

# Motion Robust High-Speed Light-weighted Object Detection with Event Camera

Bingde Liu

Electronic Information School

Wuhan University

**Abstract**—The event camera produces a large dynamic range event stream with a very high temporal resolution discarding redundant visual information, thus bringing new possibilities for object detection tasks. However, the existing methods of applying the event camera to object detection tasks using deep learning methods still have many problems. First, existing methods cannot take into account objects with different velocities relative to the motion of the event camera due to the global synchronized time window and temporal resolution. Second, most of the existing methods rely on large parameter neural networks, which implies a large computational burden and low inference speed, thus contrary to the high temporal resolution of the event stream.

In our work, we design a high-speed lightweight detector called Agile Event Detector (AED) with a simple but effective data augmentation method. Also, we propose an event stream representation tensor called Temporal Active Focus (TAF), which takes full advantage of the asynchronous generation of event stream data and is robust to the motion of moving objects. It can also be constructed without much time-consuming. We further propose a module called the Bifurcated Folding Module (BFM) to extract the rich temporal information in the TAF tensor at the input layer of the AED detector. We conduct our experiments on two typical real-scene event camera object detection datasets: the complete Prophesee GEN1 Automotive Detection Dataset and the Prophesee 1 MEGAPIXEL Automotive Detection Dataset with partial annotation. Experiments show that our method is competitive in terms of accuracy, speed, and the number of parameters simultaneously. Also by classifying the objects into multiple motion levels based on the optical flow density metric, we illustrated the robustness of our method for objects with different velocities relative to the camera.

**Index Terms**—Event Camera, Object Detection, Deep Learning

## I. INTRODUCTION

The event camera [1]–[5], as a new type of sensor, has many excellent features compared to traditional frame-based APS cameras: firstly, event cameras do not rely on synchronized global time stamps, but rather each pixel independently fires an event once an illumination change is detected, so the event stream has a very high temporal resolution, typically of the order of microseconds. Secondly, by measuring only illumination changes, the event camera triggers a spatially and temporally sparse asynchronous signal, usually encoding the edges of the moving objects, thus can automatically discard redundant visual information. Thirdly, event cameras have a large dynamic range (typically over 120 dB) due to logarithmic pixel response characteristics. Due to these properties, event cameras have great promise for applications in scenarios where traditional frame-based APS cameras are

subject to motion blur, extreme lighting, and high latency, with very low energy consumption and small data storage costs. The real-world object detection problem has extremely high requirements on the temporal resolution of the sensor and robustness in different scenarios, so the event camera brings new possibilities for object detection tasks. There are already works based on deep convolutional neural networks applying event cameras to object detection tasks [6]–[15], which are mainly used in traffic scenarios. Among them, the methods with advanced accuracy level mainly reconstruct event stream data into synchronous and dense multi-channel frame-level representation tensor, which is then fed into a convolutional neural network-based object detector. However, the existing methods still have many problems.

First, existing methods cannot take into account objects with different velocities relative to the motion of the event camera. Current methods for reconstructing event stream data into a tensor require a fixed global time window: each time a representation tensor is built, only event streams within some fixed time range before the current timestamp will be considered. This is like the globally synchronized shutter time in traditional photography. Since the event camera only measures illumination changes, objects generating events depend on the relative motion of the event camera and the target when the environment illumination is constant. The edges of objects that are moving fast relative to the event camera tend to trigger events more frequently; on the other hand, objects that are moving slow or even stationary relative to the event camera do not likely to trigger events. Therefore, if there is only one object with fast motion relative to event camera in the field of view, it does not need a long time window to obtain sufficient information for object detection, and an excessively long time window will instead reduce the temporal resolution and lead to the occurrence of motion blur; on the contrary, if there is only one target with slow motion relative to event camera in the field of view, a longer time window is required to obtain sufficient information. In the object detection problem, we often have to detect multiple objects in the field of view at the same time. When there are objects with different motion speeds relative to event camera in the field of view, we can not take them all into account. This is similar to a global shutter time that cannot take into account the dark and light parts of a photo in traditional photography.

Second, most of the existing deep learning object detection methods with high accuracy rely on large parameter neural

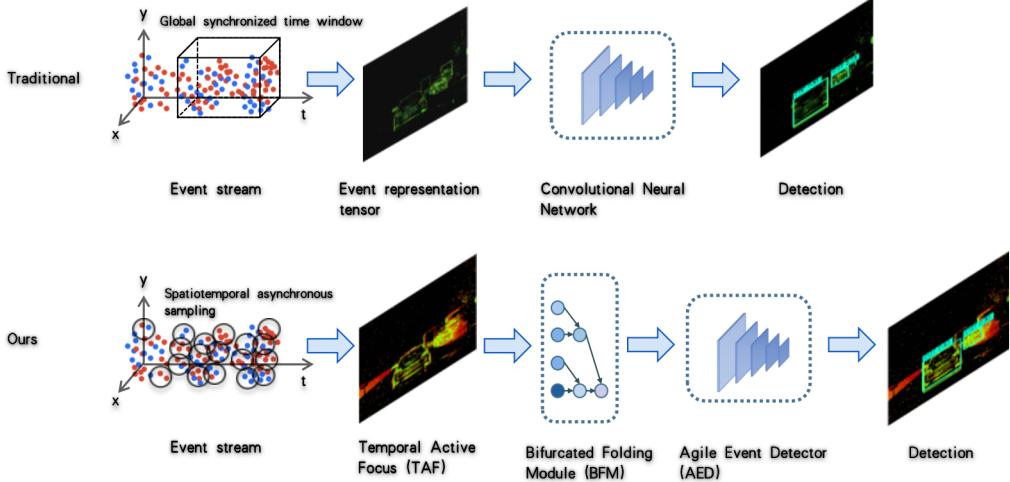


Fig. 1: Overall architecture diagram of our approach and comparison with the traditional approach

networks, which implies a large computational burden and low inference speed. Applying those algorithms to event stream data is contrary to its high temporal resolution. The state-of-the-art methods with memory mechanism in the field of event stream object detection [10], [11] use Recurrent-Convolutional neural networks, which are not designed according to the characteristics of the event stream data, so their effectiveness depends on a large number of parameters and datasets with high labeling frequency for training. Therefore, they are expensive to train and slow to run. In order to fully exploit the potential of using event cameras for object detection tasks, we need to design a lightweight detector for event stream data that balances accuracy and real-time performance.

To solve the above problems, we propose the following solutions:

Firstly, we designed a high-speed lightweight detector called Agile Event Detector (AED) based on the lightest implementation of the YOLOX [16] target detection model. We mainly modify the backbone network structure to improve its detection accuracy for event stream representation tensor data while increasing its running speed. We also design a simple but effective data augmentation method for event stream representation tensor data to improve the generalization capability of the detector.

Secondly, we propose an event stream representation tensor called Temporal Active Focus (TAF). It takes full advantage of the asynchronous generation of event stream data. By sampling the Event Measurement Field [17] in different spatial and polarity positions asynchronously in time, the generated event representation tensor is discrete and fixed in the temporal dimension, but different spatial and polarity positions contain information in different time ranges. Moreover, the TAF tensor has the property of being incrementally updated, which makes it much less computationally demanding to construct in continuous event stream detection scenarios than the state-

of-the-art methods.

Finally, we propose a module called the Bifurcated Folding Module (BFM), which is located at the input layer of the AED detector. The rich temporal information in TAF tensor will be pre-extracted by the module, instead of being fed into the convolutional neural network directly by the convolution operation.

Overall architecture diagram of our approach and comparison with the traditional approach is shown in Fig.1.

We choose two typical real-scene event camera object detection datasets: the complete Propesee GEN1 Automotive Detection Dataset (GEN1 Dataset) [18] and the Propesee 1 MEGAPIXEL Automotive Detection Dataset [10] with partial annotation (1 MEGAPIXEL Dataset (Subset)) to perform experiments. The experiments show that our proposed data augmentation method can substantially improve the accuracy of the baseline detector on the test set by large amount. With only a small increase in the number of parameters compared to YOLOX, Agile Event Detector (AED) reduces inference time while improving accuracy.

We use the optical flow calculated using the event stream data as a measure of the object's motion speed relative to the event camera, then classify the objects on both datasets into 5 motion levels according to it. We calculate the detection accuracy for each of the 5 levels. In this way, for detectors using different event representation methods with different parameters, we can measure their detection performance for objects with different motion speeds. Experiments show that compared with using the best parameters of the state-of-the-art event representation methods, using the TAF event stream representation tensor will lead to a significant improvement in the detection accuracy. Besides, adding BFM will further improve the detection accuracy for all 5 motion levels. Our proposed method achieves the best detection accuracy for all 5 motion levels in GEN1 Dataset and 4 motion levels in 1 MEGAPIXEL Dataset (Subset). This shows the motion

robustness of our method.

Compared with the current state-of-the-art deep learning object detection methods for event stream data, our method has only 24.1% of the number of parameters compared with the state-of-the-art feedforward detector without memory mechanism, but 9.5% improvement in accuracy and reduction in inference time on GEN1 Dataset, 9.6% improvement in accuracy on 1 MEGAPIXEL Dataset Sampled and inference time reduction of 37.1%. Compared with the state-of-the-art method using the memory mechanism, our method has 1.3% lower accuracy on the GEN1 Dataset, but runs at a higher speed and has only 37.4% of the number of parameters.

## II. RELATED WORKS

In this section we present related works, including event representation methods and object detection methods for event cameras.

### A. Event representation methods

Currently, representative approaches to applying event cameras to high-level tasks have mainly been done by processing events in batches, converting them into regular, tensor-like representations, making them compatible with deep learning techniques based on image data, and feeding them into traditional deep convolutional network-based target detectors. There has been much work demonstrating that the statistics of these temporal representation tensors overlap with those of natural images, thus also enabling transfer learning using networks pre-trained on image data [17], [19], [20]. These methods are divided into two main categories: representation methods based on discretizing Event Measurement Field, collectively called Event Spike Tensor, and representation methods emulating spike neural networks.

The Event Measurement Field is defined in the work of Gehrig et al. [17] by representing each event as a Dirac pulse in a continuous spatio-temporal manifold. The Event Measurement Field is generated by assigning a measurement to each event, and the representation tensor is then obtained by using kernel convolutions to aggregate the events. The representation tensor has at least two spatial dimensions and one temporal dimension, and the temporal dimension is considered the channel dimension when feeding the tensor into the detector. The polarity can be either encoded into the values or separated in the channel dimension. The manually generated Event Spike Tensor includes Event Frame [21], Event Count Image [19], [20], Event Volume [22]. Such methods require specifying hyperparameters including the measurement function, the convolution kernel function, and timestamps for executing kernel convolutions. There are also some works [11], [17] that explore the possibility of learning the measurement function or the convolution kernel function directly from the raw event stream, but the timestamps for performing kernel convolutions are still a hyperparameter that needs to be specified.

A common approach to the representation of emulating spike neural networks is to maintain a tensor with two spatial

dimensions and one channel dimension, and always ensure that the spatial locations of newly occurring events in the tensor have larger values. Surface of Active Events [20], [23] gives a 2D snapshot of the latest timestamp of the events in the field of view. In practice, a global numerical transformation is applied to it so that its values satisfy a certain value domain. Surface of Active Events requires the global numerical transformation function as the hyperparameter. Leaky Surface [8] is a process capable of accumulating events, where sparse events are integrated into a leaking surface. Whenever an event is triggered at a spatial location, the corresponding spatial location of the tensor increments the value by a fixed amount, and the values in other locations decrease. By an attenuation function, the amount of decrement is determined by the difference between the current timestamp and the timestamp of the last received event at a specific location. The hyperparameters to be specified by Leaky Surface are the incremental value and the attenuation function.

