electronics

MDPI

Article

Eye Tracking Based on Event Camera and Spiking Neural Network

Yizhou Jiang, Wenwei Wang *, Lei Yu and Chu He

School of Electronic Information, Wuhan University, Wuhan 430072, China; jiangyizhou@whu.edu.cn (Y.J.); ly.wd@whu.edu.cn (L.Y.); chuhe@whu.edu.cn (C.H.)
* Correspondence: wangww@whu.edu.cn

Abstract: An event camera generates an event stream based on changes in brightness, retaining only the characteristics of moving objects, and addresses the high power consumption associated with using high-frame-rate cameras for high-speed eye-tracking tasks. However, the asynchronous incremental nature of event camera output has not been fully utilized, and there are also issues related to missing event datasets. Combining the temporal information encoding and state-preserving properties of a spiking neural network (SNN) with an event camera, a near-range eye-tracking algorithm is proposed as well as a novel event-based dataset for validation and evaluation. According to experimental results, the proposed solution outperforms artificial neural network (ANN) algorithms, while computational time remains only 12.5% of that of traditional SNN algorithms. Furthermore, the proposed algorithm allows for self-adjustment of time resolution, with a maximum achievable resolution of 0.081 ms, enhancing tracking stability while maintaining accuracy.

Keywords: event camera; eye tracking; spiking neural network

Citation: Jiang, Y.; Wang, W.; Yu, L.; He, C. Eye Tracking Based on Event Camera and Spiking Neural Network. *Electronics* **2024**, *13*, 2879. https://doi.org/10.3390/electronics13142879

Academic Editor: Enzo Pasquale Scilingo

Received: 23 May 2024
Revised: 13 July 2024
Accepted: 18 July 2024
Published: 22 July 2024

1. Introduction

The complex relationship between eye movements and human emotional cognition underscores the role of eye movement behavior as a potential indicator of psychological states [1–3]. Researchers and practitioners frequently use eye-tracking technology to understand how variations in eye dynamics might reflect different emotional states, such as excitement, focused attention, or stress. Additionally, the application of eye-tracking technology in Virtual Reality and Extended Reality (VR/XR) devices further bridges the gap between the virtual and real world [4–6].

We focus on the use of eye tracking for VR devices. For VR glasses, eye tracking has specific characteristics: the close proximity of the camera to the eye and the relatively dark lighting environment [7,8]. Firstly, VR glasses' proximity to the eyes places the camera close as well, with the eyes' images dominating the camera's frame. On the other hand, VR glasses often have dim lighting since many VR glasses are generally made with an opaque casing that can block external light sources, with only a small amount of light provided by the display, and the lighting conditions also change with the content displayed on the screen. Additionally, for some simple VR devices, nearsighted users need to wear optical glasses at the same time, which is also a consideration for whether it will affect eye tracking.

High-speed eye-tracking devices and low-power, low-latency algorithms are crucial. Most eye-tracking devices use traditional frame-based cameras to capture images, requiring high frame rates to keep up with the rapid and subtle movements of the eye. In fact, most background information during eye movements is redundant, yet traditional cameras capture it in full, increasing bandwidth, memory consumption, and the computational burden of subsequent algorithms. This prompts the consideration of novel devices and algorithms to address these issues.

As a rapidly evolving technology, an event camera has gained recognition among eye-tracking researchers [5,9–11]. Compared to traditional cameras that capture images at a fixed frame rate, an event camera, also known as a Dynamic Vision Sensor (DVS) or

neuromorphic camera, detects brightness changes at the pixel level and outputs events independently. This results in low latency and microsecond-level refresh capabilities. By ignoring a large amount of redundant background data with minor brightness changes, event camera offers advantages in memory consumption and bandwidth over traditional cameras. Thus, they can adjust event density based on the speed of eye movement, utilizing camera bandwidth more effectively. The high temporal resolution, high dynamic range, and lack of motion blur of event camera are well applied in the field of eye-tracking [9–14].

For example, Angelopoulos et al. [11] constructed a hybrid system for eye gaze tracking using both traditional image frames and event data, achieving an update rate exceeding 10,000 Hz. The model parameterized the eye's image, fitting the pupil and eyelids with ellipses and parabolas, respectively. Stoffregen et al. [12] introduced hardware innovations by adding a ring of encodable LEDs around the event camera lens to generate Coded Differential Lighting. This retains partial specular reflections from the eye while suppressing diffuse reflection events from the surrounding scene. Their approach achieved sub-pixel precision in corneal reflection detection by encoding the flashing frequency of the LEDs.

Deep learning has already demonstrated its powerful capabilities in image processing, including object detection and tracking, which is also introduced to event-based eye tracking. Feng et al. [14] proposed a lightweight segmentation network that reduces the input size of the segmentation algorithm by continuously predicting regions of interest (ROIs) through event data. Chen et al. [9] introduced deep learning algorithms using a modified ConvLSTM [15] network structure for predicting the eye's center point. However, both approaches were evaluated using event data generated by simulators, lacking adaptability to real datasets.

The gaze-tracking study mentioned above is an important part of eye-tracking technology, which can generally be divided into two stages: positioning of eye movements and conversion between the coordinates of the pupil on the camera screen and the coordinates of the gaze point [16]. The movement of the eye is usually obtained by locating the center of the pupil, which is a key step in current gaze-tracking technology. To focus on this step, we constructing a model similar to the scenario in VR devices, recording eye movement data in real environments using event cameras. In actual scenarios, we found that the data captured by the event camera lacks some texture information relatively [17], and identifying the center of the pupil manually is challenging due to the unclear boundary of the iris and pupil in low-light environments [18]. Conversely, the boundaries of the iris and sclera are clearer, so we believe that detecting the iris region would be a more reasonable choice. The position of the iris region is also strongly correlated with the coordinates of the gaze point [19], so detecting the position of the iris region can also serve as an intermediate step in gaze tracking.

On the other hand, iris tracking has also been applied in some studies. For example, the Google research MediaPipe uses iris region detection for depth estimation [20], and similarly, iris detection is also a core part of tasks such as facial capture for virtual avatars and iris color changing. Qiu et al. introduced iris tracking to ophthalmic surgery [21], which can guide the robotic manipulator during the operation. These studies not only focus on the position of the iris but also consider its size, suggesting that even without gaze point regression, concentrating solely on the position of the iris region holds practical significance.

In conclusion, we have decided to focus our research on near-eye iris region tracking with a bounding box representing both the position and size, which can serve as an important step in gaze tracking and has a wider range of applications.

The asynchronous and temporal nature of event data makes it particularly suited for processing with spiking neural network [22], which is adept at synchronous extraction of spatio-temporal features. SNN has emerged as a branch of deep learning in recent years, building on traditional Artificial Neural Networks (ANNs) to mimic the behavior of biological neurons. In an SNN, the basic computational unit is the spiking neuron, which updates its membrane potential based on previous states and ongoing inputs, emitting

a pulse when the potential exceeds a threshold. Unlike the activation functions in ANN, spiking neurons communicate via binary pulses across network layers.

A key aspect of SNN is the incorporation of temporal processing. Information is encoded in the timing of spike trains, where high-frequency spike sequences can represent higher values and low-frequency spikes represent lower values. This gives SNN the capability to process both spatial and temporal information. Additionally, the retention of membrane potential after each computation grants SNN a degree of memory. These advances make SNN work well with event-based tracking tasks.

The two most common learning methods in SNN are ANN-to-SNN conversion [23] and gradient-based [24] back propagation with surrogate gradients. In ANN-to-SNN conversion, an ANN is first trained with the ReLU activation function. Then, by replacing ReLU with spiking neurons and adding operations like weight normalization and threshold balancing, the ANN is transformed into an SNN. The back propagation method employs a concept similar to Back Propagation Through Time (BPTT) [25], computing gradients by unfolding the network over time steps. Due to the non-differentiable nature of the threshold-based spike emission function, a surrogate gradient function is typically used during back propagation.

SNN has been widely applied in tasks such as object tracking [26–29], optical flow estimation [30], and video reconstruction [31]. There are also several combinations of event camera and SNN [27,31,32], but exploration in the field of eye tracking is still ongoing.

