Neurocomputing xxx (xxxx) xxx

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

# Neuromorphic implementation of motion detection using oscillation interference

Elishai Ezra Tsur[a,b,*], Michal Rivlin-Etzion[a,*]

[a] Department of Neurobiology, Weizmann Institute of Science, Rehovot, Israel
[b] Neuro-Biomorphic Engineering Lab (NBEL), Jerusalem College of Technology, Jerusalem, Israel

ARTICLE INFO

*Article history:*
Received 15 October 2018
Revised 29 May 2019
Accepted 25 September 2019
Available online xxx

Communicated by Duan Shukai

*Keywords:*
Neural engineering framework
Nengo
Optical flow
Neuromorphic vision sensor
Spike-based camera emulator
Motion detection

ABSTRACT

Motion detection is paramount for computational vision processing. This is however a particularly challenging task for a neuromorphic hardware in which algorithms are based on interconnected spiking entities, as the instantaneous visual stimuli reports merely on luminance change. Here we describe a neuromorphic algorithm, in which an array of neuro-oscillators is utilized to detect motion and its direction over an entire field of view. These oscillators are induced via phase shifted Gabor functions, allowing them to oscillate in response to motion in one predefined direction, and to dump to zero otherwise. We developed the algorithm using the Neural Engineering Framework (NEF), making it applicable for a variety of neuromorphic hardware. Our algorithm extends the existing growing set of approaches aiming at utilizing neuromorphic hardware for vision processing, which enable to minimize energy exploitation and silicon area while enhancing computational capabilities.



## 1. Introduction

The emergence of motion perception from spatiotemporal patterns of light has an essential role in vision [1,2]. Motion perception is also fundamental in numerous artificial vision processing, which span a wide spectrum of applications, ranging from support systems in surgical robotics [3] to mobile robot navigation [4]. To support the increasing demand of such applications for computational resources, multiprocessor System-on-Chip (SoC) platforms were developed, demonstrating a wide collection of architectures [5] including neuromorphic [6].

In neuromorphic computing architectures, computational principles of biological neural circuits are utilized to design and build artificial neural systems. These architectures process information represented with spikes and are comprised of physically (in contrast to computationally) joined networks of interconnected computing elements (e.g. silicon neurons). Neuromorphic elements are densely connected, [7], supporting energy efficient information processing with a high signal-to-noise ratio. Neuromorphic systems were shown to produce the same computational capabilities of a digital system with 10,000 times less energy and 100 times less silicon area [8].

Some of the first and greatest successes in Neuromorphic computing architectures have been in vision processing [9], including the first silicon retina by Mahowald and Mead [10], and its biologically plausible version by Zaghloul and Boahen [11]. Currently, most neuromorphic vision sensors communicate transients in luminance via the Address Event Representation (AER) protocol. They are comprised of an array of silicon neurons, each generates spikes in response to a change in luminance. Spikes are time-multiplexed over an asynchronous data bus via an address encoder, which designates each spike with a corresponding address (usually, the neuron's $x$–$y$ coordinate). These frame-less and event-driven neuromorphic Dynamic Vision Sensors (DVS) can resolve thousands of frames per second, have fine temporal resolution, high dynamic range and high signal to noise ratio. Moreover, since DVS perform sensor-level data compression, they optimize data transfer, storage, and processing, enhancing the power and size efficiency of the system [12].

However, there is a considerable gap between neuromorphic hardware and our current software and algorithmic approach, as the latter is fundamentally designed to be sequentially executed over von Neumann computing architectures [13] and is based on synchronous signals (e.g. frames). A neuromorphic approach for algorithm design based on spiking neurons is therefore a necessity. One framework for designing functional neuronal circuits

* Corresponding authors.
*E-mail addresses:* elishai@nbel-lab.com (E.E. Tsur), michal.rivlin@weizmann.ac.il (M. Rivlin-Etzion).

https://doi.org/10.1016/j.neucom.2019.09.072
0925-2312/

is the Neural Engineering Framework (NEF). NEF encodes signals with ensembles of spiking neurons and provides decoders as linear/nonlinear transformations between neuron ensembles [14].

