Retina : Low-Power Eye Tracking with Event Camera and Spiking Hardware

Pietro Bonazzi[1], Sizhen Bian[1] , Giovanni Lippolis[2], Yawei Li[1], Sadique Sheik[3] , Michele Magno[1]

[1]ETH Zürich, [2]Inivation AG, [3]Synsense AG

Abstract

This paper introduces a neuromorphic dataset and methodology for eye tracking, harnessing event data captured streamed continuously by a Dynamic Vision Sensor (DVS). The framework integrates a directly trained Spiking Neuron Network (SNN) regression model and leverages a state-of-the-art low power edge neuromorphic processor - Speck. First, it introduces a representative event-based eye-tracking dataset, "Ini-30," which was collected with two glass-mounted DVS cameras from thirty volunteers. Then, a SNN model, based on Integrate And Fire (IAF) neurons, named "Retina", is described , featuring only 64k parameters (6.63x fewer than 3ET) and achieving pupil tracking error of only 3.24 pixels in a 64x64 DVS input. The continuous regression output is obtained by means of temporal convolution using a non-spiking 1D filter slided across the output spiking layer over time. Retina is evaluated on the neuromorphic processor, showing an end-to-end power between 2.89-4.8 mW and a latency of 5.57-8.01 ms dependent on the time to slice the event-based video recording. The model is more precise than the latest event-based eye-tracking method, "3ET", on Ini-30, and shows comparable performance with significant model compression (35 times fewer MAC operations) in the synthetic dataset used in "3ET". We hope this work will open avenues for further investigation of close-loop neuromorphic solutions and true event-based training pursuing edge performance.
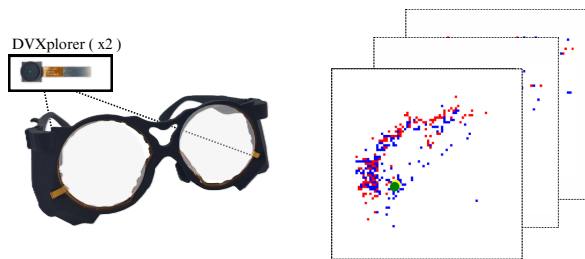

Figure 1. A picture of the hardware for data collection (left) and an example of the video recordings (right) with ground truth (green) and prediction (yellow).

Reproducibility

https://github.com/pbonazzi/retina

Acknowledgments

1. Introduction

Neuromorphic systems, mimicking the neurobiological architectures of the human brain, have emerged as a promising paradigm for sensing and processing [1, 5, 17]. Event cameras and spike-based computation present distinctive advantages compared to traditional frame-based cameras and Artificial Neuron Network (ANN), such as low power and decreased computing complexity. In the context of neuromorphic systems for eye tracking, various challenges confront the field. First, even the most recent methods either rely on frame-based input [6, 24] or focus on end-to-end tasks, such as gaze tracking learned from fixed screen-level coordinates [2]. However, pupil tracking serves as a crucial initial phase in the gaze estimation pipeline. When mastered, it can facilitate applications to extend their capabilities beyond controlled environments. Second, state-of-the-art algorithms for event-based eye tracking are model-based and require subject-specific calibration [2, 24] and fitting at the moment of use, thus limiting their generalization to consumer application [23]. Furthermore, eye-tracking algorithms and systems demand a nuanced equilibrium and trade-off between resolution, frame rate, latency and power consumption, as each pixel carries energy and bandwidth costs during the acquisition process. One of the most recent papers on event-based eye tracking without subject-dependent calibration was authored by Chen et al. [6]. Notably, in this work, they accumulated and normalized synthetically generated events to a 32-bit pixel resolution frame. This approach is inefficient as it does not leverage the asynchronous 1-bit nature of the event data. Finally, there is a need for end-to-end assessments of power consumption and latency, with the object of benchmarking the

energy efficiency and real applicability of machine learning algorithms, that leverage event input and spike-based computation.

To address these challenges, we propose an eye-tracking model, dubbed "Retina", suitable for deployment on a neuromorphic asynchronous System-on-Chip (SoC), Synsense Speck (Speck). We leverage spike-based computation and real event input to offer energy-efficient eye tracking of the pupil from near-eye events. In detail, our work introduces the following three key contributions:

1. **Event-based Eye Tracking Dataset, "Ini-30"**: We introduce the first event-based eye-tracking dataset, named "Ini-30", recorded on a glass frame. Our dataset is collected with two event cameras and features variable recording lengths and event counts from 30 volunteers, providing an ideal benchmark for modeling the heterogeneity of event-based eye tracking in real-world scenarios. Our dataset brings new challenges, such as different event temporal evolution trends across different recordings. Thus, we slice the event data based on quantity of events instead of fixed timestamps. This method enables domain gap adaptation between different DVS, solves the difference in temporal event growth between real recordings, and improves eye tracking precision.