Compared to ANN, SNN has stronger memory, which can be very useful for continuous tracking tasks. On the other hand, existing SNNs often use spike firing rates as the output [25]. For classification tasks, a higher spike rate indicates higher confidence. However, if this output is used as a feature for regression tasks, some information may be lost. Therefore, we added a temporal feature extraction module to the network, using the membrane potential of a neuron that does not fire spikes as the feature extraction result, thereby achieving better detection performance.

Due to the nascent stage of event-based eye tracking, a lack of datasets is a significant barrier to the integration of event cameras and SNN. Existing studies often use software or event data simulators [33–35] to generate event data from traditional images, or they only provide datasets captured with an event camera without manual annotation. Both situations make it difficult to assess tracking results.

In light of these issues and challenges, the innovations and contributions are outlined below:

1. A new eye-tracking dataset entirely based on real event camera footage is established and manually annotated. The dataset contains a total of 365 screened event point sequences with labels, covering various scenarios like different individuals' eyes, lighting conditions, and eye movement trajectories including saccades, smooth pursuit and circular movement by consciously requesting these trajectories in advance.
2. A novel eye-tracking algorithm combining SNN with an event camera is designed. It fully leverages the memory characteristics of SNN and utilizes membrane potential to extract temporal features, better suited to handle the asynchronous incremental output of the event camera than ANN, increasing stability while maintaining accuracy.
3. The proposed solution is validated on the custom dataset, achieving a maximum time resolution of approximately 0.081 ms and increasing results in tracking accuracy compared to ANN.

2. Methods

2.1. Event Data Representation

In the SNN, information is encoded in spike trains, with each spiking neuron's firing frequency carrying certain information. To simulate asynchronous event inputs in a computer, we divide each complete set of event points in the dataset into subsets, each with a fixed number C of points, thereby introducing temporality into the input. The parameter C refers to the number of event points contained in the input received by the network at

each time step. Since feature extraction in SNN relies on convolutional layers, we need to represent event points in a tensor format recognizable by these layers. The representation of event tensors is as follows:

$$\mathcal{E} = \{e_k\}_{k=1}^{C} = \{(x_k, y_k, t_k, p_k)\}_{k=1}^{C}, \tag{1}$$

$$E_{\pm}(x, y) = \sum_{e_k \in \mathcal{E}_{\pm}} \delta(x - x_k, y - y_k), \tag{2}$$

$$E(x, y) = \text{concat}(E_+, E_-), \tag{3}$$

Each event point is represented as a quadruple e_k, where $(x_k,\ y_k)$ are the pixel coordinates of the event, t is the time of occurrence, and $p_k \in \{-1, 1\}$ represents the polarity, determined by the decrease or increase in the pixel's brightness. If different polarities are represented by colors, as shown in Figure 1, it can be observed that the left boundary of the iris is green, indicating that this part is darkening, i.e., the pixels in this area are changing from the white sclera of the eye to the dark iris; the right boundary is the opposite, with red indicating a change from the iris to the brighter sclera of the eye. This indicates that the entire eyeball is moving to the right of the picture. $E_{\pm}$ denotes event frames generated according to event polarity, as object motion direction can be indicated by event polarity to some extent. Thus, $E_{\pm}$ are connected as different channels to form a 2-channel input event tensor E For each group of input event points.

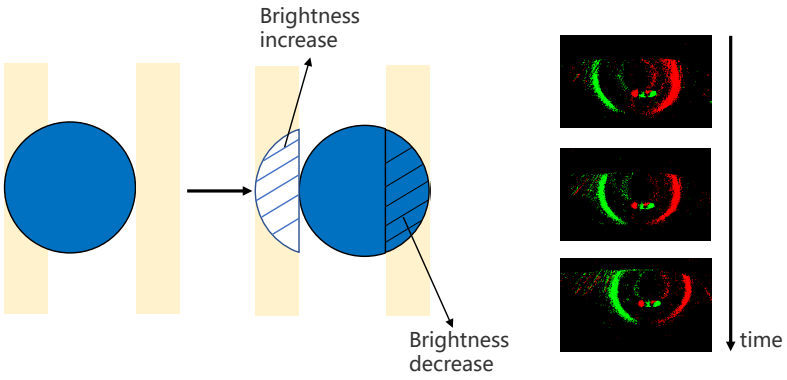

Figure 1. The polarities of events indicates the direction of eye movement.

2.2. Eye Tracking Based on Spiking Neural Network

The tracking method designed in this study consists of an SNN backbone network and a tracker. The backbone network extracts spatial features with temporal information, while the tracker generates tracking results.

SEW ResNet [36] is used as the backbone network. ResNet has shown excellent performance in various image processing fields, and SEW ResNet improves upon it by modifying the residual structure, connecting the input of the residual block to the output of the spiking layer for identity mapping:

$$O[t] = SN(\mathcal{F}(S[t]) + S[t]), \tag{4}$$

where $S[t]$ represents the spike input from the previous layer, $\mathcal{F}$ denotes convolution and pooling operations within the residual block, and SN represents the spiking layer. This adaptation makes the SEW ResNet suit a complex motion representation of spiking neurons.

A spiking neuron can be described as follows:

$$h[t] = f(u[t-1] + x[t]), \tag{5}$$

$$o[t] = \Theta(h[t] - u_{th}), \tag{6}$$

$$u[t] = h[t] \cdot (1 - o[t]) + u_r \cdot o[t], \tag{7}$$

where $x[t]$ is the input, $u[t]$ is the membrane potential, u_{th} is the emitting threshold and u_r is the reset potential. The threshold value is set to a default value of 1.0 and the reset value is set to 0. $h[t]$ means the hidden state when the neuron is after charging but before emitting. f is the charging equation in which spiking neurons differ. Θ is the Heaviside function. The spiking neural model uses the Parametric-LIF(PLIF) model [37], with a charging equation f same as the LIF model [38]:

$$\tau \frac{du}{dt} = -(u(t) - u_r) + x(t), \tag{8}$$

the difference between PLIF and the LIF neuron is that the time constant τ of PLIF is not a preset hyper parameter but is optimized during training. Each layer of spiking neurons shares τ, while different layers have different τ values.

As shown in Figure 2, to obtain the firing frequency of spikes over a period, SEW ResNet or other existing SNN structures need to run for several time steps on PC to simulate the charging state of spiking neurons continuously receiving input. Regardless of how many event points are included in each input, every pixel in the input tensor participates in the computation, so the time required for forward propagation at each time step of an SNN is equal to that of an equivalent structured ANN. Thus, each output of an SNN incurs several times the operational cost compared to ANN, which obviously contradicts the initial goal of low power consumption. To reduce the time steps of SNN simulating on PC, we designed a new algorithm fully based on SNN.

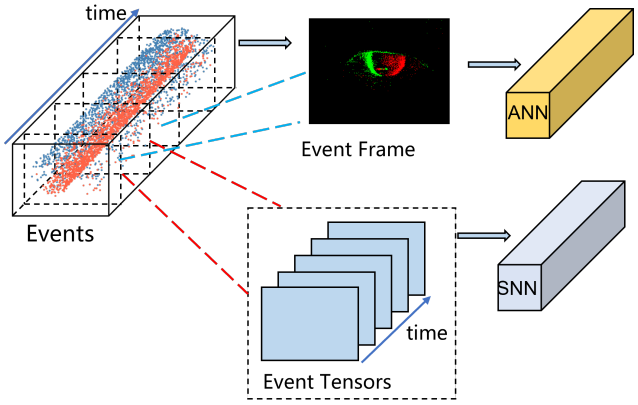

Figure 2. Comparison with ANN and SNN when processing events. The event frame is generated by compressing event points, within which red and green colors represent binary polarities. The event tensors have a time dimension which is generated by dividing the event points into several groups.

