MDOE: A Spatiotemporal Event Representation Considering the Magnitude and Density of Events

Fuqiang Gu, Yong Lee, Yuan Zhuang, You Li, Jingbin Liu, Fangwen Yu, Ruiyuan Li, Chao Chen

***Abstract*—Event-based sensors (e.g., DVS cameras) are capable of higher dynamic range, higher temporal resolution, lower time latency, and better power efficiency compared to conventional devices (e.g., RGB cameras). However, learning from these sensors remains challenging; event-based sensors output a stream of asynchronous events, which cannot be directly used by state-of-the-art convolutional neural networks (CNNs). In this paper, we present a novel event-based representation called MDOE that considers both the magnitude and density of events. Compared to existing representations, which discard one or more types of information about event polarity, temporal information, and/or density, MDOE contains richer information about events. It has two benefits: (i) it is a conceptually-simple generic representation that is task-independent; (ii) it achieves superior performance relative to existing representations on a variety of event-based datasets.**

***Index Terms*—Event-based learning, event sensors, deep learning, object recognition, convolutional neural networks**

I. Introduction

EVENT-based sensors, such as dynamic vision sensors (DVS) [1], [2] and NeuTouch tactile sensors [3], are bio-inspired devices that mimic the efficient event-driven communication mechanisms of the brain. Different from conventional sensors (e.g., RGB cameras), which synchronously capture the scene at a fixed rate, event-based sensors asynchronously report the changes (called events) of the scene. For example, DVS cameras capture brightness changes for each pixel independently rather than intensity images as RGB cameras do. Compared to conventional sensors, event-based sensors are advantageous of higher dynamic range, higher temporal resolution, lower time latency, and higher power efficiency [4].

While event-based sensors are attractive for many applications (e.g., self-driving vehicles, security and surveillance, industrial automation), they can not be directly used by state-of-the-art CNNs since their outputs are a stream of asynchronous events. Single event alone contains little information about the scene. To make use of event-based data, certain methods have been proposed to learn useful representations that can be exploited by CNNs. Figure 1 gives an example of converting events into grid-like voxels.

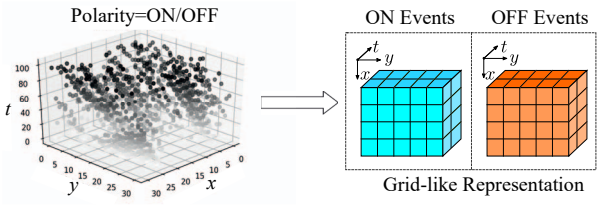


Fig. 1: Illustration of event-based representation, which converts asynchronous events into grid-like voxels

Among these representations, of particular interest for this work are Event Frame [5], Surface of Active Events (SAE) [6], Event Count [7], Voxel Grid [8], Hierarchy of Time Surfaces (HOTS) [9], Histograms of Averaged Time Surfaces (HATS) [10], Event Spike Tensor (EST) [4], and Matrix-LSTM [11]. These representations have been used for object recognition [4], [9], [10], optical flow estimation [4], [11], visual-inertial odometry [12], etc. However, existing representations discard one or more types of information about event polarity, density, and/or temporal information, which may degrade the performance of downstream applications.

This paper presents MDOE (Magnitude and Density Of Events), a novel representation for asynchronous event-based data. MDOE is obtained by concatenating the Magnitude of Events (MOE) and Density of Events (DOE). Compared to existing event-based representations, the proposed MDOE contains richer information about the polarity, temporal information, as well as density of events. Our representation has two benefits: (i) it is a conceptually-simple generic representation that is task-independent; (ii) it achieves superior performance relative to existing representations on a variety of event-based datasets.

Table I compares popular representations in terms of output dimension, and whether to contain polarity information, temporal information, and event density. From the table, one can find that the proposed MDOE contains more information about the scene than existing representations, and hence is more likely to give better performance. This is also justified by the higher test accuracy achieved by the proposed method on several public event-based datasets.

Manuscript received: February 7, 2022; Revised: May 13, 2022; Accepted: June 21, 2022. This paper was recommended for publication by Editor Markus Vincze upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the National Natural Science Foundation of China (No. 42174050, 41874031, 62172066 and 42111530064), National Key Research Development Program of China (No. 2016YFB0502204), China Postdoctoral Science Foundation (No.2021M701835), and Chongqing Innovative and Entrepreneurial Program of Returned Overseas Chinese Scholars (No. cx2021047). (*Corresponding Author: Fuqiang Gu*)

F. Gu, R. Li, and C. Chen are with the College of Computer Science, Chongqing University, China (Emails: {gufq, liruiyuan, cschaochen}@cqu.edu.cn).

Y. Lee is with the School of Mechanical and Electronic Engineering, Wuhan University of Technology, Wuhan, 430070, China (Email: yonglee@whut.edu.cn).

Y. Zhuang, Y. Li, and J. Liu are with the State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, China (Emails: {yuan.zhuang, liyou, jingbin.liu}@whu.edu.cn).

F. Yu is with the Department of Precision Instrument, Tsinghua University, China (Email: yufangwen@tsinghua.edu.cn).

Digital Object Identifier (DOI): see top of this page.