2. **Event-based Eye Tracking Algorithm, "Retina"**: Retina is a SNN structure based on IAF neurons, followed by a non-spiking temporal weighted-sum filter for regression, which converts spikes to bounding box predictions. The filter allows IAF neurons to learn temporal information without having to fall back to a voltage decay factor [4] or recurrent neurons, which are not supported in the hardware, Speck. To the best of our knowledge, Retina is the first eye tracking algorithm, suitable for deployment on neuromorphic hardware. Compared to 3ET [6], it shows superior precision (-20% centroid error) and reduced computational complexity (-30x MAC).

3. **Deployment on Neuromorphic Hardware**: Finally, the deployment of our model on an "edge" neuromorphic SoC, Speck, provides for the first time power and latency results for network inference on chip and an end-to-end evaluations using the on-board DVS camera.

2. Related Work

This section discusses the state-of-the-art explorations of eye-tracking based on a comprehensive literature survey. Depending on the utilized signal form (non-event or event input), we separate the existing eye-tracing solutions into two approaches:

2.1. Non-Event-Based Eye Tracking

Conventional non-event-based eye-tracking includes model-based and appearance-based methods. The former

one either tracks specular glint reflections for corneal curvature center detection [9, 19], or extracts salient geometrical features of the eye from frames and tracks the pupil with an optimized fitting method based on a physics eye model [11, 20]. Such an approach commonly supplies impressive tracking accuracy, achieving sub-degree tracking error [13], while stunted by the deployment complexity, such as ambient light conditions, image resolution, and calibration requirement. The latter one typically applies a trained machine-learning model to the raw eye images for tracking [3, 10, 15, 23]. This approach gives an end-to-end, deployable eye-tracking solution, which is heavily limited by the frame rate of the camera, with resulting tracking rate that can reach a maximum of 300 Hz. Differently from this line of work, the event-based eye tracking can reach beyond kHz [2] of update rate and provide energy-efficient observations [7].

2.2. Event-Based Eye Tracking

Benefiting from the sparse event stream and high dynamic range of event cameras, event-based eye-tracking has emerged as a groundbreaking solution enabling beyond-kHz and low-power consumption eye-tracking, as listed in Tab. 1. The first event-based eye-tracking work was published in 2020 [2], in which the authors proposed a hybrid frame-event-based near-eye gaze tracking system offering update rates beyond 10 kHz with an accuracy comparable to commercial tracker. The algorithm is based on a parametric pupil model and utilizes the event to update the model with a pupil-fitting method. This work is sensitive to sensor noises, as choosing the useful event that can be used to update the model is challenging. Following this work, researchers have published another three event-based pupil/gaze-tracking papers in the past three years. Stoffregen et al. [18] described the first fully event and model-based glint tracker, which is robust to background disturbances and has a sampling rate of 1 kHz with an estimated power of 35 mW (sensing components only). Here the author used coded differential lighting to enhance the glint detection with an event camera. Similar to [2], in [24], the authors presented a hybrid eye-tracking method that leverages both the near-eye grayscale images and event data for robust and high-frequency eye tracking. The proposed matching-based pupil tracking method gave a pixel error of only 1.2 px with a peak tracking frequency of up to 38.4 kHz. In [6], the authors proposed a sparse change-based convolutional Long-Short-Term-Memory (LSTM) model for event-based eye tracking, which reduces arithmetic operations by approximately 4.7×, compared to a standard convolutional LSTM, without losing accuracy when tested on a synthetic event dataset.

There are several limitations to existing methods. First, they rely on frames, either directly from the sensor output [2, 24],

Table 1. A summary of the related work in event-based pupil/gaze tracking

Year-Work	DVS Camera	Dataset	Input	Algorithm	Event Rate	Energy	Precision
2020-[2]	DAVIS346b	Customized	Frames/Events	Model-based	$\geq 10\ kHz$	N/A [a]	$0.45° - 1.75°$
2022-[18]	Prophesee G3.1	N/A	Events/LED Markers	Glint Detection	$1\ kHz$	$\approx 35\ \text{mW}$ [b]	< 0.5px
2023-[24]	DAVIS346	EV-Eye	Frames/Events	Point2Edge	$\leq 38.4\ kHz$	N/A	1.2-7.7px
2023-[6]	N/A	Synthetic	Event-frames	ConvLSTM	$95\ kHz$	N/A	N/A
Ours	DVXplorer	Ini-30	Events	SNN	$\leq 5kHz$ [c]	2.89[d]$4.8mW$ [d]	3-8px

[a] Not Available.
[b] Sensor power only.
[c] Depending on the DVS and time window lenght.
[d] End-to-end power.

which decreases the power efficiency of the system, or accumulate the events into frames and process the frames with deep neural network [6], which sacrifices the temporal resolution, introduces latency and increase the memory footprint. Second, with a purely event-based solution, auxiliary devices are used to enhance specular events [18]. Besides that, none of the existing works carried out a real deployment of their proposals, especially on a neuromorphic platform that perfectly matches the sparsity of the event stream. Thus, the system-level performance in the energy and latency that an event solution can bring is still unclear. In contrast, our work presents a SNN supported by an end-to-end neuromorphic edge system leveraging pure events stream from a DVS camera.

