Event-VPR: End-to-End Weakly Supervised Network Architecture for Event-based Visual Place Recognition

Delei Kong\textsuperscript{1}, Zheng Fang\textsuperscript{2}, Haojia Li\textsuperscript{2}, Kuanxu Hou\textsuperscript{2}, Sonya Coleman\textsuperscript{3} and Dermot Kerr\textsuperscript{3}

\textbf{Abstract—} Traditional visual place recognition (VPR) methods generally use frame-based cameras, which is easy to fail due to dramatic illumination changes or fast motions. In this paper, we propose an end-to-end visual place recognition network for event cameras, which can achieve good place recognition performance in challenging environments. The key idea of the proposed algorithm is firstly to characterize the event streams with the EST voxel grid, then extract features using a convolutional network, and finally aggregate features using an improved VLAD network to realize end-to-end visual place recognition using event streams. To verify the effectiveness of the proposed algorithm, we compare the proposed method with classical VPR methods on the event-based driving datasets (MVSEC, DDD17) and the synthetic datasets (Oxford Robot-Car). Experimental results show that the proposed method can achieve much better performance in challenging scenarios. To our knowledge, this is the first end-to-end event-based VPR method. The accompanying source code is available at https://github.com/kongdelei/Event-VPR.

I. INTRODUCTION

Visual place recognition (VPR)\textsuperscript{1} is a very challenging problem in the field of computer vision and mobile robots. In computer vision, VPR can be used to retrieve visual information or cross-time place information in a large-scale image database with geographic information annotation, or be used in interactive 3D vision applications such as augmented reality (AR). In mobile robots, the ability of robots to recognize visual places in GPS-denied environment is one of the key capabilities for autonomous localization and navigation. In simultaneous localization and mapping (SLAM), VPR is an important component of loop closure detection\textsuperscript{2} \textsuperscript{3}, which can be used to detect candidate loop-closures and eliminate accumulated errors through global optimization for globally consistent pose estimating and mapping.

At present, there are many solutions to the VPR problem in large-scale environments. For those existing solutions, monocular, binocular, panorama camera and other frame-based vision sensors are widely used. However, since frame-based vision sensors suffer from issues such as illumination changes, motion blur and redundant information, which makes traditional VPR methods difficult to deal with recognition tasks in some challenging environments. Besides, most existing methods are based on the appearance of scenes\textsuperscript{4}. Due to various reasons, such as day-and-night, weather and seasonal changes, the appearance of the same place will change greatly at different time. In addition, the appearance of some places at long distances may be very similar. These issues pose great challenges to the existing large-scale frame-based VPR methods.

In contrast to traditional frame-based VPR methods, we propose a novel VPR method using event cameras in this paper. Event cameras are neuromorphic visual sensors inspired by the biological retina, and work in a completely different way from frame-based cameras; they use address-event representation (AER) and output pixel-level brightness changes (called events) with microsecond resolution to generate sparse asynchronous event streams. Event cameras have the advantages of low latency, high temporal resolution, low bandwidth, low power consumption and high dynamic range\textsuperscript{5}, which can effectively overcome the problems existing in typical frame-based cameras. To achieve robust VPR using event cameras, we propose a novel end-to-end visual place recognition network for event cameras (Event-VPR). The key idea, as shown in Fig. 1, is to apply NetVLAD to EST voxel grid representation generated by event streams. To the best of our knowledge, this is the first end-to-end VPR method for event cameras. Experimental results on multiple datasets with different weather and scenes show that the proposed method is superior to the traditional frame-based VPR methods, and can effectively solve the challenges of large-scale scenes, high dynamic range and long-term adaptability in visual place recognition.

\textsuperscript{1}This work was supported by National Natural Science Foundation of China (No. 62073066), Science and Technology on Near-Surface Detection Laboratory (No. 6142414200208), the Fundamental Research Funds for the Central Universities (No. N182608003; No. N172608005).
\textsuperscript{2}Delei Kong is with College of Information Science and Engineering, Northeastern University, Shenyang, China.
\textsuperscript{3}Zheng Fang, Haojia Li and Kuanxu Hou are with Faculty of Robot Science and Engineering, Northeastern University, Shenyang, China. Corresponding author: fangzheng@mail.neu.edu.cn
\textsuperscript{3}Sonya Coleman and Dermot Kerr are with Faculty of Computing, Engineering and Built Environment, Ulster University, Northern Ireland, UK.
The main contributions of this paper are as follows:

- A novel end-to-end weakly supervised network pipeline of visual place recognition for event cameras is proposed, which directly uses event streams as input.
- Different event representations, network structures, and parameters of the proposed network are compared to show how it affects the total performance.
- The performance of the proposed method against frame-based methods on different datasets with different weather and seasons is thoroughly evaluated and the source code of our method is open-sourced.

