Voxel-based Multi-scale Transformer Network for Event Stream Processing

Daikun Liu, Teng Wang, Changyin Sun

***Abstract*—Event cameras are bio-inspired dynamic vision sensors that are superior to frame-based cameras in terms of low power consumption, high dynamic range, and high temporal resolution in computer vision tasks. Recent advances in voxel-based representation learning have successfully exploited the sparsity of events with low computational complexity, but face challenges in extracting spatio-temporal features within voxels and representative global dependencies between voxels, thus limiting their representation power. In this work, towards a better trade-off between accuracy and computation overhead, we propose a novel voxel-based multi-scale transformer network (VMST-Net) to process event streams. Specifically, VMST-Net projects events within voxels into multi-channel frames along the time axis, such that 2D convolutions could be leveraged to encode spatio-temporal features in voxels. Then, VMST-Net utilizes a novel multi-scale multi-head self-attention (MSMHSA) mechanism with a multi-scale fusion (MSF) module that allows different heads within each layer to attend different scale 3D neighborhoods to adaptively aggregate the coarse-to-fine voxel features with little computational costs and parameters. Moreover, to model effective global features while saving computations, we aggregate features in a local-to-global manner by enlarging the coverage of 3D neighborhoods as the network gets deeper. Extensive experimental results on benchmark datasets demonstrate that our model advances state-of-the-art accuracy with low model complexity and computational complexity in all three visual tasks, including object classification, action recognition, and human pose estimation.**

***Index Terms*—Event stream, voxelization, spatio-temporal feature extraction, multi-scale transformer, multi-scale feature fusion.**

I. Introduction

EVENT cameras are bio-inspired dynamic vision sensors (DVSs) [1] that pose a *paradigm shift* in the way visual information is acquired. Instead of producing frames at fixed rates, DVS measures asynchronous brightness changes for each pixel independently and outputs a stream of sparse and asynchronous "events" that encode the space-time coordinates (x, y, t) and polarity ("ON" and "OFF") of the brightness changes. Compared to conventional frame-based cameras, DVSs have low power consumption, high dynamic range, and low latency. These advantages make DVS sensors particularly suitable for various vision tasks, including object/gesture recognition [2, 3], object tracking [4, 5], optical flow estimation [6–8], and Simultaneous Localization and Mapping (SLAM) [9, 10]. Since event data is intrinsically sparse and asynchronous, conventional frame-based learning models, such as 2D convolutional neural networks (CNNs), are no longer compatible. As a consequence, designing novel learning-based models for event data receives a lot of attention recently.

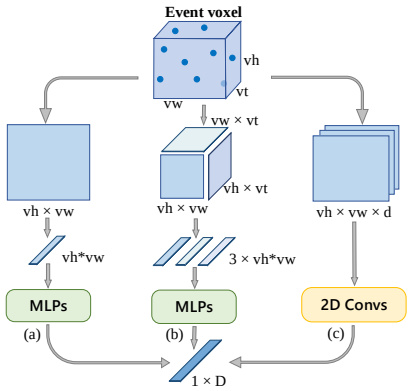

Fig. 1. Illustration of our voxel feature encoding methods in comparison with current voxel-based methods. Voxel feature encoding in (a) EV-VGCNN [11], (b) VMV-GCN [12], and (c) our STFE module. vh, vw, and vt are the size of the event voxel, and D is the feature dimension of the encoded feature vector.

To take advantage of the sparsity and high temporal resolution of the event data, some recent works turn to graph convolutional networks (GCNs) by focusing on data representation and neural network architecture design [11–15]. As for data representation, earlier work directly treats the downsampled events of the raw events as graph vertices, such as RG-CNN [13]. These point-wise representations suffer significant information loss due to the over-sparsity of the downsampled events. Although increasing the number of downsampled events could improve performance to some extent, the computational complexity is increased correspondingly. Toward a better trade-off between accuracy and computation overhead, event representation is shifting to voxel-based methods [11, 12], which voxelize raw events and select representative voxels as vertices. To generate the input features of vertices, EV-VGCNN [11] aggregates events inside each voxel by accumulating events into a 2D patch along the time dimension and feeding the flattened patch into multi-layer perceptron layers (MLPs) (Fig. 1 (a)). The

This work was in part supported by the National Natural Science Foundation of China (Grant No. 62273093, 61921004), and in part by ZhiShan Scholar Program of Southeast University. (*Corresponding author: Teng Wang.*)

Daikun Liu and Teng Wang are with the School of Automation, Southeast University, Nanjing 210096, China. (e-mail: dkliu@seu.edu.cn; wangteng@seu.edu.cn).

Changyin Sun is with the School of Automation, Southeast University, Nanjing 210096, China, and also with the Key Laboratory of Measurement and Control of Complex Systems of Engineering, Ministry of Education, Southeast University, Nanjing 210096, China. (e-mail: cysun@seu.edu.cn).

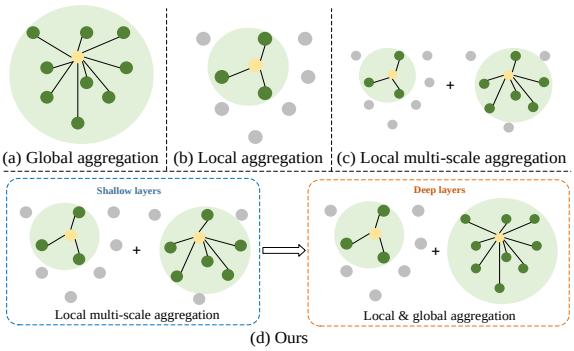

Fig. 2. Illustration of our multi-scale feature aggregation strategy in comparison with feature aggregation methods in other networks. (a) Global aggregation, each layer of the network adopts a global receptive field. (b) and (c) Local feature aggregation, where each layer of the network aggregates information in the local receptive field. (d) Our multi-scale strategy aggregates local multi-scale information at shallow layers and integrates global information and local details at deep layers. Dots represent tokens in the transformer-based network or vertices in the graph-based network. Yellow dots represent reference points, dark green dots represent neighbor dots within the receptive field, and gray dots represent dots outside the receptive field. In addition, large light green circles indicate the range of receptive fields.

EV-VGCNN could preserve more local cues than point-wise representations but ignores the temporal relations of events within voxels. VMV-GCN [12] alleviates this issue by fusing flattened multi-view 2D patches to encode the spatio-temporal information(Fig. 1 (b)), while still suffering from insufficient semantic and motion cues in voxels. To this end, we propose a spatio-temporal feature extraction (STFE) module to preserve rich semantic and motion cues inside each voxel. This is achieved by projecting events within each voxel into a multi-channel event frame with each channel representing one short time span and subsequently processing the frame with 2D convolutions (Fig. 1 (c)). Since 2D convolutions simultaneously mix information across both the spatial and channel dimensions, STFE is capable of capturing detailed spatio-temporal features. Particularly, in STFE the empty time bins of the multi-channel frames could also provide chronological information to perceive motion, thus enhancing motion cues in voxels.

Concerning the network architecture, the two SOTA models, VMV-GCN and EV-VGCNN, aggregate vertices information at the local single scale and local multiple scales in each layer, respectively (Fig. 2 (b)-(c)). However, they show a weak global spatio-temporal feature modeling capability due to limited receptive fields at different layers. More recently, multiple transformer architectures are proposed to process event data due to their strong power in global relation modeling [16, 17]. Nevertheless, these transformer-based methods share the same receptive field across different layers (Fig. 2 (a)) and thus fail to capture local multi-scale detailed features, limiting their representation power to some extent. In this work, we propose a novel voxel-based multi-scale transformer architecture (VMST-Net), as shown in Fig. 3, aiming to improve feature representation power while maintaining low computational complexity. It is demonstrated that an event stream mainly contains semantic details and subtle motions within a small space-time neighborhood, whereas obvious motions and coarse-grained spatial messages within a large neighborhood [11]. To fully exploit this variation, we introduce a multi-scale multi-head self-attention (MSMHSA) layer within each transformer block to capture both semantic and motion features for each voxel. Specifically, the proposed MSMHSA allows different self-attention heads within the same layer to respectively attend different scale 3D neighborhoods. It introduces no additional computations compared to regular MHSA. Note that instead of sharing the multi-scale 3D neighborhoods across all layers, we perform feature aggregation in a local-to-global manner by enlarging the coverage of 3D neighborhoods progressively (Fig. 2 (d)) to learn discriminative global features while saving computational costs. This local-to-global manner could be seen as another multi-scale strategy of our network. The obtained features from different heads are further adaptively weighted by the proposed multi-scale fusion (MSF) module, thus avoiding the influence of sub-optimal scales. To the best of our knowledge, this is the first attempt to apply a multi-scale transformer with dynamic fusion to the event data.

