Egomotion from event-based SNN optical flow

Yi Tian
Juan Andrade-Cetto
ytian@iri.upc.edu
cetto@iri.upc.edu
Institut de Robòtica i Informàtica Industrial, CSIC-UPC
Barcelona, Spain

ABSTRACT

We present a method for computing egomotion using event cameras with a pre-trained optical flow spiking neural network (SNN). To address the aperture problem encountered in the sparse and noisy normal flow of the initial SNN layers, our method includes a sliding-window bin-based pooling layer that computes a fused full flow estimate. To add robustness to noisy flow estimates, instead of computing the egomotion from vector averages, our method optimizes the intersection of constraints. The method also includes a RANSAC step to robustly deal with outlier flow estimates in the pooling layer. We validate our approach on both simulated and real scenes and compare our results favorably to the state-of-the-art methods. However, our method may be sensitive to datasets and motion speeds different from those used for training, limiting its generalizability.

CCS CONCEPTS

• **Computing methodologies → Vision for robotics**.

KEYWORDS

spiking neural network, event camera, optical flow, egomotion

ACM Reference Format:
Yi Tian and Juan Andrade-Cetto. 2023. Egomotion from event-based SNN optical flow. In *International Conference on Neuromorphic Systems (ICONS '23), August 1–3, 2023, Santa Fe, NM, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3589737.3605978

1 INTRODUCTION

Egomotion estimation is a crucial task in visual navigation for robots, vehicles, and wearable computing, which involves computing the 3D camera motion relative to a rigid scene. Traditionally, egomotion is estimated by measuring the apparent motion from frame to frame of image features. However, conventional visual odometry systems are prone to failure in harsh illumination conditions due to motion blur and image saturation, which impede the extraction of image correspondences [28].

In contrast, event cameras detect independently the change of luminance in each pixel and produce an asynchronous feed of pixel coordinates where intensity changes occur. Since no frame or image is produced, event cameras can detect very fast motion with low latency, down to μs time scale, and high dynamic range. Since each pixel is processed independently, event cameras have high dynamic range up to 140dB, which makes them robust also to abrupt changes in illumination. Further, their sparse output without redundant information requires low power consumption, suggesting their adoption in mobile embedded systems.

These advantages make them an ideal candidate to solve the egomotion problem in difficult navigation tasks [30, 35]. However, the task is challenging, since event-based egomotion systems must compute apparent motion from such sparse asynchronous data feed, either aggregating events temporally to build features to track or match [3], or better yet treating all events independently [7].

Spiking Neural Networks (SNNs) process discrete spikes using timing information, making them a natural match to process the spatio-temporal data from event cameras. The combination of event cameras and SNNs has raised increasing interest in different areas of computer vision. They have mostly been used to tackle low-level vision tasks such as feature extraction [13], tracking [18, 19, 32], depth estimation [26], optical flow [12, 15, 23, 24, 33], motion segmentation [8, 29], and image reconstruction [36], but also some higher-level vision tasks such as object or gesture detection and recognition [2, 9, 22]. Limited work has been done in using SNNs for camera egomotion estimation [11].

In this paper, we present a method to estimate the egomotion of an event camera with an SNN. We compute optical flow using a pre-trained network and use it to compute global egomotion estimates. The SNN was trained using unsupervised learning with spike-timing-dependent plasticity, showing selectivity for local flow. To estimate both rotational and translational motion at different altitudes, we add an optimization-based pooling technique to compute global flow estimates, addressing the aperture problem for noisy and sparse normal flows. After pooling, egomotion with outlier rejection is computed in a final step.

For validation, we perform tests in multiple datasets on both pure rotation and pure translation motion along the three axes and compare our results with the state of the art. Our contribution is two-fold: first, we have presented a framework for event-based ego-motion estimation using optical flow from a pre-trained SNN. Secondly, we have implemented an optimization-based pooling technique to solve the aperture problem for the noisy and sparse normal flow input.

2 RELATED WORK

Previous works on SNN-based dense optical flow estimation have used bio-inspired Gabor filters [23, 25] or unsupervised learning with STDP to learn these filters [4, 24]. However, these methods

ICONS '23, August 1–3, 2023, Santa Fe, NM, USA

ACM ISBN 979-8-4007-0175-7/23/08…$15.00
https://doi.org/10.1145/3589737.3605978

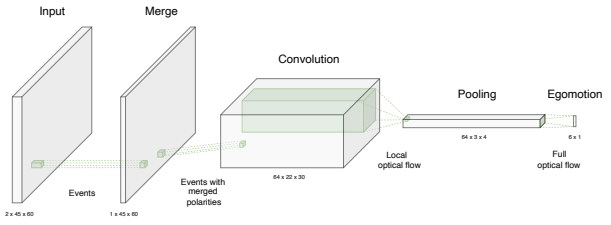


Figure 1: Proposed spiking neural network structure.

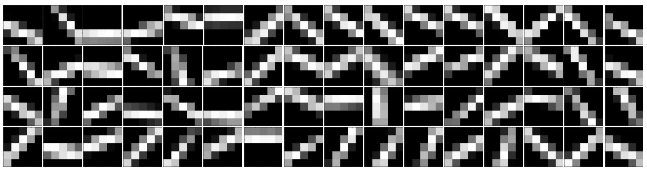


Figure 2: The 64 pre-trained spatiotemporal kernels used in the multisynaptic convolutional layer. Each kernel corresponds to a particular local velocity-tuned filter in a specific direction of motion. For instance, kernels in the first column will respond to diagonal motion, whereas kernels in the third column are sensitive mostly to vertical motion.

suffer from the common problem of spatio-temporal filter methods, which can only be deployed for test data similar to that used for training. For self-supervised event-based optical flow learning, most works follow the pipeline of their ANN counterpart and use photometric or contrast maximization losses to train the SNN with surrogate gradients [33, 34]. Some later works have combined an SNN in the encoder and an ANN in the decoder to balance performance and energy efficiency [15, 16].

While ANNs have been used to learn egomotion from event inputs [30, 34], the only notable work on SNN-based egomotion estimation to date is a framework for computing 3-axis angular velocity regression using supervised learning [11]. However, using purely SNNs to recover both rotational and translational egomotion has not yet been addressed in the literature.