### B. Object detection methods for event cameras

There are existing works on applying event cameras to target detection tasks based on deep convolutional neural networks, mainly in traffic scenarios.

Conventional approaches using feedforward target detector include that designed by Chen et al [6], using Event frame as event representation tensor and YOLO [24] model as feedforward object detector, and that designed by Hu et al [7], using Event Volume as the event representation tensor and YOLOv3 [25] model as the feedforward object detector. Some works focus on exploiting the spatial sparse nature of the Event Representation Tensor to reduce the computational effort of the model. Cannici et al. [8] designed an approach using Leaky Surface as the Event Representation Tensor and a fully convolutional YOLO [24] model with sparsely updateable feature map as a feedforward object detector. Messikommer et al. [9] designed a method using a sparse Event Count Image as the event representation tensor and an asynchronous sparse convolution-based implementation of the VGG [26] model as a feedforward object detector.

Recurrent-Convolutional neural network object detector is currently an advanced approach in the field of event camera event stream object detection. Among them, the RED algorithm designed by Perot et al [10] uses Event Volume as the event representation tensor and replaces the deep layer of SE-ResNet [27] backbone with the ConvLSTM [28] module. The ASTMnet algorithm designed by Li et al. [11] uses an asynchronous attention embedding learned directly from the original event stream and the VGG [26] backbone with a lightweight memory module called Rec-Conv. These memory mechanisms are not designed according to the characteristics of the event stream data, so their effectiveness depends on a large number of parameters and datasets with high labeling frequency for training. Therefore, they are expensive to train and slow to run.

Methods to fuse the event stream representation tensor and traditional APS camera images all use Event Frame as

the event representation tensor and the YOLOv3 [25] as feedforward target detector. Among them, Li et al. [13] post-fused the detection results using Dempster-Shafer theory [29], while Jiang et al. [12] performed fusion on the confidence map and output the detection results. The fusion method requires traditional APS cameras to be involved in data acquisition, implying more energy consumption and data storage costs. Besides, not all event camera datasets provide the corresponding traditional APS frames at the same time.

Messikommer et al. [9] have also explored maintaining asynchronous queues of events at each spatial location to sample the event stream asynchronously while using a VGG [26] model implemented based on asynchronous sparse convolution as a feedforward object detector. This approach can theoretically overcome the problem of hyperparameter setting in the event stream representation tensor, but it is not mature enough to be used for designing high-accuracy detectors.

### III. THEORETICAL FOUNDATIONS

This section focuses on the theoretical basis of our work. We first introduce event stream data and its characteristics. Then, we introduce the event stream data target detection paradigm, based on which we discuss the relationship between event streams and the detection of target motion. Next, we address the problems in event stream data target detection and the shortcomings of existing methods.

#### A. Event stream data

Event cameras bring a new computer vision paradigm by representing visual information in a different way [2]–[4]. Instead of encoding dynamic visual scenes using a series of static images acquired at a fixed frame rate like traditional APS cameras, event cameras generate data in the form of sparse and asynchronous event streams. According to the event generation mechanism, an event  $\{e_i = (x_i, y_i, p_i, t_i)\}$  will be triggered at moment  $t$  when the log scale illumination change at the pixel  $(x, y)$  exceeds a predefined threshold  $\eta$  compared to the previous moment, i.e.,

$$\log(I(x_i, y_i, t_i)) - \log(I(x_i, y_i, t_i - \Delta t)) \geq p_i - \eta \quad (1)$$

where  $\Delta t$  is a value of the order of microseconds. The polarity  $p \in \{0, 1\}$  indicates whether the illumination is increasing or decreasing. The asynchronous and sparse nature of the event stream data is explored in detail in the work of Messikommer et al. [9]; in short, when the global illumination is constant, the event camera responds mainly to the edges in the scene. This means that events are not present at the vast majority of spatial locations and moments.

#### B. Event stream data object detection

A paradigm for the event camera object detection problem was defined by Perot et al. It is briefly modified here to make it more precise and to facilitate the discussion later on.

- Event stream data: for a camera with a field of view height of  $H$  and width of  $W$ , its event stream data is defined as:

$$E = \{e_i = (x_i, y_i, p_i, t_i)\}_{i \in \mathbb{N}} \quad (2)$$

where  $x_i \in \{0, 1, \dots, W - 1\}$ ,  $y_i \in \{0, 1, \dots, H - 1\}$ ,  $p_i \in \{0, 1\}$ ,  $t_i \in [0, T_{max}]$ , in which  $T_{max}$  is the maximum duration of the event stream record.

- Annotation:

$$B^* = \{b_j^* = (x_j, y_j, w_j, h_j, l_j, t_j)\} \quad (3)$$

where  $l_j \in \{0, \dots, L\}$  is the class,  $t_j$  is the timestamp of the appearance of the annotation. The labeling is performed at a specific time interval, so  $t_j$  has only a fixed set of taken values.

- Detection result:

$$\mathbb{D}^{(k)}(E) \approx \{D(\{e_i\}_{t_i \in [0, t^{(n)}]})\} \quad (4)$$

where  $D(\cdot)$  is the detector,  $k$  means it is the  $n$ th detection on the event stream,  $n \in \{1, 2, \dots\}$ .  $t^{(n)}$  denotes the timestamp of the  $n$ th detection occurrence with  $t^{(n)} > t^{(n-1)}, \forall n$ . The detection bounding boxes with timestamp  $t^{(n)}$  can be matched with the annotation with  $t_j$  near  $t^{(n)}$ , thus applying the evaluation protocol defined under synchronized frames. Ideally, the detector will detect at each  $t_j$ .

#### C. The relationship between event stream and motion

When the global illumination of the environment is constant, an object generates events depending on its relative motion to the event camera. The edges of objects moving fast relative to the event camera tend to be more likely to trigger events, while objects moving slow or even stationary relative to the event camera do not easily, or even not, trigger events.

The most accurate measure of motion velocity is the optical flow calculated from camera imaging information. There have been many efforts to estimate optical flow from event stream data. We chose the method proposed by Nagata et al. [30] because of its computational simplicity and efficiency. They estimated the TV-L1 optical flow [31] by matching adjacent timestamps using the Surface of Active Events [20], [23] representation tensor built from the event stream data, which allows the estimation of the TV-L1 optical flow without recovering luminance or additional sensor information. At the timestamp  $t_j$  with a labeling bounding box  $b_j^*$ , the dense optical flow can be represented as a horizontal optical flow  $u_{t_j}(x, y)$  and a vertical optical flow  $v_{t_j}(x, y)$ , i.e., both a horizontal and a vertical component exist at each spatial location, to describe the short-time motion velocity.

#### D. Problem Statement

Event Spike Tensor is a data representation method for event streams based on discretizing the Event Measurement Field. According to the definition proposed in the work of Gehrig et al [17], we define the Event Measurement Field build from  $E^{(n)} = \{e_i\}_{t_i \in [0, t^{(n)}]}$  as  $S^{(n)}(x, y, p, t) = \sum_{e_i \in E^{(n)}} f(x, y, p, t) \delta(x - x_i, y - y_i, p - p_i, t - t_i)$ , where  $x \in \{0, 1, \dots, W - 1\}$ ,  $y \in \{0, 1, \dots, H - 1\}$ ,  $p \in \{0, 1\}$ ,  $f(\cdot)$  is the measurement used to extract event features, usually defined manually, but can also be learned from the event stream [11], [17].

The temporal dimension of the Event Measurement Field is indefinite and continuous, to transform it into the Event Spike Tensor with a fixed and discrete temporal dimension, it is necessary to use kernel convolutions to aggregate events at specific timestamps. The general formula of Event Spike Tensor is  $S^{(n)}(x, y, p, b) = \sum_{e_i \in E^{(n)}} f(x_i, y_i, p_i, t_i) k(x - x_i, y - y_i, p - p_i, \tau_b^{(n)} - t_i)$ .

$T^{(n)} = \{\tau_b^{(n)}\}_{b \in \{0, 1, \dots, B - 1\}}$  is a set of timestamps for performing kernel convolutions, and  $B$  is the length of the temporal dimension of the Event Spike Tensor. In the existing work on Event Spike Tensor,  $T^{(n)}$  are a set of globally synchronized timestamps independent of  $x, y, p$ . Timestamps in  $T^{(n)}$  are usually set at a fixed time interval  $\Delta\tau$ , i.e.,  $T^{(n)} = \{\tau_b^{(n)} = t^{(n)} - (B - b)\Delta\tau\}_{b \in \{0, 1, \dots, B - 1\}}$ .

$k(\cdot)$  is the convolution kernel for aggregating events, usually manually defined but can also be learned from the event stream [11], [17]. Define  $k_{upper}$  as the upper bound of  $t$  that does not satisfy  $k(x, y, p, t) \rightarrow 0$  and  $k_{lower}$  as the lower bound of  $t$  that does not satisfy  $k(x, y, p, t) \rightarrow 0$ , the currently available work on Event Spike Tensor has  $k_{upper}$  and  $k_{lower}$  as global synchronized values independent of  $x, y$ , and  $p$ .

With a dimension transformation  $S_{c,y,x} = S^{(n)}(x, y, c - p\lfloor\frac{c}{p}\rfloor, \lfloor\frac{c}{p}\rfloor)$ , the Event Spike Tensor can finally be in a form suitable for inputting to the detector.

Event Volume [22] is a typical Event Spike Tensor with  $f(x, y, p, t) = 1$  and  $k(x, y, p, t) = \delta(x, y, p) \max(0, 1 - |\frac{t}{\Delta\tau}|)$ , then its  $k_{upper} = \Delta\tau$ . Event Volume has the problem of the global synchronized time window and the global synchronized temporal resolution. The Event Volume can only consider events satisfying  $t_i \in [\tau_0^{(n)} - k_{upper}, t^{(n)}]$  at  $t^{(n)}$ , and we call this interval the global synchronized time window of the Event Volume. The global synchronized temporal resolution means that regardless of the frequency of events triggered at each spatial and polarity location at a moment, kernel convolutions will be performed at a globally consistent time interval  $\Delta\tau$ .

Since the temporal dimension will be used as the channel dimension when feeding the tensor into the object detector, the value of  $B$  should not be too large, otherwise, it may lead to the occurrence of overfitting. With constant  $B$ , the global synchronized time window length is proportional to the interval size, but a larger interval symbolizes a lower temporal resolution for sampling the event stream, implying that the time window length is inversely proportional to the tempo-

ral resolution, which means that having both a long global synchronized time window and a large global synchronized temporal resolution is not possible.

Fig.2 visualizes the problems that can result from the global synchronized time window and the global synchronized temporal resolution. It can be seen that for spatial and polarity locations recently trigger events less frequently, a short global synchronized time window cannot aggregate enough information, while a too long global synchronized time window leads to a decrease in global synchronized temporal resolution for spatial and polarity locations recently trigger events more frequently.