We evaluate the proposed VMST-Net on three different visual tasks, including object classification, action recognition, and human pose estimation. Extensive experiment results on benchmark datasets demonstrate that VMST-Net can effectively process event streams in scenes that include static objects as well as dynamic objects. The main contributions of the work could be summarized as follows:

- We propose a novel voxel-based multi-scale transformer architecture to learn discriminative feature representations from event streams in a local-to-global manner, which can effectively enhance representation power while maintaining the sparsity of event data.
- We introduce a simple but effective STFE module to encode the input features for voxels, which help retain more spatial context as well as motion cues.
- By taking advantage of spatio-temporal dependencies between voxels and their neighbors, we propose an MSMHSA layer with an MSF module to aggregate voxel information over different scale 3D neighborhoods in an adaptive manner.
- Extensive experiments demonstrate that our model achieves SOTA accuracy with low model and computational complexity on different tasks.

II. Related Work

A. Event data representation

The descriptors of corner detection [18, 19] are the early attempts of event cameras in traditional vision tasks but are difficult to generalize to complex tasks. To apply frame-based algorithms to event data, most studies preprocess event data by converting events into dense frames or maps. Time surface (TS) [20, 21] is a 2D map where each pixel stores a time value, which asynchronously processes event data and explicitly preserves spatio-temporal information. However, TSs are sensitive to noise and can lead to performance degradation when events spike frequently. Another commonly used representation is event frames. Maqueda *et al.* [22] design a simple event frame to address steering-angle prediction, where events of different

polarities are accumulated in two channels of the event frame. EV-FlowNet [7] integrates time and polarity information by converting events into a four-dimensional grid to effectively perform optical flow estimation. Considering the sacrifice of temporal resolution in previous event frames, Zhu *et al.* [23] convert events into a spatio-temporal voxel grid with more temporal information. Uddin *et al.* [24] weighted and accumulate the N most recent events in each spatial location based on their arrival time to capture rich spatio-temporal relations. Furthermore, Deng *et al.* retain more motion information by considering the multi-view 2D grid on event streams [25]. TBR accumulates binarized frames into single-bin frames, leading to high accuracy in action recognition [26]. Inspired by end-to-end training in conventional vision, EST [27] and Matrix-LSMT [28] attempt to design learning-based event representations, which can adaptively extract important information. Although these frame-based representations can be processed by conventional visual algorithms, they ignore sparsity, loss a lot of time information, and has poor robustness. Additionally, dense frame-wise representations introduce spatial redundancy, and applying CNNs on them greatly increases the model complexity. Therefore, in our approach, we avoid converting events into frames as well as utilizing CNNs to introduce high model complexity.

Recent studies, instead, leveraging the sparsity nature of event data, can retain higher time resolution and make the model more efficient. These types of representations usually treat the event data as point clouds [14, 29, 30] or graphs [11–13, 15, 31] with preprocessing. Specifically, RG-CNN [13] and EV-Gait [15] construct spatio-temporal graphs on downsampled events, resulting in spatio-temporal information loss. AEGNN preserves more event information by processing events asynchronously instead of downsampling them [31]. RG-CNN, EV-Gait, AEGNN, and works [29, 30] are point-wise representations, in which the raw event points carry little information and are sensitive to noise. In [14], a rasterized event representation is proposed to improve these limitations that aggregate events on the same position of a small time slice. Even so, point-wise inputs still lack local semantic and motion information, thus voxel-wise representations are adopted in EV-VGCNN [11] and VMV-GCN [12]. EV-VGCNN preserves local spatial relations inside voxels by flattening the 2D x-y vector. VMV-GCN further introduces the local motion feature inside voxels by utilizing the multi-view idea of [25]. In this way, more local semantic and motion cues are retained in a voxel than in an individual event and save more computational costs. However, simply flattening the multi-view 2D vector cannot encode sufficient local spatio-temporal dependencies within a voxel.

In this work, based on voxel-wise representation, the proposed STFE module with 2D convolutions captures richer local spatial relations as well as motions inside each voxel. Our informative voxels allow us to set a smaller number of voxels, maintaining the sparsity advantage while reducing a lot of computational complexity.

B. Event-based processing framework

Given the success of deep learning in other vision sensor algorithms, event-based vision tends to apply it to unlock the advantages of event data. Early attempts are filtering-based [2, 20, 32] or shallow learning [21] methods. These methods process the events asynchronously and, therefore, preserve the properties of the event stream. However, since they rely on complex descriptors, they are difficult to extend to complex tasks. Models based on CNNs are gradually developed and applied to event-based vision, such as optical flow estimation [7, 23], action recognition [3, 26], and object classification [25, 27]. Though achieving SOTA results, these CNN-based models neglect the sparse and asynchronous nature of events, leading to redundant computation. On the contrary, SNN-based models [6, 33] show an advantage in this respect but are harder to train due to the lack of efficient learning rules.

Recently, point cloud-based [30] and graph-based [11–13, 15, 34] models are proposed and outperformed most CNN- & SNN-based methods. In particular, RG-CNN [13] achieves SOTA results on three visual tasks and reduces model complexity, but ignores the detailed motion cues since the network focuses on extracting spatial relations at the shallow layers. The EV-VGCNN finds both adjacent and distant neighborhoods for its voxel-wise graph vertices, and then locally aggregates features within the neighborhoods at each layer. While improving performance, the multi-scale strategy introduces additional computational costs. The VMV-GCN aggregates features and updates coordinates of voxel-wise vertices dynamically layer by layer, which further obtains significant improvements in object classification and action recognition [12]. However, VMV-GCN does not allow for hierarchical learning which weakens the modeling ability of global information. Recently, some models introduce transformers to event data such as [14, 16, 17, 35]. Although the introduction of the transformer enhances the modeling of spatio-temporal information, there are still limitations, such as the lack of modeling of local detailed information in EvT [17] and [16]. The voxel-based transformer [36] and multi-scale transformers [37, 38] in point clouds show great progress in 3D object detection, but further design of these frameworks is required to shift to the event data.

To this end, this paper proposes the VMST-Net to aggregate spatio-temporal dependencies within different receptive fields at each layer of the network in a local-to-global manner, and finally obtain effective global features. Consequently, our model outperforms most other models with a significant improvement in accuracy and computational complexity.

III. The Proposed Method

A. Overview

The overall architecture of the proposed voxel-based multi-scale transformer network (VMST-Net) is presented in Fig. 3. It first transforms the input event stream into voxels by a voxel token generation block. Each voxel is treated as a "token". These voxel tokens are processed by a transformer block with a multi-scale multi-head self-attention (MSMHSA) layer

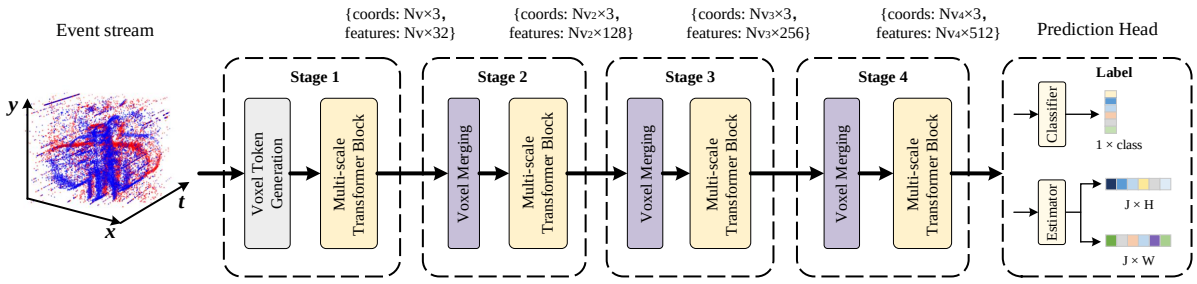

Fig. 3. The architecture of voxel-based multi-scale transformer network. An event stream is first transformed to voxels by the voxel token generation block. These voxels are further treated as tokens and processed by multi-scale transformer blocks and voxel merging blocks. The network can perform scenes that include both static and dynamic object tasks, such as object classification, action recognition, and human pose estimation tasks.

to encode both semantic and motion cues. The transformer block maintains the number of voxels and the number of feature channels, and together with the voxel token generation block is referred to as "Stage 1". To produce a hierarchical representation and save computations, the number of voxels is reduced by voxel merging layers as the network gets deeper. And the number of feature channels increases after the voxel merging block. A multi-scale transformer block is applied afterward for feature aggregation. This block of voxel merging and feature aggregation is denoted as "Stage 2". The procedure is repeated twice, as "Stage 3" and "Stage 4".

In this work, we consider three different visual tasks, including object classification, action recognition, and human pose estimation, and design a prediction head on top of the backbone network for each task. Specifically, for object classification and action recognition tasks, the prediction head is designed to first perform pooling operations to obtain a global spatio-temporal feature vector. Then, three fully connected (FC) layers are stacked as a classifier to map the feature vector into the classification logits. Each of the first two FC layers is followed by a Batch Normalization (BN) layer and a LeakyReLU function. For human pose estimation, we follow the design of the output layer in recent work [14], which first applies the global average pooling to get the global features and then introduces two FC layers with the ReLU function to transform the global features for prediction. Here, we follow SimDR [39] to output the x-axis vector x_v and y-axis vector y_v in a decoupled manner. The $argmax$ operation is further used to decode x_v and y_v into the joints location $(pred\_x, pred\_y)$ of the human body:

$$\begin{aligned} pred\_x &= \underset{j \in J}{argmax}(x_v(j)), \\ pred\_y &= \underset{j \in J}{argmax}(y_v(j)), \end{aligned} \tag{1}$$

where J refers to the total number of joints.