Observing the state equations of spiking neurons Formulas (5)–(7), one finds that the membrane potential of the spiking neurons is continuously stored internally and changes with each time step's input. This attribute gives SNN the ability to remember the current state. When an SNN continuously receives inputs, neurons emit spikes continuously according to the charging equation and the emission threshold, with the spike output result of each time step being related to the state of the previous time step. Therefore, in situations where the SNN continuously receives event inputs, we consider a real-time recording of the spike tensor output, but the information contained in the output of a single time step is limited. Accumulating spikes over all time steps would result in the output containing information from too far back. To address this, we set a record length L, with the network continuously receiving event inputs and recording the spike output SF_t (spike feature tensor) of the backbone network at each time step. Existing SNNs often use spike firing rates as the output which can be obtained from Formula (10). If we follow this method, the spike feature tensor of the current and the previous $L-1$ time steps is averaged over

the time dimension to obtain the spike rate tensor SR_t for the recent L time steps, which is considered as the feature of the current time step.

$$SF_t(x, y) \in \{0, 1\}, \tag{9}$$

$$SR_t = \frac{1}{L} \sum_{t-L}^{t} SF_t, \tag{10}$$

However, as shown in Figure 3, it depicts the membrane potential changes and output spikes of two different neurons after receiving inputs.

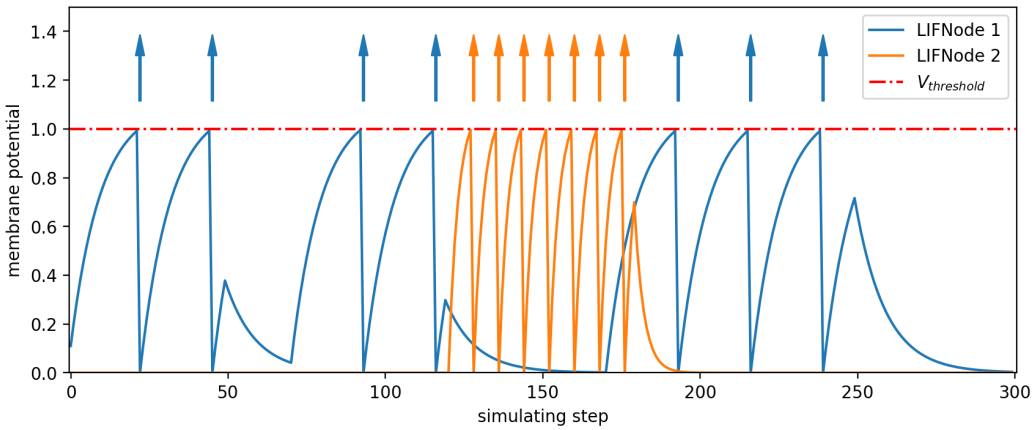

Figure 3. Membrane potential changes and output spikes (the arrows) of two different neurons.

It is evident that the inputs between these two cases differ significantly, while the differences in the inputs did not result in any change in the output. For both inputs, there were 7 spikes within 300 time steps. This suggests that using spike rate as a feature loses the relative timing information of each spike emission. For instance, there is a distinct difference in the information contained within sequences where spikes are concentrated in a short period versus those where spikes are uniformly distributed over a longer duration.

To better extract information from spike sequences, it is necessary to consider a module capable of appropriately capturing information from different spike sequences. This module should output a higher value when continuously receiving input and also respond when input ceases. This requirement aligns well with the characteristics of the LIF neuron model. Consequently, we replace the operation of computing the spike firing rate with a PLIF neuron as well as above. The time constant τ, which influences the neuron's charging curve, is treated as a learnable parameter that is updated synchronously during network training. This neuron accumulates membrane potential without emitting spikes. After continuously receiving inputs for L time steps (the feature extraction window length of our model structure), the membrane potential in the neuron is extracted as a floating-point number. The final result varies based on the presence or absence of input spikes at different times. For ease of reference, such neurons are named NSPLIF (Non-Spiking PLIF).

As shown in Figure 4, the output differences exhibited by the NSPLIF under various input conditions are clearly demonstrated. By shifting the input along the time axis, the figure displays two membrane potential change curves. When the neuron continuously receives inputs, its membrane potential experiences a rapid rise followed by a gradual stabilization; when the input ceases, the membrane potential decreases gradually due to the effect of leak voltage until the arrival of the next input. This process enables the NSPLIF to effectively extract temporal information hidden at different moments from the input spikes, thereby providing a more precise basis for the final output. Additionally, it is worth mentioning that the NSPLIF model can be derived by making appropriate modifications to the PLIF model. This feature allows the entire neural network to utilize a single type of neuron model, thus ensuring structural uniformity and consistency within the network.

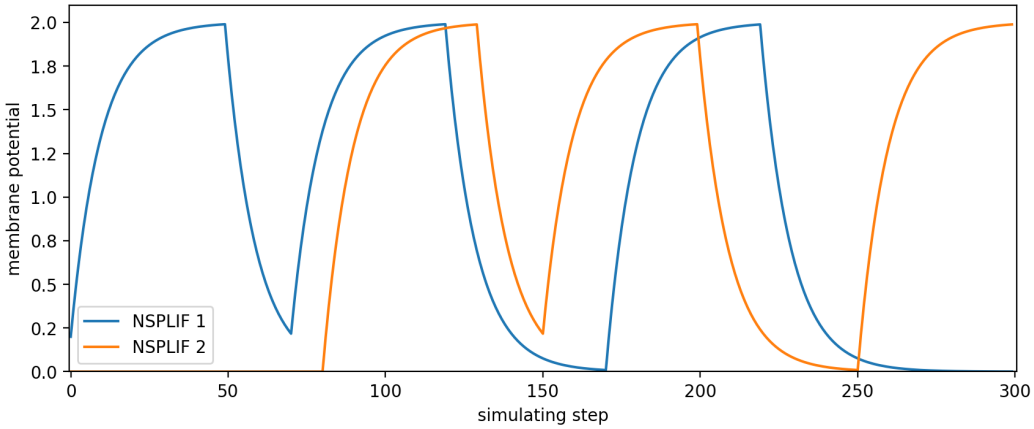

Figure 4. Membrane potential changes in NSPLIF.

Eye-tracking tasks can be viewed as single-object tracking, and thanks to the event camera not including redundant background information in its inputs, bounding box regression is handled by a fully connected layer. SR_t is passed through a pooling layer and then unrolled before being input to the fully connected layer, ultimately yielding the tracking result for the current time step. The whole structure of the algorithm is shown in Figure 5, with an SNN backbone containing residual blocks, allowing an NSPLIF to obtain the membrane potential as a temporal feature and a fully connected layer to regress the bounding box.

The algorithm network requires a warm-up period initially, as the first update result must wait for the network to run for $L-1$ time steps; from the Lth time step onward, the tracking result can be updated at each time step, with each regression containing information from the previous $L-1$ time steps. This enables the network to self-recover tracking when the target disappears and reappears.

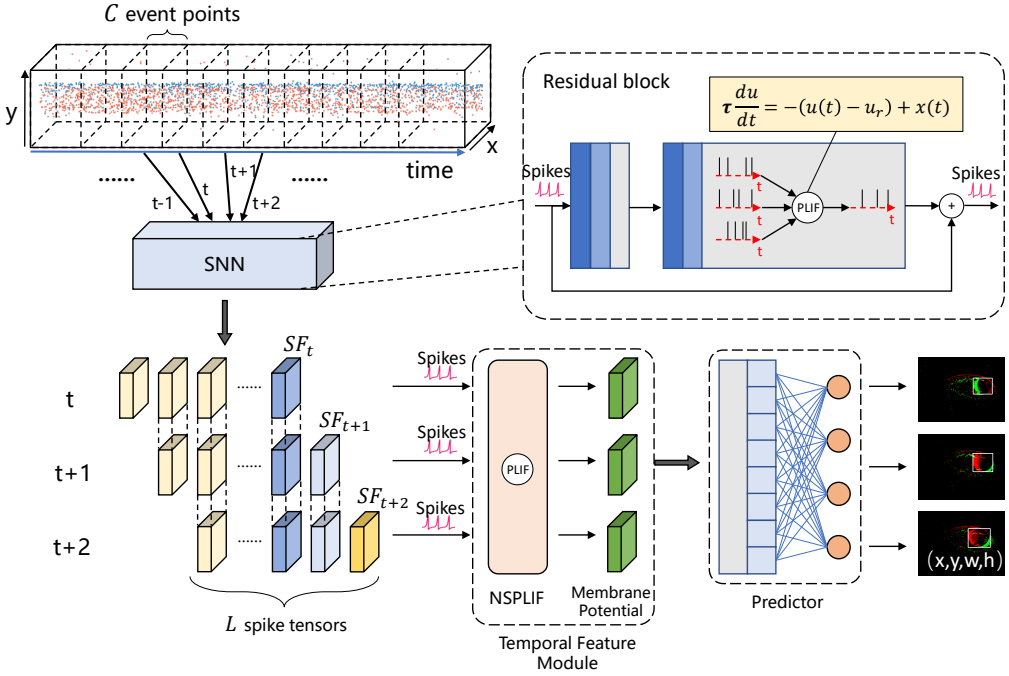

Figure 5. Structure of the Event-SNN eye-tracking model. The SNN backbone continuously receives tensors processed from event points. The NSPLIF neuron feeds the membrane potential after past L time steps inputting as a feature into a fully connected layer, predicting the bounding box of the iris region.

