RetinoSim: an Event-based Data Synthesis Tool for Neuromorphic Vision Architecture Exploration

Jonah P. Sengupta
jsengup1@jhu.edu
Johns Hopkins University
Baltimore, Maryland, USA

Susan L. Liu
sliu151@jhu.edu
Johns Hopkins University
Baltimore, Maryland, USA

Andreas G. Andreou
andreou@jhu.edu
Johns Hopkins University
Baltimore, Maryland, USA

ABSTRACT

Neuromorphic vision sensors (NVS), also known as silicon retina, capture aspects of the biological functionality of the mammalian retina by transducing incident photocurrent into an asynchronous stream of spikes that denote positive and negative changes in intensity. Current state-of-the-art devices are effectively leveraged in a variety of settings, but still suffer from distinct disadvantages as they are transitioned into high performance environments, such as space and autonomy. This paper provides an outline and demonstration of a data synthesis tool that gleans characteristics from the retina and allows the user to not only convert traditional video into neuromorphic data, but characterize design tradeoffs and inform future endeavors. Our retinomorphic model, RetinoSim, incorporates aspects of current NVS to allow for accurate data conversion while providing biologically-inspired features to improve upon this baseline. RetinoSim was implemented in MATLAB with a Graphical User Interface frontend to allow for expeditious video conversion and architecture exploration. We demonstrate that the tool can be used for real-time conversion for sparse event streams, exploration of frontend configurations, and duplication of existing event datasets.

CCS CONCEPTS

• **Computing methodologies → Modeling and simulation**.

KEYWORDS

Software Modeling, Event-based Sensors, Data Synthesis, Video Converter, Neuromorphic Vision

ACM Reference Format:
Jonah P. Sengupta, Susan L. Liu, and Andreas G. Andreou. 2022. RetinoSim: an Event-based Data Synthesis Tool for Neuromorphic Vision Architecture Exploration. In *International Conference on Neuromorphic Systems (ICONS 2022), July 27–29, 2022, Knoxville, TN, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3546790.3546805

1 INTRODUCTION

The mammalian retina's ability to compress analog information, process visual signals in parallel, and consume channel bandwidth in an energy efficient manner serves as the model for robust, energy aware sensing. Retinomorphic engineering, coined by Kwabena Boahen in 1996 [5], has sought to duplicate a vast array of the retina's characteristics and processing capabilites in silicon-based sensor arrays. The results of such efforts are known as silicon retina. First iterations of silicon translation included chips that emulated outer plexiform layer transfer characteristics and photoreceptor dynamics [18], [6]. Zaghloul and Boahen realized a silicon retina that replicated signaling seen in the cat's optic nerve. This complete, spike-accurate, inner and outer plexiform interconnections and behaviors were realized on the sensing array [28]. Lichtsteiner and Delbruck emulated behavior seen in photoreceptor, bipolar, and ganglion cells within the retina [16], [7]. This latter, event-based, biologically inspired camera, named the Dynamic Vision Sensor (**DVS**), encodes the temporal contrast of the logarathimically compressed photocurrent into an anisochronous stream of spikes. The DVS and cameras similar in functionality [23] have become appealing to the robotics, machine learning and computer vision community for their low-latency, energy-efficiency, and data compression characteristics.

Due to the relative recency of the event-based sensing paradigm, many researchers do not have access to these devices or have large scale event-based datasets that can be used for downstream algorithms. In addition, many effective deep learning techniques for high-resolution require deep learning techniques that make tradtional frame and corresponding event data essential. Thus, simulators modeling the key characteristics have been devised to aid these efforts. These models have been used to develop graph-based object detectors [20], reconstruct high dynamic range video from event streams [26], and compensate for egomotion on autonomous platforms [27].

1.1 Related Work

Numerous silicon retina software models have been reported and vary in complexity, speed, and capability. A first model of the DVS camera produced pseudo-asynchronous in address event representation (AER) format from video using frame differencing and global thresholds [13]. Improved versions of a DVS model, named *Event Camera Simulator* and *ESIM* by Rebecq et al., were constructed to synthesize spiking data from simulated graphic environments and interpolated video frames [25]. These platforms linearly interpolate between video frames to produces precise spike timings that are proportional to log intensity differences. Despite the usefulness and breadth of use within machine vision applications, the software model does not model non-idealities within the silicon retina nor demonstrates the retina's ability to adapt to low-ligh conditions. In an effort to ameliorate these issues, He and Delbruck developed

ICONS 2022, July 27–29, 2022, Knoxville, TN, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9789-6/22/07…$15.00
https://doi.org/10.1145/3546790.3546805

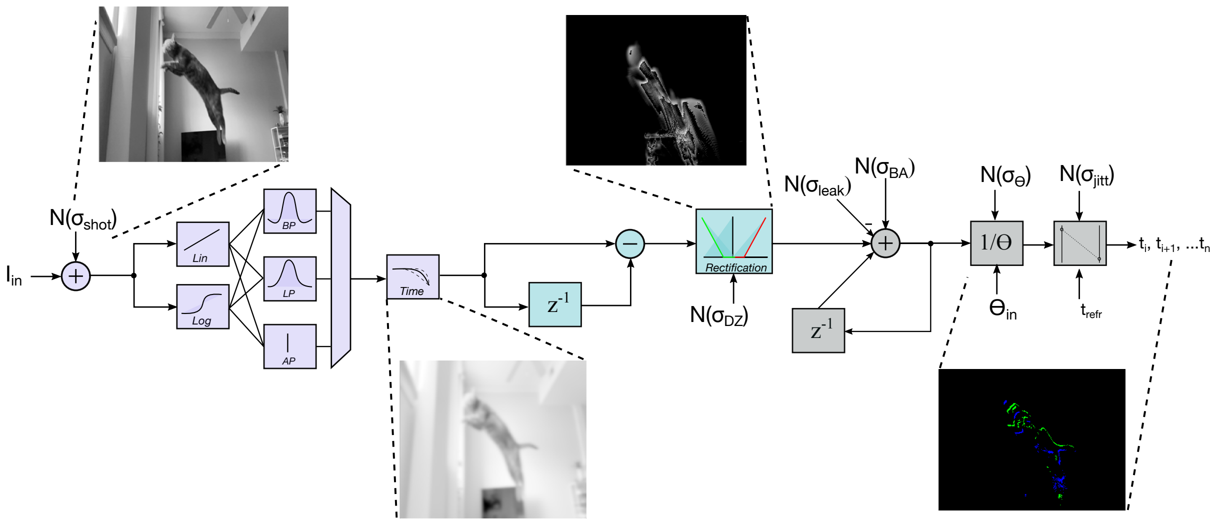


Figure 1: Block Diagram of the RetinoSim Programmable Pixel model with layer output inset- event synthesis is decomposed into three distinct submodules: the Outer Plexiform Layer (OPL-purple), Inner Plexiform Layer (IPL-blue), and Ganglion Cells (GC - gray). Shot noise is injected into the video signal before processed through a programmable compression and spatiotemporal filter step. A discrete-time high pass filter produces an output proportional to frame differences. IPL output signal is recifited and integrated onto a Leaky, Integrate-and-Fire neuron. Parameters were generated from a normal distribution ($N(\cdot)$) to model fixed pattern noise seen in sensor arrays [22]. Address events (AEs) were then from the GC if membrane potential exceeds the local threshold.