B. Voxel Token Generation

Given an event camera with a pixel grid size of $H \times W$, an event stream with N_e events could be denoted as the following four-element tuple:

$$\varepsilon = \{e_i\}_{i=1}^{N_e} = \{(x_i, y_i, t_i, p_i)\}_{i=1}^{N_e}, \tag{2}$$

where (x_i, y_i, t_i) refers to the spatio-temporal coordinate of the event e_i with $(x_i, y_i) \in [0, W-1] \times [0, H-1]$; $p_i \in \{-1, +1\}$ can be seen as the polarity of the event e_i with 1 and -1 representing "ON" and "OFF" events, respectively. The voxel token generation block, as shown in Fig. 4, is designed to organize the input event stream into voxels and calculate the input feature of each voxel.

Voxelization. Since there typically exists a large-scale difference between the spatial coordinate (x, y) and the temporal coordinate t of the event, we first normalize the timestamp t_i of each event e_i into $[0, B-1]$ by the following equation:

$$t_i^* = \frac{(t_i - t_{min})}{t_{max} - t_{min}} \times (B-1). \tag{3}$$

where t_{max} and t_{min} refer to the maximum and minimum timestamps among N_e events, respectively. After normalization, the 3D space, ranging from $[0, W-1]$, $[0, H-1]$, $[0, B-1]$ along the x, y, and t axes respectively, are divided into equally distributed voxels with the size of each voxel as (v_w, v_h, v_t). The resulting voxel grid is of size $\frac{H}{v_h} \times \frac{W}{v_w} \times \frac{B}{v_t}$. Each event is then grouped into these voxels based on its spatial coordinate (x_i, y_i) and normalized timestamp t_i^*. It's worth noting that the number of events in each voxel is different. We regard the coordinates of each voxel within the 3D grid as voxel locations, represented by $\mathbf{u} = (u^x, u^y, u^t)$.

Voxel Selection. Some empty voxels will be produced from the voxelization procedure due to the sparsity of events, as shown in Fig. 4. We filter out these empty voxels and denote the remaining number of nonempty voxels as $N_v^{'}$. Moreover, for the sake of optimizing the computational complexity, similar to [12], the non-empty voxels are further reduced to N_v voxels by downsampling. Specifically, when $N_v^{'} > N_v$, we select top-N_v non-empty voxels with the largest number of events inside. For the event stream with $N_v^{'} \leq N_v$, we select $0.95N_v$ voxels that include more events than others to filter out the noisy events and then expand the filtered voxels to N_v voxels by zero-padding.

Voxel Feature Encoding. As each voxel typically contains multiple events, an effective method to aggregate these points to derive the voxel's input feature is required. Current works calculate voxel features by projecting events within voxels into 2D image patches through contracting along single or multiple dimensions and then flattening the resulting 2D features to obtain feature vectors [11, 12], leading to the loss of spatio-temporal information. To alleviate this issue, we propose a simple but effective voxel feature encoding module, dubbed

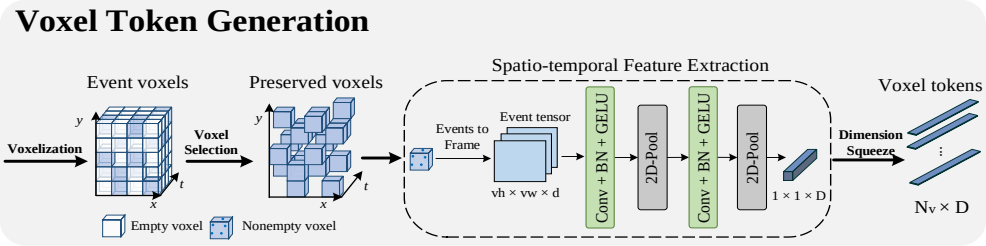

Fig. 4. The voxel token generation block. We first split the event stream into voxels in space-time 3D volume and select N_v voxels with dense internal events. Each voxel is processed by a spatio-temporal feature extraction module (STFE) to obtain the feature. Finally, the voxels are treated as voxel tokens.

a spatio-temporal feature extraction (STFE) module, whose detailed architecture is shown in Fig. 4. For each voxel v at location (u^x, u^y, u^t), STFE first reshapes the event stream within the voxel into an event tensor $\mathbf{E}_v$ of size $v_h \times v_w \times d$, where d is the number of temporal bins. For each temporal bin $n \in \{0, 1, \cdots, d-1\}$, the value $\mathbf{E}_v(x, y, n)$ at location (x, y) is computed by accumulating the polarity values of events from the specific location within the given time span. Specifically, we follow [23] to populate the event tensor using temporal bilinear interpolation, where each event $e_j = (x_j, y_j, t_j^*, p_j)$ in the voxel contributes its polarity to the two closest temporal bins as follows:

$$\mathbf{E}_v(x, y, n) = \sum_j p_j * max(0, 1 - |n - t_j^v|) * \mathbb{I}(x_j^v = x, y_j^v = y), \tag{4}$$

where

$$t_j^v = \frac{(t_j^* - u^t * v_t)}{v_t} \times (d - 1), \tag{5}$$

$$x_j^v = x_j - u^x * v_w, \tag{6}$$

$$y_j^v = y_j - u^y * v_h, \tag{7}$$

and $\mathbb{I}(\cdot)$ refers to the indicator function whose value is equal to 1 only when the condition is true, otherwise 0.

Once the event tensor $\mathbf{E}_v$ is available, we regard it as a dense 2D frame and pass it through two consecutive 2D convolution layers to learn the spatial and channel dependencies, aiming to capture the spatio-temporal dependencies between events. Here, each of the convolutional layers is followed by a BN layer and a GELU function. Then a pooling layer is introduced after each convolution layer to reduce the size of the feature maps. The output feature map of the STFE block is squeezed into a feature vector of length D, which is viewed as the input feature of the voxel. In summary, each representative voxel is regarded as one token and is associated with two attributes, namely, the coordinate $\mathbf{u} \in \mathbb{R}^{1\times 3}$ in the spatio-temporal 3D space and the feature $f \in \mathbb{R}^{1\times D}$. All voxels share the STFE module, resulting in features $F \in \mathbb{R}^{N_v \times D}$ and coordinates $U \in \mathbb{R}^{N_v \times 3}$.

C. Multi-scale Transformer Block

The proposed multi-scale transformer block consists of a multi-scale multi-head self-attention (MSMHSA) layer and a feed-forward network (FFN) layer, as shown in Fig. 5. Different from regular multi-head self-attention layer, MSMHSA first performs multi-scale multi-head self-attention on input features, which allows different heads to perform self-attention over different scale 3D neighborhoods. Then, it aggregates the obtained features from different heads through a multi-scale fusion (MSF) module, which dynamically weights the coarse-to-fine features from different heads.

Multi-scale Multi-head Self-attention. At each stage, it takes the features $F \in \mathbb{R}^{N\times C}$ and coordinates $U \in \mathbb{R}^{N\times 3}$ of voxel tokens as inputs, where N and C refer to the total number of voxel tokens and the number of feature channels, respectively. The features are first processed by a Layer Normalization (LN) layer and then projected into three different feature spaces to generate query matrix $Q \in \mathbb{R}^{N\times C}$, key matrix $K \in \mathbb{R}^{N\times C}$, and value matrix $V \in \mathbb{R}^{N\times C}$ as follows:

$$Q = FW_Q, K = FW_K, V = FW_V, \tag{8}$$

where $W_Q, W_K, W_V \in \mathbb{R}^{C\times C}$ are the learnable parameters of the linear projection.

To extract distinguishable appearance and motion features from voxels, we introduce the multi-scale strategy into the regular multi-head self-attention layer. This is motivated by the observation that adjacent 3D neighbors hold subtle motion and local details while distant neighbors carry obvious changes and large-scale objects. To avoid introducing additional computations, instead of applying different receptive field scales directly to the original input feature, we split Q, K, and V evenly along the channel dimension into two heads $\{Q_i, K_i, V_i\}_{i=1}^2$ with $Q_i, K_i, V_i \in \mathbb{R}^{N\times \frac{C}{2}}$ and perform self-attention over different scale 3D neighborhoods at different heads as described in Fig. 5 (a). Specifically, for the i-th head, we construct neighborhoods of size k_i for each voxel using the K-Nearest Neighbors (kNN) algorithm to obtain the key and value matrices $K_i^{'}, V_i^{'} \in \mathbb{R}^{N\times k_i \times \frac{C}{2}}$ from neighbors. Particularly, to capture the position relationships between voxel tokens, we add the learnable relative position bias term B_i based on coordinates U defined by [40] to the self-attention. The Q_i, $K_i^{'}$, $V_i^{'}$, and B_i of i-th head are further used to calculate the self-attention:

$$out_i = Softmax(\frac{Q_i(K_i^{'})^T}{\sqrt{d}} + \theta(B_i))(V_i^{'} + B_i), \tag{9}$$

where d and out_i are the scaling factor and the output of the self-attention step, respectively; $\theta(\cdot)$ denotes the sum function used to squeeze the feature dimension.