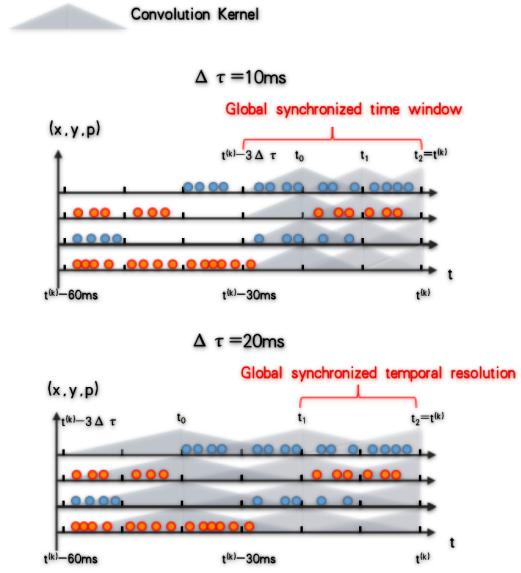


Fig. 2: The process of constructing Event Volume, which showcases the exist of global synchronized time window and global synchronized temporal resolution

Event Count Image [19], [20] is also an Event Spike Tensor. unlike Event Volume, the convolution kernel size of Event Count Image can be dynamically adjusted according to the frequency of events triggered within the recent event stream. The  $B$  of Event Count Image is usually taken as 1. The hyperparameter controlling the size of the convolution kernel is an integer  $N$ . In our implementation, we take  $f(x, y, p, t) = 0.05$  and  $k(x, y, p, t) = \begin{cases} u(x, y, p, t), & t \leq t^{(n)} - t_{i^{(n)}-N}, \\ 0, & Others \end{cases}$

where  $i^{(n)} = \arg \max_i t_i, t_i \in [0, t^{(n)}]$ , then its  $k_{upper} = t^{(n)} - t_{i^{(n)}-N}$ , which means that Event Count Image will consider the latest  $N$  events at  $t^{(n)}$ . Its global synchronized time window is  $t_i \in [t_{i^{(n)}-N}, t^{(n)}]$ .

The global synchronized time window in Event Count Image can be dynamically adjusted according to the frequency of events triggered within the recent event stream since it considers the nearest fixed-length event stream. This approach is valid if there isn't much difference between the frequency of events triggered at each spatial and polarity location.

However, in the object detection problem, we often have to detect multiple objects within the field of view at the same time. When there are objects with different motion speeds relative to the event camera in the field of view, there will be different event generation frequencies at different spatial locations simultaneously. In this case, as long as the global synchronized time window exists, it is impossible to set a reasonable time window for each spatial location.

Fig.3 further illustrates this problem. It can be seen that for spatial and polarity locations recently trigger events less frequently, too short a global synchronized time window cannot aggregate enough information, while for spatial and polarity locations recently trigger events more frequently, too long a global synchronized time window makes the convolution kernel cover excessive events at one time, resulting in a loss of information.

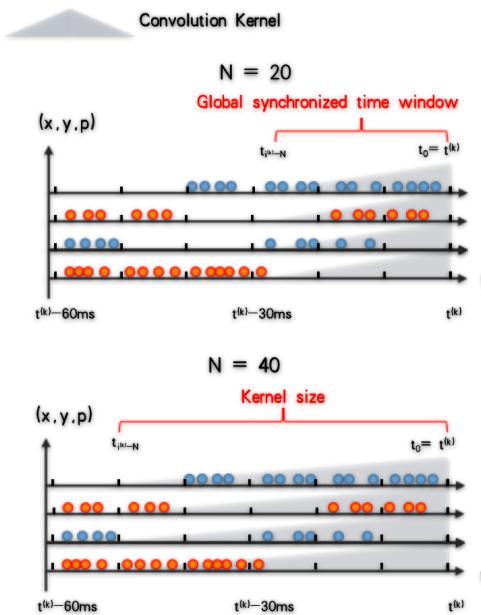


Fig. 3: The process of constructing Event Count Image, which showcases the trade-off between the kernel size and the time window

The representation method of emulating spike neural networks always ensures that the spatial locations of newly occurring events in the tensor have larger values than other positions. Surface of Active Events [20], [23] belongs to this type of method, which gives a 2D snapshot of the time surface with the latest timestamp of the event in the field of view, i.e.,  $S^{(n)}(x, y, p) = \max_{i, x_i=x, y_i=y, p_i=p} t_i$ . In practice, an exponential transforming function is applied to it to restrict its value range, i.e.  $S^{(n)}(x, y, p) = \exp[\lambda(\max_{i, x_i=x, y_i=y, p_i=p} t_i - t^{(n)})]$ , where  $\lambda$  is a hyperparameter. This gives a value range of  $(0, 1]$ , where the spatial locations of newly triggered events have larger values in the value range, while the values of the spatial locations without new events for a long time will continue to

decay, and the  $\lambda$  is the hyperparameter to controls the decay rate. By dimensional transformation  $S_{c,y,x} = S^{(n)}(x, y, c)$ , Surface of Active Events can eventually be in a form suitable for inputting to the detector.

Fig.4 shows the mapping between  $\Delta t = t - t^{(n)}$  and the value in  $S^{(n)}$  when  $\lambda$  takes different values. It can be seen that under  $\lambda = 1 \times 10^{-5}$ , when  $\Delta t$  takes  $-5 \times 10^4 \mu s$  and  $1 \times 10^4 \mu s$ , the value in  $S$  takes a larger difference, which indicates that Surface of Active Events gives the newly triggered events a larger temporal resolution. However, when the value of  $\Delta t$  is less than  $-1 \times 10^6 \mu s$ , the values in  $S$  are close to zero, which indicates that the information of the events triggered further than 1s from the current time  $t^{(n)}$  will hardly be retained. Therefore, there is also a global time window for Surface of Active Events. Define  $k_{lower}$  as the lower bound of  $t$  that does not satisfy  $e^{\lambda t} \rightarrow 0$ , then the global time window of Surface of Active Events is  $[t^{(n)} + k_{lower}, t^{(n)}]$ .

The length of the global time window can be increased by decreasing the value of  $\lambda$ , as shown in Figure.4, when  $\lambda = 1 \times 10^{-6}$ , there is still a large difference in the value of  $S$  after  $\Delta t$  takes a value smaller than  $-1 \times 10^6 \mu s$ . However, relatively, when  $\Delta t$  takes values of  $-5 \times 10^4 \mu s$  and  $1 \times 10^4 \mu s$ , the difference in the values in  $S$  decreases. If the value of  $\lambda$  is continue to reduce, the newly triggered events will be more difficult to distinguish from each other by the value in  $S$ .

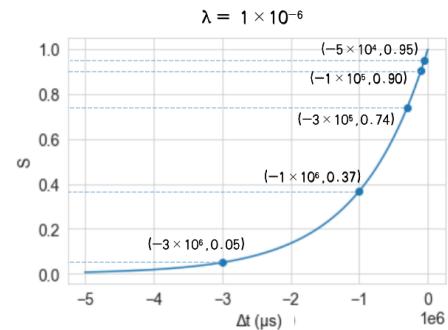
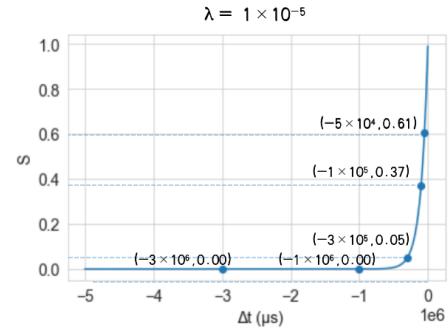


Fig. 4: The mapping between  $\Delta t$  and the value in  $S^{(n)}$  under different  $\lambda$

Column 1 in Fig.5 shows the TV-L1 optical flow plot of the scene, Columns 2 to 6 show the visualization of Event Volume, Event Count Image and Surface of Active Events

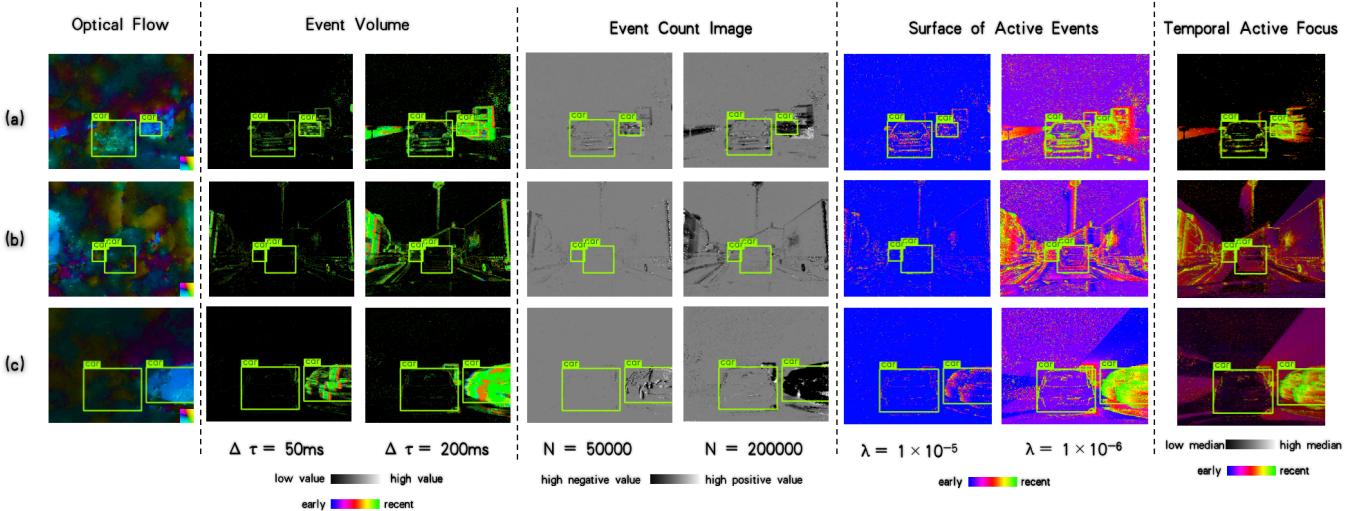


Fig. 5: Visualization of different event representation tensor with different hyperparameters. In the field of view, (a) is the case where only objects with fast motion relative to event camera are present, (b) is the case where only objects with slow motion relative to event camera are present, and (c) is the case where objects with different motion relative to event camera are present at the same time

with different hyperparameters. It can be found that for objects with slow motion relative to event camera, a long global synchronized time window is required to obtain sufficient information, but for objects with fast motion relative to event camera, an excessively long global synchronized time window instead leads to the occurrence of motion blur. When there are objects with different motion speeds relative to camera in the field of view at the same time, it is hard to find a hyperparameter that takes into account all objects.

#### IV. METHOD

This section introduces our core methods, including the high-speed lightweight detector Agile Event Detector (AED), the event representation method called Temporal Active Focus (TAF), and the Bifurcated Folding Module (BFM) for fully extracting features from the TAF representation tensor.

##### A. Agile Event Detector

The feedforward target detector in the state-of-the-art approaches are designed mainly based on the YOLO series model [6]–[8]. Also, we designed the Agile Event Detector (AED) mainly based on the YOLOX model [16], which has the characteristics of light weight and high speed. The AED model receives event representation tensors as input, and outputs the detection results. The general structure of the AED model is shown in Fig.6.