In this work, we utilized NEF for the implementation of a neuromorphic motion detector using the Oscillation Interference (OI) model [15]. Our model was developed in Nengo [16] and deployed over a GPU for efficient evaluation. Visual input was implemented via a neuromorphic spiking camera, as well as via a spiking camera emulator that we developed. Our model can be deployed over neuromorphic hardware for near real-time detection of motion, is sensitive over a wide range of velocities (up to 10 times the optimal specification) and does not heavily rely on the neuronal spatial organization and synaptic time constants. It is particularly appealing for newly developed neuromorphic hardware, which have a designated compiler that supports NEF, such as the BrainDrop [17] and Intel's Loihi chip [18].

## 2. Methods

### 2.1. Dynamics and the neural engineering framework

NEF is often referred to as a "neural compiler", being able to transform high level specifications given in terms of vectors and functions to a set of interconnected ensembles of spiking neurons. In NEF, the firing rate $\delta$ of neuron $i$ in response to stimulus $x$ (tuning curve) is defined with:

$$\delta_i(x) = G_i\left[\alpha_i e_i x + J_i^b\right] \tag{1}$$

Where $G_i$ is a spiking neuron model (here, we used leaky-integrate-and-fire (LIF)), $\alpha_i$ is the gain term, $e_i$ is the neuron's preferred direction (encoding vector) and $J_i^b$ is a fixed background current. An ensemble of neurons, in which each neuron has its own gain and preferred direction, can encode a vectorized high-dimensional stimuli. The encoded stimuli $\hat{x}$ can be decoded using:

$$\hat{x} = \sum_i^N a_i(x) d_i \tag{2}$$

Where $N$ is the number of neurons, $a_i(x)$ is the postsynaptic filtered activity of neuron $i$ to stimuli $x$ and $d_i$ is a linear decoder which was optimized to reproduce $x$ using least squared optimization. As the number of neurons N increase, the mean squared error decreases as $1/N$.

Eqs. (1) and (2) specify the encoding and decoding of vectors using distributed activity of neuronal ensembles. A key aspect in neuromorphic computing is activity propagation – the transference of data from one neuron group to another - by linking ensembles with a weighted matrix of synaptic connections. The resulted activity transformation is a function of $x$. Notably, it was shown that any function $f(x)$ can be approximated using a specific set of decoding weights $d^f$ [19].

Defining $f(x)$ in NEF can be made by connecting two neuronal ensembles A and B via neural connection weights $w_{ij}(x)$ using:

$$w_{ij} = a_j d_i^f L e_j \tag{3}$$

Where $i$ is the neuron index in ensemble $A$, $j$ is the neuron index in ensemble $B$, $d_i^f$ is a linear decoder, which was optimized to transform $x$ to $f(x)$, and $L$ is a linear relationship between $x$ and $f(x)$. $d$ matches the linear decoder in Eq. (2) and is the set of connection weights needed to decode $x$ given the activity in $A$; $e$ is the connection weights which represent $f(x)$ in ensemble $B$. That is, the optimal weights to pass the information from A to B is simply the dot-product of $d$ and $e$ as it is reflected in Eq. (3). Further details and a full explanation are in [19].

A very useful aspect of NEF, which is of a particular interest here, is its ability to resolve dynamic behavior or detect dynamic input, such as the motion of an object in the visual image and its direction. As we connect ensembles of neurons to compute $f(x)$, we introduce low-pass filtered spikes from one ensemble to the other, resulting in: $y(t) = f(x(t)) * h(t)$, where $*$ is the convolution operator and $h(t)$ is the synaptic response function (a decaying exponential), which is given in the Laplace domain as: $H(s) = 1/(1+st)$. Given an ensemble encoding $X$, an input $u(t)$, and a recursive connection for resolving $g(x(t))$ (a feedback connection from $x$ back to itself), we define in the Laplace domain $X(s) = H(s)(G(s) + U(s))$. Following substitution and rearrangement we derive $sX(s) = (G(s) - X(s))/t + U(s)/t$, which can be inversely transformed to the time domain as:

$$dx/dt = g(x(t))/t - x(t)/t + u(t)/t \tag{4}$$

Thus, we can achieve arbitrary dynamics in the form of: $dx/dt = f(x(t) + u(t))$, by scaling $u$ by $\tau$ and resolving connection weights matrix $w$ that optimize: $g(x(t)) = \tau f(x) + x$ as was described above.