v2e [9]. This provides the most accurate model of the DVS as it incorporates phenomena commonly seen in this version of the silicon retina, namely: temporal noise, mismatch, and light-dependent bandwidth. Even though a accurate model of sensor performance is desirable for users without available technology and large scale dataset synthesis, little insight is offered into pathways towards improved silicon retina and possible biological processes that can be further translated to current designs.

1.2 Contributions

Modeling of biological and embedded systems allows for deeper understanding of underlying interactions, accurate representation of target systems, and analysis of future improvements. Current software platforms afford engineers and designers the former two capabilites, but forgo the ability to forecast what improvements can be made to future silicon retina. By revisiting the functionality and structure of the biological retina, the RetinoSim tool has been cultivated. This software tool not only improves upon existing methodologies but also adds features to inform future design of future vision chips. Section 2 provides an overview of the overall model implemented in the platform and details the retinormorphic submodules. Section 3 outlines the backend software details used to expedite data synthesis and integration into a frontend user interface. Various model use cases are demonstrated in Section 4 before concluding in Section 5.

2 PROGRAMMABLE RETINOMORPHIC MODEL

Core functionality of the RetinoSim tool is derived from the programmable pixel model. A block diagram of this pipeline is seen in Figure 1. Input video frames are first processed through the Outer Plexiform Layer (**OPL**) which injects shot noise and programmatically compresses intensity data before applying a spatiotemporal filter. OPL output is then processed in the Inner Plexiform Layer (**IPL**) using a temporal highpass filter and rectified into two channels. Each channel is subsequently integrated in within the Ganglion Cell (**GC**) layer using a Leaky-Integrate and Fire (**LIF**) which outputs address event spike trains. Formally, the model, $\mathbb{M}$ performs the transformation:

$$\mathbb{M}(v) : V \longrightarrow E \mid V \in \mathbb{R}^{m \times n \times 3 \times f},\ E \in \mathbb{R}^{d \times 4} \tag{1}$$

where m and n are the dimensions of a video frame, f is the number of frames, and d is the number of events returned by the model over the stream. Further, E can be decomposed into individual AEs which have form

$$E = \{e_1 \ldots e_d\} \mid e_i = [\mathbf{x_i}, p_i, ts_i] \tag{2}$$

As seen above, the AEs are described the spatial ($\mathbf{x_i} = x_i, y_i$) and temporal (t_i) extent along with the polarity (p_i). Each submodule within the programmable pixel model will be discussed in subsequent subsections.

2.1 Outer Plexiform Layer

In the mammalian retina and within the RetinoSim pipeline, the Outer Plexiform Layer consists of elements that compress light intensity and perform spatiotemporal filtering. If the input video consist of three channels, input video frames are first compressed into a single grayscale luminance measures based on the bio-inspired BT.601 standard. Formally, output frames, I_{in}, can be expressed as

$$I_{in} = 0.299 I_{f,r} + 0.587 I_{f,g} + 0.114 I_{f,g} \tag{3}$$

Photoreceptor dynamics and solid state devices are influenced by shot noise, the temporally stochastic discrete movement of particles, such as photons or electrons [1]. This mechanism is particularly influential on image sensor performance in low-light situations [9]. Coupling this latter relationship and the root mean square of shot noise current fluctuations:

$$\lambda \quad = \quad \frac{\sqrt{2qN_{trans}I_{mag}(1/t_s)}}{I_{mag}}(\max(\mathbf{I_{in}}) - I_{in}) \tag{4}$$

where λ represents the normalized, Poisson rate, N_{trans} is the number of devices in the photoreceptor frontend (set to 2 using [16] as reference), $1/t_s$ is the inverse of the video frame rate while serving as the bandwidth to extract the root mean square, and I_{mag} is a physical quantity to map into the modeling domain. For sufficiently large values of λ, the Poisson distribution can be approximated using a Gaussian distribution with $\sigma_{shot} = \sqrt{\lambda}$.

Programmable compression and spatial filtering is used within the model to allow users to explore the effect of different front-end methodologies on event synthesis output and algorithm performance. Logarithmic compression can be selected to model transconductance transfer function implemented in the subthreshold regime[16, 23]. Alternatively, a linear encoding forgoes the extended dynamic range afforded by the log compression but allows for increased sensitivity. Such a scheme is adopted by silicon retina that integrate photocurrent [17] or operate in the current-mode regime [24]. Next, the user has the ability to choose between three variants of spatial filters: low-pass, normalized band-pass, or all-pass.

Spatial low-pass filtering models electrical coupling of photoreceptor or horizontal cells in the retina [4, 12]. These are implemented in silicon using diffusive networks composed of resistive elements that connect adjacent pixels in rectangular and hexagonal morphologies [5, 6, 18]. Within the model, this low-pass filtering is approximated using a Gaussian filter

$$G(x) = \sqrt{\frac{1}{2\pi\sigma_1^2}}e^{\frac{-x^2}{2\sigma_1^2}} \tag{5}$$

where σ_1 is standard deviation of the filter and synonymous to the space constant, $\gamma_{c,h}$, realized by the photoreceptor or horizontal cell networks. However, the outer plexiform layer in the retina does not realize a simple low-pass filter, but instead outputs a contrast enhanced signal via normalized band-pass filter. Horizontal cells have larger gap junctions and a long space constant γ_h. Thus, these cells pool average response and construct an antagonistic center-surround receptive field that, through lateral inhibition, subtracts this average from rod or cone potential [4]. When implemented in silicon, the impulse response of such a feedback system takes the shape of the Ricker wavelet, also known as the Mexican Hat kernel [6]. Such a kernel is commonly used in computer vision to extract edges and can be constructed by using a Difference of Gaussians (**DoG**) [10].