3. Dataset

Several prominent eye-tracking and gaze estimation datasets have contributed significantly to the advancement of this field using frame cameras [12, 22]. To the best of our knowledge, the only available event-based datasets have been presented by Angelopoulos et al. [2] for *gaze tracking* and in Zhao et al. [24] for *gaze and eye tracking*. As a matter of fact, the state-of-the-art methods, i.e. 3ET [6], had to generate synthetic event-based dataset to develop eye tracking algorithms. In this paper, we introduce a representative event-based *eye tracking* dataset, dubbed "Ini-30", which, for the first time, was collected with two event cameras (one per eye) mounted on a glass-frame without fixing the head of the user on a head set in a controlled environment.

3.1. Dataset Collection

The Ini-30 dataset is collected with two event cameras mounted on a glass frame. Each DVXplorer sensor (640 × 480 pixels) is attached on the side of the frame. The power supply was provided via a 2 meter cable connected from the cameras to a computer, which provided enough freedom of movement. Differently from [2, 24], the participants were

not instructed to follow a dot on a screen, but rather encouraged to look around to collect natural eye movements. As shown in Fig. 1, the event cameras were securely screwed on a 3D-printed case attached to the side of the glass frame.

The data was annotated based on accumulated linearly decayed events by defining the pixel intensity as function of the linear accumulation of previous pixel intensity. Next we labeled the position of the pupil in the DVS's array manually, using an assistive labeling tool. We discarded the first 20ms of events to ensure the eye was visible and annotations met the level of image-based annotators. The number of labels per recording was intentionally variable, spanning from 475 to 1'848 with a time per label ranging from 20.0 to 235.77 milliseconds depending on the overall duration of the sample.

This setup allows for unconstrained head movements, enables to capture event data from eye movement in a "in-the-wild" setting and allows the generation of a representative, unique, diverse and challenging dataset. In the next section, we characterize the dataset and compared further with event-based datasets for gaze tracking.

3.2. Dataset Comparaison

In this section, we present a comparative analysis of the dataset presented in [2, 24] with our proposed Ini-30 dataset. We summarize the key points in Tab. 2. To the best of our knowledge, Ini-30 is the first eye-tracking event-based dataset with pupil location labelled on the sensor. The other event-based datasets available [2, 24] consist of point coordinates on a display, instead of pupil location coordinates on the DVS's array. Labeling pupil locations provides superior precision and granularity for understanding gaze behavior compared to screen coordinates.

In addition, [2, 24] employs a lower resolution sensor, DAVIS346b (346 × 260 px), whereas our Ini-30 dataset incorporates DVXplorer sensor, 640 × 480 px. Both datasets are collected using Near-Infra Red (Illumination) (NIR) illumination technology, to highlights the events surrounding

Table 2. Comparison between our proposed dataset (Ini-30) with the one proposed in [2] and [24].

Aspect	[2] [24]	Ini-30 (Ours)
Resolution	346×260px	640×480px
Glass Frame	✗	✓
Pupil Label	✗	✓
Variability*	Low	High

* Variability refers to the difference in duration of each recordings and the event temporal distribution.

the pupil. Additionally, [2, 24] captures subjects with fixed head positions, while Ini-30 is designed for use cases with unconstrained head movements with cameras mounted directly on glass frames. Because of this, Ini-30 covers a broader range of situations. For example, temporally, [2] maintains fixed recording lengths ($30s$) and linearly growing event counts, while Ini-30 showcases richer variability, Tab. 3, with recording durations spanning from 14.64 to 193.8 seconds.

Tab. 3 provides key statistics for Ini-30, in the context of event cameras updates rates. The sampling time step ranges from 61 to 346 microseconds, and the event count per timestep spans from 3 to 5'000 events.

Table 3. Statistics for sampling times (Ts [μs]) and number of events per timestamps in Ini-30.

Name	Median	Mean	Std	Min	Max
Sampling Time	200	200	14	61	346
Events / Ts	94	175	299	3	4'799

Overall, our dataset encompasses a diverse range of recordings (30), with labels per recording spanning from 475 to 1'848, with total event data varying from 4.8 million to 24.2 million, and time per label ranging from 20.0 to 235.77 milliseconds. This information helped informed data preparation strategies in our models which consider a temporal dimension.

