: image frame, raw data layered by timestamp (the red represents the latest events and the blue is the oldest), the compensated image, and the comparison between the compensated and the original. The remaining columns are the object detection results on dataset [18].

noise ability of the algorithm should be strong enough. Our algorithm combines spatiotemporal event filtering, median filtering, and an anti-noise clustering algorithm, which is sufficient to achieve stable target detection for the above scenes. These, however, also introduce some costs. Multi-stage filtering will more or less filter out some useful target contour information, leading to a partial loss of the target (Fig. 7: Columns 4, 5, 6). Nevertheless, Once the movement of the target relative to the camera becomes more significant, the benefits of motion compensation will be greater, and the detection of the moving target will be better (Fig. 7: Columns 7, 8, 9). Besides, higher image resolution will alleviate the above problems. Given the low computational load of our algorithm, the pipeline supports running at higher image resolution (Fig. 8). In general, the proposed algorithm has obtained relatively accurate and robust fast-target detection results in both extreme datasets and real scenes.

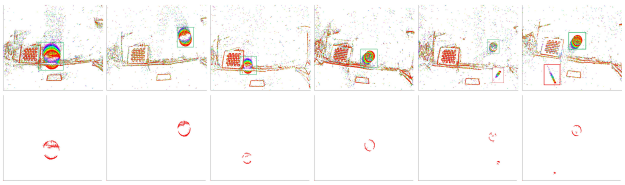


Fig. 8. Self-collected data experiments (collected by the higher resolution camera Dvxplorer, no frame output).

VI. CONCLUSION

In this paper, we present a motion compensation and fast-moving target detection pipeline based on sensor fusion, which can work in the cases of fast self-rotation or fast-moving targets with a good anti-noise ability. We have lower computation costs than optimization-based methods, enable robust sequence detection, and run in real-time on higher-resolution images. Compared with the same sensor-aided methods, the algorithm restores clearer and more detailed pattern information, providing good sources for subsequent visual tasks. After extensive experimental validation, our method has the potential for engineering application.

ACKNOWLEDGEMENT

The work is supported by the National Natural Science Foundation of China under Grant No. 62203358. The authors would like to thank Dr. Liyuan Pan for her valuable discussions and insightful suggestions.

REFERENCES

[1] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128×128 120 dB 15 μs latency asynchronous temporal contrast vision sensor. 43(2):566–576, 2008.

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

[3] Christoph Posch, Daniel Matolin, and Rainer Wohlgenannt. A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS. 46(1):259–275, January 2011.

[4] Anton Mitrokhin, Chengxi Ye, Cornelia Fermuller, Yiannis Aloimonos, and Tobi Delbruck. EV-IMO: Motion segmentation dataset and learning pipeline for event cameras. 2019.

[5] Timo Stoffregen, Guillermo Gallego, Tom Drummond, Lindsay Kleeman, and Davide Scaramuzza. Event-based motion segmentation by motion compensation. 2019.

[6] Chethan M Parameshwara, Nitin J Sanket, Chahat Deep Singh, Cornelia Fermüller, and Yiannis Aloimonos. 0-mms: Zero-shot multi-motion segmentation with a monocular event camera. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9594–9600. IEEE, 2021.

[7] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio-Hoi Ieng, and Chiara Bartolozzi. Event-based visual flow. 25(2):407–417, 2014.

[8] Alex Junho Lee and Ayoung Kim. Event-based real-time optical flow estimation. In *2017 17th International Conference on Control, Automation and Systems (ICCAS)*, pages 787–791. IEEE, 2017.

[9] Liyuan Pan, Miaomiao Liu, and Richard Hartley. Single image optical flow estimation with an event camera. pages 1669–1678. IEEE, 2020.

[10] Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. pages 3867–3876, 2018.

[11] Henri Rebecq, Guillermo Gallego, Elias Mueggler, and Davide Scaramuzza. EMVS: Event-based multi-view stereo—3D reconstruction with an event camera in real-time. 126(12):1394–1414, December 2018.

[12] Germain Haessig, Xavier Berthelon, Sio-Hoi Ieng, and Ryad Benosman. A spiking neural network model of depth from defocus for event-based neuromorphic vision. *Scientific reports*, 9(1):1–11, 2019.

[13] Raoul Hoffmann, David Weikersdorfer, and Jörg Conradt. Autonomous indoor exploration with an event-based visual SLAM system. pages 38–43, 2013.

[14] Michael Milford, Hanme Kim, Stefan Leutenegger, and Andrew Davison. Towards visual SLAM with event-based cameras. In *The Problem of Mobile Sensors Workshop in conjunction with RSS*, 2015.

[15] Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Ultimate SLAM? combining events, images, and IMU for robust visual SLAM in HDR and high speed scenarios. 3(2):994–1001, April 2018.

[16] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[17] Tobi Delbruck, Vicente Villanueva, and Luca Longinotti. Integration of dynamic vision sensor with inertial measurement unit for electronically stabilized event-based vision. pages 2636–2639, 2014.

[18] Anton Mitrokhin, Cornelia Fermuller, Chethan Parameshwara, and Yiannis Aloimonos. Event-based moving object detection and tracking. 2018.

[19] Guillermo Gallego and Davide Scaramuzza. Accurate angular velocity estimation with an event camera. *IEEE Robotics and Automation Letters*, 2(2):632–639, 2017.

[20] Yi Zhou, Guillermo Gallego, Xiuyuan Lu, Siqi Liu, and Shaojie Shen. Event-based motion segmentation with spatio-temporal graph cuts. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[21] Davide Falanga, Kevin Kleber, and Davide Scaramuzza. Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics*, 5(40):eaaz9712, 2020.

[22] 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.