Within RetinoSim, the kernel is composed of two Gaussian filters which standard deviations σ_1 and σ_2 which reflect the cone γ_c and horizontal cell γ_h space constants respectively. As in the retina and in silicon implementations of the outer plexiform layer [6, 28], filter output is normalized with respect to average photocurrent which allows the retina to respond uniformly across many decades of input. The normalized band-pass response at time t and location $\mathbf{x_i} = (x_i, y_i)$ is expressed as:

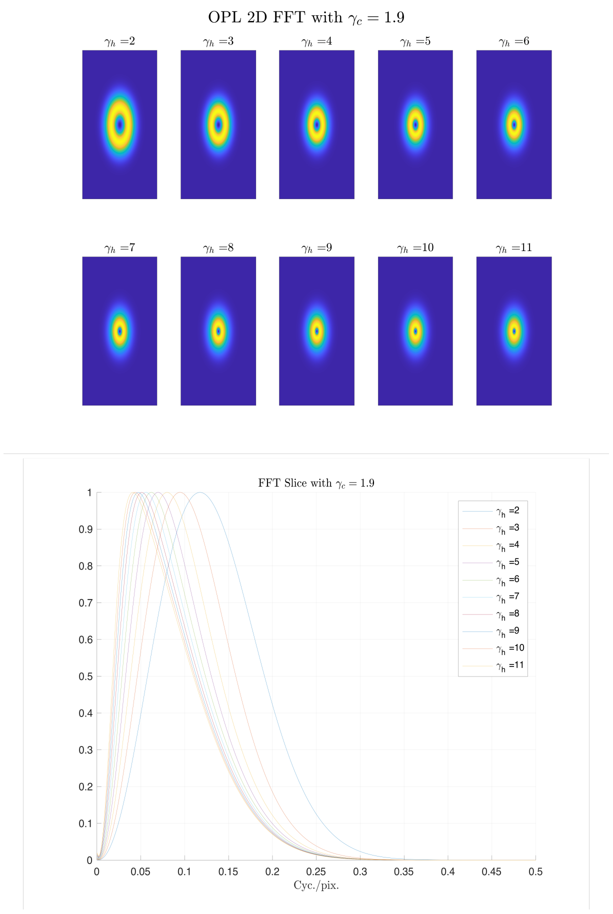


Figure 2: Outer Plexform Layer - programmable spatial response with respect to layer space constants. The bottom chart depicts slices of the 2D Filter FFT seen in the top. First column corresponds to sweeping γ_h from 2 to 10 while fixing γ_c = 1.9.

$$I^t_{osr}(\mathbf{x_i}) \quad = \quad \frac{127(K_{DoG}(\mathbf{x_i}) * I_{in}(\mathbf{x_i}))}{I_{av}} \tag{6}$$

$$K_{DoG}(\mathbf{x_i}) \quad = \quad \sqrt{\frac{1}{2\pi\gamma_c^2}}e^{\frac{-\mathbf{x_i}^2}{2\gamma_c^2}} - \sqrt{\frac{1}{2\pi\gamma_h^2}}e^{\frac{-\mathbf{x_i}^2}{2\gamma_h^2}} \tag{7}$$

where $K_{DoG}(\mathbf{x_i})$ realizes the band-pass filter and 127 maps the average value to midrange intensity with 8-bit encoding. Figure 2 exhibits how the filter spatial response can be programmed using RetinoSim. One methodology is to is to maintain $\gamma_c = 1.9$ while scaling γ_h. Peak spatial frequency shifts allows RetinoSim end users to analyze the frequency content of their scene and program an optimized filter profile, with respect to bandwidth, cutoff, or rolloff. An allpass filter, without phase distortion, can be used if no spatial processing is desired.

After the compression and spatial filtering steps, a discrete-time, infinite impulse response temporal filter is implemented and used to model OPL temporal dynamics. Filter time constant is configured to be linearly proportional to incident intensity I_{in} which models the light-dependent input impendance and bandwidth seen in many silicon retina [6, 16, 23]. Pixel-specific, filter delay is computed as the normalized deviation from mean intensity and output $I_{opl,str}$ is found by

$$\tau_{ly}(\mathbf{x_i}) = \alpha\frac{I^t_{in}(\mathbf{x_i}) - \mu_{I_{in}}}{\max(I^t_{in})} \tag{8}$$

$$\tau_{opl}(\mathbf{x_i}) = \min(\tau_{opl,in} + \tau_{ly}(\mathbf{x_i}), 1) - \beta \tag{9}$$

$$I^t_{ost}(\mathbf{x_i}) = (1 - \tau_{opl}(\mathbf{x_i}))I^{t-1}_{ost}(\mathbf{x_i}) + \tau_{opl}(\mathbf{x})I^t_{osr}(\mathbf{x_i}) \tag{10}$$

where α and β are programmed constants chosen to be 0.0625 and 0.025 respectively, $\tau_{opl,in}$ is the user specified target time constant, and $I_{ost}(\mathbf{x_i}))^{t-1}$ is the spatiotemporal OPL pixel response from the prior time sample.

2.2 Inner Plexiform Layer

Event-based vision sensing is centered around the asynchronous communication of address events that are generated upon a requisite change in input intensity. Otherwise known as temporal contrast encoding, the DVS and other similar pixels implement a voltage mode, high-pass filter that attenuates low temporal frequencies. Similar behavior is exhbited by parasol ganglion cells in the magnocellular visual pathway [15]. Current-mode pixel designs have been implemented that adopt the contrast and temporal frequency adaptation in the inner plexiform layer of the retina and ultimately produce a similar high-pass response [28, 29]. Such contrast and temporal frequency adaptation seen in the IPL is not implemented in this model. Instead, the high pass response was modeled by computing a temporal difference between successive I_{ost} samples.