The performance of the transformer- & graph-based models relies heavily on the quality of the 3D neighborhoods,

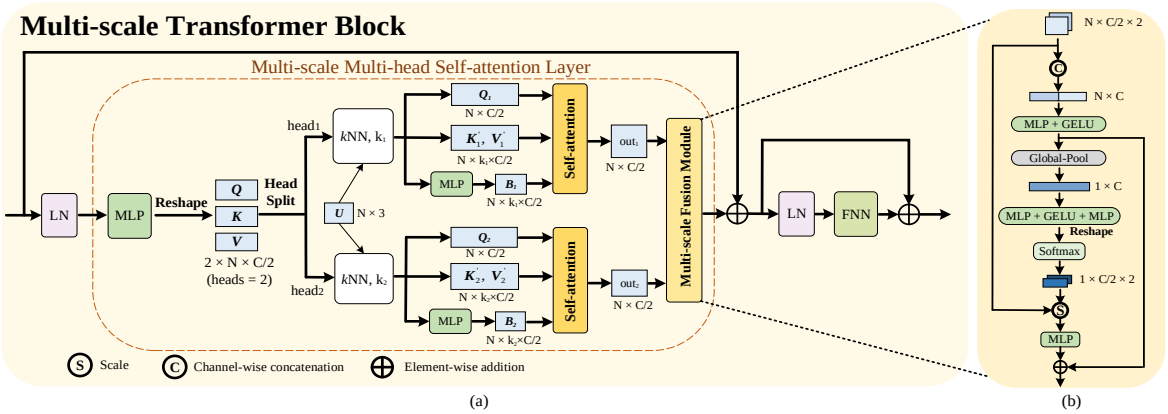

Fig. 5. (a) Multi-scale transformer block. This block applies the multi-scale multi-head self-attention mechanism (MSMHSA) on voxel tokens to capture features in different receptive fields at different heads. For ease of representation, only a 2-head design is shown. (b) Multi-scale fusion module (MSF). This module is introduced in the MSMHSA layer to fusion multi-scale features.

where the networks perform feature aggregation. In prior works [12, 17], the 3D neighborhoods across all layers are usually at the same scale, which induces a poor trade-off between accuracy and computational overhead. To this end, we aggregate features in a local-to-global manner by enlarging enlarge the coverage of 3D neighborhoods as the network gets deeper. Specifically, for the first two stages, we assign a local k_i-neighbor to each head, that is, $k_{i\in\{1,2\}} < N$. Thus, the local coarse-grained and fine-grained spatio-temporal features are extracted while low computational costs are required. In the last two stages, to provide the global information for visual tasks, we fix $k_1 = N$ for one head, while the other head still satisfies $k_2 < N$.

Multi-scale Fusion Module. Recently, channel-wise concatenation [41, 42] or element-wise addition [11] are commonly used for multi-scale feature fusion. However, it is difficult to determine the optimal receptive field scale in different stages. Simple fusion may result in a sub-optimal receptive field scale, which affects the performance of the model. Therefore, we propose a powerful multi-scale fusion (MSF) module to dynamically weight the diverse-grained features from different heads, as shown in Fig. 5 (b).

The main idea of MSF is to first fuse multi-scale features from different heads and then generate the weight vector based on the integrated features. As for feature fusion, MSF first performs channel-wise concatenation on features $O \in \mathbb{R}^{N\times\frac{C}{2}\times 2}$ from the two different heads to obtain the concatenated feature $O_c \in \mathbb{R}^{N\times C}$. Then the multi-scale features interact by applying an FC layer followed by a GELU function to O_c. Finally, the obtained feature O_f is fed into a global pooling layer to derive the global feature $O_g \in \mathbb{R}^{1\times C}$. The above feature fusion procedure could be described as follows:

$$O_f = \mathrm{GELU}(\mathrm{FC}(O_c)), \tag{10}$$

$$O_g = \mathrm{Pooling}(O_f). \tag{11}$$

After that, two FC layers with a GELU function between them and a softmax function are performed sequentially on the global feature O_g to generate the weight vector $W \in \mathbb{R}^{1\times\frac{C}{2}\times 2}$ for each scale:

$$W = \mathrm{Softmax}(\mathrm{FC}(\mathrm{GELU}(\mathrm{FC}(O_g)))). \tag{12}$$

To re-weight the original multi-scale features O, the weight vector W is applied to them by the scaling operation as the same in SE-layer [43], which refers to channel-wise multiplication between attention weight W and features O at each head. We concatenate the re-weighted multi-scale features along the channel dimension, then pass them to an FC layer to further fuse them to generate O_w as follows:

$$O_w = \mathrm{FC}(\mathrm{Concat}(\mathrm{Scale}(W, O))). \tag{13}$$

Finally, a skip connection between O_w and O_f is added to obtain the output of the MSMHSA layer as follows:

$$O_{MSMHSA} = O_f + O_w. \tag{14}$$

D. Voxel Merging

To produce a hierarchical representation while saving computational costs, it is necessary to reduce the number of voxel tokens in the network progressively. We borrow the idea of a transition down module in Point Transformer [40] to perform downsampling by the farthest point sampling (FPS) [44] in the voxel merging block. Note that to reduce information loss, we aggregate neighbor features to enhance the features of sampled voxel tokens. Given a sampled voxel token with coordinate $\mathbf{u}$, its K neighbors are constructed according to the distance of coordinates. For k-th neighbor with feature f_k and coordinate $\mathbf{u}_k$, to preserve geometric distribution and semantic information, the geometric context and feature context are combined:

$$\Delta C_k = concat(f_k, \mathbf{u}_k - \mathbf{u}), \tag{15}$$

where $\mathbf{u}_k - \mathbf{u}$ denotes the geometric relationship between the sampled token and the k-th neighbor; $\Delta C_k \in \mathbb{R}^{1\times(C_{in}+3)}$, where C_{in} denotes the number of input feature channels. Then $\{\Delta C_k\}_{k=1}^{K}$ of all neighbors goes through an FC layer, followed by an LN layer and max pooling operation to obtain the feature $\hat{f} \in \mathbb{R}^{1\times C_{out}}$ of the sampled voxel token, where

C_{out} is the number of output feature channels. The above procedure can be expressed as:

$$\hat{f} = \underset{k \in K}{max}(LN(FC(\Delta C_k))). \tag{16}$$

IV. Experiments and Results

In this section, we evaluate our voxel-based multi-scale transformer network in terms of accuracy and model complexity on event data collected in static scenes as well as dynamic scenes. Specifically, we focus on object classification in static scenes and action recognition in dynamic scenes, which require the network to provide representative semantic information and more motion cues, respectively. In addition to the classification task in dynamic scenes, we also evaluate the performance on the regression problem human pose estimation, which requires more detailed spatial and motion information. Besides, the effectiveness of our methods is further tested by ablation studies.

A. Datasets and Preprocessing

Object Classification. Several publicly available datasets, including N-MNIST [45], N-Caltech101 [45], N-Cars [21], CIFAR10-DVS [46], ASL-DVS [13], and N-ImageNet [47], are used for object classification performance evaluation. Here, N-MNIST and N-Caltech101 are converted from RGB images by moving an event camera in front of a static monitor displaying the original RGB images. N-ImageNet, the largest event-based dataset for classification to date, is generated in the same way. Instead, CIFAR10-DVS is recorded by moving an RGB image in front of a fixed event camera. Both N-Cars and ASL-DVS are directly recorded by the event camera in the real-world. For the N-MNIST, N-ImageNet, and N-CARS datasets, we use the officially split training set and testing set. For the N-Caltech101, CIFAR10-DVS, and ASL-DVS, we adopt the same split as in [13, 21], i.e., randomly select 20% of the data for testing and the rest for training.

Action Recognition. DVS128 Gesture Dataset [3], HMDB51-DVS [13], and UCF101-DVS [13] are three datasets of action recognition. DVS128 Gesture Dataset is a real-world dataset that recorded 11 different hand and arm gestures under different illumination conditions. We follow [12, 30] to use a sliding time window to select the sub-sequence of the event stream as inputs to the network. The window size is set to 0.5s and the step size is half of the sliding window size. HMDB51-DVS and UCF101-DVS are two more challenging datasets. Both of them are converted from APS-based sequences and are generated by displaying video in front of an event camera. These two datasets are the largest datasets for event-based action recognition and have 51 categories and 101 categories, respectively. Note that, similar to [13, 48], we do not perform any data preprocessing before feeding them into the model.