Compared to YOLOX, AED does not use the CSPDarknet [32] as the backbone network. AED’s backbone network is adapted from the Darknet21 used in YOLOv3 [25]. Compared to the CSPDarknet, the Darknet has a faster inference speed. The input layer of the backbone network accepts the event representation tensor as input. The event representation tensor  $S_{c,y,x}$  of shape  $(C \times H \times W)$  is resampled to  $(C \times H^* \times W^*)$

using nearest neighbor interpolation to match the network input size. We still retain the Focus layer in CSPDarknet as the input layer of the backbone network instead of the traditional  $7 \times 7$  convolution used in ResNet [33] and Darknet. The Focus layer downsamples the input tensor spatially while encoding the spatial information into the channels, which reduces the spatially redundant local information to some extent. Then the features are extracted using the  $3 \times 3$  convolution. After downsampling the feature map, the  $3 \times 3$  convolution actually possesses a larger field of perception than the traditional  $7 \times 7$  convolution. In this way, the number of parameters in the input layer is reduced, while the operation speed as well as the field of perception is increased.

In the traditional Darknet, the feature map output from the input layer often has a small number of channels. With the deepening of the network, the reduction of the feature map size, and the enhancement of the semantics, the number of the feature map channels will increase exponentially. In AED, however, the feature map output by the input layer has 64 channels, which is a large value for the model with the same number of parameters. This is because the event representation tensor depends on the channel dimension to encode time and polarity information, and its semantics is more complex than the channels of conventional RGB images. Therefore, more neurons need to be used in the early stage to fully fit the information in the event representation tensor. In order to maintain the lightness of the detector, we keep the number of output channels of Residual Layer 3 and 4 to 256. Reducing the number of channels in the last few layers of the backbone network can significantly reduce the number of parameters, improve the inference speed, while having a limited impact on the accuracy. We use the SiLU function used in the CSPDarknet of YOLOX as the activation function

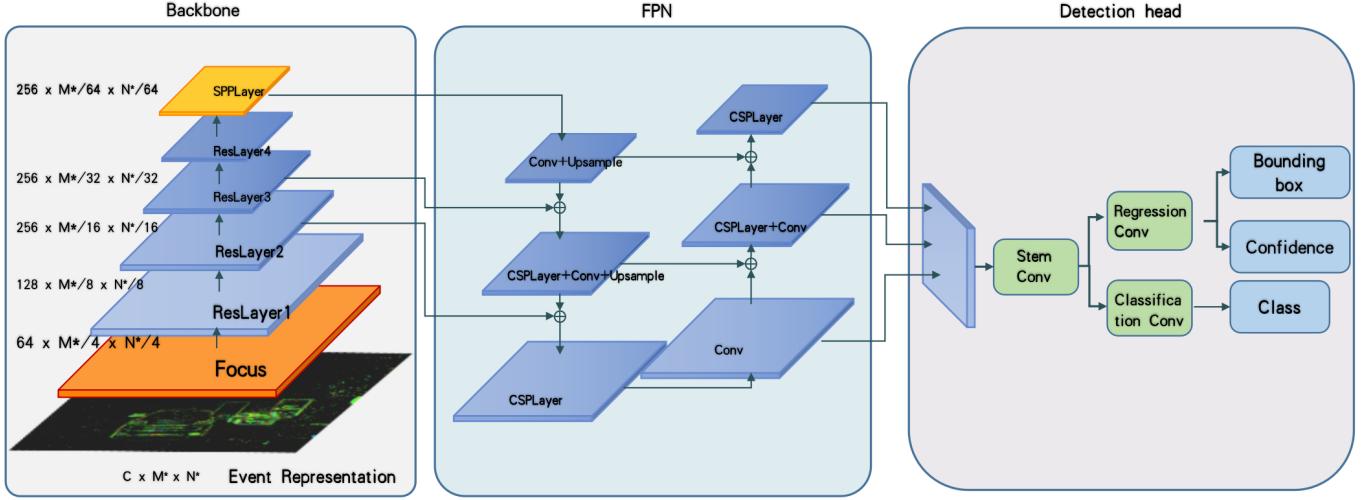


Fig. 6: The structure of the AED model

in the backbone network.

The Feature Pyramid Network (FPN) and the detection head are similar to the settings in YOLOX. The FPN mainly refers to the Path Aggregation Feature Pyramid Network (PAFPN) used in YOLOv4 [32]. The PAFPN in AED is also lighter due to the lower number of channels in the output feature map from the backbone network. The detection head network structure refers to the decoupled detection head used in YOLOX, which branches the feature maps through two convolutional layers, classification and regression, respectively. The decoupled detection head leads to better accuracy, while the increased computational effort can be compensated by the smaller number of feature map channels.

To improve the generalization capability of the lightweight detector, we also designed a simple but effective data augmentation method for event stream representation tensor data:

- 1) During training, each event representation tensor  $S$  has  $p_1$  probability to be flipped horizontally, i.e.,  $S_{c,y,x}^* = S_{c,y,W-x-1}$ .
- 2) During training, each event representation tensor  $S$  of shape  $(C \times H \times W)$  has  $p_2$  probability to be resampled to  $(C \times \lfloor \alpha H \rfloor \times \lfloor \alpha W \rfloor)$  using nearest neighbor interpolation, where  $\alpha \in [1, \infty)$ , and then randomly cropped back to  $(C \times H \times W)$ , i.e.  $S_{c,y,x}^* = S_{c,y^*:y^*+H,x^*:x^*+W}$ , where  $y^* \sim U(0, \lfloor \alpha H \rfloor - H)$ ,  $x^* \sim U(0, \lfloor \alpha W \rfloor - W)$ , and  $U(a, b)$  is the uniform distribution on  $[a, b]$ ,  $a, b \in \mathbb{R}$ .

#### B. Temporal Active Focus

Temporal Active Focus (TAF) is an event stream representation tensor that we propose, which makes full use of the asynchronous generation characteristics of the event stream data. By sampling the Event Measurement Field [17], which is continuous and of indefinite length in the temporal dimension, at different spatial and polarity locations asynchronously in time, the generated event stream representation tensor is discrete and of fixed length in the temporal dimension, but

different spatial and polarity locations all contain information of different time ranges. Moreover, the incremental update ability of the TAF tensor makes it much less computationally intensive to construct in continuous event stream detection scenarios than the state-of-the-art methods. In this subsection, we will describe the principle and the implementation of TAF respectively.

*1) Principle of Temporal Active Focus:* The timestamp  $T$  for conducting kernel convolutions in currently available works on Event Spike Tensor is a globally synchronized set of timestamps. In Temporal Active Focus, we will use an algorithm called Temporal Asynchronized Kernel Convolutions to sample events. Specifically, in Temporal Active Focus,  $T^{(n)}(x, y, p) = \{T_b^{(n)}(x, y, p)\}_{b \in \{0, 1, \dots, B-1\}}$  is a space, polarity related set, i.e., applying kernel convolution at different timestamps at different spatial, polarity positions.

Let  $T^* = \{\tau_j^* = t^{(n)} - j\Delta\tau\}_{j \in \{0, 1, \dots, B^*\}}$  for all possible values of the elements in  $T^{(n)}(x, y, p)$ , where  $B^* = \lfloor \frac{t^{(n)}}{\Delta\tau} \rfloor$ . We set a binary function  $C(x, y, p, j)$  to determine whether the kernel convolution should be applied at  $(x, y, p, \tau_j^*)$ . In order to make the generated event representation tensor have a fixed length  $B$  in the temporal dimension, for each value of  $(x, y, p)$ , the minimum  $B$  number of  $j$  satisfying  $C(x, y, p, j) > 0$  is chosen (if there are less than  $B$  number of  $j$  satisfying  $C(x, y, p, j) > 0$ , an integer  $\xi$  satisfying  $\xi\Delta\tau > T_{max}$  is used to fill the values), denoted as  $J_b^{(n)}(x, y, p)$ ,  $b \in \{0, 1, \dots, B-1\}$ , which satisfies  $J_{b-1}^{(n)}(x, y, p) > J_b^{(n)}(x, y, p), \forall (x, y, p), b \in \{1, 2, \dots, B-1\}$ .

As an exploratory attempt, we use a simple yet effective  $C(x, y, p, j)$  implementation that exploits the asynchronous and sparse nature of event stream data by applying kernel convolutions only in the time range that enables the convolution kernel to produce a valid response, i.e.,  $C(x, y, p, j) =$

$$\begin{cases} 1, \sum_{e_i} f(x_i, y_i, p_i, t_i) k(x - x_i, y - y_i, p - p_i, \tau_j^{(n)} - t_i) > 0 \\ 0, \sum_{e_i} f(x_i, y_i, p_i, t_i) k(x - x_i, y - y_i, p - p_i, \tau_j^{(n)} - t_i) = 0 \end{cases}$$

For the convolution kernel, we simply adopt the rectangular window function  $k(x, y, p, t) = \begin{cases} u(x, y, p, t), & t \leq \Delta\tau \\ 0, & \text{Others} \end{cases}$

Theoretically, as in the work of Gehrig et al. [17] and Li et al. [11] learning the measurement function and the convolution kernel directly from the original event stream,  $C(x, y, p, j)$  and  $k(x, y, p, t)$  can also be obtained using learning approach, which we take as a direction for future work.

Temporal Asynchronized Kernel Convolutions does not fix the sampling position of the Event Field, in order not to lose the absolute position information on the temporal dimension, we design a measurement based on the logarithmic transformation:  $f(x, y, p, t) = 1 - \frac{\ln[1+(t^{(n)}-t) \times 10^{-4}]}{\ln(1+T_{max})}$ , where  $t$  is calculated in microseconds. Fig.7 shows a plot of the variation of  $f(x, y, p, t)$  with  $\Delta t = t - t^{(n)}$  when  $T_{max} = 6 \times 10^7 \mu s$ . It can be seen that the values of  $f(x, y, p, t)$  on the whole  $\Delta t \in [T_{max}, 0]$  interval all reflect a high resolution: on  $\Delta t \in [-1 \times 10^6, 0]$  interval, the value of  $f(x, y, p, t)$  takes an almost vertical decline, indicating that a very high temporal resolution is assigned to the newly triggered events; even until after  $\Delta t$  takes values less than  $-3 \times 10^7 \mu s$ , the values of  $f(x, y, p, t)$  corresponding to different  $\Delta t$  still have large differences.

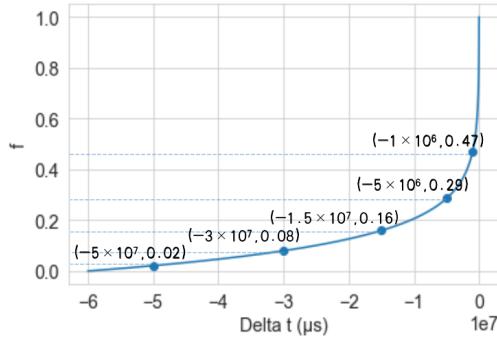


Fig. 7: The mapping between  $\Delta t$  and the value in  $f$  when  $T_{max} = 6 \times 10^7 \mu s$

Overall, Temporal Active Focus combines the characteristics of the representation methods based on discretizing Event Measurement Field and the methods emulating spike neural networks. Fig.8 visualizes the characteristics of Temporal Active Focus. It can be seen that compared with the Event Spike Tensor methods, Temporal Active Focus can adapt the interval of applying kernel convolutions according to the frequency of generating events in each period at each spatial and polarity location. A larger temporal resolution is applied for periods with a higher frequency of event generation, while a smaller temporal resolution is applied otherwise. In addition, the time interval of event streams considered at each spatial and polar location starts and ends flexibly, which means that the time window can also be adaptively adjusted asynchronously at each spatial and polarity location. Compared with the repre-

sentation method emulating spike neural networks, Temporal Active Focus has multiple samples in the temporal dimension at each spatial and polarity position. The positions and values in the temporal dimension will jointly encode information such as the elapse of the most recently triggered event at each position, the number of short-time triggered events, and the frequency of triggered events in a certain period, thus containing more information.