Bipolar cell response was modeled by recitfying the high-pass response into two channels: ON and OFF. Such rectification is accomplished using translinear circuits with a configurable deadzone where either channel does not output current [2]. This deadzone value, $I_{dz}(\mathbf{x_i})$, is sampled from a normal distribution with standard deviation σ_{dz} and mean given from the user $I_{dz,in}$ in order to model the transistor mismatch seen in sensor arrays [22]. Formally, bipolar rectification is expressed as

$$I^t_{on}(\mathbf{x_i}) = \begin{cases} |I^t_{ost}(\mathbf{x_i}) - I^{t-1}_{ost}(\mathbf{x_i})| & \text{if } I^t_{ost}(\mathbf{x_i}) - I^{t-1}_{ost}(\mathbf{x_i}) \geq I_{dz}(\mathbf{x_i}) \\ 0 & otw. \end{cases} \tag{11}$$

$$I^t_{off}(\mathbf{x_i}) = \begin{cases} |I^t_{ost}(\mathbf{x_i}) - I^{t-1}_{ost}(\mathbf{x_i})| & \text{if } I^t_{ost} - I^{t-1}_{ost} \leq -I_{dz}(\mathbf{x_i}) \\ 0 & otw. \end{cases} \tag{12}$$

When $I_{dz,in}$ is configured to be 0, the rectification simply checks the sign of the high-pass response and models the behavior seen in temporal contrast-based silicon retina designs [16, 23].

2.3 Ganglion Cells

Retinal ganglion cells are responsible for transducing incident graded potentials from the inner plexiform layer into time-encoded spike trains. Many versions of ganglion cells exist in the retina, but only the parasol variety, which are sensitive to high temporal frequencies, are realized in this model. A modified leaky-integrate and fire neuron (**mLIF**) was used to model the output comparators behavior seen in state-of-the-art NVS. The dynamics of the mLIF neuron is described by

$$V'_{mem}(\mathbf{x_i}) = \eta(I_{on|off}(\mathbf{x_i}) + I_{ba}(\mathbf{x_i}) - I_{lk}(\mathbf{x_i})) \tag{13}$$

such that I_{ba} is the background activity current that triggers false events [9, 14], I_{lk} represents the leakage current that supresses spike activity, and $i_{on|off}$ is the ON or OFF current that is integrated onto the membrane. Constant η is equal to $\frac{1}{g_{lk}\tau_m}$ where g_{lk} is the leakage conductance and τ_m is the associated time constant of the neuron. Both $I_{lk}(\mathbf{x})$ and $I_{ba}(\mathbf{x})$ are sampled from normal distributions in similar manner to $I_{dz}(\mathbf{x})$. Membrane potential is accumulated using a first-order Euler approximation of the dynamical solution. Distributions for the leakage terms are seen in Figure 3a.

Action potentials are generated from the ON or OFF neuron when V_mem exceeds the threshold. These thresholds, $\theta_{on,off}(\mathbf{x})$, are also drawn from a normal distribution with mean $\mu = \theta_{on|off,in}$ and standard deviation σ_{theta}. Example distributions for the ON and OFF threshold is seen in Figure 3b. The next state of the neuron potential is computed as

$$V^t_{mem}(\mathbf{x_i}) = \begin{cases} V^{t-1}_{mem} - \delta_t\eta(I_{on|off} + I_{ba}(\mathbf{x}) - I_{lk}(\mathbf{x})) & V^{t-1}_{mem} < \theta_{on|off} \\ V_{rst} & V^{t-1}_{mem} \geq \theta_{on|off} \end{cases} \tag{14}$$

where V_{rst} is the potential when the potential exceeds the neuron threshold, $\theta_{on|off}$. Upon exceeding the threshold, spike generation is formulated as

$$N_e = \lfloor \frac{V^t_{mem}(\mathbf{x_i})}{\theta_{on|off}(\mathbf{x_i})} \rfloor \tag{15}$$

N_e denotes the number events generated by the model unit cell. V^t_{mem} is divided by the $\theta_{on|off}$ to interpolate the number of events generate in between input video samples. Thus, N_e address events are output when this result is truncated to the nearest integer and greater than one. These spike timings are initially generated by uniformly distributing values between sampling intervals. Formally, the set $\mathbf{T_i}$ contains all spike timings from pixel located at $\mathbf{x_i}$ and global time t_G with sampling interval t_s,

$$\mathbf{T_i} = te_i \ \forall \ i \in \{1, N_e\} \tag{16}$$

$$= \begin{cases} t_G + t_s/2 & N_e = 1 \\ \{t_g, t_G + \frac{t_s}{N_e} ..., t_G + t_s - \frac{t_s}{N_e}\} & N_e > 1 \end{cases} \tag{17}$$

Each $te_i \in \mathbf{T_i}$ was injected with additive Gaussian noise with $\sigma_{ts} = \frac{t_s}{100}$ to emulate neuronal spike jitter and the non-deterministic nature of NVS asynchronous readouts. A programmable refractory period was inserted to model the requisite time before another spike can be output from the ganglion cell. An intermediate memory is used to store the timings of recent cell activations, te_p, and subsequently used to output address events with timestamps te'_i,

$$te'_i = te_i \leftarrow (te_i - te_p) > \theta_{ref} \tag{18}$$

$$t'_{s,prev} = te_i \leftarrow (te_i - te_p) > \theta_{ref} \tag{19}$$

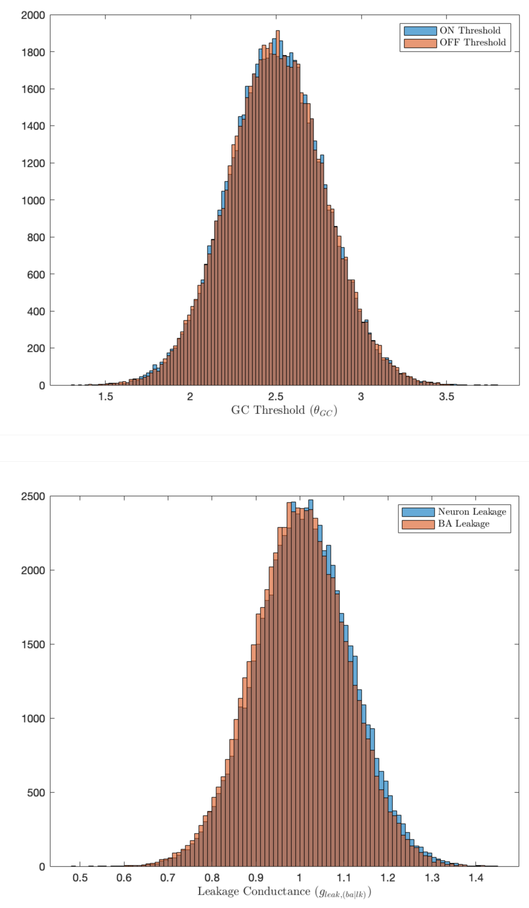
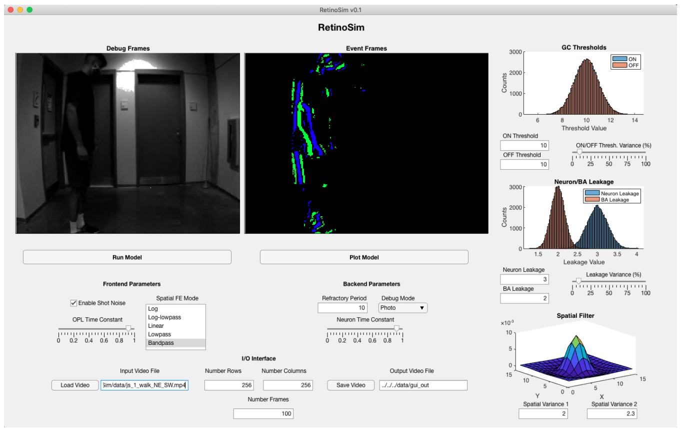