In situations where there is a high similarity between successive frames in eye tracking, the memory advantage of SNN becomes evident. Common algorithms for eye-tracking tasks fall into two categories: one based on object detection, independently computing each time an image frame is input, with the advantage being independent prediction, allowing for continued recognition when the target reappears after being lost; the other OPE tracking algorithms [39,40] fundamentally rely on similarity matching between frames, which is challenging for re-tracking lost targets. The SNN-based tracking structure proposed in this study combines the advantages of both, reasonably utilizing prior information for predicting the target's position and enabling the re-acquisition of the target without external information even after the tracking target is lost.

3. Dataset

3.1. Device Setup and Acquisition Protocol

To acquire near-range eye movement data, we designed a setup with a camera bracket, headrest, and an opaque black box as shown in Figure 6. The top of the black box is detachable to simulate low-light and natural lighting conditions. We also prepared a fixed light source that can be placed in different corners of the box to simulate various lighting directions. The event camera used for data collection is DAVIS346 [41] from iniVation(Zurich, Switzerland), with a standard resolution of 346 × 260 pixels. The camera is placed on the top layer of the camera bracket, about 10 cm from the eye, with the ability to deploy event cameras on either side (monocular data are recorded in the dataset since eyes are symmetric). The height difference between the camera center and the eyes is approximately 2 cm, and the bottom of the camera is roughly parallel to the upper border of the eyes. Each recording session lasted approximately 15–20 s.

During the recording process, we initially used a mobile phone with a 6.67-inch display mounted on the camera bracket which leads to a field of view (FoV) of 102° × 69°. For each participant, we played two videos with moving points on the display to guide participants in performing smooth pursuit and rapid saccade eye movements. For the first video, we set a red point to move back and forth horizontally on the screen at a constant speed. It moved a total of seven times one way, with the first four trips taking 2 s each and the last three trips taking 1 s each. For the second video, the red point was set to appear at each of the four corners of the screen, with a 1 s interval between each appearance. Subsequently, to increase the randomness of the data, we removed the screen and had the participants perform the aforementioned eye movements on their own. For more details about the dataset, please refer to Appendix A. The limitations of this dataset are also discussed in the Appendix A.

In total, four individuals (three male and one female) with different eye shapes participated. All of them were college students about 20 years old. A total of 91 eye movement video sequences were recorded. Table 1 shows the number of event sequences under different conditions. The internal light source was a lamp tube which can be put in the box. Note that natural light and internal light sources may coexist.

Table 1. Number of event sequences under different conditions.

Condition	Number
wearing glasses	22
low light	29
natural light	48
internal light	26

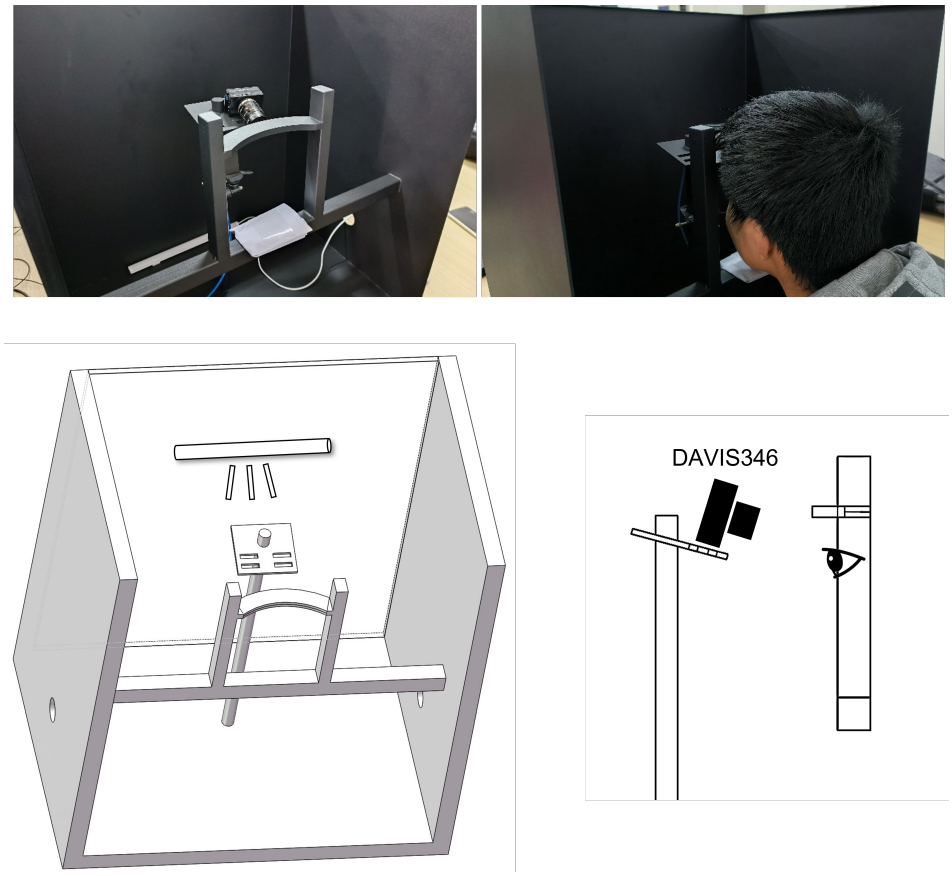

Figure 6. The schematic diagram diagram and real picture of the device for capturing event data.

3.2. Preprocessing

To facilitate manual annotation, the event data need to be converted into event frames. Given the impracticality of annotating data at microsecond-level time resolutions, the event data are grouped to ensure each group has a sufficient number of event points to form a recognizable event image. Notably, we group by fixed point count rather than time intervals. Following Equations (1) and (2), we generate the event tensors $E_{\pm}(x, y)$.

$$F_{\pm} = \Theta(E_{\pm}), \tag{11}$$

where Θ represents the Heaviside step function. $F_{\pm}$ serves as the first two channels of the event image, with the third channel completed with zeros, giving the event image F a shape of $(3 \times 260 \times 346)$. Setting C to 4000, we compared the time span between the first and last event points in each frame with the labeled x-y coordinate trajectories in Figure 7. During smooth eye movements, the time required to produce 4000 events was consistent, averaging about 25 ms (excluding over high values). However, when the eye changes direction which reduces the moving speed, the time to produce the same number of events significantly increases, aligning with the working principle of the event camera. Grouping by fixed point count ensures consistency in image distribution, aiding feature extraction by convolutional networks. It also reduces the number of frames generated in situations with reduced event rates, like when the eye moves to the corner or fixates, thus lowering subsequent computational power consumption.

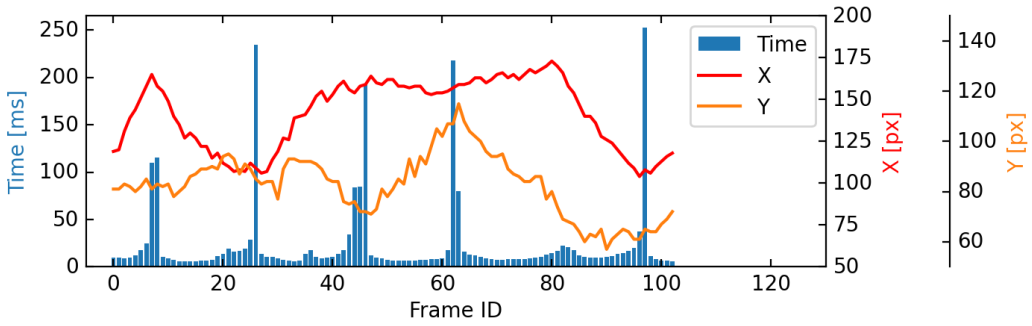

Figure 7. Comparison figure of time required to generate fixed event points and eye movement.