3 NETWORK ARCHITECTURE

We adopt the hierarchical SNN architecture proposed by [24], which utilizes an adaptive neuronal model and unsupervised STDP learning. The proposed spiking neuronal model is a modified version of the widely used Leaky Integrate and Fire (LIF) model. The LIF model assumes that spike timing, rather than spike amplitude, carries the encoded information, which aligns with the contrast threshold firing mechanism of event cameras. The LIF model is both simple and computationally efficient, enabling the integration of multiple synapses from different neurons into a presynaptic train. Moreover, the model incorporates a refractory period after the synapses.

As shown in Fig.1, our modified neural network structure retains the input, merge, and multisynaptic convolution layers from[24]. However, we do not use a feature extraction layer since we are not interested in training local motion. Instead, we use the pre-trained kernels for the multisynaptic convolution layer to obtain local motion estimates and to compute local optical flow. Furthermore, we replace the pooling layer with our own layer, which includes a more robust integration of sparse normal flows. Additionally, we add an extra module to the network for the computation of egomotion that is robust to outliers. The purpose of each layer is summarized as follows:

Input layer: The input layer consists of a 2D neural map, with one dimension per event polarity. Each neuron in the input layer receives a spike train of events from the sensor within a temporal window of size τ_s, with no overlap. To improve computational efficiency, the input size is scaled by a factor of s in both the horizontal and vertical directions.

Merge layer: Events are triggered at abrupt changes in brightness, which primarily occur at edges. However, local motion cannot be determined unambiguously when viewed locally, leading to the aperture problem. At this stage, local motion is only dependent on the spatial configuration and frequency of events, and not their polarity. Hence, we merge the two polarity input neural maps into one convolutional single-synaptic layer with no neighboring connections.

Multisynaptic convolutional layer: This layer is equivalent to the MS-Conv layer in [24] and is responsible for computing local motion estimates through velocity-selective neurons. Neural connections are multisynaptic with different constant transmission delays τ_d. The spatiotemporal convolutional kernels used in this layer are pre-trained to encode features that respond to local flow in different directions and magnitudes. Local optical flow is computed using our modified version of the EdgeFlow algorithm [20], as described in Sec. 4.1.

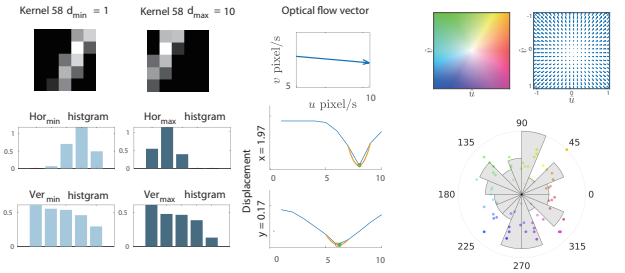
Pooling layer: Our convolutional and single-synaptic pooling layer uses the local optical flow estimates to compute a fused full flow estimate, incorporating two different pooling mechanisms for improved geometric consistency. More details are available in Sec. 4.2. The full flow output of this layer serves as input to the egomotion estimation module.

4 OPTICAL FLOW ESTIMATION

4.1 Local flow

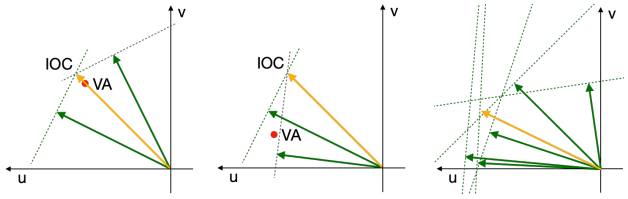
Local optical flow is computed using 64 pre-trained kernels from the rotating disk dataset in [24] (Fig. 2). These kernels are 3D filters (x, y, and τ) that identify motion at different directions and speeds. To estimate local velocities, histograms of the horizontal and vertical directions are computed for the two synapses with the most activity in the temporal window. Then, the sum of absolute differences for both horizontal and vertical pairs is obtained, and least squares fitting is applied to refine the solution [21].

An example of the computation for one of the kernels is shown in Fig. 3. The flow field is color-coded, with direction encoded in hue and speed in saturation. The bottom part of the plot shows a polar distribution of the flow vectors for the 64 kernels. It is noteworthy that the pre-trained kernels have a larger sensitivity for the estimation of vertical motion.



(a) SAD fitting (b) Kernel distribution

Figure 3: Optical flow computation from pre-trained kernels: a) the first two columns plot the histograms of the sum of intensity values along the horizontal and vertical directions. d_min and d_max indicate the corresponding kernel with the minimum and maximum delay. The last column calculates the sum of absolute differences (blue line) and least squares fit (red line). b) Color map and polar distribution of the pre-trained kernels.



(a) Type I (b) Type II (c) LSQ IOC

Figure 4: (a)-(b) Type I and Type II stimulus; (c) Least squares solution for intersection of constraints. The green arrows represent local flow vectors. The dotted lines indicate the constraints. Their intersection in the least squares sense is indicated by the yellow arrow.

4.2 Full flow

Local flow estimates are combined in the pooling layer to compute a more compact full flow estimate, similar to how biological systems integrate local motion information. However, local motion estimation suffers from the aperture problem, which means that flow parallel to local edges cannot be observed. To overcome this, optical flow is pooled across a larger receptive field using different pooling mechanisms.

Two commonly used mechanisms are geometric pooling [10] and intersection of constraints [1]. Geometric pooling computes a vector average (VA) from the set of flow data values, while intersection of constraints (IOC) computes a consistent solution compatible with each individual flow vector. For instance, the IOC solution for two flow estimates is at the intersection of their constraint lines, which are perpendicular to the direction of motion (see Fig. 4).

Research in neuroscience has shown that the pooling mechanism used in the human visual system depends on the type of stimuli received [5, 31]. Stimuli can be classified into two types: Type I stimuli have IOC solutions inside the cone described by the flow vectors, while Type II stimuli have IOC solutions outside this cone. The VA solution generates a veridical (albeit scaled) estimate of the velocity vector for Type I stimuli, while it deviates substantially from the IOC solution for Type II stimuli (see Fig. 4b). However, for noisy input data from an event camera, the IOC solution may also deviate substantially from the true value.