Figure 3: (Top) BA and Neuron Leakage Distributions with $\mu = 1.0$ and $\sigma = 0.1$ (Bottom) Ganglion Cell Threshold Distributions with $\mu = 2.5$ and $\sigma = 0.25$



Frontend Parameters

[✓] Enable Shot Noise

OPL Time Constant

Spatial FE Mode: Log, Log-lowpass, Linear, Lowpass, [✓ Bandpass]

Backend Parameters

Refractory Period: 10

Debug Mode: Photo

Neuron Time Constant

I/O Interface

Input Video File: Sim/data/js_1_walk_NE_SW.mp4

Number Rows: 256

Number Columns: 256

Output Video File: ../../../data/gui_out

Number Frames: 100

ON Threshold: 10

OFF Threshold: 10

ON/OFF Thresh. Variance (%)

Neuron Leakage: 3

BA Leakage: 2

Leakage Variance (%)

Spatial Variance 1: 2

Spatial Variance 2: 2.3

Figure 4: Snapshot of the RetinoSim Graphical User Interface with populated parameters.

Resulting spikes from the ganglion cell layer are collated into an address event stream and form the basis of the synthesized output data.

3 SOFTWARE OVERVIEW

RetinoSim is the realization of the data synthesis pipeline using the model components outlined in Section 2. The objective was to create a platform to expeditiously convert video frames into event-based data while allowing the user to quickly iterate different model parameters. Backend utilization of data arrays in the pipeline accomplishes the former task while a graphical user interface addresses the latter.

3.1 Backend Implementation Details

Data vectorization was deployed to fully utilize the computational abilities of MATLAB. Computations associated with the Outer and Inner Plexiform Layers were intuitive to perform on an array level as they consist of convolutions, element-wise subtractions, and other linear operations. However, bipolar rectification and subsequent membrane potential integration had to be translated from a two-dimensional iterative approach into a vectorized operation that utilized matrix logical indexing. A similar technique was further exploited to find neurons in the array whose $V_{mem}^{t} > \theta_{on,off}$. The following code snippet illustrates this point.

```
[neuronObj.spikeLocs(:,1), neuronObj.spikeLocs(:,2)] =
find(floor(neuronObj.state./params.gc_threshold) > 0);
```

This latter operation provides a list of neuron locations that exceed the Ganglion Cell threshold. This list is ultimately iterated element-wise to produce event stream. However, these vectorized pre-processing steps forgo unnecessary computational steps and allow for processing times upwards of 30-40 fps under sparse event conditions. Furthermore, since the spike generation algorithm only queries spiking neurons, it also provides a means to emulate the event-based sensing realized by silicon retina.

The model is sped-up further by initializing arrays prior to iterative loops to allow for contiguous memory access and reducing unnecessary variable assignments. Additional acceleration could be achieved by leveraging MATLAB's Parallel Computing Toolbox to utilize **parfor** loops and **gpuArray** deployment for spatiotemporal filtering steps.

3.2 User Interface Integration

RetinoSim was integrated into a graphical frontend that allows the user to easily configure the model, interface with input/output data, and visualize results and parameters. Figure 4 depicts the RetinoSim GUI with an example video snapshot and model parameters inset. Upon starting the program, the user would import a video and customize size and number of frames. Videos can be imported by either pressing the **Load Video** button which launches a file navigator window to find the video or by entering the path manually in the **Input Video File** text edit field. Upon completing the import, the video is displayed in the **Debug Frames** subwindow.

Before running RetinoSim to convert the video to events, the user has the option to customize the model frontend and backend parameters. Frontend parameters refer to those the configure the behavior of the Outer Plexiform Layer. The user can choose to enable shot noise, specify the average OPL time constant $\tau_{opl,in}$, choose the spatial filter, and configure Gaussian filter variances σ_1 and σ_2. The GUI also visualizes the spatial filter impulse response to allow the user to configure filter characteristics more effectively.

Backend parameters refer to those that configure the Ganglion Cell layer. These paramaters include the membrane time constant (τ_m) and refractory period (θ_{ref}). In addition, the user can specify the mean and variance of the distributions from which the GC ON/OFF neuron thresholds and background activity/neuron leakage currents are sampled. Variances for these distribtuions are configured by setting $\sigma = v_p\mu$ where v_p is a percentage specified by the user. NVS contrast threshold mismatch is typically presented in terms of percent variance thus this GUI feature allows the user to mirror parameters seen in literature [7, 16].

Upon fully configuring the model, the user simply needs to press the **Run Model** button to start the event stream synthesis. When the button reverts back to it's original shade, the model has completed processing and the results can be plotted. A dropdown menu, labelled **Debug Mode**, was included that allows the user to observe the different layer responses within the model. Options include the input with shot noise injection, OPL spatial or spatiotemporal response, and the ON or OFF membrane potential. Outputs can be stored locally using the **Save Video** button where the user navigates to a desired folder and specifies a file prefix. An video containing events and blended events/frames is saved in addition to a *.mat* file containing synthesized address events.

4 MODEL DEMONSTRATION

RetinoSim v0.1.1 has been released as an open-source platform that can be downloaded from Github (https://github.com/js3ng/RetinoSim). Software has been verified in MATLAB 2022.a on a laptop running MacOS Mojave (10.14.6) with a 2.8GHz Intel Core i7 CPU and 16 GB of RAM. Two input videos have been included in the repository to enable the user to demonstrate model capabilites. These include a video with an individual walking across the scene (*js_1_walk_NE_SW.mp4*) and another sample of a camera panning across a room (*room_pan.mp4*). Debugging scripts are also included to facilitate model usage outside of the GUI.