4. Methodology

We propose a low latency and lightweight architecture and learning rule for an eye-tracking algorithm based on event data. Our network is a single SNN featuring spiking spatial convolutions and fusible batch normalization layers. The spiking outputs are converted to continuous values by means of a 1D convolution layer with fixed weights. An overview of our network configurations can be found in Tab 5, while other implementation details are described in the next subsections.

4.1. Data Preparation

Since the algorithm is deployed on the neuromorphic SoC Speck, which has two-channel support for a 64x64 DVS's resolution, we prepared the dataset to bridge this domain gap. First, we transformed the rectangular resolution of the original data to a squared array of 512x512, by shifting the y-axis to 16 pixels and discarding 128 X-coordinates in correspondence to the spatial location where fewer events are present and no label appeared ($x < 96$ and $x > 608$).

Next, we applied sum pooling to reach the Speck compatible resolution and proceeded to dynamically slice the event temporally. Every video was sliced, from a point in time corresponding to a pupil label timestamp, until we obtained a desired pixel activations. This results in better input data for convolution layers, (Tab. 4), Fig. 2 shows an illustrated example of the working principles of the two techniques. In case multiple events from the same pixels are found during this slicing step, we keep the event polarity with the highest number of events and then clip the event frame back to 0 and 1, ensuring only one channel is active at every timebin. Finally, since in our dataset, the sampling time of events has a median of 200 μs, Tab. 3, while pupil labels have a frequency of around 30 ms, we weight-interpolated new labels at bin time using the two closest labels in the source recordings. We trained our network with 64 timebins, shuffling each slice of recording only during training.

4.2. Network Architecture

After the data preparation step, events are processed sequentially at fixed timestamps by kernel of the convolution, batch normalization layers and IAF neurons in our model. We designed each layer of our network to fit the memory limitation of the available cores in the targeted platform,
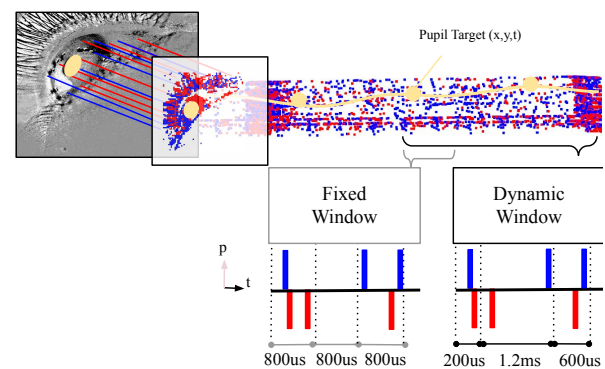

Figure 2. An example illustrating the different techniques for slicing events video recordings (red, blue) in time: A) dt = $800us$, B) events count = 2.

Table 4. An example of the qualitative improvements of slicing video recording with a dynamic time window (1) compared to fixed time (2).

Method	t	t_{i+1}	t_{i+2}
(1)			
(2)			

Speck.

We computed the total kernel memory available K_{MT} for each layer following the basic formula:

$$K_{MT} = c2^{log_2^{k_x k_y} + log_2^f} \quad (1)$$

Where c is the input channel of M, f is the output channel number, k_x and k_y are the kernel size. The necessary neuron memory N_M entries are computed by solving the following formulas:

$$N_M = f f_x f_y \quad (2)$$

Where f is the output channel number and f_x and f_y depend on the input feature map size c_x, c_y, stride s_x, s_y, and padding p_x, p_y, following the relationships:

$$f_x = \frac{c_x - k_x + 2p_x}{s_x} + 1, f_y = \frac{c_y - k_y + 2p_y}{s_y} + 1 \quad (3)$$

In Tab. 5, we present an overview of the network. The events are processed by Layer ID "1" and sequentially transmitted to the following layers. Every convolution operation is followed by a spiking neuron with a spiking threshold of 1 and a minimum voltage membrane of -1. Given that spike generation is non-differentiable, we use a surrogate gradient [14] periodic exponential function [21].

The spikes generated by the final layer are converted to continuous values using a 1-dimensional non-spiking temporal weighted-sum filter with fixed weights, discussed in Sec. 4.4. Next, akin to other grid-based methodologies such as those presented by Redmon et al. [16], we modify the output layer to consist of 4×4 cells, each containing two 5-dimensional vectors representing the top right and bottom left coordinates of the bounding box, along with a confidence score. To refine the bounding box localization, we employ a post-processing step using Non-Maximum Suppression (NMS) [8], which helps eliminate redundant detections by retaining only the highest-scoring bounding boxes while discarding overlapping alternatives. In section Sec. 4.5, we present how we compare the prediction to a generated target bounding box synthetically gener-

ated around the original 1-pixel label by expanding it to 2 pixels in each direction.