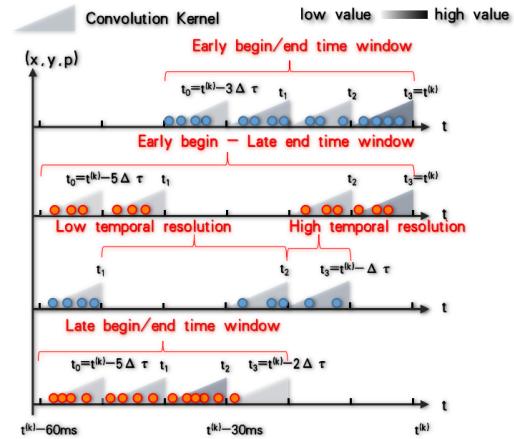


Fig. 8: The process of constructing Temporal Active Focus, which shows the flexibly adjusted time windows and temporal resolutions as well as the rich information encoded

Column 8 in Fig.5 shows the visualization of the information considered by Temporal Active Focus, where the hue is encoded using the latest value in the temporal dimension and the brightness is encoded using the median in the temporal dimension. It can be found that Temporal Active Focus can consider equal high-resolution information for objects with different motion speeds relative to the camera.

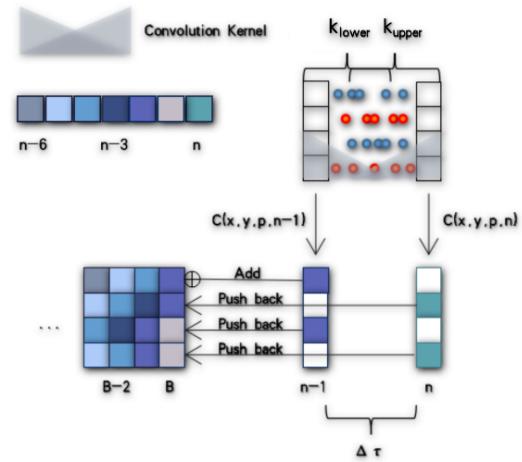


Fig. 9: The process of implementing Temporal Active Focus using FIFO queues

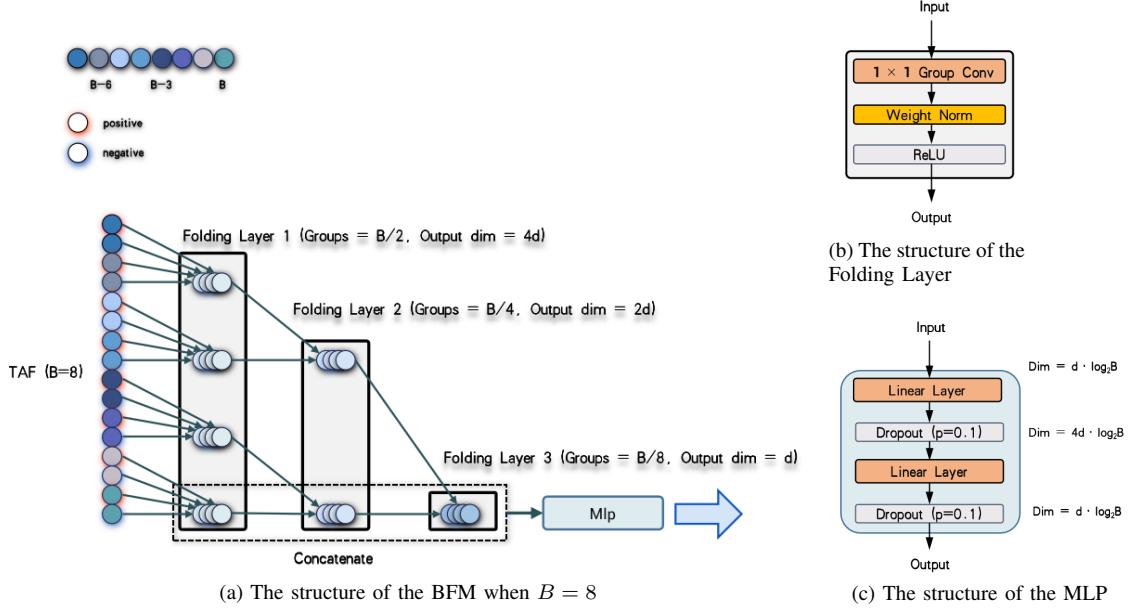


Fig. 10: Structures of the BFM and its components

2) *Implementation of Temporal Active Focus*: The normal event stream representation tensor is limited by the global time window, so only a finite length of event stream needs to be loaded at each inference to construct the event representation tensor. However, the TAF tensor is sampled from the infinite-length event stream. To optimize the effect of TAF, at each inference, it is necessary to load all events from the beginning of the event stream to the inference timestamp to construct the Event Measurement Field, which is impractical in terms of time and storage cost.

In fact, since the object detection on the event stream is performed at a certain frequency, if the detection period is set to  $\Delta\tau$ , we can greatly reduce the computational burden by using the FIFO queue, whose necessary condition is that  $k(\cdot)$  satisfies either causality or linearity.

First, a FIFO queue of length  $B$  is maintained at each  $(x, y, p)$  location. If  $k(\cdot)$  satisfies linearity, then the detection of  $t^{(n)} = n\Delta\tau$  timestamps requires only  $E^{(n)} = \{e_i\}_{t_i \in [n\Delta\tau - k_{upper}, n\Delta\tau] \cup [(n-1)\Delta\tau, (n-1)\Delta\tau - k_{lower}]}$ . For  $E^{(n)}$ , only  $S^{(n)}(x, y, p, n)$  and  $S^{(n)}(x, y, p, n-1)$  need to be computed. Then compute  $C(x, y, p, n-1)$  and  $C(x, y, p, n)$ , keeping the values in  $S^{(n)}(x, y, p, n-1)$  that satisfying  $C(x, y, p, n-1)$  and the values in  $S^{(n)}(x, y, p, n)$  that satisfying  $C(x, y, p, n)$ . If the queue head at  $(x, y, p)$  is the value in  $S^{(n-1)}(x, y, p, n-1)$ , then update the queue head to  $S^{(n-1)}(x, y, p, n-1) + S^{(n)}(x, y, p, n-1)$ . Otherwise, push  $S^{(n)}(x, y, p, n-1)$  directly into the queue at  $(x, y, p)$ . Finally, push the values kept in  $S^{(n)}(x, y, p, n)$  to the queue at  $(x, y, p)$ .

In particular, if  $k(\cdot)$  satisfies causality, i.e.,  $k_{lower} = 0$ , then it is sufficient to load  $E^{(n)} = \{e_i\}_{t_i \in [n\Delta\tau - k_{upper}, n\Delta\tau]}$  to compute  $S^{(n)}(x, y, p, n)$  for  $E^{(n)}$ . Then compute  $C(x, y, p, n)$ , keeping the values in  $S^{(n)}(x, y, p, n)$  that satisfy  $C(x, y, p, n)$

and pushing them to the queue at  $(x, y, p)$ . Our implementation takes a causal convolution kernel and thus falls into this case.

The process of implementing Temporal Active Focus using FIFO queues is visualized in Fig.9.

### C. Bifurcated Folding Module

The input layer of conventional detectors is often a convolutional layer, which not only extracts information in the channel but also models spatial dependencies to extract spatial information. It works well when using the normal RGB image as the input since the semantics in channels are often weak. However, the TAF tensor has very rich information in the temporal dimension, and it is important to note that the TAF tensor also belongs to a kind of Event Spike Tensor, so before being fed into the detector, its temporal dimension will be transformed to the channel dimension first by the dimensional transformation. Therefore, the semantics of the TAF tensor in channels is much stronger than the normal RGB image. Bearing this in mind, we designed a module called Bifurcated Folding Module (BFM) to fully extract information in the temporal dimension of the TAF tensor before being fed into the detector, which is point-wisely applied at each spatial location.

The design of the BFM follows two main concepts: first, gradually aggregating values at adjacent times; second, assigning greater importance to information from events that are triggered more recently. Following the first concept, we design the Folding Layer, whose structure is shown in Fig.10(b). the Folding Layer gradually reduces the number of input channels using the  $1 \times 1$  depthwise convolution, whose number of output channels is smaller than the number of input channels. Since the temporal dimension is transformed to the channel dimension by dimensional transformation, the  $1 \times 1$  depthwise convolution is actually a local connection

TABLE I: The statistics of the annotations in the GEN1 Dataset

Set	Size(GB)	Annotations	Car	Pedestrain
Train	473	148,432	131,593	16,839
Validation	122	40,732	37,909	2,823
Test	143	66,304	58,512	7,792
Total	738	255,468	228,014	27,454

TABLE II: The statistics of the annotations in the 1 MEGAPIXEL Dataset (Subset)

Set	Size(GB)	Annotations	Pedestrain	Two wheeler	Car	Truck	Bus	Traffic sign	Traffic light
Train	3,327	390,924	85,932	12,626	192,173	21,040	5,038	37,057	37,058
Validation	657	68,921	12,529	2,082	37,906	4,021	1,265	6,316	4,802
Test	669	72,873	19,403	2,698	32,257	3,310	1,438	6,902	6,865
Total	4,653	532,718	117,864	17,406	262,336	28,371	7,741	50,275	48,725

of adjacent times. We also use Weight Normalization [34] to make the Folding Layer converge better during training while playing a similar role to Batch Normalization during inference. Following the second idea, we are inspired by the Cross-Stage-Partial-connections (CSP) [32] mechanism. After each temporal aggregation, we make slicing in channel dimensions to get the channels encoding temporally latest information, and finally connect all sliced channels together. This will make the output channels contain more information about recent events, and also act as a residual connection to a certain extent, making the module easier to converge during training. We finally fuse the output channels with a Multi-Layer Perceptron (MLP), the structure of which is shown in Fig.10(c).

The overall structure of the Bifurcated Folding Module is shown in Fig.10(a). For a TAF tensor with temporal dimension length  $B$ , we set  $\log_2 B$  Folding Layers. The first Folding Layer will use  $B/2$  convolution groups to reduce the number of channels from  $2B$  to  $2^{\log_2 B-1}d$ , where  $d$  is a hyperparameter. Then the  $i, i \in \{2, \dots, \log_2 B\}$ th Folding Layer will use  $B/2^i$  convolution groups to reduce the number of channels from  $2^{\log_2 B-i+1}d$  to  $2^{\log_2 B-i}d$ . The channels  $c \in \{2^{\log_2 B-i}d - d, 2^{\log_2 B-i}d - d + 1, \dots, 2^{\log_2 B-i}d - 1\}$  of the output of  $i$ th Folding Layer will be taken and connected altogether in the channel dimension, serving as the input of the Multi-Layer Perceptron (MLP).