To solve the aperture problem for noisy SNN data, we propose a pooling method that combines both the VA and IOC methods depending on the type of stimuli received. Our pooling mechanism assumes that the motion is adequately encoded in the receptive field, which should be small enough to capture a single general motion direction and large enough to include responses to different kernels (different edge directions).

Sliding temporal window: To address the sparseness of local flow data, we employ a sliding temporal window integration mechanism at the output of the MS-Conv layer, assuming that flow changes smoothly over time. This increases the number of local flow vectors in the receptive field without the need to increase the receptive field size or the step size τ_s.

Vector average flow: When all local flow vectors in the receptive field lie in the same orientation bin (as shown in Fig. 3b), we assume a Type II stimulus and compute the pooling flow using the vector average method.

Least squares intersection of constraints: When neurons in the same receptive field fire corresponding to at least two different direction bins, we assume a Type I stimulus and compute the pooling flow using a least squares solution with robust outlier rejection to the intersection of constraints as described in section 4.3.

Data richness: If all local flow vectors in the receptive field come from the same kernel after update the temporal window, we assume that the observed features are not rich enough to describe motion. In that case, we choose not to update the pooling for that receptive field.

4.3 Least squares solution for the intersection of constraints

The problem of computing the least squares intersection of constraints can be formulated as follows: given n normal flow vectors $\mathbf{v}_i = (u_i, v_i)$, the goal is to find the flow vector whose constraint line minimizes the intersection error of all constraint lines. A point (x, y) in any given constraint line l_i must satisfy the equation $-u_i x - v_i y + (u_i^2 + v_i^2) = 0$.

To find the point X in the constraint line closest to all n lines A in the least squares sense, we can represent A as a matrix and solve for the eigenvector associated with the smallest eigenvalue of $A^T A$ subject to the constraint that $|X| = 1$. Specifically, we have:

$$A = \begin{bmatrix} -u_1 & -v_1 & u_1^2 + v_1^2 \\ \vdots & \vdots & \vdots \\ -u_n & -v_n & u_n^2 + v_n^2 \end{bmatrix} \qquad X = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{1}$$

Our least squares solution for the intersection of constraints is exemplified in Figure 4c. To obtain optimal results, the method requires local flow estimates from different directions. Furthermore, the method is robust to the presence of flow vectors with similar direction and different magnitudes since the quantity being minimized is the squared sum of the perpendicular distance to each constraint line, not the intersection point of all constraints.

5 EGOMOTION FROM OPTICAL FLOW

5.1 Motion field of a static scene

Up to this point, we have an array of full flow estimates at the output of the pooling layer from which to compute egomotion. We resort to the Longuet-Higgins model [17] independently to compute rotational and translational (although scale) velocity estimates that are robust to outliers. To provide a complete understanding of the process, let's review the technique. To project a 3D point $\mathbf{P} = (X, Y, Z)^T$ onto an ideal pinhole camera with focal length f, we have: $\mathbf{p} = (x, y)^T = (fX/Z, fY/Z)^T$. Taking the time derivative of $\mathbf{p}$ gives the relation between the velocity $\mathbf{V} = (V_x, V_y, V_z)^T$ of point $\mathbf{P}$ and the flow vector $\mathbf{v} = (u, v, 0)^T$ at $\mathbf{p}$: $\mathbf{v} = f(Z\mathbf{V} - V_z\mathbf{P})/Z^2$. To express the relation between $\mathbf{P}$ and the moving camera's translational and angular velocities, $\mathbf{T} = (T_x, T_y, T_z)^T$ and $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^T$, when P is static in the scene, we have: $\mathbf{V} = -(\mathbf{T} + \boldsymbol{\omega} \times \mathbf{P})$. By plugging $\mathbf{V}$ into $\mathbf{v}$, we get the motion flow at pixel i for a moving camera with velocities $\mathbf{T}$ and $\boldsymbol{\omega}$:

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \underbrace{\begin{bmatrix} -\frac{f}{Z_i} & 0 & \frac{x_i}{Z_i} \\ 0 & -\frac{f}{Z_i} & \frac{y_i}{Z_i} \end{bmatrix}}_{A_{\mathbf{T}_i}} \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} + \underbrace{\begin{bmatrix} \frac{x_i y_i}{f} & -\frac{f^2+x_i^2}{f} & y_i \\ \frac{f^2+y_i^2}{f} & -\frac{x_i y_i}{f} & -x_i \end{bmatrix}}_{A_{\boldsymbol{\omega}_i}} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad (2)$$

Note that only the translational component is depth-dependent. If we know the scene depth Z_i in at least three non-collinear flow field locations, one can estimate translational and rotational camera velocities. Otherwise, egomotion can only be determined up to a scale factor.

5.2 Pure rotational motion

To estimate angular velocities for pure rotation when the scene points are in front of the camera, we can use Eq. 2 to solve a linear system via least squares with at least two image points and their corresponding flow components:

$$\begin{bmatrix} A_{\boldsymbol{\omega}_1} \\ \vdots \\ A_{\boldsymbol{\omega}_n} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_n \end{bmatrix} \quad (3)$$

5.3 Pure translational motion

In cases where depth information is available, both translational and rotational velocities can be estimated. However, without depth information at flow vector locations, only the direction of the translational velocity can be determined. In such cases, the vanishing point (focus of expansion or contraction) can be used to estimate the direction of flow. The vanishing point represents the point where the 3D velocity vector $\mathbf{T}$ intersects the image plane (see Fig.5). The 3D coordinates of the image point $\mathbf{p}3D = (x, y, f)^T$, the flow vector $\mathbf{v} = (u, v, 0)^T$, and the linear camera velocity $\mathbf{T}$ are coplanar, i.e., $(\mathbf{p}3D \times \mathbf{v})^T\mathbf{T} = 0$. By using at least two image points and their corresponding flow vectors, a linear system can be constructed

$$\mathbf{H}\mathbf{T} = \begin{bmatrix} (\mathbf{p}_{3D_1} \times \mathbf{v}_1)^T \\ \vdots \\ (\mathbf{p}_{3D_n} \times \mathbf{v}_n)^T \end{bmatrix} \mathbf{T} = 0 \quad (4)$$

and the translational direction can be estimated by constraining the estimate of $\mathbf{T}$ to have unit magnitude and choosing the eigenvector corresponding to the smallest eigenvalue of $\mathbf{H}^T\mathbf{H}$ [6].