Since quick run-time was an objective of the model, RetinoSim was profiled with a variety of configurations, shown in Figure 5. Because of the nature of spike queueing outlined in Section 3.1, more time is consumed as more neurons are activated. By increasing neuron thresholds or leakages, the number of events generated per frame decreases and reduces run-time. For sparse (<1000) event generation per frame, the model is able to process upwards of 33 frames-per-second. With large event generation rates (<5000), throughput is throttled and decreased to <5 frames-per-second. Under certain model parameterizations, this suggests that the model can function in real-time event generation applications.

4.1 OPL Frontend Exploration

Different OPL configurations were used to demonstrate the model's ability as a NVS architecture exploration platform. An example of such a comparison is seen in Figure 6. Using a bandpass spatial filter enhances the contrast of image so subsequent temporal processing produces events that encode negative and positive spatiotemporal

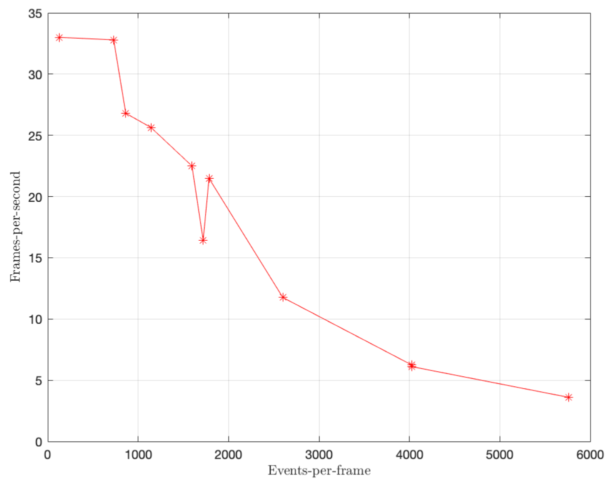


Figure 5: Model Performance Profile- relationship between RetinoSim processing frames per second and. synthesized events per frame running on a laptop with 2.8GHz Intel Core i7 CPU

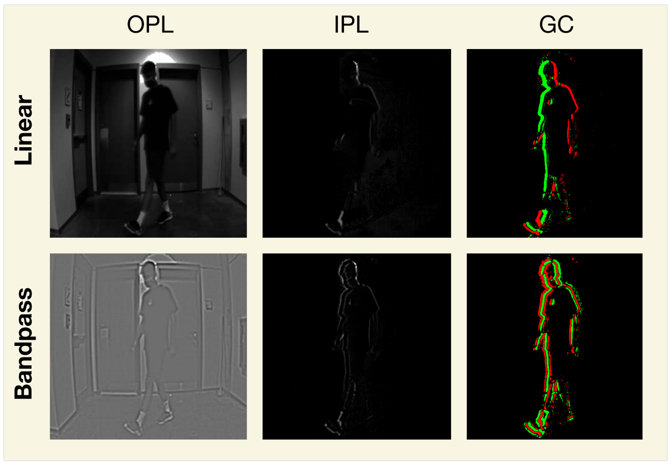
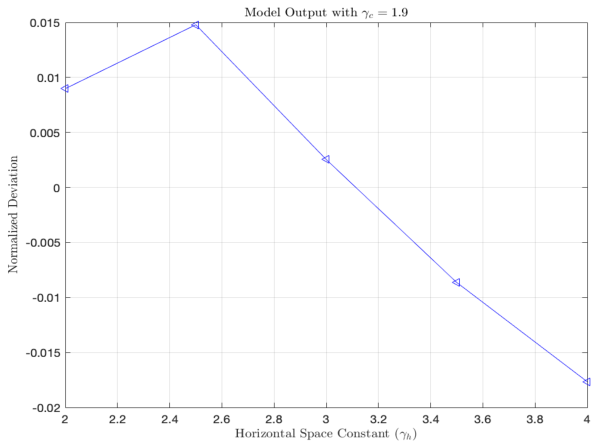


Figure 6: Simulation results from *js_1_walk_NE_SW.mp4* - the first row is an example of layer output using linear encoding with the second leveraging a bandpass filter configuration. No temporal noise (BA leakage or shot) was added, but a 10% FPN for GC thresholds was configured. Other model parameters include $\theta_{on|off} = 10$, $I_{leak} = 1$, $\tau_{opl,in} = 0.9$, $\tau_m = 0.9$,$\theta_{ref} = 100$.

contrast. In addition, the filter reduces effects of high-frequency fixed pattern noise while eliminating low frequency spatial detail. The effect of OPL space constants γ_c and γ_h were then explored using two methods. First, γ_h was increased while setting $\gamma_c = 1.9$ and second, the γ_c was swept while maintaining the difference $\gamma_c - \gamma_h = 0.1$. A normalized deviation ($\frac{N_{ev}-\mu_{trial}}{\mu_{trial}}$) of the number events from each trial was gathered and used to track filter performance, shown in Figure 7. As seen in Figure 2, peak spatial frequency and



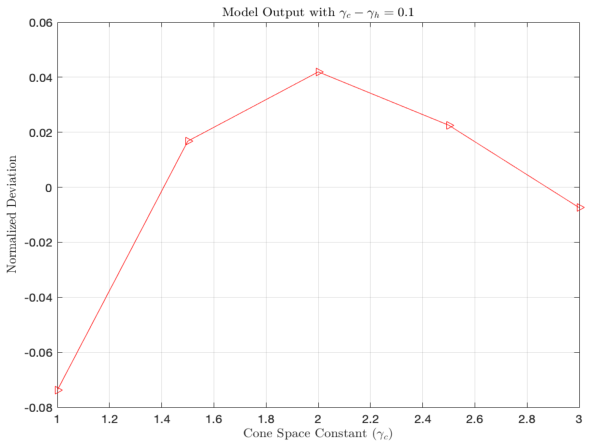


Figure 7: Normalized deviation of synthesized events from *js_1_walk_NE_SW.mp4* input while sweeping γ_c (top) and γ_h (bottom). Other model parameters include $\theta_{on|off} = 10$, $I_{leak} = 1$, $\tau_{opl,in}$ = 0.9, $\tau_m = 0.9$, $\theta_{ref} = 100$.

bandwidth of the spatial filter shifts as the space constants are modulated. Therefore, the number of events is reduced when the image spatial frequencies reside outside of the filter pass band. This allows the user to programmatically resolve what OPL parameters optimally filter their desired input.