## V. EXPERIMENTAL RESULTS

In this section, we present the results of our experiments. We will present our experimental settings, including details of the dataset used for the experiment, the selection of hyperparameters, and the evaluation method, then present the results of comparing our method with state-of-the-art methods, and finally illustrate the effectiveness of each component of our method through ablation studies.

### A. Experiment Settings

1) *Dataset*: We choose the Prophesee GEN1 Automotive Detection Dataset (GEN1 Dataset) [18] and the Prophesee 1 MEGAPIXEL Automotive Detection Dataset (1 MEGAPIXEL Dataset) [10] to conduct the experiments. The 1 MEGAPIXEL Dataset and the GEN1 Dataset are two typical event camera object detection datasets for real scenarios, both of which

focus on road object detection in traffic scenarios. When collecting the datasets, the event camera was placed behind the front windshield of a moving vehicle to collect the data from the road ahead. Each sample is a video stream of up to 60 seconds captured with the event camera and stored as the event stream, with each event recording the x-coordinate and y-coordinate of the trigger location, as well as the polarity and timestamp. The timestamps are in microsecond resolution, and the video stream start time is 0 in each sample, increasing up to the duration of the video stream  $T_{max} = 6 \times 10^7 \mu s$ . Each sample is accompanied by a separate file recording all the annotations, and the information recorded in each annotation includes the timestamp, upper left  $x$  coordinate of the bounding box, upper left  $y$  coordinate of the bounding box, width and height of the bounding box, class, confidence level, and track number.

The 1 MEGAPIXEL Dataset is a dataset of 4.65 TB in size and is the largest event camera object detection dataset to date. The picture size of the event camera used is of 720 pixels high and 1280 pixels wide. The annotation classes including pedestrians, bicycles, cars, trucks, buses, traffic signs, traffic signals, a total of more than twenty-five million annotations with labeling frequency of 60 Hz. Compared to the 1 MEGAPIXEL Dataset, the GEN1 Dataset has a smaller size of 738G, whose resolution is lower, using a event camera with picture size of 240 pixels high and 304 pixels wide. The number of annotation classes in the GEN1 Dataset is smaller, including only pedestrians and cars. It also has fewer annotations, including only 255,781 annotations, and the labeling frequency is only 1-4Hz.

Due to the limitation of time and computational resources, our experiments are conducted on the complete GEN1 Dataset and a downsampled 1 MEGAPIXEL Dataset (called 1 MEGAPIXEL Dataset (Subset)) by reducing the annotation frequency to 1 Hz while preserving the complete event stream data. This downsampling approach preserves as much diversity of scenes and objects as possible, thus preserving the quality of the dataset. However, since the number of samples used for training is only 1/60 of the original dataset, the performance of the trained model may be degraded. The detailed statistics of the two datasets are shown in TABLE I and II.

2) *Implementation Details*: All methods based on YOLOX and AED models are trained using Adam optimizer and cosine learning rate with 5 preheat epochs [16]. In the preheat epochs, the learning rate will grow from 0 to  $2.1 \times 10^{-4} \times \text{Batchsize}$ , and then decreases. The Batchsize is taken as 30 on GEN1 Dataset, training for 35 epochs. The Batchsize is taken as 16 on 1 MEGAPIXEL Dataset (Subset), training for 50 epochs.

For the GEN1 Dataset, the input layer is taken as  $H^* = 256$  and  $W^* = 320$ . For 1 MEGAPIXEL Dataset (Subset), the input layer is taken as  $H^* = 512$  and  $W^* = 640$ . For the data augmentation, we take  $p_1 = 0.5$ ,  $p_2 = 0.5$ ,  $\alpha = 1.5$ . For all Event Volume event representation tensors we take  $B = 5$  and  $\Delta\tau = 50ms$ . For the TAF event representation tensor, we set the hyperparameter  $B = 4$  on the GEN1 Dataset and  $B = 8$  on the 1 MEGAPIXEL Dataset (Subset), and  $\Delta\tau = 10ms$  on the both dataset. For the BFM module, we set the hyperparameter  $d = 4$ .

In addition, we compute the TV-L1 dense optical flow in the GEN1 Dataset and 1 MEGAPIXEL Dataset (Subset) for all timestamps where the annotations are present. The difference between adjacent timestamps which is considered a hyperparameter to calculate the optical flow in the work of Nagata et al. [30] is set to  $50ms$ .

3) *Evaluation Method*: We select the trained models that perform best on the validation set and apply them to the test set. The performance metrics include accuracy, run time, and the number of parameters.

The accuracy evaluation protocol is consistent with the evaluation protocol provided by the 1 MEGAPIXEL Dataset [10]. The target detection algorithm needs to output a timestamp while generating the detection result. Since the event stream data has annotations with timestamps, then, based on a predefined tolerance, the detection results can be matched against the annotations within the tolerance based on their timestamps, thus equating the evaluation to evaluate the frame-based detection algorithm. Specifically, the mAP (0.50:0.95) in the COCO protocol [35] is used as the accuracy metric. For algorithms that use the TAF tensor as input, we set the detection period equal to  $\Delta\tau$  and the tolerance to  $\Delta\tau/2$ ; for algorithms that use other event representation methods, we specify detection at the timestamp where the annotations appear. In addition, we also follow the same procedure of results filtering. On the GEN1 Dataset, we filter the annotations and detection results whose bounding box with diagonal less than 30 pixels and height or width less than 10 pixels. On the 1 MEGAPIXEL Dataset (Subset), we filter the annotations and detection results whose bounding box with diagonal less than 60 pixels and height or width less than 20 pixels. We also filter annotations and detection results with timestamps less than 500ms.

For timestamp  $t_j$  with an annotation  $b_j^*$  presents, for a camera with a picture size of  $H$  in height and  $W$  in width, let the estimated horizontal optical flow be  $u_{t_j}(x, y)$  and the vertical optical flow be  $v_{t_j}(x, y)$ , where  $x \in \{0, 1, \dots, W - 1\}$ ,  $y \in \{0, 1, \dots, H - 1\}$ , then we have the optical flow intensity denoted as:

$$\|V(x, y)\|_{t_j} = \sqrt{u_{t_j}^2(x, y) + v_{t_j}^2(x, y)} \quad (5)$$

The we can define a metric called Bounding Box Optical Flow Density:

$$\|V(x, y)\|_j = \frac{\sum_{x=x_j}^{x_j+w_j-1} \sum_{y=y_j-h_j+1}^{y_j} \|V(x, y)\|_{t_j}}{w_j \times h_j} \quad (6)$$

which can be a effective metric for the motion speed relative to the event camera of the object corresponding to the annotation  $b_j^*$ . The larger the  $\|V(x, y)\|_j$ , the faster its corresponding object relative to the the event camera at  $t_j$ . We set 5 intervals for the Bounding Box Optical Flow Density of all the annotations in each of the two datasets at each 20% quantile, which can classify the annotations and detection results into 5 classes according to their corresponding objects' speed relative to the camera. We call the 5 classes the 5 motion levels, which are numbered from 1-5, indicating the speed relative to the camera from low to high. We evaluate the annotations and detection results under each of the five classes. To make the metric accurate, we filtered out all overlapping bounding boxes and cropped all bounding boxes into the camera picture size.

All methods in our work were trained on a server with a single GeForce RTX 3090 GPU and an 8-core Intel(R) Core(TM) i7-7820X CPU @ 3.60GHz. We test all methods on a server with a single Titan Xp GPU and a 16-core Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz.

## B. Comparison with the State-of-the-art

We compare our method with the state-of-the-art methods. We first compare our overall method with other existing works in terms of accuracy, model inference speed, and model parameters. Then, for the event representation methods, we conduct a separate comparison experiment, using AED as the detector, comparing the accuracy and the speed of TAF with other methods.

1) *Comparison with the State-of-the-art methods* : TABLE III shows the results of comparing our method with the state-of-the-art methods. It can be seen that among the methods using feedforward detectors without the memory mechanism [6], [7], [9], [12], [13], our method achieves the best performance in terms of accuracy and speed at the same time. Compared with the NGA-events method using Event Volume as the event representation method and YOLOv3 as the detector, our method using the AED detector needs only 24.1% of parameters but achieves 9.5% mAP improvement and model inference time reduction on the GEN1 Dataset, while 9.6% mAP improvement and inference time reduction on the 1 MEGAPIXEL Dataset (Subset). ASTMNet [11] is currently the most advanced method on GEN1 Dataset, which uses a recurrent network structure and thus makes the network memorable. Comparing our method with ASTMNet, the mAP is 1.3% lower on GEN1 Dataset, but the model inference speed is improved and the number of model parameters is only

TABLE III: Performance comparison with the State-of-the-art methods

Method	Event Representation	Detector	Memory	GEN1 Dataset		1 MEGAPIXEL Dataset (Subset)		Params(M)
				mAP	Inference Time(ms)	mAP	Inference Time(ms)	
Chen et al. [6]	Event Frame	YOLO	-	0.322	21.47	-	-	45.3
Jiang et al. [12]	Event Frame	YOLOv3	-	0.326	22.34	0.207	15.53	61.5
JDF-events [13]	Two Polarities Event Frame	YOLOv3	-	0.332	22.34	0.224	15.53	61.5
NGA-events [7]	Event Volume	YOLOv3	-	0.359	26.11	0.248	15.84	61.5
Sparse-conv [9]	Raw Events	YOLO	-	0.145	-	-	-	-
RED [10]	Event Volume	SSD	ConvLSTM	0.400	-	-	-	24.1
ASTMNet [11]	Raw Events	SSD	Rec-conv	<b>0.467</b>	35.61	-	-	39.6
Our baseline	Event Volume	YOLOX	-	0.350	<u>9.84</u>	0.213	<u>12.06</u>	<b>14.4</b>
Ours	Temporal Active Focus	AED	-	<u>0.454</u>	<b>8.94</b>	<b>0.344</b>	<b>9.97</b>	<u>14.8</u>

TABLE IV: Performance comparison with the State-of-the-art event representation methods

Method	$\Delta\tau(ms)$	N	$\lambda$	GEN1 Dataset		1 MEGAPIXEL Dataset (Subset)	
				mAP	Representation Time(ms)	mAP	Representation Time(ms)
Event Volume	50	-	-	0.426	1.94	0.299	5.82
	100			0.424	2.09	<u>0.305</u>	8.31
	200			0.422	2.44	0.290	10.60
Event Count Image	$5 \times 10^4$	-	-	0.386	<b>0.67</b>	-	-
				0.376	<u>0.75</u>	-	-
				0.368	0.89	-	-
				-	-	0.276	<b>0.98</b>
				-	-	0.268	<u>1.00</u>
Surface of Active Events	$1.2 \times 10^6$	-	$1 \times 10^{-5}$	-	-	0.278	1.11
				0.400	0.74	0.294	2.77
				0.404	0.75	0.296	3.20
Temporal Active Focus	10	-	-	0.403	0.79	0.303	3.32
				<b>0.454</b>	1.07	<b>0.344</b>	2.11

37.4%. It can be seen that our method is competitive among existing methods when considering the metrics of accuracy, speed, and lightness together.