TABLE I: Comparison of different representations for event-based data. C denotes the number of temporal bins, H and W represent the height and width of image, respectively. ✓ denotes a representation contains such information, while $\times$ means a representation discards such information

Representation	Description	Dimension	Polarity	Time	Density
Event Frame [5]	Image of event polarities	$H \times W$	$\times$	$\times$	$\times$
Event Count [7]	Image of event counts	$2 \times H \times W$	✓	$\times$	$\times$
SAE [6]	Most recent timestamp	$2 \times H \times W$	✓	$\times$	$\times$
HOTS [9]	Hierarchy of event time surfaces	-	$\times$	$\times$	$\times$
HATS [10]	Histogram of average time surfaces	$2 \times H \times W$	✓	$\times$	$\times$
EV-FlowNet [13]	Event counts and most recent timestamps	$4 \times H \times W$	✓	$\times$	$\times$
Voxel Grid [8]	Summation of event polarities	$C \times H \times W$	$\times$	✓	$\times$
EST [4]	Sample event point-set into a grid	$2 \times C \times H \times W$	✓	✓	$\times$
Matrix-LSTM [11]	LSTM-based learned surfaces	$C \times H \times W$	$\times$	✓	$\times$
MDOE (Ours)	Magnitude and density of events	$4 \times C \times H \times W$	✓	✓	✓

II. Related Work

Event-based sensors have been successfully applied for a variety of tasks such as object recognition [4], [9], [10], [14], gesture recognition [15], [16], optical flow estimation [4], [11], and visual-inertial odometry [12]. Despite their advantages in terms of dynamic range, temporal resolution, and time latency, event-based sensors are still not in the mainstream due mainly to the difficulty of learning good representations from sparse and asynchronous stream of events. In this section, we provide a brief overview of related works on event-based representations.

Handcrafted Representations A naive way to represent asynchronous events is to use the number of events occurred at each pixel [5], but such method discards the temporal information and polarity information of events, which may degrade the accuracy of downstream applications. A similar representation is also used in [7], but the work in [7] considers event polarity information in addition to event counts. Instead of counting events for each pixel, SAE [6] uses the most recent timestamps as event-based representation. In HOTS [9], the concept of *Time Surface* is introduced to describe the recent history of events in the spatial neighborhood of an event. However, HOTS suffers from high latency since computing time surfaces is relatively expensive, and is sensitive to noisy events as it considers only the timestamps of the last events in the neighborhood of an event. A popular extension to HOTS is HATS [10], which uses a local memory mechanism. Instead of using the last event in the neighborhood of an event, HATS computes time surfaces by considering all past events in a spatiotemporal window. In this way, noisy events have less impact on the task. Another variant of time surface is introduced [13], in which a four-channel grid is built containing the number of positive and negative events occurred at each pixel as well as the most recent timestamps.

Spike-based Representations Due to their event-driven property, SNNs have been applied in several event-based tasks such as edge detection [17], object classification [18], [14], and gesture recognition [19], [15]. SNNs are more biologically plausible than deep learning models and can be run on power-efficient neuromorphic processors such as the IBM TrueNorth [20] and Intel Loihi [21]. However, SNNs are non-differentiable and hence cannot directly use well-established back-propagation methods to train the models, limiting the usability of SNNs in real-world scenarios. A typical method to address this problem is training a conventional neural network with frame-based data and converting the learned parameters to a SNN for event-based data [22]. In recent years, several works tried to approximate the derivative of the spike function [23], [24]. However, the achieved accuracy of spike-based methods is still not comparable to state-of-the-art deep learning methods.

Learning-based Representations In recent years, several learning-based representations have been proposed to deal with asynchronous events. In [4], a grid-based representation called EST is proposed to learn end-to-end features directly from asynchronous event-based data through differentiable kernel convolution and quantization. A multi-layer perceptron is used to learn a trilinear kernel that generates voxel-grid features, which can be tuned to the downstream task. In Matrix-LSTM [11], a grid of Long Short-Term Memory (LSTM) cells are used to learn end-to-end an event-based representation, which can capture local temporal features while retaining spatial structures. In [25], a variant of LSTM network called PhaseLSTM that uses a time gate is proposed to learn the precise timings of incoming events, which extracts relevant features with a word embedding layer. While PhaseLSTM achieves good performance on simple datasets, it may not capture general features well due to the limited representation ability of its embedding layer [26]. In [16], the event stream is treated as a set of 3D points in space-time, and a neural network called PointNet++, which is originally developed for recognizing 3D point clouds, is used to extract event-based features. In [27], a general frame is proposed to deal with event data by using the asynchronous sparse convolutional network that is converted from a synchronous model.

Most closely related to this paper are methods that map raw asynchronous events into voxel or 4D representations, mainly including Voxel Grid [8] and EST [4]. The Voxel Grid can capture the full spatiotemporal distribution of the events by discretizing the time domain and then accumulating the events in a linear way, but it discards the polarity information about events. EST [4] extends the Voxel Grid by separating the polarity and building a 4D representation with a learnable trilinear kernel. Nevertheless, EST does not consider all past events

nor the density of events. In addition, EST is an end-to-end representation, which needs to be tuned to downstream tasks. Compared to these representations, our representation contains richer information and is a more generic representation that is task-independent.