3.3. Labeling

After preprocessing, the event frames underwent manual selection and annotation, with eye regions enclosed in bounding boxes. As shown in Figure 8, the yellow boxes represent the annotated bounding boxes. We observed that the pupils and iris of participants are similar in color, making it difficult to distinguish the boundaries of the pupils in low light conditions. In contrast, the boundaries of the iris are clearer. To ensure the accuracy of annotation, we chose to use bounding boxes to annotate the visible iris region. The bounding boxes contain both position and size information, which can effectively express the different shapes of the iris during movement. The position information of the bounding box is strongly correlated with the gaze point coordinates, while its size information can be used in previously mentioned fields such as depth estimation and facial capture. When the participant looks down, blinks, or when the iris moves to the corner of the eye, the visible size of the iris changes significantly. These eye movements can sometimes be related to emotional states, such as increased blinking frequency or involuntary squinting, which may indicate emotional changes [42]. In such cases, the size information of the bounding box may become useful. Our dataset, composed entirely of event data, opts for bounding box annotation, balancing accuracy and efficiency. Figure 8 shows some labeled data. The RGB images were taken by DAVIS346 at the same time when recording event data. The images in the first and second columns correspond to each other which describe different lighting conditions.

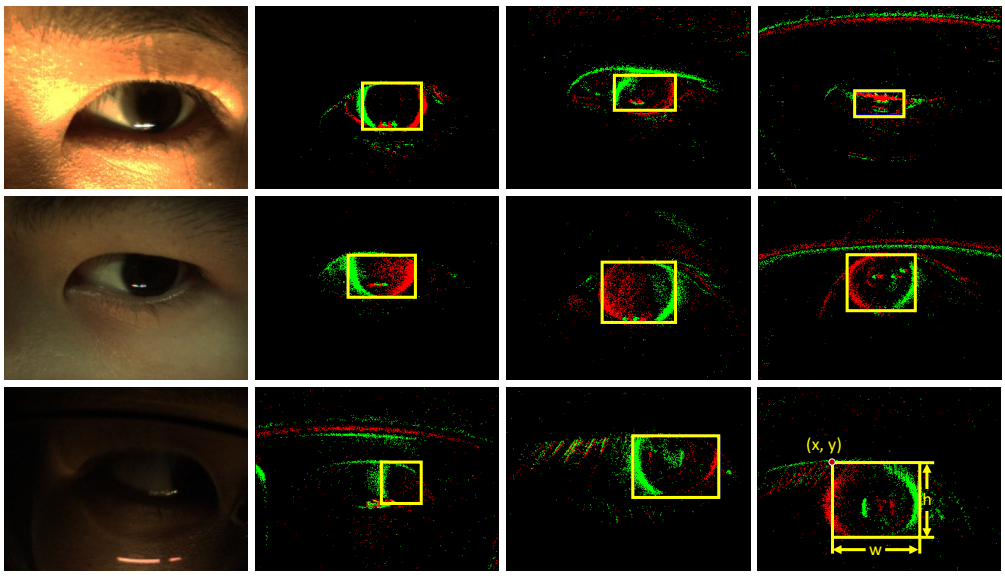

Figure 8. Labeled eye data in different sequences. The first column contains RGB images in different lighting conditions. Other images are processed event frames with red and green pixels representing binary event polarities. The yellow bounding boxes are labeled manually.

For the sake of training, the 91 original sequences in the dataset were segmented into a total of 365 shorter sequences, each containing around 100 labels, and all event points within them occur continuously. A label is a bounding box with four parameters: (x, y, w, h). Here (x, y) represent the coordinates of the upper left point of the bounding box, and (w, h) represent the width and height of the bounding box.

Among the total 91 sequences and 39,572 labeled event frames, the shortest time span between frames is only 0.648 ms, with a movement of approximately 8.32 pixels compared to the previous frame. The longest time span between frames is 1278 ms, with a movement of about 10 pixels compared to the previous frame. Clearly, grouping events by a fixed number of points can adapt to changes in eye movement speed.

4. Experiments and Results

4.1. Training Setup

The whole experiment is based on PyTorch, while the training framework for SNN uses SpikingJelly [43], also based on PyTorch. All model training and testing were conducted on an NVIDIA GeForce RTX 3090 GPU. The dataset's 365 sequences were divided into 296 training sequences and 69 testing sequences, with no overlap between the two.

SNN training employed the BPTT scheme, with the surrogate gradient function set to the arctan function. Following the dataset processing method in Section 3.2, event sets generated after dividing by the fixed point number 4000 were further divided into time steps. To align training time steps with the record length L mentioned in the method, we stipulated:

$$C \times L = 4000, \tag{12}$$

we used the Adam Optimizer with a learning rate of 0.005 for a total of 50 training epochs. For ANN, the comparison model was trained on the MMtracking [44] platform using the single-object tracking network siamrpn [40], also with a ResNet backbone.

4.2. Evaluation Metrics

Since our labels are in the form of bounding boxes, we base our tracking results' evaluation metrics on the intersection over union (IoU) and the precision error (PE) between the predicted and actual bounding boxes:

$$IoU = \frac{P \cap G}{P \cup G}, \tag{13}$$

$$PE = \sqrt{(x_P - x_G)^2 + (y_P - y_G)^2} \tag{14}$$

Here, P and G represent the prediction box and the ground truth, respectively, $(x_P,\ y_P)$ and $(x_G,\ y_G)$ represent the center coordinates of the predicted and actual bounding boxes, respectively.

Given our dataset's fixed interval annotations, not every prediction made by our SNN tracking algorithm corresponds to a manual label. Hence, we employ linear interpolation for unlabeled frames, as eye movements in short intervals can be approximated as uniformly linear. For ANN, inputs are adapted to overlapped frames generated from the total data of L time steps involved in each SNN operation, and evaluations are performed using such a method.

4.3. Performance

The test set comprises 69 independent eye movement data sequences. Both our proposed algorithm and the comparison ANN algorithm are applied to each sequence, calculating the average IoU and PE for each sequence. With C set to 500 and L to 8, our algorithm achieves the highest average IoU of 0.8072. As shown in Table 2 and Figure 9, compared to other ANN algorithms, our SNN performs better in terms of success rate and accuracy. The success rate is evaluated based on the IoU metric and accuracy is based on

PE Note that the cb-convlstm scheme directly predicts the center points of the detection boxes, so only precision error is calculated.

Table 2. Comparison of models based on average IoU and PE values.

Network	Average IoU	Average PE
SiamRPN	0.702	16.574
CB-ConvLSTM [9]	-	12.136
ResNet18	0.797	6.962
ESET (ours)	0.807	6.121

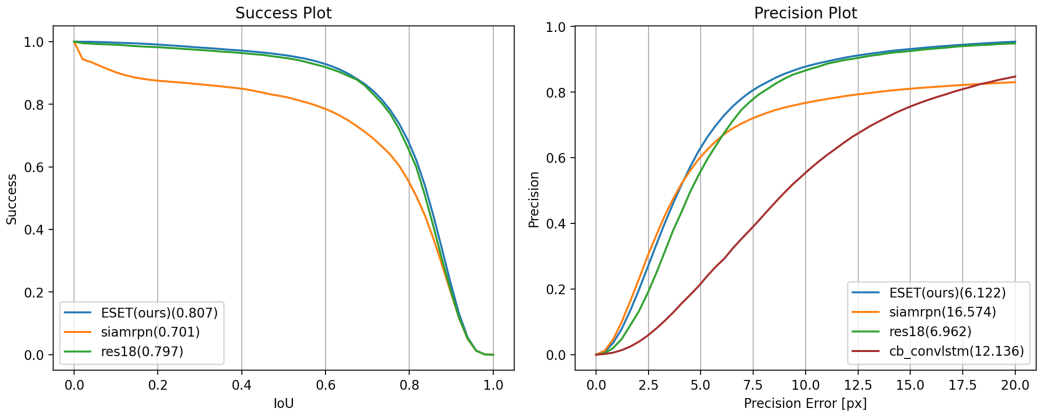

Figure 9. Success plot and precision plot of results based on IoU and precision error, respectively.