Human Pose Estimation. The DHP19 dataset [49] is the only real-world public dataset used for human pose estimation, which consists of four event streams generated by event cameras from four views. Similarly, we use the $MeanLabel$ described in [49] as 3D human pose labels. Then, the 3D labels are transformed into 2D labels by projection matrices, and the number of joints J is 13. The dataset contains 17 subjects of which S1-S12 of them are used for training and S13-S17 for validation. We follow [14, 49] to use the event data from the cameras fixed at two front views, and perform the same denoising and filtering methods on these event streams.

B. Implementation Details

Voxel Token Generation Block. According to the duration of the event streams, we set the normalization parameter $B = 9$ in equation (3) for N-Caltech101, ASL-DVS, N-MNIST, and DHP19 dataset, $B = 6$ for N-ImageNet and N-Cars, $B = 30$ for CIFAR10-DVS, and $B = 50$ for all action recognition datasets. The major difference within the used datasets lies in their spatial resolutions and the spatio-temporal distribution of events. On account of it, we set different sizes and numbers of voxels for them. For simple datasets, N-Cars and N-MNIST, we set (v_h, v_w, v_t) as $(5, 5, 3)$ and $(2, 2, 3)$, respectively, and set N_v as 256 and 512, respectively. For ASL-DVS, N-Caltech101, CIFAR10-DVS, DHP19 dataset, and all action recognition datasets, the (v_h, v_w, v_t) is $(10, 10, 3)$, while $(20, 20, 3)$ for N-ImageNet. Additionally, we set $N_v = 1024$ for N-Caltech101, CIFAR10-DVS, and N-ImageNet, $N_v = 512$ for ASL-DVS, DHP19 dataset, and DVS128 Gesture dataset, and $N_v = 2048$ for HMDB51-DVS and UCF101-DVS. In the STFE module, we set $d = 3$ of the event tensor for all datasets and only use a 2D convolution layer following a 2D pooling layer for N-MNIST and N-Cars. Particularly, we set different parameters for the convolution layer and the pooling layer according to the voxel size of different datasets. Furthermore, we add an LN layer following the 2D pooling to normalize the output of the STFE module for all datasets and a dropout with a probability of 0.1.

Network Details. For all datasets, the number of input feature channels for the four stages is $32, 128, 256$, and 512. We set the output number of the tokens for the three downsampling blocks as 128, 64, and 16 for N-Cars. For N-MNIST, ASL-DVS, DHP19 dataset, and DVS128 Gesture dataset, the number of downsampling voxel tokens is 256, 64, and 16, respectively. The downsampling rates of the three voxel merging blocks are $[4, 4, 4]$ for the remaining datasets. We aggregate information of $K = 16$ neighbors for each sampled voxel token. We take $[20, 25]$ as the kNN-based multi-scale size in the first two stages, using $[20, N_{v3}]$ and $[8, N_{v4}]$ for stage 3 and stage 4, respectively. Note that, we don't explore the impact of different multi-scale sizes on performance, and therefore a better combination of multiple scales may exist. Moreover, for object classification and action recognition, to alleviate the overfitting problem, we add dropout layers with a probability of 0.5 following the first two FC layers.

Training Settings. For object classification and action recognition, we train the network by optimizing the cross-entropy loss for 250 epochs, except for the N-ImageNet which uses 200 epochs. The SGD optimizer with momentum 0.9 and weight decay 1e-4 is used in the training process. We use 0.05 as the initial learning rate and decay it to 5e-4 by cosine annealing scheduler. The batch size is 32 for N-Caltech101 and CIFAR10-DVS, 256 for ImageNet, and 64 for other datasets.

TABLE I
THE OBJECT CLASSIFICATION ACCURACY OF OUR METHOD AND OTHER METHODS.

Method	Type	N-MNIST	N-Caltech101	N-CARS	CIFAR10-DVS	ASL-DVS	N-ImageNet
EST [27]	Frame	0.99	0.753	0.919	0.634	0.979	0.489
MVF-NET [25]	Frame	0.981	0.687	0.927	0.599	0.971	-
Matrix-LSTM [28]	Frame	0.986	0.738	0.927	0.631	0.98	0.322
DisT [47]	Frame	-	0.681	0.908	0.626	-	0.484
AMAE [50]	Frame	0.983	0.694	0.936	0.62	0.984	-
H-First [2]	Point	0.712	0.054	0.561	0.077	-	-
Gabor-SNN [33]	Point	0.837	0.196	0.789	0.245	-	-
HOTS [20]	Point	0.808	0.21	0.624	0.271	-	0.443
HATS [21]	Point	0.991	0.642	0.902	0.524	-	0.471
EventNet [29]	Point	0.752	0.425	0.750	0.171	0.833	-
RG-CNNs [13]	Point	0.99	0.657	0.914	0.54	0.901	-
ECSNet [48]	Point	0.992	0.693	0.946	0.727	0.997	-
EV-VGCNN [11]	Voxel	0.994	0.748	**0.953**	0.651	0.983	-
VMV-GCN [12]	Voxel	**0.995**	0.778	0.932	0.69	0.989	-
EVSTr [51]	Voxel	-	0.797	0.941	-	0.997	-
VMV-PointTrans [12, 40]	Voxel	**0.995**	0.773	0.931	0.75	**0.999**	-
VMST-Net (Ours)	Voxel	**0.995**	**0.822**	0.944	**0.753**	0.998	**0.603**

TABLE II
THE COMPUTATIONAL COMPLEXITY (GMACS) AND MODEL SIZE (MB) ON THE N-CALTECH101 DATASET.

Method	Type	GMACs†	Size (MB)	Runtime (ms)‡
EST [27]	Frame	4.28	21.38	10.8
MVF-NET [25]	Frame	5.62	33.62	14.3
Matrix-LSTM [28]	Frame	4.82	21.43	16.6
EventNet [29]	Point	0.91	2.81	-
RG-CNNs [13]	Point	-	19.46	-
ECSNet [48]	Point	-	0.18	-
EV-VGCNN [11]	Voxel	0.7	0.84	14.8
VMV-GCN [12]	Voxel	1.3	0.86	11.6
EVSTr [51]	Voxel	0.81	0.93	13.3
VMV-PointTrans [12, 40]	Voxel	9.44	3.69	77.7
VMST-Net (Ours)	Voxel	0.44	3.61	42.5

† 1 GMACs = 10^9 MACs. ‡ Runtime includes data loading and input construction time on the CPU and inference time on the GPU.

For human pose estimation, we train our model every 30 epochs with the Kullback–Leibler divergence (KLD) loss, and the batch size is set to 16. We use Adam optimizer with an initial learning rate of 0.001 which is dropped by 2× at epochs 10 and 20.

C. Results

Object Classification. The classification accuracy of our method and that of counterparts on six datasets is shown in Table I. These methods are roughly divided into three categories, i.e., frame-based [25, 27, 28, 47, 50], point-based [2, 13, 20, 21, 29, 33, 48], and voxel-based [11, 12, 51] methods, according to how they process event data. The frame-based methods are all trained from scratch. Additionally, we construct a voxel-based transformer method for performance comparison, dubbed VMV-PointTrans, by taking the voxels of VMV-GCN [12] as inputs to the Point Transformer [40] and training it from scratch. Similar to EV-VGCNN [11], we report the model complexity in terms of the number of parameters and multiply-accumulate operations (MACs) on the N-Caltech101 dataset and the average inference time for processing each sample on the N-Cars dataset in Table II. All methods are implemented on a workstation with a CPU (Intel Xeon E5-2640 v4), a GPU (NVIDIA TITAN Xp), and 251GB of available RAM.

Results in Table I show that the proposed VMST-Net obviously outperforms those frame-based models on all datasets. Meanwhile, it achieves superior performances on most of the datasets compared to those point- & voxel-based models. Specifically, on the simple datasets N-MNIST and ASL-DVS, our network achieves comparable results with other counterparts in accuracy. However, on the more challenging datasets N-Caltech101 and CIFAR10-DVS, VMST-Net shows significant improvements in accuracy with only 1024 voxels. Similar superiority of our model is found on N-ImageNet. We owe the advantages of our network to the following four points: (i) Compared with the frame-based methods, our VMST-Net is capable of capturing and encoding motion information with high temporal resolution between events by projecting events within each voxel into a multi-channel frame along the time dimension and subsequently processing it through 2D convolutions. (ii) Compared with the point-based methods, voxels carry more spatio-temporal information than raw event points. In addition, voxel-based methods could filter out noisy events during the voxel selection process. (iii) Compared with the voxel-based methods, the voxel features encoded by the STFE module not only preserve temporal information but also model the relative spatial position information, which retains more spatial information during voxelization. (iv) Our transformer-based backbone with the multi-scale mechanism captures coarse-to-fine features at each stage in a local-to-global manner. In the MSMHSA layer, the proposed MSF module can adaptively select useful information from multi-scale features. Therefore, our network can model a variety of local appearance information as well as global semantic information.