4.3. Neuron Model

The IAF neuron model, operating after every convolution layer, is characterized by a straightforward mathematical formulation. It involves the integration of incoming synaptic inputs (convolution operations) and the generation of a spike once a certain membrane potential threshold is reached. The dynamics of the IAF neuron are described by the following differential equation:

$$\tau_m \frac{dV}{dt} = -V(t) + R_m I_{syn}(t) \quad (4)$$

where $V(t)$ is the membrane potential at time t, τ_m is the membrane time constant, R_m is the membrane resistance, and $I_{syn}(t)$ represents the synaptic input current. The neuron fires when $V(t)$ surpasses a predefined threshold $V(th) = 1$, at which point the membrane potential is reset to a resting value $V(reset) = 0$. One of the key features of the IAF neuron model is its integration mechanism for incoming synaptic inputs. The synaptic input $Isyn(t)$ is often modeled as a sum of weighted contributions from different synapses:

$$I_{syn}(t) = \sum_j w_j \cdot I_j(t - t_j) \quad (5)$$

where w_j represents the synaptic weight, $I_j(t-t_j)$ is the synaptic input spike train arriving at time t from synapse j with a spike at t_j.

4.4. Temporal Weighted-Sum Filter

The implemented temporal weighted-sum filter can be described as follows:

$$y(t) = \sum_{i=1}^{N} w_i \cdot x(t - i) \quad (6)$$

where $y(t)$ represents the filtered output at time t, N is the length of the convolutional kernel, w_i denotes the filter weight at position i in the kernel, and $x(t - i)$ is the input value at time $t - i$.

The filter weights (w_i) are determined by a 'synaptic kernel' S(t) and 'membrane kernel' M(t), which in turn are computed based on a membrane constant (τ_{mem}) and a synaptic constant (τ_{syn}):

$$S(t) = \exp\left(-\frac{t}{\tau_{syn}}\right), M(t) = \exp\left(-\frac{t}{\tau_{mem}}\right) \quad (7)$$

The membrane kernel initializes the weights of a 1D convolution applied to the synaptic kernel. The weights w_i of the temporal weighted-sum filter w_i are initialized as a result of the synaptic and membrane kernel convolution.

Table 5. The network configuration, memory footprint and core compatibility on Speck for each layer.

Layer ID	SNN	c_{in}	c_{out}	$k_x * k_y$	$s_x * s_y$	N_M	K_{MT}	Cores ID
1	BatchConv IAF Pool	2	16	5×5 2×2	2×2 1×1	0.78 KiB	15.02 KiB	all
2	BatchConv IAF Pool	16	64	3×3 2×2	1×1 1×1	9.00 KiB	64.00 KiB	0, 1, 2
3	BatchConv IAF Pool	64	16	3×3 2×2	1×1 1×1	9.00 KiB	4.00 KiB	all
4	BatchConv IAF	16	16	3×3	1×1	2.25 KiB	1.00 KiB	all
5	BatchConv IAF	16	8	3×3	1×1	1.12 KiB	0.50 KiB	all
6	BatchConv IAF	8	16	3×3	1×1	1.12 KiB	1.00 KiB	all
7	BatchConv IAF	144	128	1×1	1×1	18.00 KiB	1.12 KiB	3, 4, 5, 6
8	BatchConv IAF	128	160	1×1	1×1	34.37 KiB	2.42 Ki	5, 6

4.5. Loss Function

Our loss function $\mathcal{L}$ is a combination of several components: a box loss $\mathcal{L}_{box}$, a confidence loss $\mathcal{L}_{conf}$, a synaptic loss $\mathcal{L}_{syn}$.

The box loss $\mathcal{L}_{box}$ measures how well the model can localize the pupil within the image, by minimizing the mean squared error distance between the predicted p_i and target t_i bounding boxes, represented as:

$$\mathcal{L}_{\text{box}} = \sum_{i=1}^{N}(p_i - t_i)^2 \qquad (8)$$

The confidence loss $\mathcal{L}_{conf}$ measures how confident the model is about its prediction, by penalizing low confidence scores for correct predictions and high confidence scores for incorrect predictions. This component is calculated as a mean squared error distance between the predicted and target confidence scores (c_i and g_i):

$$\mathcal{L}_{\text{conf}} = \sum_{i=1}^{N}(c_i - g_i)^2 \qquad (9)$$

The synaptic loss $\mathcal{L}_{syn}$ is the first of our regularization terms and it ensures that the number of multiply-accumulate operations performed by the neurons in the network at each layer is within a range the neuromorphic SoC Speck can handle (1e6). $\mathcal{L}_{syn}$ is formulated as the squared difference between the target synaptic operations with each layer

synaptic operations, normalized by the square of the target synaptic operations. The total loss $\mathcal{L}$ is a weighted sum of these components. To summarize, the $\mathcal{L}_{box}$ and $\mathcal{L}_{conf}$ are tasks losses which are used to detect the pupil in the event array. The $\mathcal{L}_{syn}$ is a regularization component in order to deploy the network on the neuromorphic device.