II. RELATED WORK

Visual sensors are the main sensor types for place recognition due to their low cost, low power consumption and abundant scene information. Nowadays, most popular VPR methods generally use frame-based visual sensors and appearance-based [4] approaches to realize large-scale place recognition. In this case, the VPR problem can generally be transformed into a large-scale geo-tagged image retrieval problem, and that can be solved by image matching. Extensive researches on how to represent and match images have been carried out [1] [6]. Those methods usually use traditional feature extraction techniques (such as SIFT [7] and ORB [8]), and local descriptor aggregation techniques (such as BoW [2] [3] and VLAD [9] [10]) to establish a higher-order statistical model of image features. With the rise of deep learning, some works begin to use off-the-shelf convolutional networks (such as Overfeat [11], VggNet and AlexNet [12]) as trainable feature extractors. Some researchers also revised VLAD into a trainable pooling layer (such as NetVLAD [13] and DenseVLAD [14]) to obtain image descriptor vectors as compact image representations. In the retrieval and matching process, sequence-based matching is a widely recognized method. The most known work is SeqSLAM [15], which searches highly similar short image sequences for VPR. Recently, researchers also tried to further improve the recognition performance from different aspects. For example, most structure-based methods use structural information such as repeated edges and semi-dense maps of the scene [4], [16], [17] for place recognition. In recent years, there are also some semantic-based works for VPR that mainly use semantic information such as landmarks, texts and objects in the scene. [18]–[20].

Though traditional frame-based VPR methods have developed rapidly over the past decade, they are still difficult to solve some problems in challenging scenarios (such as illumination changes and motion blur) due to the inherent defects of frame-based cameras. Compared to standard frame-based cameras, event cameras have many advantages such as high dynamic range, high temporal resolution and low latency [5]. Due to the above advantages, event cameras have drawn more and more attention. However, to the best of our knowledge, there are still few works related to the event-based VPR problem. Among them, Milford et al. first tried to migrate SeqSLAM to the event camera in 2018 and completed a relatively basic recognition experiment based on event frames [21]. Recently, they proposed an event-based VPR scheme with ensembles of spatio-temporal windows (Ensemble-Event-VPR) [22]. This method uses event bins with different numbers of events and sizes of temporal windows to reconstruct a group of intensity frame sequences using E2Vid [23] for place recognition. However, this method is not a direct event-based method but needs to convert the events into intensity frames, which is still a frame-based VPR method in essence. Different from that, in this paper, we propose a novel end-to-end event-based visual place recognition network (Event-VPR) that directly uses event streams, which achieves excellent recognition results even in challenging environments.

III. METHODOLOGY

In this section, we will describe the network architecture of Event-VPR in detail, including the various module components of our algorithm and the main steps of network training.

A. Problem and Pipeline

Problem Definition. Define a database of events $D = \{P, E\}$ that contains $n$ geo-location coordinated $P = \{P_1, \cdots, P_n\}$ under a fixed reference system and $n$ groups of event bins $E = \{E_1, \cdots, E_n\}$, where each place coordinate $P_i$ corresponds to several event bins $E_i = \{E_{i1}, \cdots, E_{in}\}$ and each event bin $E_{ij}$ is collected by using an event camera in the area near the place coordinate $P_i$. The area of coverage (AOC) of all sub-areas is approximately the same, i.e. \(\text{AOC}(P_1) \approx \cdots \approx \text{AOC}(P_n)\). The problem of event-based VPR can be formally defined as follows: Given a query event bin denoted as $E_q$, our goal is to retrieve several event bins from the database $D$ that are most similar to $E_q$, thus obtaining their corresponding geographic location coordinates $P_{q_i}$.