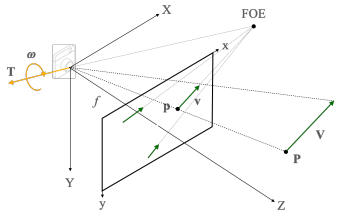


Figure 5: Camera velocity as a consequence of scene point projections and their flow vectors.

5.3.1 RANSAC for robust estimation. The full flow output may include noisy values, which can lead to errors in the estimation. To address this issue, we propose a RANSAC-based outlier rejection algorithm for robust estimation of the egomotion components. The algorithm considers both endpoint error (EE) and angle error (AE) between the input flow vectors $\mathbf{v}_i$ and the model flow vectors $\bar{\mathbf{v}}_i$. The model flow vectors are computed using Eq. 2 and the model estimates $\bar{\mathbf{T}}$ and $\bar{\boldsymbol{\omega}}$.

$$EE_i = \|\bar{\mathbf{v}}_i - \mathbf{v}_i\| , \qquad AE_i = \cos^{-1} \frac{\bar{\mathbf{v}}_i^T \mathbf{v}_i}{\|\bar{\mathbf{v}}_i\| \|\mathbf{v}_i\|} . \quad (5)$$

6 EXPERIMENTS

6.1 Datasets and implementation details

To evaluate our method, we generated synthetic datasets using the planar scene render of the ESIM simulator [27], using two patterns: a checkerboard and a windmill. The planar scene was positioned 1 meter high in the camera reference frame, and the camera intrinsic parameters were $(200, 200, 120, 90)$. We first tested our method on constant roll rotation at a speed similar to that used for training, followed by simulated constant diagonal translational motion perpendicular to the principal axis at 1 meter depth. We also validated the system with multiple 3-axis rotation datasets, and for translational motion with unknown depth, we used a simple 3D scene involving depth variations.

We implemented our method in C++ using the cuSNN library and ran all tests with a simulation step size of 1 *ms* on 500 *ms* data sequences on a desktop equipped with an Intel(R) Core(TM) i7-7700 CPU and a NVIDIA Quadro K420 GPU. The SNN architecture and RANSAC parameters are listed in Table 1. The input layer size was scaled by $s = 4$. The optimal pooling receptive field size r_f depends on the pattern texture and the assumption of consistent motion. Increasing r_f would provide more accurate flow estimates by including more edges in the estimation of each flow component, while a smaller size would produce more flow vectors in the pooling layer, allowing for more robustness in the egomotion estimation. We set the r_f size to 7×7.

We also modified some parameters of the original SNN. Since α is an inhibition term in the neural network that's related to the firing time, we set a small value to avoid losing important events in the motion pattern. However, to compensate for the larger amount of firings, we increased the firing threshold V_{th} and decay time constant τ_m.

Table 1: Parameters used for the windmills dataset tests

layer	V_{th} mV	τ_m ms	α –	rf -	bins –	W ms
MS-Conv	0.8	40	0.1	5×5	–	–
Pooling	0	5	1	7×7	8	10

(a) Parameters for SNN

motion type	steps –	inliers –	EE pixels	AE rad
translation	20	7	22	0.18
rotation	20	8	50	0.5

(b) Parameters for RANSAC

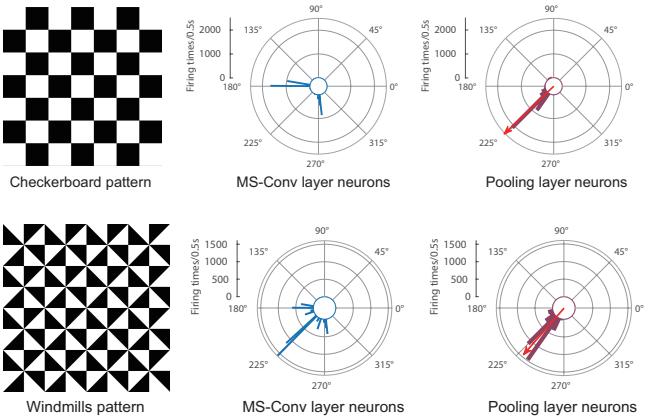


Figure 6: Polar histograms of neurons firing counts in the MS-Conv and pooling layers for pure translation motion tests. The red arrow shows the average flow direction.

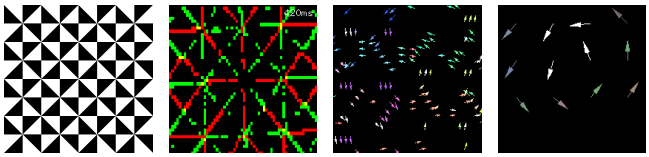
6.2 Pooling flow evaluation

In Fig. 6, we present the response of the MS-Conv layer neurons and pooling layer neurons for the constant translational motion test, where the ground truth flow direction is 225°. While the checkerboard pattern contains only horizontal and vertical edges, the windmill pattern includes edges with eight different orientations. For the checkerboard pattern, there are no normal flow vectors in the MS-Conv layer that match the motion direction. This is because the training sequence did not have kernels that correspond to the executed motion direction, resulting in a deviation of T_y compared to T_x due to unequal distribution of kernels describing different motions. Nevertheless, the pooling layer provides satisfactory full flow results in both cases, despite the limitations of the available kernels.

Table 2 reports the evaluation of the average endpoint error (AEE) and average angle error (AAE) between the pooling flow and the ground truth flow. Our method is compared with EV-FlowNet [33], and outliers flows with EE larger than 3 pixels and 5% of their magnitude are indicated. For the pure translation case, our results outperform those of EV-FlowNet under the same experimental conditions. This is due to the high contrast synthetic images used in our test datasets, which contain relatively dense event bursts compared to the natural scene datasets that EV-FlowNet was trained with. In an attempt to improve EV-FlowNet's performance for the translation case, we either increased frame windows or decreased the speed of motion. The results improved but were still less accurate than ours, highlighting the advantage of the firing patterns in the MS-Conv layer better matching those of the pre-trained kernels in our case.