It can also be seen that compared to our baseline method, with only a 2.8% increase in the number of model parameters, our method reduces model inference time by 9.2% while improving mAP by 10.4% on the GEN1 Dataset, reduces model inference time by 17.3% while improving mAP by 13.1% on the 1 MEGAPIXEL Dataset. This illustrates the effectiveness of our proposed method. We will further demonstrate the effectiveness of each component in the subsequent ablation studies.

2) *Comparison with the State-of-the-art event representation methods:* Under the condition of using AED as the detector with data augmentation, our TAF method is compared with three state-of-the-art event representation methods, i.e. Event Volume [22], Event Count Image [19], [20], and Surface of Active Events [20], [23]. We set three sets of hyperparameters for each of the three event representation methods, resulting in global synchronized time windows of different lengths.

TABLE IV shows the results of the comparison experiment. It can be seen that, overall, using Event Volume achieves better accuracy than using Event Count Image and Surface of Active Events on both datasets, but it also takes the longest time to

build the event representation tensor. In contrast, using Event Count Image achieves the worst accuracy but takes the shortest time to build the event representation tensor. According to this result, our TAF method is highly competitive in terms of accuracy and speed. In terms of accuracy, the TAF method achieves the best accuracy on both datasets. Compared to the Event Volume with the best parameters, using TAF results in a 2.8% improvement in mAP on the GEN1 Dataset and a 3.9% improvement on the 1 MEGAPIXEL Dataset (Subset). In terms of speed, TAF is slower than Event Count Image and Surface of Active Events on GEN1 Dataset, but faster than Event Volume; slower than Event Count Image on 1 MEGAPIXEL Dataset (Subset), but faster than Event Volume and Surface of Active Events. The high running speed of TAF is due to the incremental update mechanism based on the FIFO queue, and the high accuracy is due to its breakthrough in the time window and temporal resolution. We will further illustrate this in detail with the accuracy results refined to 5 motion levels and the qualitative analysis results.

Fig.11 and 12 show the comparison of accuracy between different event representation methods with different parameters at 5 motion levels in the GEN1 Dataset and the 1 MEGAPIXEL Dataset (Subset) respectively. In general, no matter what event representation method is used, the detection

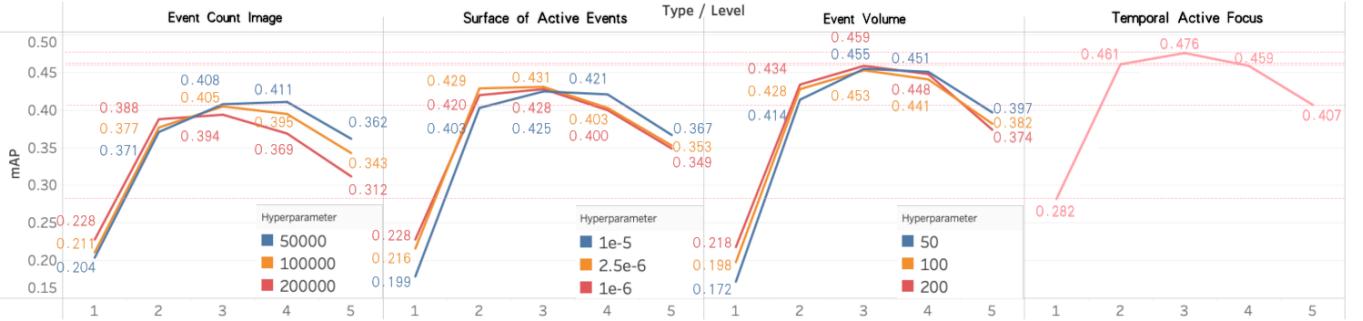


Fig. 11: Comparison of accuracy between different event representation methods with different parameters at 5 motion levels in GEN1 Dataset

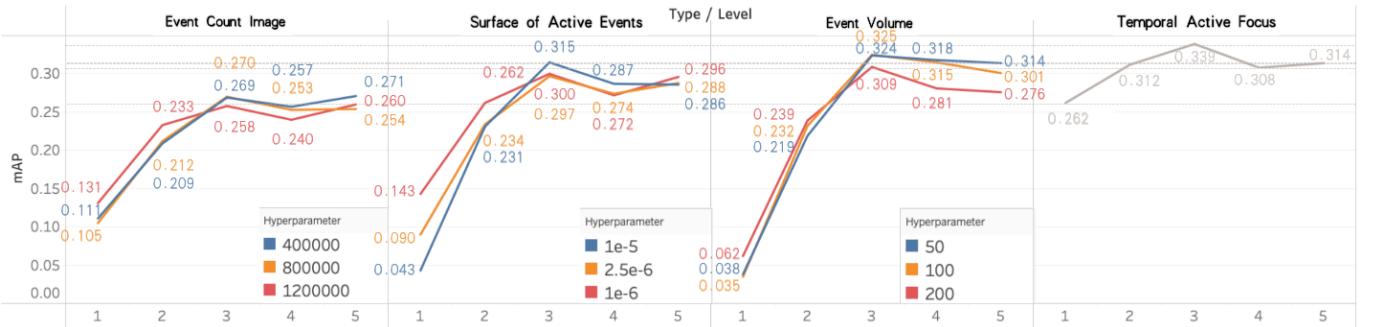


Fig. 12: Comparison of accuracy between different event representation methods with different parameters at 5 motion levels in 1 MEGAPIXEL Dataset (Subset)

accuracy of the detector for the object changes in an inverse parabola with the increase of the motion speed of the object relative to the camera, which means that the detector cannot achieve good results for the object with extremely fast or slow motion speed relative to the camera.

When using Event Count Image, Surface of Active Events and Event Volume as the event representation methods, the detection accuracy for objects moving slowly relative to the event camera is higher when using longer global synchronized time windows, while the accuracy is gradually surpassed by methods using shorter global synchronized time windows as the speed of objects moving relative to the camera increases. This demonstrates the tradeoff that exists in these event representation methods.

However, the breakthrough in the time window and temporal resolution makes the TAF method achieves high detection accuracy at all motion levels in an astonishingly parameter-free manner. The TAF method achieves the best detection accuracy for all 5 motion levels in the GEN1 Dataset and 4 motion levels in the 1 MEGAPIXEL Dataset (Subset). Especially when detecting objects in slow motion relative to the event camera, the accuracy improvement achieved by using the TAF method is groundbreaking. On the GEN1 Dataset, the mAP results obtained by the TAF method are 5.4%, 2.7%, 1.7%, 0.8%, 1.0% higher than the other results with the best accuracy at each motion level from 1 to 5, respectively. On the 1 MEGAPIXEL Dataset (Subset), except for the accuracy result

at motion level 4, which is 1.0% lower than the other results with the best accuracy, and at motion level 5, which is equal to the other results with the best accuracy, the mAP results are 11.9%, 5.0%, and 1.4% higher than the other results with the best accuracy in the remaining motion levels 1, 2, and 3, respectively.

The visualization in Fig.13 further demonstrates the robustness of the TAF method. In case (a), the first object on the left has a high motion speed relative to the camera, so when Event Volume is taken as  $\Delta\tau = 200ms$  and Surface of Active Events is taken as  $\lambda = 1 \times 10^{-6}$ , both cannot detect the object due to the motion blur. Although the object can be detected by Event Count Image under both parameters, the estimation of the height is inaccurate. In case (d), the two objects on the right side have high motion speed relative to the camera, and also due to the motion blur, the estimation of the size is inaccurate when Event Volume is taken as  $\Delta\tau = 200ms$ , and the localization is inaccurate when Event Count Image is taken as  $N = 200000$ , while in Surface of Active Events taking  $\lambda = 1 \times 10^{-6}$  the object is not detectable. On the other hand, in both cases (b)(c)(d), there are objects with low motion speed relative to the camera. It can be seen that when Event Volume is  $\Delta\tau = 50ms$  and Surface of Active Events is  $\lambda = 1 \times 10^{-5}$ , the first object on the left in case (b) and the second object on the left in case (c) are not detected, and the size estimation of the first object on the left in case (d) is not accurate. When Event Count Image takes  $N = 50000$ , the second object from

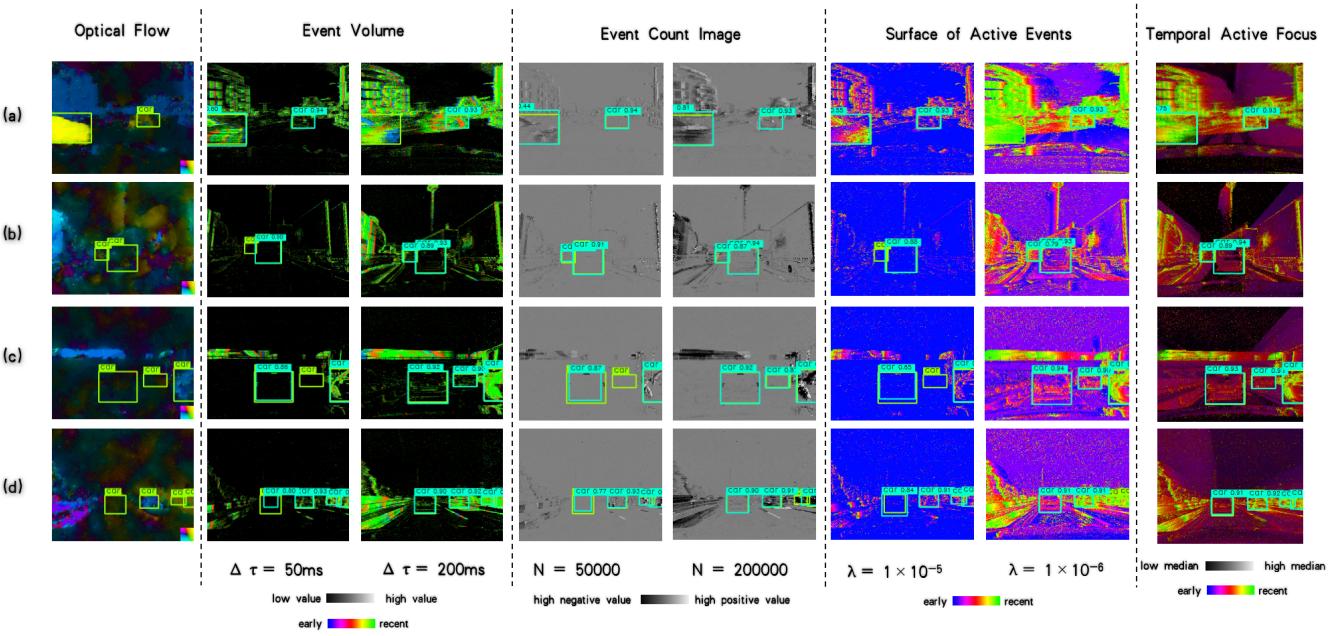


Fig. 13: Qualitative analysis results. The green bounding boxes indicate the annotation, while the cyan bounding boxes indicate the detection results

TABLE V: The performance of our methods w/o specific components

Detector	Data Augmentation	Event Representation	GEN1 Dataset		1 MEGAPIXEL Dataset (Subset)		Params(M)
			mAP	Runtime(ms)	mAP	Runtime(ms)	
YOLOX	-	Event Volume	0.350	11.78	0.213	17.88	<b>14.4</b>
YOLOX	✓	Event Volume	0.410	11.78	0.269	17.88	<b>14.4</b>
AED	✓	Event Volume	0.426	10.58	0.299	15.93	14.8
AED	✓	TAF	<b>0.454</b>	<b>10.01</b>	<b>0.344</b>	<b>12.08</b>	14.8

the left in case (c) is not detected and the size of the first object from the left in case (d) is not estimated accurately. In contrast, the TAF method can detect all the objects mentioned above while estimating the size accurately.