To this end, we design a deep neural network to learn a function $f_{\text{Event-VPR}}(\cdot)$, which is used to map a given event bin $E_q$ to a global descriptor vector $v_q = f(E_q)$ such that $d(v_q, v_{q_i}) < d(v_q, v_s)$ if $v_q$ is similar to $v_{q_i}$ but different from $v_s$. Here $d(\cdot)$ is a distance function (such as euclidean distance function). Our problem then is simplified to find the place coordinates $P_{q_i}$ of the sub-area such that the global descriptor vector $v_i$ from one of its event bin gives the minimum distance with the global descriptor vector $v_q$ from the query, i.e. $d(v_i, v_q) \leq d(v_{q_i}, v_s)$, where $i \neq s$.

Our Pipeline. The event-based visual place recognition (Event-VPR) proposed in this paper is a novel end-to-end VPR method using event-based camera. The key idea of the algorithm is as follows: Firstly, we divide consecutive event stream into event bins and convert event bins into EST voxel grid representations using MLP-based kernel. Then, a cropped ResNet34 network [24] is used to extract visual features of EST voxel grids. Next, a VLAD-based local aggregated description layer is used for feature descriptor aggregation. Finally, the triplet ranking loss is used to train the network with weakly supervised training. Corresponding to

1. https://github.com/kongdelei/Event-VPR
the aforementioned key idea, our pipeline is mainly divided into the following 4 parts: EST voxel grid representation, feature extraction convolution network, feature aggregated description layer and triplet ranking loss. Its architecture is shown in Fig. 2.

### B. Events and EST Voxel Grid

#### Event-based Data.

The pixel array of the event camera is capable of independently and logarithmically responding to pixel-level brightness changes (i.e. \( L = \log(I) \), where \( I \) is photocurrent) and triggering sparse asynchronous events \( E = \{ e_1, \ldots, e_N \} \). To be exact, without considering fixed pattern noise (FPN), the brightness change at the pixel \((x_k, y_k)\) at time \( t_k \) is given by:

\[
\Delta L(x_k, y_k, t_k) = L(x_k, y_k, t_k) - L(x_k, y_k, t_k - \Delta t_k)
\]

where \( \Delta t_k \) is the time interval between last triggered event and current triggered event of an pixel. When the brightness change of a pixel reaches the contrast threshold \( \theta \) (here \( \theta > 0 \)), the pixel triggers an event \( e_k = (x_k, y_k, t_k, p_k) \). Here, \( p_k \) is event polarity which is given by:

\[
p_k = \frac{\Delta L(x_k, y_k, t_k)}{|\Delta L(x_k, y_k, t_k)|} = \{-1, 1\}
\]

In real sensor, positive events (ON) and negative events (OFF) can be triggered according to different contrast thresholds \( \theta \). Furthermore, the events can be summarized as an event measurement field with polarity defined on the 3D continuous spatio-temporal manifold:

\[
S(x, y, t) = \sum_{e_k \in E} p_k \delta(x - x_k, y - y_k) \delta(t - t_k)
\]

where \( \delta(\cdot) \) denotes the dirac pulse defined in the event domain and is used to replace each event.

#### EST Voxel Grid.

The output of an event camera is the consecutive event stream. In order to use deep neural networks to extract features from event stream, we need to convert the event streams into a representation that convolutional network could process. Nowadays, typical representation methods of events roughly include motion-compensated event frame (EF) [25], 4-channel event count and last-timestamp image (4CH) [26], and event spike tensor (EST) [27]. In addition, the events can also be converted into traditional frame-based video (e.g. E2Vid) [23]. In this paper, we firstly divide event stream into event bins, then we use the voxel grid of event spike tensor (EST) to represent event bins. In order to obtain most meaningful visual feature information from the event measurement field, we convolve it with a trilineral voting kernel \( k(x, y, t) \). Therefore, the convolution signal becomes:

\[
(k * S)(x, y, t) = \sum_{e_k \in E} p_k k(x - x_k, y - y_k, t - t_k)
\]

\[
k(x, y, t) = \delta(x, y) \max(0, 1 - \frac{|t - t_k|}{\Delta t})
\]