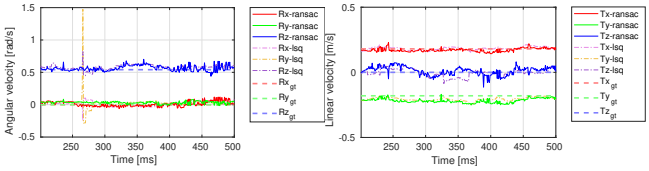
Table 2: Results for the computation of pooling flow with and without RANSAC compared to EV-FlowNet on the windmills dataset for pure roll rotation and pure diagonal translation.

	Windmills pure roll rotation			**Windmills pure translation**		
	AEE	% Outliers	AAE	AEE	% Outliers	AAE
EV-FlowNet [33] dt = 1 frame	1.67	8.73	-	2.44	29.03	-
EV-FlowNet [33] dt = 2 frames				1.77	12.04	-
EV-FlowNet [33] half speed				1.71	10.37	-
our method without RANSAC	1.20	2.94	0.33	0.48	0	0.17
our method with RANSAC	0.82	0	0.27	0.40	0	0.14



(a) Pattern (b) Input (c) Normal flow (d) Full flow

Figure 7: Pooling flow results for pure rotation: (a) windmills pattern; (b) on and off polarity in the input pattern; (c) flow output from the MS-Conv layer, normalized and color coded according to Fig. 3b; and (d) full flow from the pooling layer with RANSAC inliers shown with random color and outliers shown in white.



(a) Pure rotation (b) Pure translation

Figure 8: Motion estimation results for pure rotation and pure translation in a 500 ms interval in the windmills dataset. The results with and without RANSAC are shown with solid and dashed dot lines, respectively. Ground-truth values are plotted with dotted lines.

6.3 Pure rotation and pure translation on a planar scene

The results of the different layers in the SNN are depicted in Fig.7 for the case of constant fronto-parallel rotation with the windmill pattern. It is worth noting that in the full flow plot (Fig.7d), the flow vectors located in the center of the image have smaller ground truth magnitudes than those of the best matching kernels trained, leading to outliers.

The estimation results of both angular velocity and translation direction on the entire 500 ms sequence of the windmills dataset are shown in Fig.8 and Table 3. Depending on the number of bins in the pooling layer, the IOC method proposed may fail to produce accurate optical flow estimates, as exemplified in the rotation plot at step 266. However, this situation can be easily corrected with the aid of our RANSAC module. Additionally, the results in the table are consistent with the observation made in Sec.6.2 regarding the unequal distribution of kernels.

Table 3: Motion estimation results for pure rotation and pure translation in a 500 ms interval in the windmills dataset.

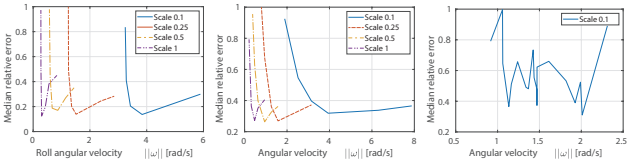
	Windmills pure roll rotation			Windmills pure translation		
	R_x rad/s	R_y rad/s	R_z rad/s	T_x m/s	T_y m/s	T_z m/s
true value	0	0	0.5409	0.18	- 0.18	0
mean estimate	0.0101	0.0333	0.5625	0.172	- 0.218	0.0128
RMSD	0.0381	0.0187	0.0425	0.0131	0.0162	0.0294
RMSE	0.0303	0.0346	0.0357	0.0156	0.0413	0.0321

6.4 3D rotations

To evaluate the effectiveness of our method in estimating 3D rotations, we created a dataset of the windmills pattern subject to constant angular velocities around the three axes. We varied the time-scale of the events to generate a range of angular velocities. We then adjusted the maximum delay for all pre-trained kernels and tested the method for different ranges of angular velocities. We also scaled other parameters such as the time delay, the sliding integration temporal window, and the RANSAC thresholds according to the corresponding scale factor. Fig. 9 (a and b) show the mean relative error for the pure roll and 3-axes rotation tests with varying angular velocities in the windmills dataset, and (c) for SO(3) with scaled kernels in the 19 natural scene sequences from the dataset in [11]. We observed that slower rotations led to larger errors since they are more likely to cause erroneous firing of the neurons. Our method performed better on pure roll rotation than on SO(3) rotation, as the kernels were trained from similar motion patterns and lacked richness to explain other types of motion. Additionally, we found that the MS-Conv layer neurons had better direction selectivity than speed selectivity, which is consistent with the findings of [24].

To compare our results with the only other work in the literature that attempted 3D rotation estimation with SNNs [11], we tested on their synthetic dataset. The dataset contains 100 ms sequences generated from panorama images, with slightly varying random angular velocities. Without fine-tuning the parameters, our method worked on approximately 19 percent of the 100 sequences randomly selected from the test set. We scaled the angular velocities between 0.5-2.5 rad/s, and decreased the input scale by a factor of 2 and the receptive field for the pooling flow to 5 to ensure sufficient optical flow vectors in the output. Fig.9c shows the median relative error for all sequences, which ranged from 0.4 to 0.6. This value is slightly higher than the results reported in [11] for angular velocities between 0° and 120° (see the last block in Table 4). In the Table, the first column in each set indicates the speeds with smaller relative errors (minima in the plots), and the second column indicates the average relative error $\|\bar{\omega} - \omega_{gt}\| \cdot \|\omega_{gt}\|^{-1}$ for the entire speed sensitivity ranges shown. We also present the results for some of these sequences in Fig. 10.

Furthermore, we tested their pre-trained model in our windmills rotation dataset and found that the median relative error ranged from 0.7 to 0.9. These poor results indicate the limited generalization ability of the optical flow model used, which had difficulty estimating 3D rotation in sequences with speeds and patterns different from those used to train the MS-Conv kernels.



(a) Roll / windmills **(b) SO(3) / windmills** **(c) SO(3) / [11]**

Figure 9: Rotation estimation for varying angular velocities and scaled filters.

Table 4: Rotation estimation results for kernels scaled at 0.1 (blue lines in Fig. 9).