As shown in Table II, our proposed VMST-Net significantly outperforms the frame-based methods in both computational costs and model size at the expense of a relatively low inference efficiency. The relatively longer run time could be mainly explained by the kNN-based neighbor searching for feature aggregation and FPS-based voxel downsampling in the voxel merging procedure. Compared to the point- & voxel-based methods, the VMST-Net achieves better accuracy while maintaining the lowest computational complexity (0.44 G).

TABLE III
THE ACTION RECOGNITION ACCURACY, COMPUTATIONAL COMPLEXITY, AND MODEL SIZE OF OUR METHOD AND OTHER METHODS.

Method	Type	DVS128 Gesture	UCF101-DVS	HMDB51-DVS	GMACs	Size (MB)
Event Frames + I3D [13]	Frame	0.965	0.635	0.466	33.53	12.28
Event Frames + 3D-ResNet [13]	Frame	0.955	0.579	0.438	14.99	63.5
EvT [17]	Frame	0.962	-	-	0.1	0.48
PointNet++ [44]	Point	0.941	-	-	0.43	1.47
Wang WAC2019 [30]	Point	0.953	-	-	-	-
RG-CNNs + res3d [13]	Point	0.972	0.673	0.497	-	12.43
ECSNet [48]	Point	**0.986**	0.702	-	-	-
EV-VGCNN [11]	Voxel	0.957	-	-	0.46	0.82
VMV-GCN [12]	Voxel	0.975	-	-	0.33	0.84
EVSTr [51]	Voxel	**0.986**	-	-	1.1	2.88
VMV-PointTrans [12, 40]	Voxel	0.956	-	0.461	2.34	3.43
VMST-Net (Ours)	Voxel	0.978	**0.787**	**0.595**	0.31	3.61

TABLE IV
THE HUMAN POSE ESTIMATION ERROR AND MODEL COMPLEXITY OF OUR METHOD AND OTHER METHODS ON THE DHP19 DATASET.

Method	Type	MPJPE_{2D}	MPJPE_{3D}	Size (MB)	GMACs
Pose-Res50 [52]	Frame	**5.28**	**59.83**	34	12.91
LeVit-128S [53]	Frame	7.68	87.79	7.87	0.2
DHP19 [49]	Frame	7.67	87.9	0.22	3.51
Ras-PointNet [14]	Point	7.29	82.46	4.46	1.19
Ras-DGCNN [14]	Point	6.83	77.32	4.51	4.91
Ras-PointTrans [14]	Point	6.46	73.37	3.65	5.03
VMV-PointTrans [12, 40]	Voxel	9.13	103.23	3.67	5.98
VMST-Net (Ours)	Voxel	6.45	73.07	3.59	0.38

The low computations of VMST-Net could be attributed to the delicately designed STFE module and MSMHSA layer, which allow us to preserve rich semantic and motion features with fewer voxels. The computational costs are further reduced as all transformer blocks have only one layer, and only a little additional computation is introduced in the MSF module of the MSMHSA layer. However, the VMST-Net shows limitations in model size and inference efficiency compared to other voxel-based models. Our parameters mainly come from the last stage, where we set the length of the input vertex feature vector to be 512, which is larger than 128 of EV-VGCNN, VMV-GCN, and EVSTr, resulting in a larger model size. The relatively lower inference efficiency could be mainly due to the FPS-based voxel downsampling procedure. In comparison, VMV-GCN does not perform any voxel merging, while EV-VGCNN and EVSTr employ random sampling (RS) to merge voxels. Although FPS requires longer processing times to generate sampling points iteratively, it provides better spatial coverage and uniformity than RS.

Action Recognition. We compare our model with several existing frame-based [17], point-based [13, 30, 44, 48], and voxel-based [11, 12, 51] counterparts. In Table III, we list the classification accuracy on three datasets as well as the MACs and the parameters of all models on the DVS128 Gesture Dataset. As in object classification, we train the VMV-PointTrans network from scratch on three datasets. Note that since the VMV-PointTrans has a sharp drop in accuracy on UCF101-DVS, we do not report its results in the table.

It is observed from Table III that on the simple gesture dataset DVS128, both ECSNet and EVSTr achieve the best performance, which could be explained by the fact that these two methods design the specific long-term message encoding modules tailored for this task to fully exploit the temporal dependency. Although not the best, the proposed VMST-Net still achieves impressive performance due to its strong power in encoding spatial-temporal dependency between events. It is worth noting that our proposed VMST-Net brings significant improvements on the challenging datasets UCF101-DVS and HMDB-DVS compared to all other baseline models. The relatively poor performances of RG-CNN and ECSNet on these two challenging datasets might be due to the fact that their representations carry insufficient information to describe the complex scenes. Specifically, RG-CNN fails to capture detailed motion cues at shallow layers, while ECSNet shows limitations in encoding multi-scale motion and semantic cues. In contrast, our VMST-Net retains more local details inside voxels due to the STFE module. Our hierarchical architecture also extracts multi-scale appearance and motion features from local to global. In addition, the results also show that our method shows advantages in the computational complexity (0.31G) and model complexity (3.61M) over all counterparts except for EvT. The low model complexity of frame-based EvT could be explained by the fact it processes event frames asynchronously, but the focus on extracting global spatio-temporal features results in low accuracy. In general, our network is capable of capturing comprehensive local and global spatial information as well as motion cues in dynamic scenes with low computational complexity.

Human Pose Estimation. We employ the commonly used Mean Per Joint Position Error (MPJPE) as the metric to evaluate the performance of our model on human pose estimation in both 2D and 3D space. The definition of MPJPE is described by the following equation:

$$\text{MPJPE} = \frac{1}{J}\sum_{i}^{J} \parallel pred_i - gt_i \parallel_2, \tag{17}$$

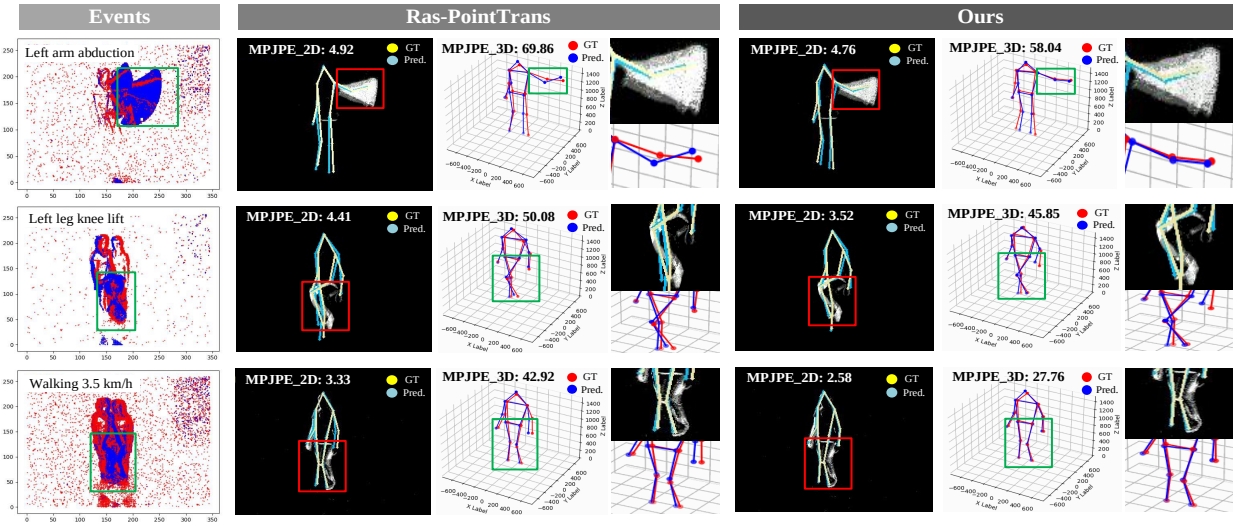

Fig. 6. Visualization of human pose estimation results of the proposed VMST-Net and Ras-PointTrans on the DHP19 dataset. The first column is the event stream visualized in 2D. The second to fourth columns are Ras-PointTrans's 2D pose estimation results (yellow for ground truth, blue for prediction), 3D pose estimation results (red for ground truth, blue for prediction), and the detailed results of the major moving parts of the body (marked by boxes), respectively, and the fifth to the last columns are our results. Our method achieves great results compared to Ras-PointTrans on three different movements, including "Left arm abduction", "Left leg knee lift", and "Walking 3.5 km/h" from top to bottom.

where $pred_i$ and gt_i represent the prediction and ground truth of the i-th joint, respectively; J refers to the total number of joints. Note that, the measurement unit of MPJPE for 2D human pose estimation is pixel and is millimeter in 3D space. Similar to [14], our network first predicts the 2D human pose, then transform it into a 3D human pose by triangulation [49] based on two projection matrices and positions of the front two cameras.