III. Proposed Method

The output of event-based sensors is a stream of asynchronous events that encode the location, time, polarity (sign) of the changes. It can be described by a sequence:

$$\varepsilon = \{e_i\}_{i=1}^{I} = \{\mathbf{x}_i, t_i, p_i\}_{i=1}^{I} \tag{1}$$

where $\mathbf{x}_i$ is the location (for DVS cameras, $\mathbf{x}_i = (x_i, y_i)$ is the coordinates of the pixel triggering the event), t_i is the timestamp when the event is generated, and p_i is the polarity of the event. The polarity takes two values: 1 and -1, representing ON and OFF events, respectively. I is the number of events in the sequence.

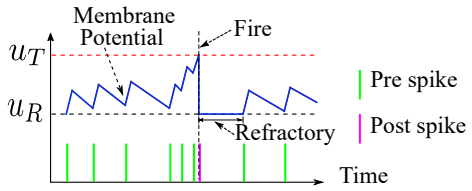
To process these asynchronous events using feature extractors (e.g., CNN), we need to convert the event sequence into a fixed-size representation. Although there are already many representation methods proposed in the literature, such as HATS [10] and EST [4], they discard certain information about the events (e.g., temporal information, polarity, or density), which may affect the performance of downstream applications (e.g., object recognition, optical flow estimation).

In this paper, we propose a novel representation called MDOE, which contains richer information about the events. MDOE is the combination of the MOE and DOE. The MOE can be considered as an unbounded membrane potential that accumulates all past events from the same pixel. By contrast, DOE reflects the event density information over a temporal bin. Combining MOE and DOE allows to capture the information about all past events and the events occurred during each temporal bin.

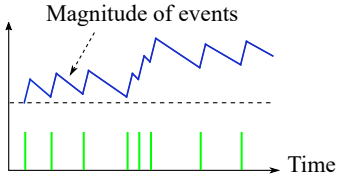
Specifically, the MOE is inspired by the dynamics of neurons in a spiking neural network (SNN) where a post-neuron integrates the inputs from all the pre-neurons and fires when its membrane potential surpasses a threshold. The dynamics of neurons' membrane potential in SNNs [28] can be described as:

$$\tau \frac{du(t)}{dt} = -u(t) + \sum_i w_i x_i' \tag{2}$$

where $u(t)$ denotes the internal membrane potential of a neuron at time t, $\sum_i w_i x_i'$ is the weighted summations of the inputs from pre-neurons, and τ is a time constant. Figure 2(a) visualizes the dynamic of a neuron's membrane potential. It can be seen that during a period of time, the denser the spikes, the larger the membrane potential. Figure 2(b) shows the proposed MOE for the same event sequence. The main difference is that in the MOE, there is no firing process and hence the magnitude will not be reset. The motivation behind this is that resetting the membrane potential (or magnitude) may lose certain information about the event stream. Thus, by sampling the MOE at regular time interval for each polarity, we can obtain the representation of dimension $2 \times C \times H \times W$,



(a) The dynamics of a neuron in the SNN



(b) MOE

Fig. 2: (a) The dynamics of a neuron's membrane potential in a SNN, where u_T and u_R are the firing threshold and reset value. When the membrane potential of a neuron surpasses the threshold u_T, it will fire (output a spike) and its membrane potential will be reset to u_R; (b) The magnitude of events of a pixel is similar to the membrane potential, but it has no firing process and refractory period

where H and W are the spatial height and width of the event image, respectively, and C is the number of temporal bins.

Apart from the MOE, we also integrate the DOE of each temporal window into the MDOE representation. An earlier work [7] has considered the total number of triggered events for each pixel, which may not be informative enough since it can not reflect the event distribution. Instead of considering the total number of events during a time period, we divide the time period into different time windows and count the events over each temporal window pixel by pixel, resulting a vector of size C for each pixel. By doing this for each polarity, we can obtain the DOE representation with dimension $2 \times C \times H \times W$.

Both MOE and DOE are 4-dimensional (namely, $2 \times C \times H \times W$), concatenating them results in a 4-dimensional MDOE representation ($4 \times C \times H \times W$). Let E be the MOE, we describe it as follows:

$$E_{\pm}(x_l, y_m, c_n) = \lambda E_{\pm}(x_l, y_m, c_{n-1}) + \sum_{e_i \in \varepsilon_{\pm}\{t_{n-1} \le t_i \le t_n\}} \delta(x_l - x_i)\delta(y_m - y_i)k(t_n - t_i) \tag{3}$$

$$t_n = t_1 + (c_n + 1)\Delta T \tag{4}$$

$$\delta(a) = \begin{cases} 1, & \text{if } a = 0 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$k(a) = max(0, 1 - |a|) \tag{6}$$

where ε_+ and ε_- are event sequences with positive polarity and negative polarity, respectively. λ is the decaying factor that determines how much past information is considered, and it ranges from 0 to 1. $k(a)$ is a bilinear kernel that emulates the neuron dynamics, (x_l, y_m, c_n) is the spatiotemporal coordinates on a voxel grid, and $x_l \in \{0, 1, \cdots, W-1\}$, $y_m \in \{0, 1, \cdots, H-1\}$, and $c_n \in \{0, 1, \cdots, C-1\}$. ΔT is temporal bin size. The MOE can be considered as

an unbounded membrane potential that accumulates all past events from the same pixel. The closer the events distributed during a time period, the larger the value of MOE.