After the kernel convolution, the signal (4) can be periodically sampled on the spatio-temporal coordinates:

\[
V_{EST}[x_t, y_m, t_n] = \sum_{e_k \in E} p_k \delta(x_t - x_k, y_m - y_k) \max(0, 1 - \frac{|t_t - t_k|}{\Delta t})
\]

where \((x_t, y_m, t_n)\) are the sampled grid space-time coordinates, \(x_t \in \{0, 1, \ldots, W - 1\} \), \(y_m \in \{0, 1, \ldots, H - 1\} \), \(t_n \in \{t_0, t_0 + \Delta t, \ldots, t_0 + (C - 1)\Delta t\} \). Here, \((W, H)\) is the pixel size, \(t_0\) is the start timestamp, \(\Delta t\) is the size of time blocks, \(C\) is the number of time blocks (that is, the number of channels). We replace the manually designed kernel in (5) with a multi-layer perceptron (MLP) to generate EST voxel grid as an end-to-end events representation, as shown in Fig. 3, where \(C\) is the number of channels, \(B\) is the batch size. The MLP receives the normalized timestamp of the event as input and has 2 hidden layers, each with 30 neurons. The value generated by MLP for each event is put into the corresponding voxel grid coordinates. EST voxel grid can be optimized according to specific tasks to optimize the performance of the whole network.

![Fig. 2. Overview of Our Proposed Pipeline. Firstly, the event bins are converted into EST voxel grids by MLP-based kernel. Then, the cropped ResNet34 convolution network is used to extract visual features of EST voxel grids. Next, VLAD-based local aggregated description layer is used for feature descriptor aggregation. Finally, the triplet ranking loss is used to train the network with weakly supervised training.](image)

![Fig. 3. Overview of EST Voxel Grid for Event-based Data. The value generated by MLP-based kernel for each event is put into the corresponding coordinates, and kernel is learnable through end-to-end training.](image)
C. Feature Extraction and Aggregation

Feature Extraction Network. To extract features from the EST voxel grid, we use the ResNet34 pre-trained in event-based hand-written number recognition tasks [24]. In order to migrate it to our VPR task, the original network needs to be cropped. We modify the number of input channels of the first convolution layer of ResNet34 to make it suitable for EST voxel grid as network input. Meanwhile, in order to connect the feature description aggregation layers, we remove the last average pooling layer and the full connection layer at the end of the ResNet34. The network output $x = f_{\text{ResNet34}}(V_{\text{EST}})$ is the feature maps of the EST voxel grid. Here, $x$ is the $w \times h \times D$-dimensional feature tensor obtained by the feature extraction network. For ResNet34, $w \times h = 7 \times 7$.

Feature Aggregation Network. After getting features for the EST voxel grid, we need to aggregate the features for descriptor matching. We use vector of locally aggregated descriptor (VLAD) [9] [10] which is a trainable descriptor pooling method commonly used in the fields of image retrieval. As shown in Fig. 4, we interpret the $w \times h \times D$-dimensional feature maps $x$ output by the feature extraction network as M-dimensional local descriptors $\{x_1, \cdots, x_M \mid x_i \in \mathbb{R}^D\}$ as input, and K-dimensional cluster centers $\{c_1, \cdots, c_K \mid c_k \in \mathbb{R}^D\}$ as VLAD parameters. The normalized output of descriptor vector $V_{\text{VLAD}}$ is a $D \times K$-dimensional matrix, which is given by:

$$V_{\text{VLAD},k}(x) = \sum_{i=1}^{M} \tilde{a}_k(x_i)(x_i - c_k)$$

where $(x_i - c_k)$ is the residual vector of descriptor $x_i$ to cluster center $c_k$, and $\tilde{a}_k(x_i)$ denotes the soft assignment of descriptor $x_i$ to cluster center $c_k$, which is given by:

$$\tilde{a}_k(x_i) = \frac{e^{-\alpha \|x_i - c_k\|^2}}{\sum_{t} e^{-\alpha \|x_t - c_k\|^2}} = \frac{e^{w_k^T x_i + b_k}}{\sum_{t} e^{w_t^T x_i + b_t}}$$

It assigns the weight of descriptor $x_i$ to cluster center $c_k$ according to their proximity distance. Where, $w_k = 2\alpha c_k$, $b_k = -\alpha \|c_k\|^2$, $\alpha > 0$, and $\tilde{a}_k(x_i) \in [0, 1]$. According to (7), soft assignment can be decomposed into a convolution layer and a soft-max layer, and the weight $\{w_k\}$ and bias $\{b_k\}$ of the convolution layer are taken as independent trainable parameters together with the clustering center $\{c_k\}$. Finally, the aggregated vector needs to be intra-normalized and L2-normalized to produce the final global descriptor vector $v \in \mathbb{R}^D$, $\|v\|_2 = 1$, $\Omega = D \times K$ for event bins that can be used for efficient retrieval.