5. Experiments

5.1. Setup

Our precision results are based on the centroid error in pixels extracted from the predicted bounding box. We evaluated the efficiency of our algorithm by measuring the power (P) [mW] consumption, energy (E) [mJ] consumption and the latency (L) [ms] on the neuromorphic SoC. In addition, we evaluated the network complexity with the number of parameters and Multiply and Accumulate (MAC) operations. Regarding our implementation, we trained Retina's convolution layer M with 8-bit weight parameters (kernel memory K_{MT}) and a 16-bit spiking neuron states (neuron memory N_M). Batch normalization layers are fused to the convolution blocks in inference. The models are trained for 576 iterations using the ADAM optimizer and a step learning rate scheduler with a gamma of 0.8. Furthermore, we reset the states of the neuron at every iteration. The weights of the losses are $\lambda_{box} = 7.5$, $\lambda_{conf} = 1.5$ and 1e-7 for λ_{syn}. Finally, we train our models with a batch size of 16, a sequence length of 64 , and an initial learning rate of 1e-

3. The training takes 1 hour on a single NVIDIA GeForce RTX 4090.

5.2. Ablation Studies

This section examines the two main components of our methodology: the dynamic event windows, the temporal weighted-sum filter as well as the loss function. The ablation studies are performed on the Ini-30 Dataset and using the best models.

5.2.1 Event-Based Video Recording Slicing

Precision Results: In Tab. 6, we evaluate the effects of fixed time windows (dt) on the precision of pupil localisation. The evaluation considers the median, the minimum and maximum number of events/time-window per bin, see Tab. 3. The comparison is carried out considering the average time window. The dynamic event time window consistently outperform the fixed time window on the validation set.

Table 6. The performance of different event slicing methods.

Method	Event Count	Time Window	Error (px) ↓
Fixed	198 [52, 401]	$1.1ms$	4.40 ($\pm$ 2.69)
Dynamic	100	$1.1ms$ [0.41, 3.1ms]	**3.54 ($\pm$ 1.43)**
Fixed	242 [69, 481]	$1.6ms$	5.18 ($\pm$ 1.37)
Dynamic	150	$1.6ms$ [0.63, 4.0ms]	**3.49 ($\pm$ 1.18)**
Fixed	277 [84, 541]	$2.1ms$	3.39 ($\pm$ 1.02)
Dynamic	200	$2.1ms$ [0.86, 4.6ms]	**3.46 ($\pm$ 1.36)**
Fixed	331 [110, 630]	$3.0ms$	3.71 ($\pm$ 1.40)
Dynamic	300	$3.0ms$ [1.10, 8.4ms]	**3.24 ($\pm$ 0.79)**

Firing Rates: In Fig. 3, we provide insights into the firing rates of the SNN at different network depths. Results show the dynamic time window of events has considerably lower firing rate (10% instead of 20%) in the first layer.

5.2.2 Temporal Weighted-Sum Filter

The temporal weighted-sum filter plays a crucial role in enhancing the performance of our model. In Tab. 7, we present
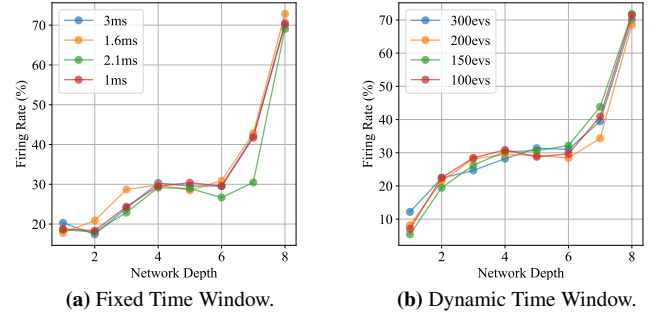

(a) Fixed Time Window. **(b)** Dynamic Time Window.

Figure 3. The firing rates of the trained SNN at different network depths with different slicing methods and time windows.

an evaluation of the impact of two key parameters, namely τ_{mem} and τ_{syn}, on the overall effectiveness of the filter. Notably, the without the filter yields an error of 24.46 pixels (± 3.17). Adjusting the values of τ_{mem} and τ_{syn} allows for a tailored optimization of the filter's performance.

Table 7. The effect of the components of the temporal filter.