Beyond parameter search capabilities for a specific frontend, RetinoSim enables the user to experiment with different OPL configurations to potentially resolve which are most capable for their desired application. Figure 8 shows the result of a comparitive experiment using the five OPL configurations while tracking model noise immunity, ni_m. This latter metric conveys the model's ability to reject additive noise:

$$ni_m = 100\frac{N_{ev,i} - N_{ev,n}}{N_{ev,i}} \tag{20}$$

where $N_{ev,i}$ and $N_{ev,n}$ are the amount of events generated in the cases with and without additive noise. With the given model parameters, additive noise, and video streams, Figure 8 suggests that using bandpass filter would reduce the most amount of noise events.

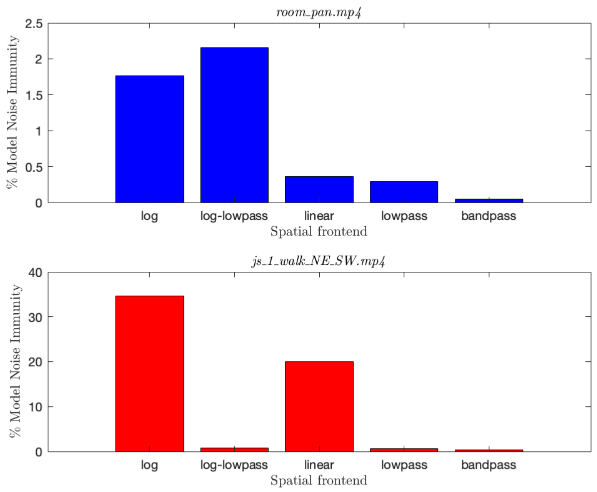


Figure 8: Observed Model Noise Immunity with model spatial frontends. Model parameters include: $\sigma_1 = \gamma_c = 2$, $\sigma_2 = \gamma_h = 2.3$, $\theta_{on|off} = 10$, $I_{leak} = 1$, $\tau_{opl,in}$ = 0.9, $\tau_m = 0.9$, $\theta_{ref} = 100$. Parameters used to inject noise are: $I_{BA} = 2$, enabled shot noise, $v_{leak} = v_\theta = 0.1$.

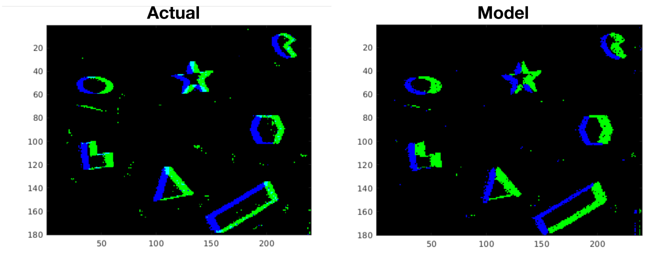


Figure 9: Sampled event streams from the *shape_6dof* sample and from RetinoSim using the accompanying frames from the sample. Model thresholds ($\theta_{on|off}$) leakages $I_{lk|ba}$, and time constants ($\tau_{m|opl,in}$) were optimized to match the total amount of events between the streams.

4.2 Matching Event Camera Output

RetinoSim also enables the user the ability to match the event output from an existing dataset that contains events streams and frame-based video. The Event Camera dataset [21], compiled by the Robotics and Perception Group at ETH-Zurich, provides such data as it leverages the DAVIS240 [7] camera streams events and frames simultaneously. Of the various example scenes provided, the *shape_6dof* sample was selected to demonstrate the platform's ability to match an existing dataset. Original events and those generated by RetinoSim are seen side-by-side in Figure 9. Frame-based data was processed using RetinoSim with a time step, t_s, that matches the frame rate of the DAVIS240. Model parameters (namely

$\theta_{on|off}$, $I_{lk|ba}$, and $\tau_{m|opl,in}$), were then manually iterated in order to match the total event output of the source event stream. To note, the matching of model output to the source can be accomplished using other metrics, such as event-rate or inter-spike-interval, and in an automatic fashion using learning rules and gradient-descent methods [8, 9].

After converging on model parameters, the user can deploy RetinoSim to see how events would be synthesized from new video using the dataset configuration. For example, if video frames are captured on a UAV platform, similar to that seen in the *shape_6dof* sample, but the user does not have an NVS, data can be synthesized using the dataset configuration and used to see if an event camera would be advantegeous for their application.

4.3 Future Steps

As the platform is open-source and in development, RetinoSim will continue to mature and adopt new features to optimize current capabilities and reflect emerging new directions that improve upon current NVS architectures. For instance, the Ganglion Cell layer can be generating spikes using a Poisson distribtution by setting $N_{ev} = \lambda$. This effectively models the stochastic nature of AP generation in neurons [11] while also further capturing non-uniformity effects observed in sensor arrays [16]. In addition, neuronal behavior can be enhanced using a feedback configuration which realizes spike-rate adaptation: the ability to shunt excitatory current and inhibit output spike rates using slowly accumulated potassium currents. Such behavior has been realized in other silicon retina [5, 28] and is another route to realize an adaptive high-pass temporal filter.

Amacrine cells are a key component of the mammalian retina that have been unrealized in NVS architectures. Similar to horizontal cells in the outer plexiform layer of the retina, amacrine cells laterally spread signals from their bipolar cells in the inner plexiform layer onto ganglion neurons to allow for frequency and contrast adaptation [29]. Specifically, wide field amacrine cells play a key role in suppressing global, coherent motion from stimulating ganglion cells during microsaccades [3]. Hardware modeling of this object motion sensitive response has been modeled in hardware and promises to be a solution to suppress event stimulation during egomotion [19]. Integration of such behavior into RetinoSim will provide a pathway to understanding OMS impact on NVS performance.

5 CONCLUSION

RetinoSim, an event-based data synthesis tool for NVS architecture exploration, was detailed and demonstrated. It is a MATLAB software platform that is composed of processing states that model the Outer Plexiform, Inner Plexiform, and Ganglion Cell layers in the retina. The OPL injects temporal shot noise, compresses image data, and applies a programmable spatiotemporal filter. Inner Plexiform layer consists of a temporal high-pass filter with rectification into the ON and OFF channels. Spikes are ultimately generated by a modified, leaky integrate-and-fire neuron model within the Ganglion cell layer. Vectorized data programmaing allows for expedient processing of video data while a GUI enables the user to visualize results and rapidly configure the model. Real-time synthesis of event data from video frames can be acheived given low (<1000 events/frame) conversion rates. In addition, RetinoSim demonstrated the ability to optimize spatial filter parameters with respect to noise immunity and output event rate. It also provides the means to match the output from existing neuromorphic datasets. Given the model's current capabilities and future features detailed in 4.3, RetinoSim hopes to not only provide a means to generate event-based data in a robust manner, but also inform present design efforts and allow the neuromorphic community to explore NVS architectures that extend our current capabilities.