Similarly, we can write the DOE (denoted by V) as:

$$V_{\pm}(x_l, y_m, c_n) = \sum_{e_i \in \varepsilon_{\pm}\{t_{n-1} < t_i \leq t_n\}} \delta(x_l - x_i)\delta(y_m - y_i)f(e_i) \tag{7}$$

where $f(e_i)$ is the indicator function that ensures to count only events falling between the specified time period. $f(e_i)$ is written as

$$f(e_i) = \begin{cases} 1, & \text{if } e_i \in \varepsilon_{\pm}\{t_{n-1} < t_i \leq t_n\} \\ 0, & \text{otherwise} \end{cases} . \tag{8}$$

To avoid overflow when the MOE makes sum operation, we normalize the timestamps events to the range of [0, 1] before calculation. For real-time applications, one can convert the timestamps into seconds. DOE reflects the number of events over a time window. The more events triggered during the time period, the larger the value of DOE.

To reduce the effect of sensor noises (e.g., reporting different number of events for the exact scene at different time), the resulting MOE ($E_{\pm}$) and DOE ($V_{\pm}$) are normalized via dividing them by their respective maximum values across pixels and temporal bins. The MDOE (denoted by S) is the concatenation of MOE and DOE along the polarity, namely

$$S(p, x_l, y_m, c_n) = Cat_{\pm}(E_{\pm}(x_l, y_m, c_n), V_{\pm}(x_l, y_m, c_n)) \tag{9}$$

where Cat represents the concatenation operation, p indicates the polarity of MOE and DOE, and $p \in \{0, 1, 2, 3\}$, where $p = 0$ and $p = 1$ represent the ON and OFF polarity for MOE, while $p = 2$ and $p = 3$ correspond to the ON and OFF polarity for DOE. For better clarity, we provide the pseudo-code of calculating the MDOE in Algorithm 1.

Algorithm 1: Calculation of MDOE

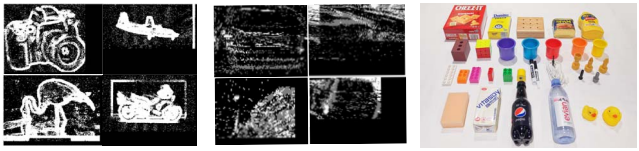
Input : A stream of events $\varepsilon = \{\mathbf{x}_i, t_i, p_i\}_{i=1}^{I}$, the spatiotemporal dimension of output dimension (C, H, W), and decay factor λ
Output: MDOE representation with dimension $4 \times C \times H \times W$

Separate the event sequence by their polarity and obtain ε_+ and ε_-;
Calculate the temporal bin by $\Delta T = \frac{t_I - t_1}{C}$;
for $x_l \in \{0, \cdots, W-1\}, y_m \in \{0, \cdots, H-1\}, c_n \in \{0, 1, \cdots, C-1\}$ **do**
    Initialize $E_{\pm}(x_l, y_m, c_n)$ and $V_{\pm}(x_l, y_m, c_n)$ to 0;
end
for $x_l \in \{0, \cdots, W-1\}, y_m \in \{0, \cdots, H-1\}, c_n \in \{0, 1, \cdots, C-1\}$ **do**
    $t_{n-1} = t_1 + c_n \Delta T$; //Compute t_{n-1}
    $t_n = t_1 + (c_n + 1)\Delta T$; //Compute t_n
    for $e_i \in \varepsilon_+$ **do**
        if $t_i \in (t_{n-1}, t_n]$ **then**
            $E_+(x_l, y_m, c_n) = \lambda E_+(x_l, y_m, c_{n-1}) + \delta(x_l - x_i)\delta(y_m - y_i)k(t_n - t_i)$;
            $V_+(x_l, y_m, c_n)$ += $\delta(x_l - x_i)\delta(y_m - y_i)f(e_i)$;
        end
    end
    for $e_i \in \varepsilon_-$ **do**
        if $t_i \in (t_{n-1}, t_n]$ **then**
            $E_-(x_l, y_m, c_n) = \lambda E_-(x_l, y_m, c_{n-1}) + \delta(x_l - x_i)\delta(y_m - y_i)k(t_n - t_i)$;
            $V_-(x_l, y_m, c_n)$ += $\delta(x_l - x_i)\delta(y_m - y_i)f(e_i)$;
        end
    end
end
for $x_l \in \{0, \cdots, W-1\}, y_m \in \{0, \cdots, H-1\}, c_n \in \{0, 1, \cdots, C-1\}$ **do**
    $E_{\pm}(x_l, y_m, c_n) = E_{\pm}(x_l, y_m, c_n)/max(E_{\pm})$;
    $V_{\pm}(x_l, y_m, c_n) = V_{\pm}(x_l, y_m, c_n)/max(V_{\pm})$;
end
$S = Cat_{\pm}(E_{\pm}, V_{\pm})$; //Concatenate MOE and DOE to generate MDOE;
Return S;

IV. Experiments and Results

A. Datasets

We evaluate the proposed method using three publicly available event-based datasets: N-Caltech101 [29], N-Cars [10], and EvTouch-Objects [3], [14]. N-Caltech101 (Neuromorphic-Caltech101) is an event-based version of the popular Caltech101 dataset [30]. To convert the images to event sequences, an ATIS event camera was installed on a motorized pan-tilt unit, and automatically moved while pointing at images from the original dataset (Caltech101) that were shown on a LCD monitor. N-Cars (Neuromorphic-Cars) is a real-world event-based dataset for recognizing whether a car is present in a scene. It was recorded using an ATIS camera that was mounted behind the windshield of a car. EvTouch-Objects dataset is an event-driven tactile dataset for object recognition. It was collected using a 7-DoF Franka Emika Panda arm equipped with a Robotiq 2F-140 gripper, equipped with a NeuTouch event-based tactile sensor [3] and an ACES decoder [31] to decode the sensor signals into spikes. This dataset comprises tactile data from 36 object classes. For each object class, 20 samples were collected, yielding a total of 720 samples. Figure 3 gives some sample examples of the used datasets. For better visualization, the Event Frame method is used to convert events into images.