Figure	τ_{mem}	τ_{syn}	Kernel Size	Error (px) ↓
-	Not Used			24.46 ($\pm$3.17)
B	5	1	20	21.70 ($\pm$4.54)
C	1	5	20	8.72 ($\pm$4.60)
D	5	5	20	**3.24 ($\pm$ 0.79)**
E	10	10	20	3.52 ($\pm$ 0.89)
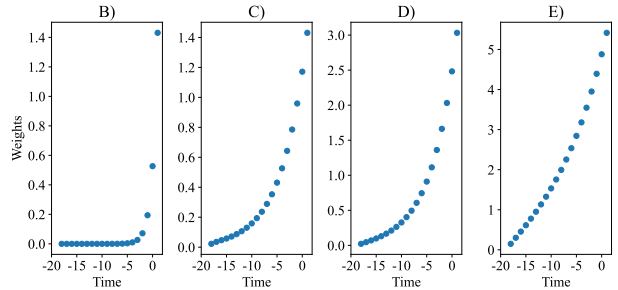
5.2.3 Architecture Components

We conducted additional experiments on variants of the Retina model, specifically one without neuron state resets ("Retina w/o resets") during training and another trained with 3ET's loss function ("Retina w/o box"). The latter variant shares the same network architecture as Retina, except for the output layer, which directly predicts pupil coordinates.

Tab. 8 shows the importance of resetting the neuron states during training and the efficacy of predicting bounding boxes instead of single pixels coordinates.

5690

Table 8. The performance of our model on the validation set.

Method	Error (px) ↓
Retina w/o *box*	5.89 (± 1.71)
Retina w/o *reset*	7.99 (± 5.79)
Retina	**3.24 (± 0.79)**

5.2.4 Latency & Power Consumption

In this section, we present a comprehensive analysis of the power consumption, energy and latency metrics on the Speck platform for the two time windows of events, namely the Dynamic Window (300 events) and the Fixed Window (3ms). We injected the events bins at sufficiently distant timestamps ($100ms$) in order to isolate the response of the chip. Tab. 9 provides an overview of the power consumption across various components contributing to the overall power profile in the Speck namely: Vdd, Vda, (referring to the digital and analog steps on the DVS), Logic (the power usage of the SNN), RAM, and I/O. The Dynamic Window can be used to stabilize the power consumption in case higher rates of events are expected, however in our test the Fixed Window was able to utilize the resources available more efficiently. We computed the latency by calculating the difference between the timestamp for any given bin from the input timestamp of the events to the generation of spikes (on average 1'700) for a prediction.

Table 9. The average and peak latency (L - ms), power consumption (P - mW) and energy (E - mJ) on Speck for the validation dataset using two time windows of events.

Device	Channel	Unit	Dynamic	**Fixed**
DVS	Vdd		0.03 [0.01-0.05]	
	Vda		0.6 [0.58-0.63]	
Processor	Logic	mW	2.43 [0.28-14.34]	**1.26** [0.28-15.08]
	RAM		1.64 [0.01-9.44]	**0.90** [0.03-8.95]
	I/O		0.10 [0.08-0.23]	
Total	End-to-End	mW	4.80	**2.89**
		ms	8.01	**5.57**
		mJ	38.40	**16.10**

5.3. Benchmark Comparison

We quantified the efficacy of our system using the Centroid Error metric. The validation protocol employed a leave-two out participant scheme for Ini-30. Tab. 10 presents the centroid error results for a DVS resolution of 64x64x2. The results of 3ET on the synthetic dataset are different compared to those reported in the original manuscript because we modified the data pipeline to transmit only 1-bit event data.

Table 10. The performance of our model on the validation set.

Dataset	3ET [6]	Retina
Ini-30	4.48 (± 1.94)	**3.24 (± 0.79)**
Synthetic (LPW) [6]	**5.33 (± 1.59)**	6.46 (± 2.49)

In Tab. 11, we present a comprehensive comparison between our proposed model and the state-of-the-art 3ET [6], focusing on both the number of parameters and the volume of MAC operations. Notably, our model demonstrates a remarkable reduction in complexity, underscoring its efficiency.

Table 11. A breakdown of the network complexity.

Method	MAC Operations ↓	Parameters ↓
3ET [6]	107M	418k
Retina	3.03M	63k

6. Discussion and Future Work

As shown in Tab. 8, Retina is less precise when trained without the continuous resetting of neuron states. The ongoing reset of neuron states may introduce potential disruptions in continuous tracking on a neuromorphic chip. Exploring strategies to mitigate this challenge and optimize model performance under those additional hardware constraints is a promising avenue for future research.

7. Conclusion

Our work describes an energy-efficient ($5mW$, end-to-end), low latency ($6ms$, end-to-end), and accurate (3-4px) neuromorphic approach for eye tracking, leveraging the strengths of the neuromorphic form of both sensor and processor, and a truly lightweight and deployable spiking neural network model. The presented model demonstrates better than baseline precision with significantly reduced computational complexity. Finally, we hope the introduced event-based eye-tracking dataset Ini-30 could promote further exploration in the realm of real-world ultra-low power wearable eye-tracking technology.