### C. Ablation Study

In the ablation experiments, we focus on the effectiveness of the different components of our method and the factors that affect the performance of TAF.

1) *Components*: The ablation study results of components are shown in TABLE V, where the runtime is the sum of the representation time and the model inference time. Experiments show that our proposed data augmentation method can substantially improve the accuracy of the baseline detector: mAP improves by 6% on the GEN1 Dataset and by 5.6% on the 1 MEGAPIXEL Dataset (Subset). With only a 2.8% increase in the number of parameters compared to YOLOX, the AED improves mAP by 1.6% and reduces runtime by 12.2% on the GEN1 Dataset, improves mAP by 3% and reduces runtime by 16.2% on the 1 MEGAPIXEL Dataset (Subset). Using the TAF method instead of the Event Volume will result in a 2.6% improvement in mAP with a 5.4% reduction in runtime on the

GEN1 Dataset, and a 4.5% improvement in mAP with a 31.9% reduction in runtime on the 1 MEGAPIXEL Dataset (Subset).

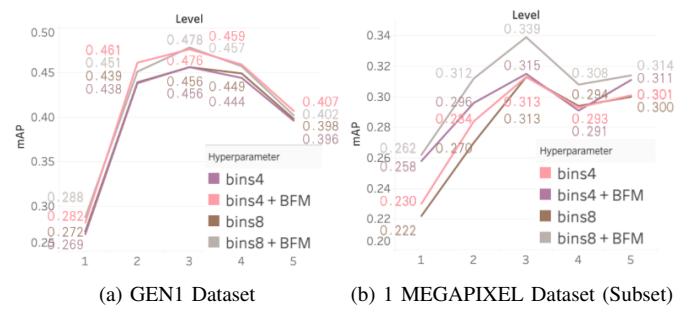


Fig. 14: Comparison of accuracy between TAF with different settings at 5 motion levels

2) *Temporal Active Focus*: TABLE VI shows the performance of the TAF under different settings, while Fig.14 shows the comparison results of accuracy at 5 motion levels. Overall, the use of BFM for pre-extracting features in TAF will result in effective accuracy improvement at all five motion levels on both datasets. And on the 1 MEGAPIXEL Dataset (Subset),

TABLE VI: The performance of the TAF under different settings

$B$	BFM	GEN1 Dataset			1 MEGAPIXEL Dataset (Subset)		
		mAP	Representation Time(ms)	Inference Time(ms)	mAP	Representation Time(ms)	Inference Time(ms)
4	-	0.444	<b>1.07</b>	<b>8.55</b>	0.326	<b>1.31</b>	<b>8.71</b>
4	✓	<b>0.454</b>	<b>1.07</b>	8.94	<u>0.333</u>	<b>1.31</b>	9.58
8	-	0.445	1.31	<b>8.71</b>	0.323	2.11	<b>9.24</b>
8	✓	<u>0.451</u>	1.31	9.15	<b>0.344</b>	2.11	9.97

we can see a higher performance improvement for detecting objects that are slow relative to the camera. However, the running speed is slightly reduced because feature extraction needs to be performed point by point in space. On GEN1 Dataset, a value of 4 or 8 for  $B$  has little impact on accuracy, while it takes more time to maintain the queue when the value is 8. Therefore,  $B = 4$  is a better hyperparameter on GEN1 Dataset. On the 1 MEGAPIXEL Dataset (Subset), when BFM is not added, the accuracy is slightly less when  $B = 8$  than when  $B = 4$ , but when BFM is added, the accuracy is much greater when  $B = 8$ . This further illustrates the effectiveness of the BFM module: The 1 MEGAPIXEL Dataset (Subset) has a higher resolution, and the camera used is more sensitive to changes in illumination, requiring larger  $B$  values to aggregate events over a larger time range in order to fully retain event information. However, larger  $B$  values bring richer semantics to the temporal dimension, which makes it difficult for the model to extract this information by convolutional neural networks alone, resulting in a degradation of performance. The BFM can extract the rich semantics of the temporal dimension when  $B = 8$  as much as possible, thus fully exploiting the effect of the TAF method.

## VI. DISCUSSIONS AND FUTURE WORKS

In our work, address to the problem that most of the existing deep learning object detection methods rely on large parameter neural networks, which implies a large computational burden and low inference speed, we designed a high-speed lightweight detector called Agile Event Detector (AED), with a simple but effective data augmentation method. Then address to the problem that existing methods cannot take into account objects with different velocities relative to the motion of the event camera, we proposed an event stream representation tensor called Temporal Active Focus (TAF), which takes full advantage of the asynchronous generation of event stream data, thus robust to the motion of moving objects while can be constructed without much time-consuming. We further proposed a module called the Bifurcated Folding Module (BFM) to extract the rich temporal information in the TAF tensor at the input layer of the AED detector.

We conducted our experiments on two typical real-scene event camera object detection datasets: the complete Prophesee GEN1 Automotive Detection Dataset and the Prophesee 1 MEGAPIXEL Automotive Detection Dataset with partial annotation. Experiments show that our method is competitive in terms of accuracy, speed, and the number of parameters

simultaneously. Also by classifying the objects into multiple motion levels based on the optical flow density metric, we illustrated the robustness of our method for objects with different velocities relative to the camera.

However, our work still leaves something to be desired. And in future work, we will continue to explore more possibilities of our existing work. First, as we mentioned, the execution position of the Temporal Asynchronized Kernel Convolutions required to implement the TAF can be learned directly from the event stream, and in addition, we will try to have the convolutional kernels learned directly, which will reduce the workload of manual hyperparameters tuning. Second, our work uses feedforward detectors, which cannot apply a priori information from previous detection results when performing continuous detection on event streams. Existing models using memory mechanisms do not consider the characteristics of event generation. Since they implement recurrent detection by means of learning using large parametric number models, the models are bulky. Therefore, we will also consider the design of a lightweight memory module to implement recurrent detection. Finally, if time and computational resources allow, we will also try to conduct experiments on the full 1 MEGAPIXEL Automotive Detection Dataset and more other event camera object detection datasets to fully demonstrate the effectiveness of our methods.

## REFERENCES

- [1] T. Serrano-Gotarredona and B. Linares-Barranco, “A  $128 \times 128$  1.5% contrast sensitivity 0.9% fpn  $3 \mu\text{s}$  latency 4 mw asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers,” *IEEE Journal of Solid-State Circuits*, vol. 48, no. 3, pp. 827–838, 2013.
- [2] P. Lichtsteiner, C. Posch, and T. Delbrück, “A  $128 \times 128$  120 db  $15 \mu\text{s}$  latency asynchronous temporal contrast vision sensor,” *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.
- [3] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbrück, “Retinomorphic event-based vision sensors: bioinspired cameras with spiking output,” *Proceedings of the IEEE*, vol. 102, no. 10, pp. 1470–1484, 2014.
- [4] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis *et al.*, “Event-based vision: A survey,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 1, pp. 154–180, 2020.
- [5] T. Finateu, A. Niwa, D. Matolin, K. Tsuchimoto, A. Mascheroni, E. Reynaud, P. Mostafalu, F. Brady, L. Chotard, F. LeGoff *et al.*, “5.10 a  $1280 \times 720$  back-illuminated stacked temporal contrast event-based vision sensor with  $4.86 \mu\text{m}$  pixels, 1.066 geps readout, programmable event-rate controller and compressive data-formatting pipeline,” in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2020, pp. 112–114.
- [6] N. F. Chen, “Pseudo-labels for supervised learning on dynamic vision sensor data, applied to object detection under ego-motion,” in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 644–653.

- [7] Y. Hu, T. Delbruck, and S.-C. Liu, "Learning to exploit multiple vision modalities by using grafted networks," in *European Conference on Computer Vision*. Springer, 2020, pp. 85–101.
- [8] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, "Asynchronous convolutional networks for object detection in neuromorphic cameras," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.
- [9] N. Messikommer, D. Gehrig, A. Loquercio, and D. Scaramuzza, "Event-based asynchronous sparse convolutional networks," in *European Conference on Computer Vision*. Springer, 2020, pp. 415–431.
- [10] E. Perot, P. de Tournemire, D. Nitti, J. Masci, and A. Sironi, "Learning to detect objects with a 1 megapixel event camera," vol. 33, 2020, pp. 16 639–16 652.
- [11] J. Li, J. Li, L. Zhu, X. Xiang, T. Huang, and Y. Tian, "Asynchronous spatio-temporal memory network for continuous event-based object detection," *IEEE Transactions on Image Processing*, vol. 31, pp. 2975–2987, 2022.
- [12] Z. Jiang, P. Xia, K. Huang, W. Stechele, G. Chen, Z. Bing, and A. Knoll, "Mixed frame-/event-driven fast pedestrian detection," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8332–8338.
- [13] J. Li, S. Dong, Z. Yu, Y. Tian, and T. Huang, "Event-based vision enhanced: A joint detection framework in autonomous driving," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2019, pp. 1396–1401.
- [14] H. Cao, G. Chen, J. Xia, G. Zhuang, and A. Knoll, "Fusion-based feature attention gate component for vehicle detection based on event camera," *IEEE Sensors Journal*, vol. 21, no. 21, pp. 24 540–24 548, 2021.
- [15] M. Liu, N. Qi, Y. Shi, and B. Yin, "An attention fusion network for event-based vehicle object detection," in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 3363–3367.
- [16] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.
- [17] D. Gehrig, A. Loquercio, K. G. Derpanis, and D. Scaramuzza, "End-to-end learning of representations for asynchronous event-based data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5633–5643.
- [18] P. de Tournemire, D. Nitti, E. Perot, D. Migliore, and A. Sironi, "A large scale event-based detection dataset for automotive," *arXiv preprint arXiv:2001.08499*, 2020.
- [19] A. I. Maqueda, A. Loquercio, G. Gallego, N. García, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5419–5427.
- [20] 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.
- [21] H. Rebecq, T. Horstschafer, and D. Scaramuzza, "Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization," 2017.
- [22] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "Unsupervised event-based learning of optical flow, depth, and egomotion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 989–997.
- [23] 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.
- [24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
- [25] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.
- [26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [27] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.
- [28] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *Advances in neural information processing systems*, vol. 28, 2015.
- [29] H. Lee, H. Kwon, R. M. Robinson, W. D. Nothwang, and A. M. Marathe, "Dynamic belief fusion for object detection," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016, pp. 1–9.
- [30] J. Nagata, Y. Sekikawa, and Y. Aoki, "Optical flow estimation by matching time surface with event-based cameras," *Sensors*, vol. 21, no. 4, p. 1150, 2021.
- [31] A. Chambolle and T. Pock, "A first-order primal-dual algorithm for convex problems with applications to imaging," *Journal of mathematical imaging and vision*, vol. 40, no. 1, pp. 120–145, 2011.
- [32] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.
- [33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [34] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," *Advances in neural information processing systems*, vol. 29, 2016.
- [35] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.