(a) N-Caltech101 (b) N-Cars (c) EvTouch-Object

Fig. 3: Sample examples of the used three public event-based datasets. For better visualization, the Event Frame is used to demonstrate the sample of N-Caltech101 and N-Cars datasets. For EvTouch-Object dataset, which is an event-based tactile dataset, we demonstrate the images of objects used in the data collection

B. Implementation Details

We evaluate the proposed method using a ResNet-34 architecture [32] for easier comparison with baseline methods that share the same network architecture. We also evaluate our method using other state-of-the-art networks including VGG-19 [33], MobileNet-V2 [34], and Inception-V3 [35]. All the networks are pretrained on ImageNet [36]. Since the number of input channels and output classes for our case is different from these pre-trained models, we adopt the approach used in [7], [4] and replace the first and last layer of the pre-trained models with random weights, and then fine-tune all

the parameters on the task. Note that the original ResNet-34 is for 2D images, while our event presentation is a 4D tensor. To adapt to the ResNet-34, we concatenate the event-based representation along the polarity and temporal dimension as channels.

The Adam optimizer [37] is used to train the model by minimizing the cross-entropy loss. The initial learning rate is set to 1×10^{-4} and is reduced by a factor of 0.5 every 10 iterations. The total number of iterations is set to 200. We use a batch size of 4 for all the datasets. To conduct a robust evaluation, we run the model on each dataset for multiple rounds with different random seeds, and report the mean and standard deviation values. We perform early stopping [38] on a validation set using the splits provided by the EST [4] on N-Caltech101 and 20% of the training data on N-Cars. For EvTouch-Objects, there is no validation dataset provided online or in publications, we report the best test accuracy during the 200 epochs. The default value of decay factor in MDOE is set to 0.1.

We have implemented the MDOE in PyTorch and the source code will be available at *https://github.com/uisc-cqu/MDOE*.

C. Ablation Study

Contribution of MOE and DOE To evaluate the contribution of each component of the proposed MDOE, we first compare the object recognition accuracy of using MDOE, MOE, and DOE, respectively. We use ResNet-34 as the classifier, and let all the methods share the same configuration (e.g., training epochs, learning rate). A temporal bin of 9 is used for all the methods. Table II demonstrates that MDOE performs better than MOE and DOE. Specifically, the MOE achieves a higher accuracy on N-Cars but a lower accuracy on N-Caltech101 than the DOE. The proposed method that combines the MOE and DOE outperforms the MOE and DOE alone on both datasets. This might because MDOE contains richer information about the events, including not only the polarity and temporal information, but also the density. This makes it more robust when conducted on different datasets.

TABLE II: Test accuracy (%) comparison of using MOE, DOE, and MDOE

Representation	N-Caltech101 (Std)	N-Cars (Std)
MOE	84.9 (0.9)	93.5 (0.5)
DOE	85.6 (0.4)	92.9 (0.5)
MDOE	**85.9 (0.1)**	**94.2 (0.4)**

Analysis of different temporal bins We then analyze the effect of different temporal bins on the performance of our method with the number of temporal bins varying from 1 to 64. Table III shows that the number of temporal bins has an impact on the proposed method. The achieved accuracy increases from 83.4% to 85.9% on N-Caltech101 and from 93.6% to 94.2% on N-Cars when the temporal bin rises from 1 to 9. The test accuracy continues to rise until the temporal bin reaches 32 for N-Caltech101. However, it is not the same case for N-Cars dataset where the number of temporal bins has a negligible impact on the proposed method after it reaches to 9. In the following, we set the temporal bin to 9 to analyze the effect of other factors and compare with baseline methods.

TABLE III: The test accuracy (%) of using MDOE with different number of temporal bins

Temporal Bin	N-Caltech101 (Std)	N-Cars (Std)
1	83.4 (0.4)	93.6 (0.3)
9	85.9 (0.1)	94.2 (0.4)
16	86.2 (0.7)	94.1 (0.4)
32	86.6 (0.8)	94.2 (0.4)
64	86.4 (0.6)	94.3 (0.4)

Performance across different models We also evaluate the proposed method across four state-of-the-art deep learning architectures, namely ResNet-34 [32], VGG-19 [33], MobileNet-V2 [34], and Inception-V3 [35]. Table IV shows that the test accuracy of the proposed method using different network architectures on N-Caltech101 and N-Cars datasets. It can be seen that MobileNet-V2 and ResNet-34 can result in a higher accuracy than VGG-19 and Inception-V3 on both datasets. Specifically, ResNet-34 performs the best on N-Caltech101, which is followed by MobileNet-V2 and Inception-V3. By contrast, MobileNet-V2 achieves the highest accuracy on N-Cars, followed by ResNet-34 and Inception-V3. VGG-19 witnesses the lowest accuracy on both datasets among the four architectures.