We present the MPJPE, the number of model parameters, and the number of computations (GMACs) of our model and other comparable models in Table IV. On one hand, compared to the frame-based Pose-Res50 [52], although the estimation errors of our model are relatively larger on both 2D and 3D pose estimation tasks, it requires much fewer model parameters and computational costs (3.59M parameters and 0.38G MACs). On the other hand, compared to the transformer-based method VMV-PointTrans and Ras-PointTrans, our model achieves the lowest estimation error on both 2D and 3D pose estimation tasks while maintaining much lower computational complexity, suggesting that our model is capable of capturing more detailed semantic and motion cues at a low computational cost. Such superiority could be attributed to the proposed voxel-based representation technique and the multi-scale feature aggregation mechanism. In addition, we also visualize the human pose estimation performances of both our model and Ras-PointTrans [14] in Fig. 6. It is observed that our model is capable of providing high-precision estimates for most joints compared to Ras-PointTrans, especially for those with movements that are marked by boxes. The above observations demonstrate that the proposed voxel-based multi-scale transformer network is also suitable for dynamic scenes with higher requirements for motion details.

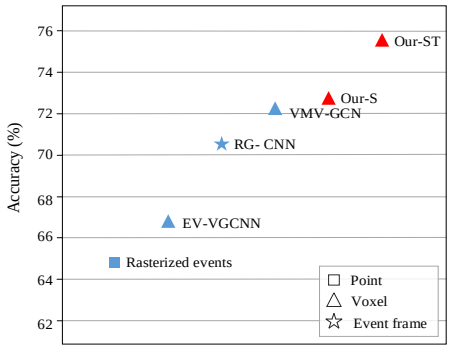

Fig. 7. Ablation study of the STFE Module on CIFAR10-DVS. The blue symbols indicate other methods, and the red ones indicate the methods of this work.

D. Ablation Study

In this part, we conduct ablation studies for object classification on the CIFAR10-DVS dataset to verify the effectiveness of the four key ideas of our network, including the spatio-temporal feature extraction (STFE) module, multi-scale (MS) strategy, multi-scale feature fusion (MSF) module, and the local-to-global (L-G) strategy.

STFE Module. We evaluate six different event representations with the same backbone and compare their classification accuracy in Fig. 7, including rasterized events [14], voxels in EV-VGCNN [11] and VMV-GCN [12], event frames generated in RG-CNN [13], our voxel-based representation and its variant. To make the frame-based, point-based, and existing voxel-based representations compatible with ours, we make the following modifications: (1) we use the feature embedding layer in [40] to project rasterized event points to tokens; (2) we split the event frames into patches, and apply the patch embedding used in [54] on representative patches to generate tokens. In addition, our proposed model is briefly denoted

TABLE V
ABLATION STUDIES OF EACH COMPONENT OF OUR VMST-NET ON DATASETS CIFAR10-DVS AND DHP19.

Structure	Stage1-2		Stage3-4		CIFAR10-DVS			DHP19			
	MS	MSF	MS	MSF	Accuracy	GMACs	Size(MB)	MPJPE_2D	MPJPE_3D	GMACs	Size(MB)
L-G	✗	✗	✗	✗	0.749	0.43	2.92	6.891	78.48	0.36	2.9
	✓	✗	✓	✗	0.744	0.43	2.92	7.124	79.88	0.36	2.9
	✓	✓	✗	✗	0.742	0.43	2.96	6.7	76.46	0.37	2.93
	✗	✗	✓	✓	0.745	0.44	3.58	7.2	81.86	0.37	3.55
	✓	✓	✓	✓	0.753	0.44	3.61	6.45	73.07	0.38	3.59
L-L	✓	✓	✓	✓	0.75	0.44	3.61	7.23	81.78	0.38	3.59
G-G	✓	✓	✓	✓	0.758	0.51	3.61	6.75	76.41	0.40	3.59

Ours-ST, while Ours-S refers to the variant model without time information encoding in the STFE module by setting d to be 1. The number of tokens is set to 1024 for all representations, except rasterized 2048. It shows that voxel-wise and frame-wise input work better than point-wise input because individual points carry less spatio-temporal information. The frame-based representation loses a lot of information as events accumulate, resulting in worse performance than ours. Comparing the results of Ours-S and EV-VGCNN shows that our STFE module has strong spatial information modeling capabilities. And we only adapt spatial feature extraction to achieve comparable results with VMV-GCN. When more temporal information is encoded, our model Ours-ST achieves a significant gain of 2.5% compared to Ours-S. Thus, the STFE module can also extract temporal information at the same time, which can provide more motion information.

MS Strategy, MSF Module, and L-G strategy. We perform ablation studies on object classification (CIFAR10-DVS) and human pose estimation (DHP19 dataset), corresponding to the static and dynamic scenes, respectively. Experiment details are the same as described in Section IV.B, except for changing the network structure and components. Table V shows the results under different settings. "L-L" and "G-G" indicate that the network uses local multi-scale and local-global scale at all layers, respectively. "✗" and "✓" denote with and without the module, respectively. Note that we apply an FC layer to fuse multi-head information when MSF is removed. Take dataset CIFAR10-DVS from a static scene as an example for illustration. According to the results of the "L-G" structure, using only the MS mechanism in all stages (the second row), the accuracy drops from 0.749 to 0.744, showing that the MS mechanism cannot be simply introduced due to the impact of the sub-optimal scale among multiple scales. After applying the MSF module (the last row), the accuracy is improved by about 1%, verifying that the MSF module can effectively fuse multi-scale features, thus learning useful information. Comparing the model complexity of the first, second, and last rows, it can be found that our MS mechanism introduces no calculations and model parameters, and the MSF module only increases the 0.01G MACs and 0.69M parameters.The power of the proposed MSMHSA layer is also demonstrated on the DHP19 dataset from the dynamic scene. In addition, we also remove the multi-scale settings in the local and global stages and replace them with single local scales and global scales, respectively, both of which result in a decrease in accuracy on these two datasets. In conclusion, applying MS and MSF mechanisms to both local and global stages can effectively encode useful coarse-to-fine features, thereby improving the performance of the model. The results of "L-L", "G-G", and the last row of "L-G" show that the network processes the event stream in a local-to-global manner could save computational costs while capturing discriminative global features, achieving a trade-off between performance and computational overhead. Especially on DHP19, "L-G" can capture valuable global and detailed motion cues in addition to appearance semantics for human pose estimation tasks, bringing significant performance improvements.

V. CONCLUSION

The present work introduces a novel end-to-end framework, a voxel-based multi-scale transformer Network (VMST-Net), for event stream processing in a local-to-global manner. VMST-Net develops a simple but effective spatio-temporal feature extraction (STFE) block that encodes more spatio-temporal relations inside the voxels than other representations. VMST-Net captures coarse-to-fine features over voxels by the multi-scale transformer block at each layer across our network, modeling representative global features. The multi-scale features are dynamically fused by the multi-scale fusion (MSF) module. We demonstrate that our network achieves better or comparable performance in accuracy than the SOTA methods for different tasks in both static and dynamic scenes. Moreover, our method requires less computational overhead than prior methods, facilitating its use on low-power hardware. As a future line of research, we plan to explore more efficient downsampling algorithms and extend the network to dense prediction tasks, such as optical flow estimation and depth estimation.

REFERENCES

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

[2] G. Orchard, C. Meyer, R. Etienne-Cummings, C. Posch, N. Thakor, and R. Benosman, "HFirst: A temporal approach to object recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 10, pp. 2028–2040, 2015. 1, 3, 8

[3] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau,

M. Mendoza, J. Kusnitz, M. Debole, S. Esser, T. Delbruck, M. Flickner, and D. Modha, "A low power, fully event-based gesture recognition system," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7388–7397. 1, 3, 7

[4] B. Ramesh, S. Zhang, H. Yang, A. Ussa, M. Ong, G. Orchard, and C. Xiang, "E-TLD: Event-based framework for dynamic object tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3996–4006, 2021. 1

[5] J. Huang, S. Wang, M. Guo, and S. Chen, "Event-guided structured output tracking of fast-moving objects using a celex sensor," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 9, pp. 2413–2417, 2018. 1

[6] C. Lee, A. Kosta, A. Z. Zhu, K. Chaney, K. Daniilidis, and K. Roy, "Spike-FlowNet: Event-based optical flow estimation with energy-efficient hybrid neural networks," in *European Conference on Computer Vision*, 2020. 1, 3

[7] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "EV-FlowNet: Self-supervised optical flow estimation for event-based cameras," *ArXiv*, vol. abs/1802.06898, 2018. 3

[8] M. Liu and T. Delbruck, "EDFLOW: Event driven optical flow camera with keypoint detection and adaptive block matching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 9, pp. 5776–5789, 2022. 1

[9] H. Rebecq, T. Horstschaefer, G. Gallego, and D. Scaramuzza, "EVO: A geometric approach to event-based 6-dof parallel tracking and mapping in real time," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 593–600, 2017. 1