D. Network Training

Training Triplet Building. We use the metric learning to train Event-VPR in an end-to-end manner to learn the function $f_{\text{Event-VPR}}(\cdot)$ that maps the input of event bins $E$ into global descriptor vectors $v \in \mathbb{R}^D$. In the process of network training, the training query $E_q$ and its geographical location $P_q$ is given. It is necessary to select suitable best positive and hard negatives from the database $D = \{P, E\}$. In order to improve the search efficiency, firstly, all samples within the range of geographical distance $\lambda$ are selected as potential positives $\{E_{\text{pos},i}\}$ according to the location $P_q$ of the query:

$$d_{\text{geo}}(P_q, P_{\text{pos},i}) \leq \lambda, \forall E_{\text{pos},i} \subseteq E$$

Then, among these potential positives, the positive with the smallest descriptor vector distance is selected as the best positive, and its descriptor vector is:

$$v_{\text{best-pos}} = \arg \min_i (v_q, v_{\text{pos},i})$$

When selecting negatives, firstly, all samples outside the range of geographical distance $\delta$ are selected according to the location $P_q$ of the query, and then $n_{\text{sample}}$ of them are selected as randomly sampled negatives $\{E_{\text{neg},i}\}$:

$$d_{\text{geo}}(P_q, P_{\text{neg},i}) > \delta, \forall E_{\text{neg},i} \subseteq E$$

Then, among these randomly sampled negatives, the samples that violates the margin condition are selected as the candidate hard negatives $\{E_{\text{hard-neg},i}\}$:

$$d_{\text{geo}}(P_q, P_{\text{hard-neg},i}) \leq \delta, \forall E_{\text{hard-neg},i} \subseteq E$$

where $m$ is a constant parameter, representing the margin between $d_{\text{geo}}(v_{q}, v_{\text{pos},i})$ and $d_{\text{geo}}(v_q, v_{\text{neg},j})$. In order to improve the training efficiency, we select the $n_{\text{neg}}$ samples with the smallest descriptor vector distance as the hard negatives for training, where $n_{\text{neg}} \in [0, N_{\text{neg}}]$, $n_{\text{neg}} \ll n_{\text{sample}}$.

Loss Function. Based on the above method, we use the data from the event camera datasets to obtain a set of training tuples from the training set, where each tripole is represented as $\xi = (E_q, E_{\text{best-pos}}, \{E_{\text{hard-neg},j}\})$. Here, $E_q$ is the query, $E_{\text{best-pos}}$ is the best positive, and $\{E_{\text{hard-neg},j}\}$ is a group of hard negatives. If their global descriptor vector is $\gamma = (v_q, v_{\text{best-pos}}, \{v_{\text{hard-neg},j}\})$, the descriptor vector distances of query $E_q$ to the best positive $E_{\text{best-pos}}$ and the hard negatives $\{E_{\text{hard-neg},j}\}$ are defined as $d_{\text{f}}(v_q, v_{\text{best-pos}})$ and $d_{\text{f}}(v_q, v_{\text{hard-neg},j})$ respectively. The loss function is designed to minimize the global descriptor distance between the query and the best positive, and to maximize the distance between the query and the hard negatives. We use weakly supervised triplet ranking loss which is defined as follows:

$$L_{\text{triplets}}(\gamma) = \sum_{j} \left[ d_{\text{f}}(v_q, v_{\text{best-pos}}) - d_{\text{f}}(v_q, v_{\text{hard-neg},j}) + m \right]_+$$

where $m$ denotes the margin, and $[\cdot]_+ = \max(\cdot, 0)$ means the loss takes a positive number, i.e. $L_{\text{triplets}}(\gamma) \geq 0$. 

![Fig. 4. Overview of VLAD-based Pooling Layer. Residual vector and soft assignment are statistics from descriptors to clusters, and the convolution layer and clusters are learnable through end-to-end training.](image-url)
IV. EXPERIMENTS

In this section, we carried out experiments on multiple datasets, including the MVSEC [28], DDD17 [29] and Oxford RobotCar datasets [30] to verify the effectiveness of the proposed method through quantitative and qualitative experimental results (A detailed description of the experimental configuration is given in APPENDIX). We conducted four experiments to evaluate our proposed method. Firstly, we evaluated the performance of the Event-VPR in different driving scenes, and verified its long-term robustness. Then, we compared the Event-VPR with frame-based VPR algorithms on the same dataset, and provided numerical analysis to show the performance of Event-VPR. Finally, we performed network ablation study on Event-VPR from two aspects (event representation and feature extraction network), and the experimental results proved the advantages of each module in our method.