TABLE IV: The test accuracy (%) of using MDOE across different models

Model	N-Caltech101 (Std)	N-Cars (Std)
VGG-19	81.9 (0.8)	91.9 (0.6)
Inception-V3	84.9 (0.6)	93.6 (0.8)
MobileNet-V2	85.3 (0.6)	94.8 (0.8)
ResNet-34	85.9 (0.1)	94.2 (0.4)

Analysis of time latency Time latency is the consumed time period to accumulate evidence in making a decision on the object class [10], which is crucial for the applications requiring fast reaction time. We compare the time latency of the proposed method with Event Frame [5], Event Count [7], and Voxel Grid [8] on N-Caltech101. As shown in Figure 4, our method outperforms all the baseline methods and can achieve about 82% accuracy using only the first 30 ms of the samples. This makes the proposed method a good match for applications with highly real-time demand such as autonomous navigation. We can also find that a larger time latency does not necessarily mean a higher accuracy. This is especially true for the Event Frame and Event Count methods where the achieved accuracy degrades when the time latency reaches up to 60 ms and 100 ms respectively. This might be that Event Frame and Event Count do not consider the temporal information of events, and a larger time latency may introduce more noise.

Ratio of used training data We analyze the effect of using different ratios of training data on the performance of the proposed method, and compare it with Event Frame [5], Event Count [7], and Voxel Grid [8] on N-Caltech101. For fair comparison, we let our method and all the baseline methods share the same configuration (e.g., ResNet-34 network, random seed, and learning rate). As shown in Figure 5, the classification

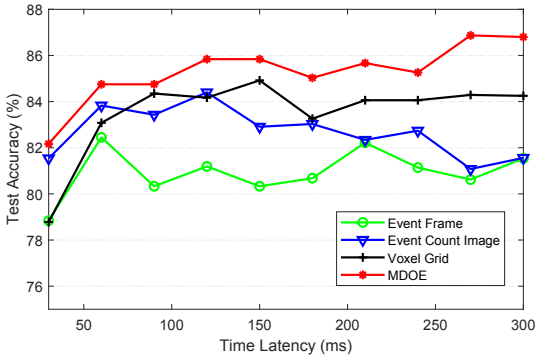


Fig. 4: Time latency as a function on N-Caltech101

accuracy of all the methods increases as the ratio of used training data goes, and the proposed method is consistently more accurate than the baselines. It can be seen that our method achieves an accuracy of about 63% when using only 10% training data, which is much higher than Voxel Grid (about 59%), Event Count (55%) and Event Frame (53%). This implies that the proposed method can better deal with small datasets, and such feature is very useful for cases where collecting a large amount of data is troublesome or impractical.

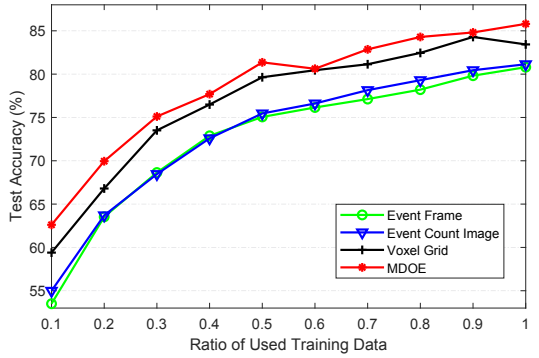


Fig. 5: Effect of using different ratios of training data (N-Caltech101)

Impact of Decay Factor The impact of the decay factor on the performance of the proposed method is also investigated. The considered value of the decay factor ranges from 0 to 1, where 0 represents the MOE at the current time window does not use the MOE from previous time windows, and 1 means that all the MOE from past time windows are considered at the current time window. Figure 6 shows that the decay factor has an impact on the performance of MDOE. The best performance of about 85.9% can be achieved when the decay factor is set to 0.1, 0.2, 0.5, and 0.9, and the smallest standard deviation is witnessed at the value of 0.1. It is also observed that discarding (the corresponding decay factor is 0) or keeping (decay factor is 1) all the information from past time windows does not result in the best performance. This implies that we should keep the information from past windows to some extent to achieve better performance.

Complexity Analysis The storage cost and computational cost of the proposed method are also analyzed. We compare the proposed method with Event Frame [5], Event Count Image [7], Voxel Grid [8], and EST [4]. The results are given in the Table V, where B is the batch size, C is the number of time windows (or temporal bins), N is the number of events, H and W are the height and width of events, respectively. From the table, we can see that the storage and computational complexity of the MDOE is similar to Voxel Grid and EST, all of which consider the temporal information of events. However, the achieved accuracy of MDOE is much higher than other methods.