ACKNOWLEDGMENTS

This work was supported by the Science, Mathematics, and Research for Transformation (SMART) fellowship, Defense Advanced Research Projects Agency (DARPA) Microsystems Technology Office (MTO) ReImagine project under Contract HR0011-17-C0071 and in part by the Northrop Grumman Faculty Award to support graduate student research (GT).

REFERENCES

[1] Pamela Abshire and Andreas G Andreou. 2001. Capacity and energy cost of information in biological and silicon photoreceptors. *Proc. IEEE* 89, 7 (2001), 1052–1064.

[2] Andreas G Andreou and Kwabena A Boahen. 1996. Translinear circuits in subthreshold MOS. *Analog Integrated Circuits and Signal Processing* 9, 2 (1996), 141–166.

[3] Stephen A Baccus, Bence P Ölveczky, Mihai Manu, and Markus Meister. 2008. A retinal circuit that computes object motion. *Journal of Neuroscience* 28, 27 (2008), 6807–6817.

[4] Horace Basil Barlow, Horace Basil Barlow, and JD Mollon. 1982. *Senses*. Vol. 3. CUP Archive.

[5] Kwabena Boahen. 1996. Retinomorphic vision systems. In *Proceedings of Fifth International Conference on Microelectronics for Neural Networks*. IEEE, 2–14.

[6] Kwabena A Boahen and Andreas G Andreou. 1992. A contrast sensitive silicon retina with reciprocal synapses. In *Advances in neural information processing systems*. 764–772.

[7] C. Brandli, R. Berner, M. Yang, S. Liu, and T. Delbruck. 2014. A 240 × 180 130 dB 3 µs Latency Global Shutter Spatiotemporal Vision Sensor. *IEEE Journal of Solid-State Circuits* 49, 10 (2014), 2333–2341. https://doi.org/10.1109/JSSC.2014.2342715

[8] Tobi Delbruck, Rui Graca, and Marcin Paluch. 2021. Feedback control of event cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1324–1332.

[9] Tobi Delbruck, Yuhuang Hu, and Zhe He. 2020. V2E: From video frames to realistic DVS event camera streams. *arXiv preprint arXiv:2006.07722* (2020).

[10] David Forsyth and Jean Ponce. 2011. *Computer vision: A modern approach*. Prentice hall.

[11] David Heeger et al. 2000. Poisson model of spike generation. *Handout, University of Standford* 5, 1-13 (2000), 76.

[12] David H Hubel. 1995. *Eye, brain, and vision*. Scientific American Library/Scientific American Books.

[13] Matthew L Katz, Konstantin Nikolic, and T Delbruck. 2012. Live demonstration: Behavioural emulation of event-based vision sensors. In *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 736–740.

[14] Alireza Khodamoradi and Ryan Kastner. 2018. $O(N)$ O (N)-Space Spatiotemporal Filter for Reducing Noise in Neuromorphic Vision Sensors. *IEEE Transactions on Emerging Topics in Computing* 9, 1 (2018), 15–23.

[15] Helga Kolb, Eduardo Fernandez, and Ralph Nelson. 1995. Webvision: the organization of the retina and visual system [Internet]. (1995).

[16] P. Lichtsteiner, C. Posch, and T. Delbruck. 2008. A 128× 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE Journal of Solid-State Circuits* 43, 2 (2008), 566–576. https://doi.org/10.1109/JSSC.2007.914337

[17] Udayan Mallik, M Clapp, Edward Choi, G Cauwenberghs, and R Etienne-Cummings. 2005. Temporal change threshold detection imager. In *ISSCC. 2005 IEEE International Digest of Technical Papers. Solid-State Circuits Conference, 2005.* IEEE, 362–603.

[18] Carver A Mead and Misha A Mahowald. 1988. A silicon model of early visual processing. *Neural networks* 1, 1 (1988), 91–97.

[19] Diederik Paul Moeys, Tobias Delbrück, Antonio Rios-Navarro, and Alejandro Linares-Barranco. 2016. Retinal ganglion cell software and FPGA model implementation for object detection and tracking. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1434–1437.

[20] Anindya Mondal, Jhony H Giraldo, Thierry Bouwmans, Ananda S Chowdhury, et al. 2021. Moving Object Detection for Event-based Vision using Graph Spectral Clustering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 876–884.

[21] Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza. 2017. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *The International Journal of Robotics Research* 36, 2 (2017), 142–149.

[22] Aleksandra Pavasović, Andreas G Andreou, and Charles R Westgate. 1994. Characterization of subthreshold MOS mismatch in transistors for VLSI systems. *Journal of VLSI signal processing systems for signal, image and video technology* 8, 1 (1994), 75–85.

[23] Christoph Posch, Daniel Matolin, and Rainer Wohlgenannt. 2010. A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS. *IEEE Journal of Solid-State Circuits* 46, 1 (2010), 259–275.

[24] Philippe O Pouliquen, Andreas G Andreou, C Cauwenber, and CW Terrill. 2000. A CMOS smart focal plane for infra-red imagers. In *2000 IEEE International Symposium on Circuits and Systems (ISCAS)*, Vol. 4. IEEE, 329–332.

[25] Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza. 2018. ESIM: an open event camera simulator. In *Conference on Robot Learning*. 969–982.

[26] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. 2019. High speed and high dynamic range video with an event camera. *IEEE transactions on pattern analysis and machine intelligence* 43, 6 (2019), 1964–1980.

[27] Timo Stoffregen, Guillermo Gallego, Tom Drummond, Lindsay Kleeman, and Davide Scaramuzza. 2019. Event-based motion segmentation by motion compensation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7244–7253.

[28] Kareem A Zaghloul and Kwabena Boahen. 2006. A silicon retina that reproduces signals in the optic nerve. *Journal of neural engineering* 3, 4 (2006), 257.

[29] Kareem A Zaghloul and Kwabena A Boahen. 2005. An on-off log domain circuit that recreates adaptive filtering in the retina. *IEEE Transactions on Circuits and Systems I: Regular Papers* 52, 1 (2005), 99–107.