Motion type	Roll		SO(3)				
Dataset	windmills		windmills			natural scene [11]	
	ours best	ours range/mean	ours best	ours range/mean	[11]	ours range/mean	[11]
$\bar{\omega}$ rad/s	3.838	3.245 - 5.891	3.938	1.890 - 7.876	-	0.91 - 2.312	-
relative error	0.137	0.377	0.319	0.482	0.977	0.560	0.209
relative error-LSQ	0.120	0.511	0.585	4.274	-	3.683	-
RMSE	0.925	1.950	1.255	1.772	2.959	0.849	0.993

Table 5: Translational direction estimation for 2D and 3D scenes.

Motion type	Linear translation			Linear translation		
Dataset	Windmills planar			3D checkerboard		
Normalized direction	T_x	T_y	T_z	T_x	T_y	T_z
Ground truth	0.3208	-0.3208	-0.8912	0.3208	-0.3208	-0.8912
Mean	0.3027	-0.3530	-0.8821	0.2749	-0.2801	-0.9151
RMSD	0.0511	0.0473	0.0305	0.0571	0.0667	0.0304
RMSE	0.0541	0.0571	0.0318	0.0732	0.0780	0.0386

6.5 3D translation direction

We generated two datasets with a planar scene of the windmill pattern and a 3D scene made up of checkerboard pattern boxes, both moving at a constant linear velocity in the camera reference frame. Using the method described in Section 5.3 and without any depth information, we estimated the normalized translational velocity for both datasets.

Figure 11 illustrates the simulated 3D scene and its corresponding optical flow. The camera's direction of motion intersects the image plane at the vanishing point. We normalized the estimates and compared them to the ground-truth values in Table 5. However, our pooling flow method struggles to accurately estimate the magnitude of flow vectors near the vanishing point, where the magnitude of flow vectors is small. Moreover, for the 3D scene, flow vectors are also difficult to estimate near object boundaries, making the estimation of motion direction less accurate for cluttered 3D scenes. As shown in the table, the mean, RMSD, and RMSE values confirm this observation.

7 DISCUSSION

In this study, we investigated the potential of using pre-trained SNNs for egomotion estimation on event data. We tested our method on various datasets involving pure translations and rotations on both 2D and 3D planes. We assessed the SNN's ability to generalize by using pre-trained kernels to compute the flow field.

Our approach's performance is constrained by the pre-trained kernel's statistics. The main obstacle to testing more complex motion sequences than those trained or in more complex datasets is

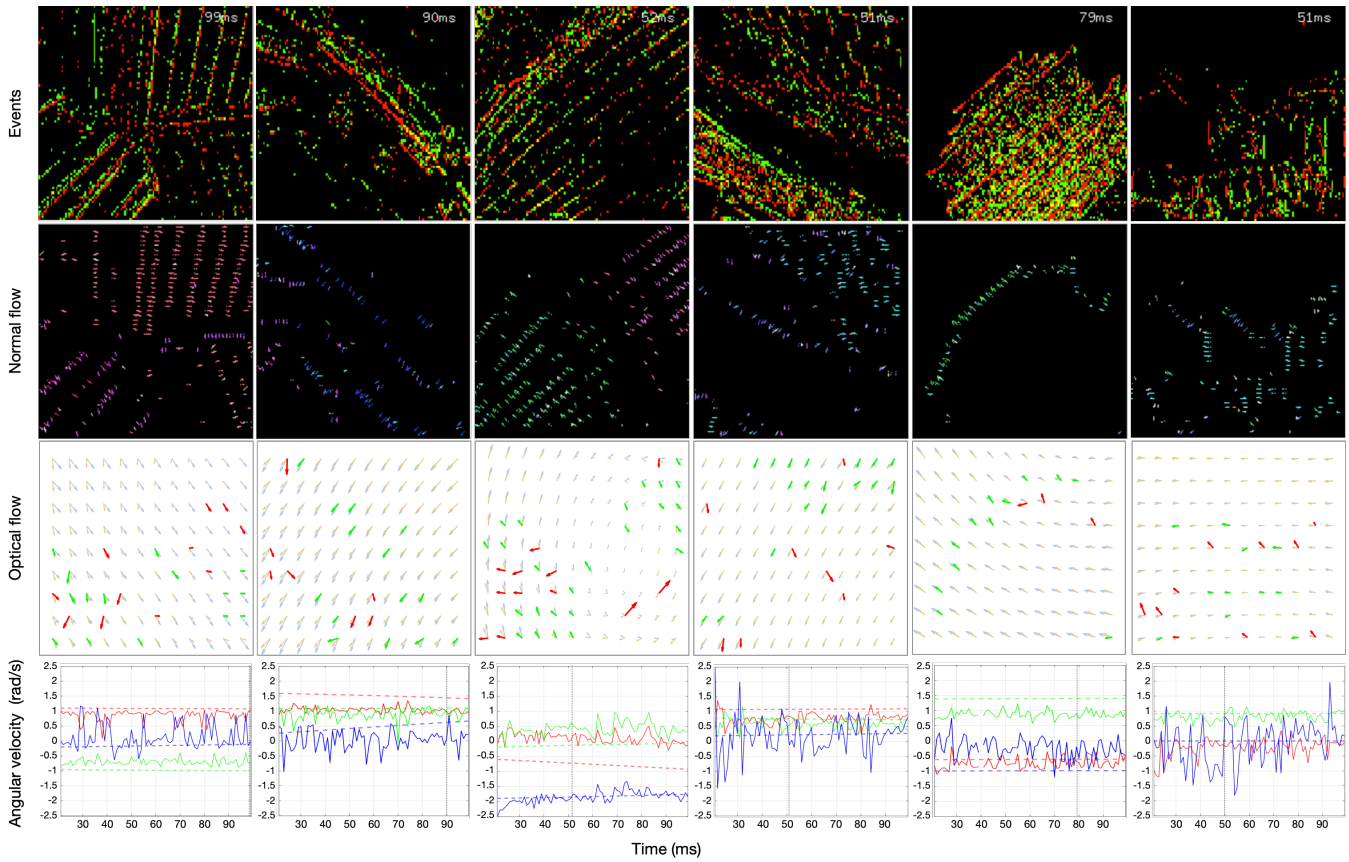
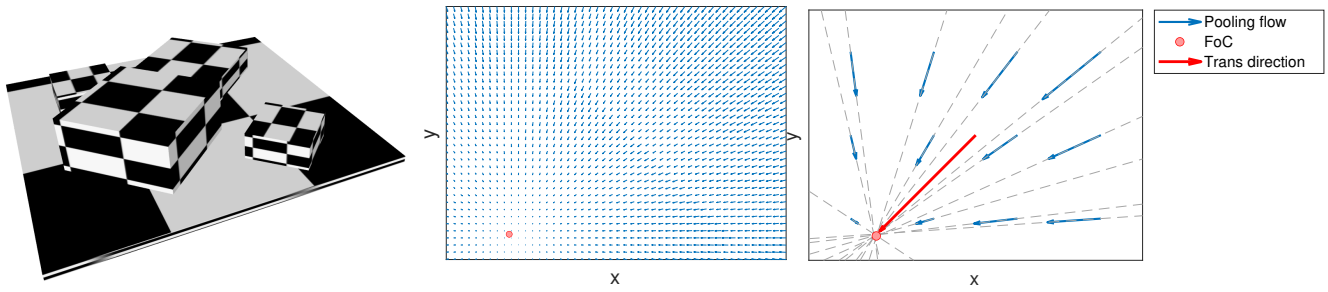