A. Performance Comparison in Different Scenarios

In the first experiment, we validated the performance of our Event-VPR in different scenarios of MVSEC, DDD17 and Oxford RobotCar datasets. We selected several challenging sequences for our experiments. For example, the night scene sequence in the MVSEC dataset is dim with little light. There are many similar scenes and high dynamic range scenes (e.g. clouds and tunnels) in the expressway scenes in the DDD17 dataset. As shown in Fig. 5, there are also changing weather and seasonal conditions in the Oxford RobotCar dataset. For instance, there are light reflection from the surface of water on rainy day and the bright street lamps at night, which result in a high dynamic range scenario for visual cameras.

Results on MVSEC. As shown in Table I, the experimental results on the MVSEC dataset show the recognition performance (Recall@N) of the method in daytime and nighttime scenes. On the MVSEC dataset, we trained 2 daytime and 3 nighttime sequences together, and then tested each sequence separately. The results show that the Recall@1 of our model in night sequences can achieve 97.05% on average, which is almost the same as that in the daytime sequences (Please refer to APPENDIX to see how we select training and test sets).

Table I: Results on Event-VPR(EST) on MVSEC Dataset

| Sequences | Recall@1 | Recall@5 | Recall@10 | Recall@20 |
|-----------|----------|----------|-----------|-----------|
| All-day1  | 99.51%   | 99.84%   | 100.00%   | 100.00%   |
| All-day2  | 91.52%   | 97.57%   | 99.17%    | 99.58%    |
| All-night1| 98.67%   | 99.33%   | 100.00%   | 100.00%   |
| All-night2| 95.11%   | 98.22%   | 99.11%    | 100.00%   |
| All-night3| 97.37%   | 98.68%   | 99.34%    | 100.00%   |

Results on Oxford RobotCar. Fig. 6 shows the recognition performance of our method under various weather and seasons on the Oxford RobotCar dataset. The experimental results of model trained on all sequences and tested on different weather sequences are shown in Fig. 6(a) (Please refer to APPENDIX for details). Our model can achieve high performance in different weather conditions. We trained models in day and night sequences separately, and tested on untrained scene sequences (e.g. overcast, rain and snow). The experimental results of model trained on all daytime sequences (including sun and cloud) but tested on different weather sequences are shown in Fig. 6(b). We noted that the test results on overcast and snow sequences are almost the same as those on sun and cloud sequences, and the difference between Recall@1 is only about 2.15%. And the test results in rain and night sequences are poor, with Recall@1 being 14.23% lower on average. The experimental results of model trained on all night sequences and tested on night and day sequences respectively is shown in Fig. 6(c). Due to the large noise of the events in the night sequences, the trained model has poor performance in the daytime scenes.

Results on DDD17. In addition, in order to verify the generalization performance of our method, we use the network model trained on the synthesis event streams of Oxford RobotCar dataset, and then we test the trained model on freeway sequences of DDD17 dataset recorded by real DAVIS camera (Please refer to APPENDIX for details). Surprisingly, the Recall@5 of our method on the night freeway sequences in DDD17 dataset is up to 92.2% without any transfer learning or fine-tuning of the trained network. It means that just using synthesis events for model training can obtain excellent results on events recorded by real event camera. All sequences results are shown as Table II.

Table II: Results on DDD17 of Models Trained with Synthesis Events

| Sequences | Recall@1 | Recall@5 | Recall@10 | Recall@20 |
|-----------|----------|----------|-----------|-----------|
| Night1    | 80.18%   | 88.52%   | 91.76%    | 94.72%    |
| Night2    | 79.50%   | 92.22%   | 95.57%    | 97.39%    |
| Night3    | 73.31%   | 88.82%   | 93.21%    | 95.64%    |
| Day1      | 76.80%   | 90.43%   | 93.65%    | 95.97%    |
| Day2      | 76.26%   | 91.07%   | 94.14%    | 96.36%    |
| Day3      | 78.06%   | 91.10%   | 94.27%    | 96.65%    |