We also conducted experimental evaluations for different eye movement behaviors, as shown in Table 3. The numbers on either side of the slash are the average IoU and PE. Our model performs optimally across all categories. It excels particularly in smooth pursuit and saccade movements.

Table 3. Comparison of models with different eye movements based on the average IoU and PE.

Network	Smooth Pursuit	Saccade	Circle	Random
SiamRPN	0.776/11.26	0.739/12.818	0.705/15.285	0.681/20.372
CB-ConvLSTM	-/11.891	-/9.439	-/14.231	-/13.212
ResNet18	0.819/6.859	0.840/5.998	0.769/11.652	0.774/7.389
ESET (ours)	0.822 /6.072	0.846/5.265	0.779/10.327	0.788/6.496

Figure 10 shows a part of the test results, where (a) represents the tracking results of the ANN algorithm, (b) represents the tracking results of our SNN algorithm, and (c) compares the IoU of the tracking results of both algorithms for this sequence. As can be seen from the figure, the ANN algorithm loses track at a certain moment, but our algorithm's tracking results are relatively stable. After a period, our algorithm can re-acquire the target, whereas the ANN cannot continue tracking. This also proves that the proposed SNN algorithm has decent re-tracking performance.

To better verify the re-tracking performance of our algorithm, we selected 20%, 30%, and 40% of consecutive event points in each sequence and transformed them into pure 0 values to simulate situations where no events are generated at all. The algorithm's tracking results are shown in Table 4, including predictions with and without excluding the aforementioned 0 value outputs. Obviously, these 0 values have a minor impact on the prediction results of other event points.

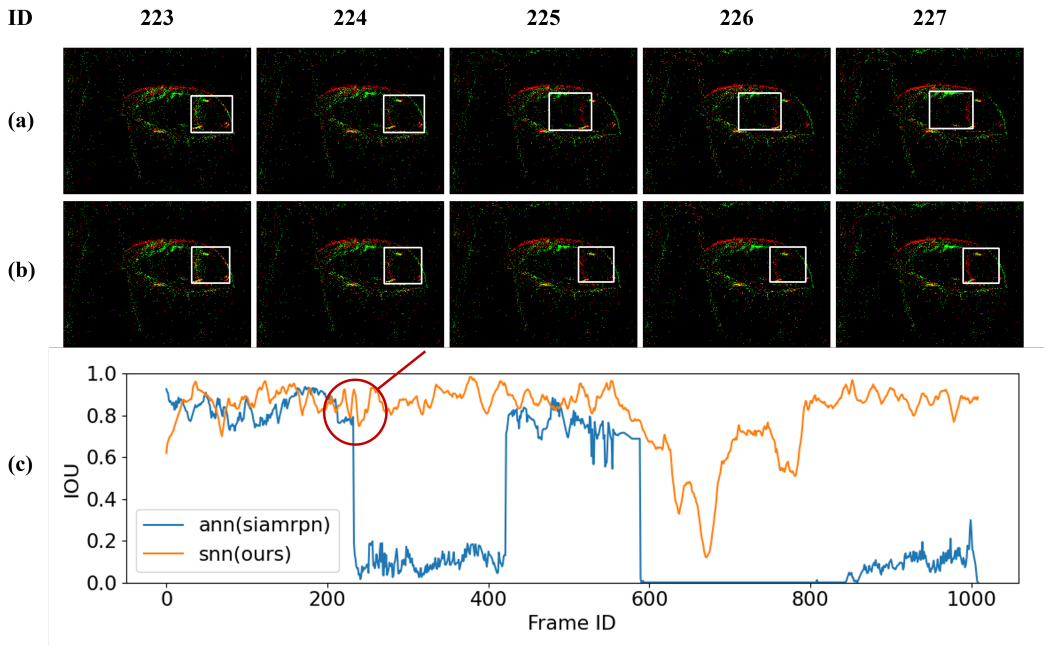

Figure 10. Comparison of re-tracking performance. (**a**) The tracking results of the ANN algorithm. (**b**) The tracking results of the proposed SNN algorithm. (**c**) The IoU of each frame result from tracking both algorithms. Red circle in (**c**) indicates the IoU value in (**a**,**b**).

Next, we analyze the performance and power consumption of our algorithm. The methodology in this paper can be divided into two parts: data acquisition with an event camera and algorithmic processing with a spiking neural network. For high-speed eye-tracking tasks, traditional cameras inevitably rely on high frame rate outputs to discern subtle movements of the eye. Taking a resolution of 346 × 260 as an example, a single frame of RGB color image output by a traditional camera occupies about 100 KB in a computer. Assuming a frame rate of 300 fps, the total data of 1500 frames over 5 s would occupy about 150 MB. In contrast, for an event camera with the same resolution, about 742,000 event points are generated in 5 s in the dataset proposed in this paper. Each event point is stored as a quadruple, occupying only 9 MB in total, saving about 94% compared to RGB images. For RGB images produced by traditional cameras, all pixel information is retained during data transmission, regardless of its relevance. An event camera, however, only records changes when a certain pixel brightness changes, which is clearly more resource-efficient for tracking tasks focused on the position of dynamic objects.

Table 4. Re-tracking performance of our ESET model with comparing the average IoU between adding distracting data or not.

Zero Value Percentage	Average IoU (without Zero Value)	Average IoU
20%	0.800	0.768
30%	0.791	0.684
40%	0.789	0.640

SNN is more energy-efficient than ANN primarily because it operates on an event-driven mechanism, meaning computations and signal transmissions only occur when a spike is received. This significantly reduces unnecessary computations and data transmissions. Additionally, SNN exhibits sparse activation, with neurons firing only when their membrane potential crosses a threshold, leading to less overall activation. Unlike ANN, which updates neurons at every computational cycle, SNN remains in a quiescent state without consuming energy in the absence of spikes. Moreover, neuromorphic hard-

ware is optimized for the event-driven, sparse operations of SNN, further enhancing their energy efficiency.

Due to the inability to process asynchronous inputs on PC and graphics card hardware, the effect of SNN can only be simulated through multiple incremental inputs, using several time steps to mimic asynchronous input. Common algorithms using SNN for visual tasks treat each input as independent and subdivide time steps within each input to simulate the temporal input of SNN. This approach significantly increases the computational consumption of SNN, as each output of an SNN requires several network cycles to be obtained, equating to a time cost several times that of an ANN with the same structure. Our algorithm, however, reasonably utilizes the prior states and temporally encoded outputs stored in the network from past time steps. A single time step's input can produce an output, thereby saving the need for seven forward passes, significantly enhancing computational speed, reducing time loss, and only amounting to 12.5% of the previous cost (if L is set to 8).

4.4. Ablation Study

We conducted ablation experiments for different fixed point numbers C and time steps L shown in Table 5. Besides the value of 8 mentioned above, we also chose values of 2, 4, 5, 10, and 16, with corresponding partition point numbers of 2000, 1000, 800, 400, and 250, respectively.

The results demonstrate that a larger C provides more spatial information in each event image, but the short time step makes it difficult to encode enough information in spike trains. Conversely, a smaller C reduces spatial information but enhances information contained in spike trains, further improving tracking performance. Clearly, the choice of C and L plays a critical role in SNN performance. The best results in our experiment were obtained with $C = 500$ and $L = 8$.

We recommend determining the value of C and L based on the actual camera settings and recording results. First, when determining C and L, we ensure that $C \times L = 4000$. This number, 4000, is derived from calculating how many event points the camera generates on average within 25 ms during uniform eye movement. We chose 4000 to make C and L as close to integers as possible. The 25 ms, or 40 fps, represents the highest frame rate of the RGB camera in DAVIS346. On the other hand, observation shows that under the scenarios set in this paper, images generated with 4000 event points are relatively clear, with boundaries that are neither too coarse nor missing, which is conducive to manual annotation.