Figure 10: Angular velocity estimation for the sequences in [11]. 1st row: input events. color shows polarity (yellow is for overlap of positive and negative events in the same sampling interval). 2nd row: color-coded optical flow. 3rd row: pooling flow output (green - RANSAC inliers; red - outliers; light blue - ground truth; light yellow - estimated model) with green and red indicating inliers and outliers picked by the RANSAC algorithm, ground truth flow in light blue, and estimated RANSAC model flow in light yellow. Last row: angular velocity estimation throughout the 100 ms simulation. (RGB - xyz; dotted line - ground truth). Red, green and blue indicate rotation about the x, y and z axes respectively. Ground truth values in dotted lines. Motion estimation starts at 20ms since the MS-layer needs a delay period to produce a reliable output.



(a) 3D scene with varying depths of checkerboard pattern boxes

(b) Ground truth simulated translation flow

(c) Pooling translation flow for a 3D scene

Figure 11: Translation flow for a 3D scene: (a) 3D scene used for simulation; (b) simulated ground truth flow; and (c) estimated pooling flow. The red point represents the focus of contraction, i.e., the location in the image where the camera motion direction intersects the image plane.

the lack of available flow vectors in the pre-trained kernel set. Additionally, since the SNN was trained in a fully unsupervised manner from a synthetic sequence, noise affects the read-out mechanism for the pre-trained model, and erroneous neuron firing can occur during the inference phase.

The poor speed selectivity of the MS-Conv layer neurons poses a challenge to our IOC method for full flow estimation since it is sensitive to variations in the normal flow's magnitude with similar directions. Moreover, the sparse data produced by event cameras results in flow vectors that are texture and motion-dependent. We balanced simulation speed, accuracy, and sparsity in tuning the SNN, implementing a pooling technique with robust motion estimation to cope with noise and input sparsity. Our method assumes rigid motion only. Despite the pre-trained kernel's inherent limitations in flow estimation, the method produced reliable results on tests involving pure rotational and translational egomotion. However, it does not generalize favorably for sequences and motion speeds different from those trained.

Our work was based on the state-of-the-art at the moment we initialized the project. Rapid development has taken place in the neuromorphic field in the recent years. In future studies, we will explore novel architectures and learning methods to address this problem [14], as well as utilize richer datasets from real-world scenes for both learning and validation.

ACKNOWLEDGMENTS

This work received support from projects EBCON (PID2020-119244GB-I00) and AUDEL (TED2021-131759A-I00) funded by MCIN/ AEI/ 10.13039/ 501100011033 and by the "European Union NextGenerationEU/PRTR"; the Consolidated Research Group RAIG (2021 SGR 00510) of the Departament de Recerca i Universitats de la Generalitat de Catalunya; and by an FI AGAUR PhD grant to Yi Tian.