B. Performance Comparison with Different VPR Methods

In order to compare the performance of our Event-VPR with conventional frame-based method, we compared Event-VPR with NetVLAD [13] and SeqSLAM [15] on MVSEC and Oxford RobotCar datasets, and the results are shown in Table III. For fair comparison, we adjusted the spatial
resolution of the image frames to be the same as that of the event bins. The results show that our method is better than NetVLAD and much better than SeqSLAM on MVSEC dataset, since most images recorded by DAISY camera are dim and Event-VPR has advantages under this condition. On Oxford RobotCar dataset, the recall@1 of Event-VPR is about 26.02% higher than SeqSLAM but about 7.86% lower than NetVLAD. We think the reason is that a lot of fixed pattern noise is added to the event streams manually generated by V2E [31]. Considering that the event cameras are not mature enough (e.g. low spatial resolution, high noise level, etc.), therefore Event-VPR shows similar performance to frame-based method.

### TABLE III

| Methods                      | Recall@1 (%) |
|-----------------------------|--------------|
| Event-VPR(Ours)             | 99.51 91.52 98.67 95.11 97.37 91.81 90.95 93.29 91.80 |
| NetVLAD [13]                | 98.57 96.74 96.00 92.44 82.89 99.34 98.57 99.84 99.67 |
| SeqSLAM [15]                | 72.61 68.42 55.37 58.68 57.72 73.10 63.37 69.98 57.32 |

### C. Impact of Different Event Representations

In this experiment, we explored the Event-VPR performance with different event representations. In addition to the EST voxel grid described in our method, we have also tried several other event representations, including event frame (EF), unipolar event voxel grid (EVG), and 4-channel event count and last-timestamp image (4CH). We all used ResNet34 as the feature extraction network in this experiment. As shown in Fig. 7, since EF discards timestamps and EVG discards event polarity, EST voxel grid has great advantages over other representations. Due to 4CH discards earlier timestamps, it is also not as good as EST voxel grid. EST voxel grid not only retains the local spatio-temporal neighborhood information of events, but also extracts effective features through using learnable kernel to suppress the noise of events.

### D. Impact of Different Network Structures

In this experiment, we explored the Event-VPR performance of different network structures. In addition to the ResNet34 described in our method, we also trained VGG-16, AlexNet and two other ResNet with different network capacities. We used EST voxel grid as the event representation in the whole experiment. As shown in Fig. 8, experimental results show that different feature extraction networks have no significant impact on the network performance, while a larger network increases the computing resources without improving the accuracy. We guess that a larger network needs more parameters, which makes the network easier to overfitting. We finally choose ResNet34 as the feature extraction network of our pipeline. However, we can replace it with other different convolutional networks according to our own needs and computing resources.

### V. CONCLUSIONS

We proposed an end-to-end method (Event-VPR) to solve the problem of large-scale place recognition by using event cameras. We showed that our Event-VPR method based on events is more robust to changes in the surrounding environment caused by weather and time changes than conventional frame-based VPR methods. Though we get very good results, it should be noted that the spatial resolution of event cameras is still very low, so there are still some deficiencies compared to frame-based VPR methods. In the future, we will try to combine standard cameras and event cameras to implement a hybrid network architecture for visual place recognition based on frames and events.
B. Dataset Configuration

In our experiments, we use MVSEC [28], DDD17 [29] and Oxford RobotCar datasets [30]. The MVSEC and DDD17 datasets are event-based driving datasets recorded in the real environment. We select intensity images and event bins of 5 outdoor driving sequences from MVSEC dataset (including day-and-night scenes, using left-eye DAVIS camera) and 6 outdoor driving sequences from DDD17 dataset (including urban and freeway scenes), which include dramatic illumination changes and scene structure changes. In addition, Oxford RobotCar dataset is a standard dataset commonly used in VPR. We use the event synthesizer V2E [31] to convert the image sequences recorded by the center camera of the three-eye vision sensor (Bumblebee XB3) in Oxford RobotCar dataset into corresponding event streams. We tried our best to select sequences with the same trajectory under different weather conditions, including sun, cloud, rain, snow and night scenes. We think that such scenarios are more challenging for testing the performance of VPR methods.