[10] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate SLAM? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 994–1001, 2018. 1

[11] Y. Deng, H. Chen, H. Liu, and Y. Li, "A voxel graph cnn for object classification with event cameras," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 1162–1171. 1, 2, 3, 4, 6, 8, 9, 10

[12] B. Xie, Y. Deng, Z. Shao, H. Liu, and Y. Li, "VMV-GCN: Volumetric multi-view based graph cnn for event stream classification," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1976–1983, 2022. 1, 2, 3, 4, 6, 7, 8, 9, 10

[13] Y. Bi, A. Chadha, A. Abbas, E. Bourtsoulatze, and Y. Andreopoulos, "Graph-based spatio-temporal feature learning for neuromorphic vision sensing," *IEEE Transactions on Image Processing*, vol. 29, pp. 9084–9098, 2020. 1, 3, 7, 8, 9, 10

[14] J. Chen, H. Shi, Y. Ye, K. Yang, L. Sun, and K. Wang, "Efficient human pose estimation via 3d event point cloud," in *2022 International Conference on 3D Vision (3DV)*, 2022. 3, 4, 7, 9, 10

[15] Y. Wang, X. Zhang, Y. Shen, B. Du, G. Zhao, L. Cui, and H. Wen, "Event-stream representation for human gaits identification using deep neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3436–3449, 2022. 1, 3

[16] Z. Wang, Y. Hu, and S.-C. Liu, "Exploiting spatial sparsity for event cameras with visual transformers," *ArXiv*, vol. abs/2202.05054, 2022. 2, 3

[17] A. Sabater, L. Montesano, and A. C. Murillo, "Event Transformer. A sparse-aware solution for efficient event data processing," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2022, pp. 2676–2685. 2, 3, 6, 9

[18] V. Vasco, A. Glover, and C. Bartolozzi, "Fast event-based harris corner detection exploiting the advantages of event-driven cameras," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4144–4149. 2

[19] X. Clady, S.-H. Ieng, and R. B. Benosman, "Asynchronous event-based corner detection and matching," *Neural networks : the official journal of the International Neural Network Society*, vol. 66, pp. 91–106, 2015. 2

[20] 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, 2017. 2, 3, 8

[21] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman, "HATS: Histograms of averaged time surfaces for robust event-based object classification," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1731–1740. 2, 3, 7, 8

[22] 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 *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5419–5427. 2

[23] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "Unsupervised event-based learning of optical flow, depth, and egomotion," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 989–997. 3, 5

[24] S. M. N. Uddin, S. H. Ahmed, and Y. J. Jung, "Unsupervised deep event stereo for depth estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 11, pp. 7489–7504, 2022. 3

[25] Y. Deng, H. Chen, and Y. Li, "MVF-Net: A multi-view fusion network for event-based object classification," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2021. 3, 8

[26] S. U. Innocenti, F. Becattini, F. Pernici, and A. Del Bimbo, "Temporal binary representation for event-based action recognition," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 10 426–10 432. 3

[27] D. Gehrig, A. Loquercio, K. Derpanis, and D. Scaramuzza, "End-to-end learning of representations for asynchronous event-based data," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp.

5632–5642. 3, 8

[28] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, "Matrix-LSTM: a differentiable recurrent surface for asynchronous event-based data," in *European Conference on Computer Vision*, 2020. 3, 8

[29] Y. Sekikawa, K. Hara, and H. Saito, "EventNet: Asynchronous recursive event processing," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3882–3891. 3, 8

[30] 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)*, 2019, pp. 1826–1835. 3, 7, 9

[31] S. Schaefer, D. Gehrig, and D. Scaramuzza, "AEGNN: Asynchronous event-based graph neural networks," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 12 361–12 371. 3

[32] G. Gallego, J. E. Lund, E. Mueggler, H. Rebecq, T. Delbruck, and D. Scaramuzza, "Event-based, 6-dof camera tracking from photometric depth maps," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 10, pp. 2402–2412, 2018. 3

[33] J. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in Neuroscience*, vol. 10, 08 2016. 3, 8

[34] Y. Li, H. Zhou, B. Yang, Y. Zhang, Z. Cui, H. Bao, and G. Zhang, "Graph-based asynchronous event processing for rapid object recognition," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 914–923. 3

[35] Y. Alkendi, R. Azzam, A. Ayyad, S. Javed, L. Seneviratne, and Y. Zweiri, "Neuromorphic camera denoising using graph neural network-driven transformers," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2022. 3

[36] J. Mao, Y. Xue, M. Niu, H. Bai, J. Feng, X. Liang, H. Xu, and C. Xu, "Voxel transformer for 3d object detection," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 3144–3153. 3

[37] Z. Zhou, X. Zhao, Y. Wang, P. Wang, and H. Foroosh, "Centerformer: Center-based transformer for 3d object detection," in *ECCV*, 2022. 3

[38] X. Feng, H. Du, Y. Duan, Y. Liu, and H. Fan, "Seformer: Structure embedding transformer for 3d object detection," *ArXiv*, vol. abs/2209.01745, 2022. 3

[39] Y. Li, S. Yang, S. Zhang, Z. Wang, W. Yang, S. Xia, and E. Zhou, "Is 2d heatmap representation even necessary for human pose estimation?" *ArXiv*, vol. abs/2107.03332, 2021. 4

[40] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 259–16 268. 5, 6, 8, 9, 10

[41] D. Lu, Q. Xie, K. Gao, L. Xu, and J. Li, "3DCTN: 3d convolution-transformer network for point cloud classification," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 24 854–24 865, 2022. 6

[42] D. Feng, Z. Wu, J. Zhang, and T. Ren, "Multi-scale spatial temporal graph neural network for skeleton-based action recognition," *IEEE Access*, vol. 9, pp. 58 256–58 265, 2021. 6

[43] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 8, pp. 2011–2023, 2020. 6

[44] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, 2017, pp. 5099–5108. 6, 9

[45] G. Orchard, A. Jayawant, G. Cohen, and N. V. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in Neuroscience*, vol. 9, 2015. 7

[46] H. Li, H. Liu, X. Ji, G. Li, and L. Shi, "CIFAR10-DVS: An event-stream dataset for object classification," *Frontiers in Neuroscience*, vol. 11, 2017. 7

[47] J. Kim, J. Bae, G. Park, D. Zhang, and Y. M. Kim, "N-ImageNet: Towards robust, fine-grained object recognition with event cameras," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 2146–2156. 7, 8

[48] Z. Chen, J. Wu, J. Hou, L. Li, W. Dong, and G. Shi, "Ecsnet: Spatio-temporal feature learning for event camera," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 2, pp. 701–712, 2023. 7, 8, 9

[49] E. Calabrese, G. Taverni, C. Awai Easthope, S. Skriabine, F. Corradi, L. Longinotti, K. Eng, and T. Delbruck, "DHP19: Dynamic vision sensor 3d human pose dataset," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019. 7, 9, 10

[50] Y. Deng, Y. Li, and H. Chen, "AMAE: Adaptive motion-agnostic encoder for event-based object classification," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4596–4603, 2020. 8

[51] B. Xie, Y. Deng, Z. Shao, H. Liu, Q. Xu, and Y. Li, "Event voxel set transformer for spatiotemporal representation learning on event streams," *ArXiv*, vol. abs/2303.03856, 2023. 8, 9

[52] B. Xiao, H. Wu, and Y. Wei, "Simple baselines for human pose estimation and tracking," in *European Conference on Computer Vision (ECCV)*, 2018. 9, 10

[53] B. Graham, A. El-Nouby, H. Touvron, P. Stock, A. Joulin, H. Jégou, and M. Douze, "LeViT: a vision transformer in convnet's clothing for faster inference," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 12 239–12 249. 9

[54] K. Li, Y. Wang, J. Zhang, P. Gao, G. Song, Y. Liu, H. Li, and Y. Qiao, "Uniformer: Unifying convolution and self-attention for visual recognition," *arXiv preprint arXiv:2201.09450*, 2022. 10

Daikun Liu obtained the B.E. degree in Electrical Engineering from Southwest University, Chongqing, China, in 2019. Currently, she is pursuing a Ph.D. with the School of Automation, Southeast University, Nanjing, China. Her research focuses on event-based cameras for pattern recognition and visual navigation.

Teng Wang received the B.S. and M.S. degree from Shandong University and University of Science and Technology of China respectively in 2009 and 2012, and received the Ph.D. degree from Iowa State University, Ames, USA, in 2016. She is currently an associate professor with the School of Automation at Southeast University, Nanjing, China. Her research interests including pattern recognition, machine vision, cross-view image retrieval and visual navigation.

Changyin Sun received the B.S. degree from Sichuan University, Chengdu, in 1996 and Ph.D. degrees from Southeast University, Nanjing, China, in 2003. He is currently a professor with the School of Automation, Southeast University. His current research interests include deep learning, machine vision, and deep reinforcement learning.