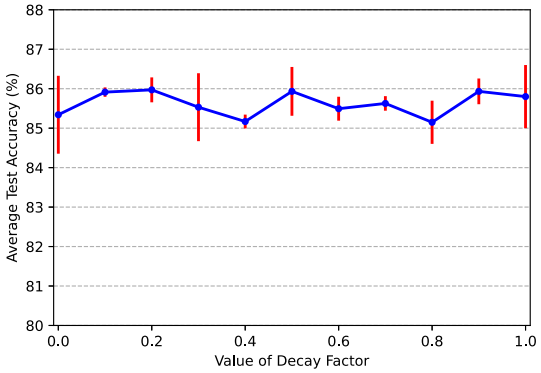


Fig. 6: The average test accuracy with standard deviation using different values of the decay factor in MDOE

TABLE V: The storage and time complexity of different methods.

Method	Storage	Time
Event Frame	$\mathcal{O}(BHW)$	$\mathcal{O}(N)$
Event Count Image	$\mathcal{O}(BHW)$	$\mathcal{O}(N)$
Voxel Grid	$\mathcal{O}(BCHW)$	$\mathcal{O}(CN)$
EST	$\mathcal{O}(BCHW)$	$\mathcal{O}(CN)$
MDOE	$\mathcal{O}(BCHW)$	$\mathcal{O}(CN)$

D. Results on Visual Object Recognition

We compare the proposed method with state-of-the-art methods on two visual object recognition datasets (N-Caltech101 and N-Cars). The baseline methods include HATS [9], Matrix-LSTM [11], E2VID [12], Event Frame [5], Event Count Image [7], Voxel Grid [8], and EST [4]. All the baseline methods except E2VID (that uses ResNet-18) use ResNet-34 as the classifier. For HATS, Matrix-LSTM, E2VID, and EST, we directly report the recognition accuracy provided in the corresponding reference. For other methods including Event Frame, Event Count, and Voxel Grid, we implement these methods and report the mean accuracy with standard deviation after running the model multiple times. To conduct a fair comparison, we use the same number of channels in the event representations (Voxel Grid and EST) and data augmentation procedures (random shifts and random horizontal flips).

Table VI summarizes the results of these methods. Our method outperforms all the baseline methods on both datasets. In particular, we can see that the proposed method can achieve an accuracy of 85.9% on N-Caltech101 and 94.2% on N-Cars, which is much higher than Voxel Grid (83.9% and 92.9% respectively), Event Count Image (81.1% and 93.5%), and

TABLE VI: The test accuracy (%) comparison of the proposed MDOE with baseline methods. Both our method and baseline methods use ResNet-34 as the classifier by default. Note that for fair comparison with E2VID, we conduct a separate experiment using ResNet-18 classifier and directly report the best test accuracy without using a validation dataset for early stopping

Method	Classifier	N-Caltech101 (Std)	N-Cars (Std)
HATS	ResNet-34	69.1	90.9
EST		81.7	92.5
Matrix-LSTM		83.5 (1.2)	92.2 (0.7)
Event Frame		80.8 (0.6)	93.4 (0.5)
Event Count		81.1 (0.4)	93.5 (0.6)
Voxel Grid		83.9 (0.3)	92.9 (0.6)
MDOE (Ours)		**85.9 (0.1)**	**94.2 (0.4)**
E2VID	ResNet-18	86.6	91.0
MDOE (Ours)		**88.2 (0.4)**	**94.6 (0.3)**

Matrix-LSTM (83.5% and 92.2%). It can be also seen that there is not a baseline method that consistently exceed the other baseline methods on both datasets. For the comparison with E2VID, we conduct a separate experiment using ResNet-18 classifier and directly report the best test accuracy without using a validation dataset for early stopping. We can see that our method outperforms E2VID by about 1.6% and 3.6% on N-Caltech101 and N-Cars, respectively. It can also be observed that the achieved accuracy of both our method and E2VID is much higher than other methods. This is because the best test accuracy is reported without using early stopping via a validation dataset.

E. Results on Tactile Sensing

We also evaluate the performance of the proposed MDOE on the EvTouch-Objects dataset. Since the EvTouch-Objects data were collected using irregularly-placed NeuTouch taxels, we cannot directly use state-of-the-art CNNs to deal with them. To make use of CNNs, we organize the tactile data in a grid structure of 11×11. Figure 7 shows that the spatial distribution of NeuTouch taxels, and the corresponding grid-based representation.

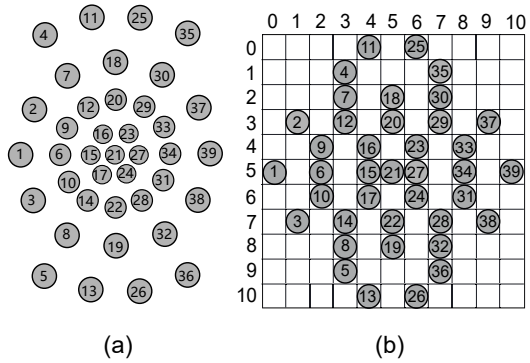


Fig. 7: (a) The spatial distribution of 39 NeuTouch taxels; (b) Grid-based representation of irregular tactile data

We compare the proposed MDOE with a spiking-based method called TactileSGNet [14], and four frame-like methods, namely Event Frame [5], Event Count [7], Voxel Grid [8], and EST [4]. All the frame-like methods and the proposed MDOE use ResNet-34 as the classifier and share the same configuration (e.g., training epochs, learning rate). Table VII shows the test accuracy of using different methods for event-based tactile object recognition. From the table, we can see that the proposed method significantly outperforms all the baseline methods. The test accuracy achieved by MDOE is about 95.8%, which is much higher than Voxel Grid (94.3%), EST (93.1%), and TactileSGNet (89.4%). It is a bit surprising to see that Event Frame and Event Count can only achieve an accuracy of 62.9%, which is about 33% lower than our method. This might be that both Event Frame and Event Count discard the temporal information of events, which significantly affect the achieved accuracy.