Indeed, the temporal resolution of the network varies with different values of L. As mentioned in Section 3.2, the dataset used in this paper generates 4000 event points approximately every 25 ms. When L is set to 8, the temporal resolution of the algorithm is 25/8 = 3.125 ms. Although there is a slight decrease in tracking accuracy when L is at its maximum value of 16, the temporal resolution of the algorithm improves to 25/16 = 1.5625 ms. However, this is not the limit of the event camera. We have statistically analyzed the event generation rate in the dataset, where the shortest time taken to generate 4000 event points is 0.648 ms. This means when $L = 8$, the time resolution can reach 0.648/8 = 0.081 ms, which exceeds 10 kHz.

We also considered an extreme case of $C = 4000$, $L = 1$. In this scenario, the feature sequence length output by the SNN backbone is only 1. The average IoU, in this case, drops to 0.6546, indicating a decrease in tracking accuracy without temporal encoding, emphasizing the memory effect of SNN in this task.

Experiments also demonstrated the importance of NSPLIF in extracting temporal information. Compared to obtaining spike firing rates, the membrane potential from NSPLIF clearly better expresses features, leading to improved results for subsequent predictors.

Table 5. Results with different C/Ls and temporal feature modules.

Temporal Feature Module	C	L	Average IoU	Average PE
NSPLIF	4000	1	0.655	12.646
	2000	2	0.773	7.960
	1000	4	0.798	6.748
	800	5	0.799	6.522
	500	8	0.807	6.122
	400	10	0.789	7.587
	250	16	0.7740	7.799
Spike Rate	4000	1	0.710	10.727
	2000	2	0.712	10.253
	1000	4	0.738	9.003
	800	5	0.720	10.197
	500	8	0.760	8.206
	400	10	0.751	9.111
	250	16	0.7350	9.239

5. Conclusions

This study explored a new eye-tracking algorithm based on an event camera and a spiking neural network, addressing limitations in traditional cameras and algorithms. The algorithm effectively merges the strengths of both technologies, combining the temporal continuity of event data with the memory properties of SNN. Experiments demonstrate improvements in tracking accuracy, stability, and significant reductions in system power consumption and complexity. The envisioned use case is to implement eye tracking based on event cameras and SNN on VR glasses or other wearable eye devices. In terms of hardware, smaller and more integrated event cameras are key to combining with VR glasses. In terms of algorithms, SNN needs to run on neuromorphic chips to fully leverage their advantages and exhibit its low-power characteristics more effectively, while the development of neuromorphic chips is still immature. Future research could focus on the deployment of actual hardware, and designing a fully independent eye-tracking system. Eye movements that VR head-mounted devices permit a vestibulo-ocular reflex and the opto-kinetic reflex could be taken into consideration. The dataset can also be further refined with more balanced and diverse data. We believe that there will be significant breakthroughs in this field in the future.

Author Contributions: Conceptualization, Y.J. and L.Y.; methodology, Y.J.; software, Y.J.; validation, L.Y., W.W. and C.H.; formal analysis, Y.J.; writing—original draft preparation, Y.J.; writing—review and editing, Y.J., L.Y. and C.H.; visualization, W.W.; supervision, L.Y. and W.W.; funding acquisition, L.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China grant number 62271354.

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to participants' privacy.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Table A1 shows the some details of the dataset. Here, "FoV" and "Peak velocity" were at their maximum value among all sequences. The bold text indicates that this part of the data was captured with moving points on the screen; thus, the FoV was calculated using actual device size. However, for those without screens, the FoV was estimated based on the recorded pixel values. Note that only horizontally data were considered in "smooth pursuit" and circular and random movements were not recorded with a display.

Table A1. The number of sequences, duration, maximum FoV and peak velocity of each eye movement.

Eye Movement	Number of Sequences	Total Duration	FoV	Peak Velocity
smooth pursuit	12	140 s	112°	120°/s
smooth pursuit	12	148 s	102°	102°/s
saccade	12	124 s	110° × 70°	162°/s
saccade	12	132 s	102° × 69°	151°/s
circle	6	72 s	122° × 72°	70°/s
random	37	702 s	120° × 72°	132°/s

Table A2 shows some statistical data of Table A1, including mean value and variance of FoV and Peak velocity of all sequences. Since the movements guided by the screen should be the same, we used this as a baseline to estimate the statistics of other sequences.

Table A2. The statistical data of FoVs and PVs (Peak velocity).

Eye Movement	Mean of FoVs	Std of FoVs	Mean of PVs	Std of PVs
smooth pursuit	107.33°	4.7°	109.75°/s	6.18°/s
saccade	97.83° × 63.58°	5.35° × 3.11°	142.83°/s	11.14°/s
circle	113.16° × 65.33°	4.13° × 2.18°	67.23°/s	4.42°/s
random	91.32° × 63.58°	16.51° × 6.24°	102.83°/s	15.21°/s

We would like to reiterate that we tend to let participants perform eye movements autonomously (even if the type of eye movement was specified in advance) during the recording process. This is because this study focuses more on identifying the position of the iris in the camera's field of view rather than the gaze point. As we cannot determine the exact range of the participants' gaze, the aforementioned data are estimates with the baseline calculating from the data recording with a screen. These estimates are for reference only. Our dataset includes a considerable amount of randomness, covering various types of eye movements, making it suitable for deep learning training. However, since the participants predominantly looked at a black background during the recordings, there might be occurrences of unnatural eye movements. This could pose certain issues when analyzing specific eye movement behaviors, which is an aspect we need to improve in our future research. Please take this into consideration.

References

1. Poletti, B.; Solca, F.; Carelli, L.; Diena, A.; Colombo, E.; Torre, S.; Maranzano, A.; Greco, L.; Cozza, F.; Lizio, A.; et al. Association of Clinically Evident Eye Movement Abnormalities with Motor and Cognitive Features in Patients with Motor Neuron Disorders. *Neurology* **2021**, *97*, e1835–e1846. [CrossRef] [PubMed]
2. Diao, Y.; Geng, M.; Fu, Y.; Wang, H.; Liu, C.; Gu, J.; Dong, J.; Mu, J.; Liu, X.; Wang, C. A Combination of P300 and Eye Movement Data Improves the Accuracy of Auxiliary Diagnoses of Depression. *J. Affect. Disord.* **2022**, *297*, 386–395. [CrossRef] [PubMed]
3. Covers, M.L.V.; De Jongh, A.; Huntjens, R.J.C.; De Roos, C.; Van Den Hout, M.; Bicanic, I.A.E. Early Intervention with Eye Movement Desensitization and Reprocessing (EMDR) Therapy to Reduce the Severity of Post-Traumatic Stress Symptoms in Recent Rape Victims: A Randomized Controlled Trial. *Eur. J. Psychotraumatol.* **2021**, *12*, 1943188. [CrossRef] [PubMed]
4. Adhanom, I.B.; MacNeilage, P.; Folmer, E. Eye Tracking in Virtual Reality: A Broad Review of Applications and Challenges. *Virtual Real.* **2023**, *27*, 1481–1505. [CrossRef] [PubMed]
5. Li, N.; Bhat, A.; Raychowdhury, A. E-Track: Eye Tracking with Event Camera for Extended Reality (XR) Applications. In Proceedings of the 2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS), Hangzhou, China, 11–13 June 2023; pp. 1–5.
6. Plopski, A.; Hirzle, T.; Norouzi, N.; Qian, L.; Bruder, G.; Langlotz, T. The Eye in Extended Reality: A Survey on Gaze Interaction and Eye Tracking in Head-worn Extended Reality. *ACM Comput. Surv.* **2022**, *55*, 1–39. [CrossRef]
7. Vasylevska, K.; Yoo, H.; Akhavan, T.; Kaufmann, H. Towards Eye-Friendly VR: How Bright Should It Be? In Proceedings of the 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), Osaka, Japan, 23–27 March 2019; pp. 566–574.
8. Kim, J.-H.; Jeong, J.-W. Gaze in the Dark: Gaze Estimation in a Low-Light Environment with Generative Adversarial Networks. *Sensors* **2020**, *20*, 4935. [CrossRef] [PubMed]
9. Chen, Q.; Wang, Z.; Liu, S.-C.; Gao, C. 3ET: Efficient Event-based Eye Tracking using a Change-Based ConvLSTM Network. In Proceedings of the 2023 IEEE Biomedical Circuits and Systems Conference (BioCAS), Toronto, ON, Canada, 19–21 October 2023.