REFERENCES

[1] Edward H. Adelson and J. Anthony Movshon. 1982. Phenomenal coherence of moving visual patterns. *Nature* 300, 5892 (1982), 523–525.
[2] Himanshu Akolkar, Stefano Panzeri, and Chiara Bartolozzi. 2015. Spike time based unsupervised learning of receptive fields for event-driven vision. In *IEEE Int. Conf. Robotics Autom.* Seattle, WA, 4258–4264.
[3] Ignacio Alzugaray and Margarita Chli. 2020. HASTE: Multi-hypothesis asynchronous speeded-up tracking of events. In *British Mach. Vis. Conf.* Virtual.
[4] Thomas Barbier, Céline Teulière, and Jochen Triesch. 2021. Spike timing-based unsupervised learning of orientation, disparity, and motion representations in a spiking neural network. In *2021 IEEE CVPR Workshops*. Nashville, TN, 1377–1386.
[5] Linda Bowns and David Alais. 2006. Large shifts in perceived motion direction reveal multiple global motion solutions. *Vision Research* 46, 8-9 (2006), 1170–1177.
[6] Anna R. Bruss and Berthold K.P. Horn. 1983. Passive navigation. *Comput. Vis. Graph. Image Process.* 21, 1 (1983), 3–20.
[7] William Chamorro, Juan Andrade-Cetto, and Joan Solà. 2020. High-speed event camera trackng. In *British Mach. Vis. Conf.* Virtual.
[8] Guillaume Debat, Tushar Chauhan, Benoit R. Cottereau, Timothée Masquelier, Michel Paindavoine, and Robin Baures. 2021. Event-based trajectory prediction using spiking neural networks. *Front. Comput. Neurosci.* 15 (2021).
[9] Lei Deng, Kai Huang, Yannan Xing, Gaetano Di Caterina, and John Soraghan. 2020. A new spiking convolutional recurrent neural network (SCRNN) with applications to event-based hand gesture recognition. *Front. Neurosci.* 14 (2020), 590164.
[10] Walter Gander, Gene H. Golub, and Rolf Strebel. 1994. Least-squares fitting of circles and ellipses. *BIT Num. Math.* 34, 4 (1994), 558–578.
[11] Mathias Gehrig, Sumit Bam Shrestha, Daniel Mouritzen, and Davide Scaramuzza. 2020. Event-based angular velocity regression with spiking networks. In *IEEE Int. Conf. Robotics Autom.* Paris, 4195–4202.
[12] Germain Haessig, Andrew Cassidy, Rodrigo Alvarez, Ryad Benosman, and Garrick Orchard. 2018. Spiking optical flow for event-based sensors using IBM's TrueNorth neurosynaptic system. *IEEE Trans. Biomed. Circ. Syst.* 12, 4 (2018), 860–870.
[13] Germain Haessig, Francesco Galluppi, Xavier Lagorce, and Ryad Benosman. 2019. Neuromorphic networks on the SpiNNaker platform. In *IEEE Int. Conf. Artif. Intell. Circuits Syst.* Taiwan, 86–91.
[14] Jesse Hagenaars, Federico Paredes-Vallés, and Guido de Croon. 2021. Self-Supervised Learning of Event-Based Optical Flow with Spiking Neural Networks. *Conf. Neural Inf. Process. Syst.* 34 (2021).
[15] Chankyu Lee, Adarsh Kosta, Alex Zihao Zhu, Kenneth Chaney, Kostas Daniilidis, and Kaushik Roy. 2020. Spike-FlowNet: Event-based optical flow estimation with energy-efficient hybrid neural networks. In *Eur. Conf. Comput. Vis.* Springer, Glasgow, UK, 366–382.
[16] Chankyu Lee, Adarsh Kumar Kosta, and Kaushik Roy. 2021. Fusion-FlowNet: Energy-efficient optical flow estimation using sensor fusion and deep fused spiking-analog network architectures. arXiv:2103.10592
[17] Hugh C. Longuet-Higgins and K. Prazdny. 1980. The interpretation of a moving retinal image. *Proc. Royal Soc. London - Biological Sci.* 208, 1173 (1980).
[18] Yihao Luo, Min Xu, Cahiong Yuan, Xiang Cao, Liangqi Zhang, Yan Xu, Yianjiang Wang, and Qi Feng. 2021. SiamSNN: Siamese spiking neural networks for energy efficient object tracking. In *Int. Conf. Artif. Neural Netw.* Springer, Bratislava, Slovakia, 182–194.
[19] Yihao Luo, Quanzheng Yi, Tianjiang Wang, Ling Lin, Yan Xu, Jing Zhou, Caihong Yuan, Jingjuan Guo, Ping Feng, and Qi Feng. 2019. A spiking neural network architecture for object tracking. In *Int. Conf. Image and Graphics*. Springer, Beijing, 118–132.
[20] Kimberly McGuire, Guido de Croon, Christophe de Wagter, Bart Remes, Karl Tuyls, and Hilbert Kappen. 2016. Local histogram matching for efficient optical flow computation applied to velocity estimation on pocket drones. In *IEEE Int. Conf. Robotics Autom.* Stockholm, 3255–3260.
[21] Kimberly McGuire, Guido De Croon, Christophe De Wagter, Karl Tuyls, and Hilbert Kappen. 2017. Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone. *IEEE Robotics Autom. Lett.* 2, 2 (2017), 1070–1076.
[22] Nico Messikommer, Daniel Gehrig, Antonio Loquercio, and Davide Scaramuzza. 2020. Event-based asynchronous sparse convolutional networks. In *Eur. Conf. Comput. Vis.* Springer, Glasgow, 415–431.
[23] Garrick Orchard and Ralph Etienne-Cummings. 2014. Bioinspired visual motion estimation. *Proc. IEEE* 102, 10 (2014), 1520–1536.
[24] Federico Paredes-Vallés, Kirk Yannick Willehm Scheper, and Guido De Croon. 2020. Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: From events to global motion perception. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 8 (2020), 2051–2064.
[25] Francesca Peveri, Simone Testa, and Silvio P Sabatini. 2021. A cortically-inspired architecture for event-based visual motion processing: From design principles to real-world applications. In *IEEE CVPR Workshops*. Nashville, TN, 1395–1402.
[26] Ulysse Rançon, Javier Cuadrado-Anibarro, Benoit R. Cottereau, and Timothée Masquelier. 2021. StereoSpike: Depth learning with a spiking neural network. In *Spiking Neural Netw. Universal Funct. Approx. Workshop*. Virtual.
[27] Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza. 2018. ESIM: An open event camera simulator. In *2nd Conf. Robot Learning*, Vol. 87. Zürich, Switzerland, 969–982.
[28] Davide Scaramuzza and Friedrich Fraundorfer. 2011. Visual odometry. *IEEE Robotics Autom. Mag.* 18, 4 (2011), 80–92.
[29] Timo Stoffregen, Guillermo Gallego, Tom Drummond, Lindsay Kleeman, and Davide Scaramuzza. 2019. Event-based motion segmentation by motion compensation. In *IEEE Int. Conf. Comput. Vis.* Seoul, 7244–7253.
[30] Chenxi Ye, Anton Mitrokhin, Cornelia Mermüller, James A. Yorke, and Yiannis Aloimonos. 2020. Unsupervised learning of dense optical flow, depth and egomotion with event-based sensors. In *IEEE/RSJ Int. Conf. Intell. Robots Syst.* Las Vegas, NV, 5831–5838.
[31] Christopher Yo and Hugh R. Wilson. 1992. Perceived direction of moving two-dimensional patterns depends on duration, contrast and eccentricity. *Vision Research* 32, 1 (1992), 135–147.
[32] Jiqing Zhang, Kai Zhao, Bo Dong, Yingkai Fu, Yuxin Wang, Xin Yang, and Baocai Yin. 2021. Multi-domain collaborative feature representation for robust visual object tracking. *The Visual Computer* 37 (2021), 2671–2683.
[33] Alex Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. 2018. EV-FlowNet: Self-supervised optical flow estimation for event-based cameras. In *Robotics Sci. Syst. Conf.* Pittsburgh, PA.
[34] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. 2018. Unsupervised event-based optical flow using motion compensation. In *ECCV Workshops*. Munich, 711–714.
[35] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. 2019. Unsupervised event-based learning of optical flow, depth and egomotion. In *IEEE Conf. Comput. Vis. Pattern Recognit.* Long Beach, CA, 989–997.
[36] Lin Zhu, Siwei Dong, Jianing Li, Tiejun Huang, and Yonghong Tian. 2020. Retina-like visual image reconstruction via spiking neural model. In *IEEE Conf. Comput. Vis. Pattern Recognit.* Seattle, WA, 1438–1446.