TABLE VII: The test accuracy (%) of using different representations for tactile object classification

Method	Average Accuracy (Std)
TactileSGNet	89.4 (0.6)
Event Frame	62.9 (1.1)
Event Count	62.9 (1.6)
Voxel Grid	94.3 (1.5)
EST	93.1 (1.6)
MDOE (Our)	**95.8 (1.1)**

V. Conclusion and Discussion

We have proposed a novel representation for asynchronous event-based data which contains not only the polarity and temporal information of events, but also the event density. While the proposed method has demonstrated superior classification accuracy across different deep learning models, there are still some open issues or challenges remained. First, we integrate events into MDOE pixel-by-pixel, and the resulted representation might be computationally expensive for high-resolution event-based data. Actually, to improve the computational and storage efficiency, one can combine events from multiple pixels into a group, and generate representation group-by-group. Second, we have not evaluated the performance of the proposed method on regression tasks, such as flow estimation, simultaneous localization and mapping, and depth estimation. In the future, we will evaluate our work to these regression tasks.

References

[1] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120 db 15μs latency asynchronous temporal contrast vision sensor," *IEEE journal of solid-state circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[2] T. Delbrück, B. Linares-Barranco, E. Culurciello, and C. Posch, "Activity-driven, event-based vision sensors," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. IEEE, 2010, pp. 2426–2429.

[3] T. Taunyazov, W. Sng, H. H. See, B. Lim, J. Kuan, A. F. Ansari, B. Tee, and H. Soh, "Event-driven visual-tactile sensing and learning for robots," in *Robotics: Science and Systems*, 2020.

[4] 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 International Conference on Computer Vision*, 2019, pp. 5633–5643.

[5] H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization," in *British Machine Vision Conference*, 2017.

[6] 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.

[7] 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.

[8] 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 Conference on Computer Vision and Pattern Recognition*, 2019, pp. 989–997.

[9] X. Lagorce, G. Orchard, F. Galluppi, B. E. Shi, and R. B. Benosman, "Hots: a hierarchy of event-based time-surfaces for pattern recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 7, pp. 1346–1359, 2016.

[10] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman, "Hats: Histograms of averaged time surfaces for robust event-based object classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1731–1740.

[11] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, "Matrix-lstm: a differentiable recurrent surface for asynchronous event-based data," *arXiv preprint arXiv:2001.03455*, 2020.

[12] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza, "Events-to-video: Bringing modern computer vision to event cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3857–3866.

[13] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "Ev-flownet: Self-supervised optical flow estimation for event-based cameras," in *Proceedings of Robotics: Science and Systems*, 2018.

[14] F. Gu, W. Sng, T. Taunyazov, and H. Soh, "Tactilesgnet: A spiking graph neural network for event-based tactile object recognition," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[15] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza *et al.*, "A low power, fully event-based gesture recognition system," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7243–7252.

[16] Q. Wang, Y. Zhang, J. Yuan, and Y. Lu, "Space-time event clouds for gesture recognition: From rgb cameras to event cameras," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 1826–1835.

[17] Q. Wu, M. McGinnity, L. Maguire, A. Belatreche, and B. Glackin, "Edge detection based on spiking neural network model," in *International Conference on Intelligent Computing*. Springer, 2007, pp. 26–34.

[18] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in neuroscience*, vol. 10, p. 508, 2016.

[19] J. Botzheim, T. Obo, and N. Kubota, "Human gesture recognition for robot partners by spiking neural network and classification learning," in *The 6th International Conference on Soft Computing and Intelligent Systems, and The 13th International Symposium on Advanced Intelligence Systems*. IEEE, 2012, pp. 1954–1958.

[20] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[21] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[22] J. A. Pérez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, and B. Linares-Barranco, "Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward convnets," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2706–2719, 2013.

[23] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems*, 2018, pp. 1412–1421.

[24] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, "Direct training for spiking neural networks: Faster, larger, better," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 1311–1318.

[25] D. Neil, M. Pfeiffer, and S.-C. Liu, "Phased lstm: Accelerating recurrent network training for long or event-based sequences," in *Advances in neural information processing systems*, 2016, pp. 3882–3890.

[26] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, "Attention mechanisms for object recognition with event-based cameras," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 1127–1136.

[27] 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.

[28] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[29] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in neuroscience*, vol. 9, p. 437, 2015.

[30] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," in *2004 conference on computer vision and pattern recognition workshop*. IEEE, 2004, pp. 178–178.

[31] W. W. Lee, Y. J. Tan, H. Yao, S. Li, H. H. See, M. Hon, K. A. Ng, B. Xiong, J. S. Ho, and B. C. Tee, "A neuro-inspired artificial peripheral nervous system for scalable electronic skins," *Science Robotics*, vol. 4, no. 32, p. eaax2198, 2019.

[32] 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.

[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[34] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[35] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[38] L. Prechelt, "Early stopping-but when?" in *Neural Networks: Tricks of the trade*. Springer, 1998, pp. 55–69.