We randomly divide the sequences of the same trajectory into geographically non-overlapping training and test sets (detailed results are shown in Table IV V VI VII VIII). In the MVSEC dataset, we select about 40k training samples and 10k test samples from 5 sequences. In the DDD17 dataset, we select about 120k test samples from 6 sequences. In Oxford RobotCar dataset, we also select about 50k training samples and 12k test samples from 11 sequences. For parameters, we choose potential positive distance threshold $\lambda = 10m$, potential negative distance threshold $\delta = 25m$, and true positive geographical distance threshold $\phi = 20m$ in MVSEC and Oxford Robocar datasets. Moreover, all parameters in DDD17 dataset are the same except that the true positive geographical distance threshold is $\phi = 60m$.

Generally, the performance of VPR methods is evaluated by Recall. In this paper, we use Recall@N to evaluate the performance of our algorithm. Specifically, for each query, the N-nearest neighbor samples are retrieved as positive samples according to the descriptor distance. If at least one of samples is less than $\phi$ away from the query according to their geographical distance, it is considered to be correctly identified. Then, we calculate the Recall@N with different N. For the query set, we traverse descriptor vectors of all queries to calculate Recall@N which is the percentage of correctly identified query descriptors. In our experiments, we restrict our analysis to N = \{1, 5, 10, 20\}.

![Fig. 9. The Scenes of MVSEC and DDD17 Datasets in Our Experiments.](image)

### Table IV

| Parameters | Day1 | Day2 | Night1 | Night2 | Night3 |
|------------|------|------|--------|--------|--------|
| Total Data $N$ | 11937 28583 2704 3863 2854 |
| Training Database $N_{db,train}$ | 7140 17199 1597 2274 1678 |
| Training Database $N_{d,train}$ | 2428 5760 577 752 620 |
| Test Database $N_{db,test}$ | 1804 4303 412 590 445 |
| Test Database $N_{d,test}$ | 615 1440 151 226 153 |
| Random Negatives $N_{r,n}$ | 300 400 100 100 100 |
| Hard Negatives $N_{h,h}$ | 10 10 10 10 10 |

### Table V

| Parameters | Day1 | Day2 | Day3 | Night1 | Night2 | Night3 |
|------------|------|------|------|--------|--------|--------|
| Total Data $N$ | 20949 29846 31336 14023 12072 14571 |
| Test Database $N_{db,test}$ | 16759 23876 25068 11218 9657 11656 |
| Test Database $N_{d,test}$ | 4190 5970 628 2805 2415 2915 |
| Random Negatives $N_{r,n}$ | 500 500 500 500 500 |
| Hard Negatives $N_{h,h}$ | 10 10 10 10 10 |

### Table VI

| Parameters | Freeway (Day) | Freeway (Night) |
|------------|---------------|-----------------|
| Total Data $N$ | rec1487417411 / rec1487419513 / rec1487430438 |
| Test Database $N_{db,test}$ | rec1487419513 / rec1487608147 / rec1487609463 |
| Test Database $N_{d,test}$ | 500 500 500 500 500 |
| Hard Negatives $N_{h,h}$ | 10 10 10 10 10 |

### Table VII

| Parameters | Freeway (Day) | Freeway (Night) |
|------------|---------------|-----------------|
| Total Data $N$ | 20949 29846 31336 14023 12072 14571 |
| Test Database $N_{db,test}$ | 16759 23876 25068 11218 9657 11656 |
| Test Database $N_{d,test}$ | 4190 5970 628 2805 2415 2915 |
| Random Negatives $N_{r,n}$ | 500 500 500 500 500 |
| Hard Negatives $N_{h,h}$ | 10 10 10 10 10 |

### Table VIII

| Parameters | Total Data $N$ | Training Database $N_{db,train}$ | Training Database $N_{d,train}$ | Test Database $N_{db,test}$ | Test Database $N_{d,test}$ | Random Negatives $N_{r,n}$ | Hard Negatives $N_{h,h}$ |
|------------|---------------|----------------------------------|----------------------------------|----------------------------|--------------------------|-------------------------|-------------------------|
| Total Data $N$ | 23539 12002 10909 6001 12002 |
| Training Database $N_{db,train}$ | 13284 7291 6627 3578 7125 |
| Training Database $N_{d,train}$ | 4384 2356 2142 1218 2451 |
| Test Database $N_{db,test}$ | 3268 1797 1634 908 1810 |
| Test Database $N_{d,test}$ | 1212 558 507 298 622 |
| Random Negatives $N_{r,n}$ | 500 500 500 500 500 |
| Hard Negatives $N_{h,h}$ | 10 10 10 10 10 |