References

[1] Steven Abreu, Muhammed Gouda, Alessio Lugnan, and Peter Bienstman. Flow cytometry with event-based vision and spiking neuromorphic hardware. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 4139–4147, 2023. 1

[2] Anastasios N Angelopoulos, Julien NP Martel, Amit PS Kohli, Jorg Conradt, and Gordon Wetzstein. Event based, near eye gaze tracking beyond 10,000 hz. *IEEEVR*, 2020. 1, 2, 3, 4

[3] Pietro Bonazzi, Thomas Rüegg, Sizhen Bian, Yawei Li, and Michele Magno. Tinytracker: Ultra-fast and ultra-low-power edge vision in-sensor for gaze estimation. *IEEE Sensors*, 2023. 2

[4] Hannah Bos and Dylan Muir. Sub-mw neuromorphic snn audio processing applications with rockpool and xylo. *Embedded Artificial Intelligence: Devices, Embedded Systems, and Industrial Applications*, page 69, 2023. 2

[5] Hugo Bulzomi, Marcel Schweiker, Amélie Gruel, and Jean Martinet. End-to-end neuromorphic lip-reading. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 4101–4108, 2023. 1

[6] Qinyu Chen, Zuowen Wang, Shih-Chii Liu, and Chang Gao. 3et: Efficient event-based eye tracking using a change-based convlstm network. *IEEE Biomedical Circuits and Systems Conference (BioCAS*, 2023. 1, 2, 3, 8

[7] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J. Davison, Jörg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based vision: A survey. *CoRR*, abs/1904.08405, 2019. 2

[8] Jan Hendrik Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. *CoRR*, abs/1705.02950, 2017. 5

[9] Benedikt Hosp, Shahram Eivazi, Maximilian Maurer, Wolfgang Fuhl, David Geisler, and Enkelejda Kasneci. Remoteeye: An open-source high-speed remote eye tracker: Implementation insights of a pupil-and glint-detection algorithm for high-speed remote eye tracking. *Behavior research methods*, 52:1387–1401, 2020. 2

[10] Petr Kellnhofer, Adria Recasens, Simon Stent, Wojciech Matusik, and Antonio Torralba. Gaze360: Physically unconstrained gaze estimation in the wild. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6912–6921, 2019. 2

[11] Muhammad Qasim Khan and Sukhan Lee. Gaze and eye tracking: Techniques and applications in adas. *Sensors*, 19 (24):5540, 2019. 2

[12] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba. Eye tracking for everyone. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[13] Clara Mestre, Josselin Gautier, and Jaume Pujol. Robust eye tracking based on multiple corneal reflections for clinical applications. *Journal of biomedical optics*, 23(3):035001–035001, 2018. 2

[14] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks. *CoRR*, abs/1901.09948, 2019. 5

[15] Cristina Palmero, Javier Selva, Mohammad Ali Bagheri, and Sergio Escalera. Recurrent cnn for 3d gaze estimation using appearance and shape cues. *arXiv preprint arXiv:1805.03064*, 2018. 2

[16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016. 5

[17] Yannick Schnider, Stanisław Woźniak, Mathias Gehrig, Jules Lecomte, Axel von Arnim, Luca Benini, Davide Scaramuzza, and Angeliki Pantazi. Neuromorphic optical flow and real-time implementation with event cameras. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4129–4138, 2023. 1

[18] Timo Stoffregen, Hossein Daraei, Clare Robinson, and Alexander Fix. Event-based kilohertz eye tracking using coded differential lighting. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2515–2523, 2022. 2, 3

[19] Engin Türetkin, Sareh Saeedi, Siavash Bigdeli, Patrick Stadelmann, Nicolas Cantale, Luis Lutnyk, Martin Raubal, and Andrea L Dunbar. Real time eye gaze tracking for human machine interaction in the cockpit. In *AI and Optical Data Sciences III*, pages 24–33. SPIE, 2022. 2

[20] Kang Wang and Qiang Ji. Real time eye gaze tracking with 3d deformable eye-face model. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1003–1011, 2017. 2

[21] Philipp Weidel and Sadique Sheik. Wavesense: Efficient temporal convolutions with spiking neural networks for keyword spotting, 2021. 5

[22] Ruohan Zhang, Zhuode Liu, Luxin Zhang, Jake A Whritner, Karl S Muller, Mary M Hayhoe, and Dana H Ballard. Agil: Learning attention from human for visuomotor tasks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 663–679, 2018. 3

[23] Xucong Zhang, Yusuke Sugano, and Andreas Bulling. Evaluation of appearance-based methods and implications for gaze-based applications. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pages 1–13, 2019. 1, 2

[24] Guangrong Zhao, Yurun Yang, Jingwei Liu, Ning Chen, Yiran Shen, Hongkai Wen, and Guohao Lan. Ev-eye: Rethinking high-frequency eye tracking through the lenses of event cameras. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. 1, 2, 3, 4