10. Zhao, G.; Yang, Y.; Liu, J.; Chen, N.; Shen, Y.; Wen, H.; Lan, G. EV-Eye: Rethinking High-frequency Eye Tracking through the Lenses of Event Cameras. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 62169–62182.
11. Angelopoulos, A.N.; Martel, J.N.P.; Kohli, A.P.; Conradt, J.; Wetzstein, G. Event-Based Near-Eye Gaze Tracking Beyond 10,000 Hz. *IEEE Trans. Vis. Comput. Graph.* **2021**, *27*, 2577–2586. [CrossRef] [PubMed]
12. Stoffregen, T.; Daraei, H.; Robinson, C.; Fix, A. Event-Based Kilohertz Eye Tracking Using Coded Differential Lighting. In Proceedings of the 2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, USA, 3–8 January 2022; pp. 3937–3945.
13. Kagemoto, T.; Takemura, K. Event-Based Pupil Tracking Using Bright and Dark Pupil Effect. In Proceedings of the Adjunct Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, New York, NY, USA, 29 October 2023; Association for Computing Machinery; pp. 1–3.
14. Feng, Y.; Goulding-Hotta, N.; Khan, A.; Reyserhove, H.; Zhu, Y. Real-Time Gaze Tracking with Event-Driven Eye Segmentation. In Proceedings of the 2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), Christchurch, New Zealand, 12–16 March 2022; pp. 399–408.
15. Shi, X.; Chen, Z.; Wang, H.; Yeung, D.-Y.; Wong, W.; WOO, W. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2015; Volume 28.
16. Liu, J.; Chi, J.; Yang, H.; Yin, X. In the Eye of the Beholder: A Survey of Gaze Tracking Techniques. *Pattern Recognit.* **2022**, *132*, 108944. [CrossRef]
17. Gehrig, D.; Scaramuzza, D. Low-latency automotive vision with event cameras. *Nature* **2024**, *629*, 1034–1040. [CrossRef]
18. Yiu, Y.-H.; Aboulatta, M.; Raiser, T.; Ophey, L.; Flanagin, V.L.; zu Eulenburg, P.; Ahmadi, S.-A. DeepVOG: Open-Source Pupil Segmentation and Gaze Estimation in Neuroscience Using Deep Learning. *J. Neurosci. Methods* **2019**, *324*, 108307. [CrossRef] [PubMed]
19. Sheela, S.V.; Abhinand, P. Iris Detection for Gaze Tracking Using Video Frames. In Proceedings of the 2015 IEEE International Advance Computing Conference (IACC), Banglore, India, 12–13 June 2015; pp. 629–633.
20. MediaPipe Iris: Real-Time Iris Tracking and Depth Estimation. Available online: https://research.google/blog/mediapipe-iris-real-time-iris-tracking-depth-estimation/ (accessed on 17 July 2024).
21. Qiu, H.; Li, Z.; Yang, Y.; Xin, C.; Bian, G. Real-Time Iris Tracking Using Deep Regression Networks for Robotic Ophthalmic Surgery *IEEE Access* **2020**, *8*, 50648–50658. [CrossRef]
22. Maass, W. Networks of spiking neurons: The third generation of neural network models. *Neural Netw.* **1997**, *10*, 1659–1671. [CrossRef]
23. Cao, Y.; Chen, Y.; Khosla, D. Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition. *Int. J. Comput. Vis.* **2015**, *113*, 54–66. [CrossRef]
24. Neftci, E.O.; Mostafa, H.; Zenke, F. Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks. *IEEE Signal Process. Mag.* **2019**, *36*, 51–63. [CrossRef]
25. Lee, J.H.; Delbruck, T.; Pfeiffer, M. Training Deep Spiking Neural Networks Using Backpropagation. *Front. Neurosci.* **2016**, *10*, 508. [CrossRef] [PubMed]
26. Zheng, Y.; Yu, Z.; Wang, S.; Huang, T. Spike-Based Motion Estimation for Object Tracking Through Bio-Inspired Unsupervised Learning. *IEEE Trans. Image Process.* **2023**, *32*, 335–349. [CrossRef] [PubMed]
27. Ji, M.; Wang, Z.; Yan, R.; Liu, Q.; Xu, S.; Tang, H. SCTN: Event-based object tracking with energy-efficient deep convolutional spiking neural networks. *Front. Neurosci.* **2023**, *17*, 1123698.
28. Luo, Y.; Xu, M.; Yuan, C.; Cao, X.; Zhang, L.; Xu, Y.; Wang, T.; Feng, Q. SiamSNN: Siamese Spiking Neural Networks for Energy-Efficient Object Tracking. In Proceedings of the Artificial Neural Networks and Machine Learning—ICANN 2021, Bratislava, Slovakia, 14–17 September 2021; Farkaš, I., Masulli, P., Otte, S., Wermter, S., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 182–194.
29. Yang, Z.; Wu, Y.; Wang, G.; Yang, Y.; Li, G.; Deng, L.; Zhu, J.; Shi, L. DashNet: A Hybrid Artificial and Spiking Neural Network for High-Speed Object Tracking. *arXiv* **2019**, arXiv:1909.12942.
30. Hagenaars, J.; Paredes-Valles, F.; de Croon, G. Self-Supervised Learning of Event-Based Optical Flow with Spiking Neural Networks. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 7167–7179.
31. Zhu, L.; Wang, X.; Chang, Y.; Li, J.; Huang, T.; Tian, Y. Event-Based Video Reconstruction via Potential-Assisted Spiking Neural Network. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 3584–3594.
32. Zhang, J.; Dong, B.; Zhang, H.; Ding, J.; Heide, F.; Yin, B.; Yang, X. Spiking Transformers for Event-Based Single Object Tracking. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 8791–8800.
33. Hu, Y.; Liu, S.-C.; Delbruck, T. v2e: From Video Frames to Realistic DVS Events. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Nashville, TN, USA, 19–25 June 2021; pp. 1312–1321.
34. Gehrig, D.; Gehrig, M.; Hidalgo-Carrio, J.; Scaramuzza, D. Video to Events: Recycling Video Datasets for Event Cameras. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 3583–3592.

35. Rebecq, H.; Gehrig, D.; Scaramuzza, D. ESIM: An Open Event Camera Simulator. In Proceedings of the 2nd Conference on Robot Learning, Zürich, Switzerland, 29–31 October 2018; pp. 969–982.
36. Fang, W.; Yu, Z.; Chen, Y.; Huang, T.; Masquelier, T.; Tian, Y. Deep Residual Learning in Spiking Neural Networks. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 21056–21069.
37. Fang, W.; Yu, Z.; Chen, Y.; Masquelier, T.; Huang, T.; Tian, Y. Incorporating Learnable Membrane Time Constant to Enhance Learning of Spiking Neural Networks. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 2641–2651.
38. Brunel, N.; Latham, P.E. Firing Rate of the Noisy Quadratic Integrate-and-Fire Neuron. *Neural Comput.* **2003**, *15*, 2281–2306. [CrossRef] [PubMed]
39. Xu, Y.; Wang, Z.; Li, Z.; Yuan, Y.; Yu, G. SiamFC++: Towards Robust and Accurate Visual Tracking with Target Estimation Guidelines. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 12549–12556. [CrossRef]
40. Li, B.; Yan, J.; Wu, W.; Zhu, Z.; Hu, X. High Performance Visual Tracking with Siamese Region Proposal Network. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8971–8980.
41. iniVation. DAVIS346 Event Camera Datasheet. Available online: https://www.inivation.cn/template/pc/pdf/DAVIS346.pdf (accessed on 17 July 2024).
42. Goshvarpour, A.; Goshvarpour, A. Eye-blinking analysis as a marker of emotional states. *Multimed Tools Appl.* **2021**, *80*, 33727–33746. [CrossRef]
43. Fang, W.; Chen, Y.; Ding, J.; Yu, Z.; Masquelier, T.; Chen, D.; Huang, L.; Zhou, H.; Li, G.; Tian, Y. SpikingJelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Sci. Adv.* **2023**, *9*, eadi1480. [CrossRef]
44. MMTracking: OpenMMLab Video Perception Toolbox and Benchmark. Available online: https://github.com/open-mmlab/mmtracking (accessed on 17 July 2024).