Detailed description of NEF is given in [19], a recent overview is given in [14], and a focused description of dynamics in NEF is given in [20]. NEF is the foundation upon our entire model is built. Most importantly, it is utilized here to define neuromorphic oscillators, which constitute our fundamental component for direction sensitivity.

### 2.2. 2D oscillation model

A 2D oscillator, which alternates the values of $x_1$ and $x_2$, at a rate $r$, can be defined recurrently using:

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 & r \\ -r & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = Ax \tag{5}$$

When initial values to $x_1$ and $x_2$ are zeroed, the oscillator stands still. When a stimulus is applied, the oscillator is driven to oscillate indefinitely.

A dumped oscillator can be defined by introducing $\lambda$ as a dumping factor (see Section 3.2 and 3.5):

$$\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = (A - \lambda I)x \tag{6}$$

Where $I$ is the identity matrix. Below, we use Eq. (4) to optimize Eqs. (5) and (6) which describe oscillatory dynamics, in order to resolve the connectivity weighted matrix $w$.

### 2.3. Gabor functions

Gabor functions are filter kernels inspired from the primary visual cortex (V1) which can be used to extract the locations and directions in space where the utmost changes in intensity appear [21].

They are mathematically interpreted as an exponentially decaying 2-dimentional sinusoidal wave and are specified with $\sigma$ (Gaussian width), $\zeta$ (ellipticity), $\theta$ (orientation), $\kappa$ (wavenumber) and $\phi$ (phase). The Gabor filter is defined with:

$$G(x,\ y) = exp\left(x_d^2 + \zeta^2 y_d^2 / 2\sigma^2\right)\cos(2\pi\kappa x_d + \phi) \tag{7}$$

Where, $x_d = x\cos(\theta) + y\sin(\theta)$ and $y_d = y\cos(\theta) - x\sin(\theta)$. The Gabors orientation and phase set the perceived edge orientation and its movement direction. This formulation is similar to the elliptic 2D Gabor function proposed in [21], with the exception of formulating the decay parameters (originally defined separately for each spatial dimension) as a Gaussian width. This simpler form is

Please cite this article as: E.E. Tsur and M. Rivlin-Etzion, Neuromorphic implementation of motion detection using oscillation interference, Neurocomputing, https://doi.org/10.1016/j.neucom.2019.09.072

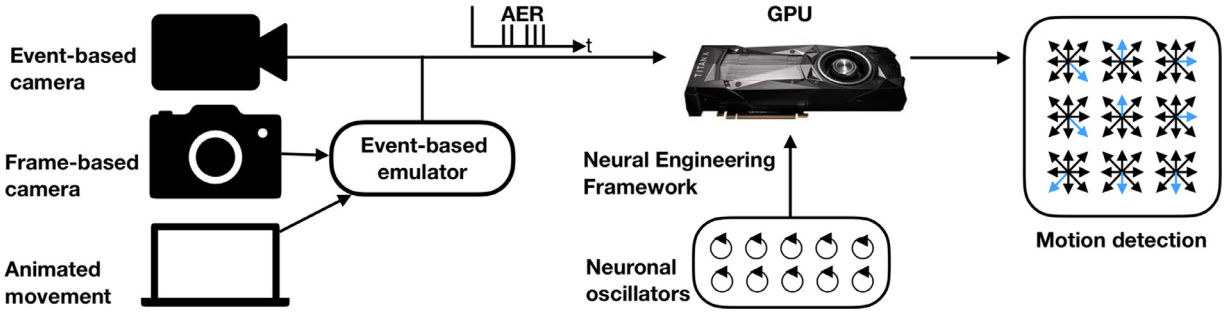



**Fig. 1.** System schematic. Our GPU deployed oscillators-based neuromorphic algorithm generates motion detection signals from asynchronous visual stimuli. Stimuli can be generated by either a commercial event-based camera or using an emulator. The emulator can convert regular frame-based camera or an animation file inputs to a neuromorphic event-based visual representation. The system reports on motion in the visual field, and if motion occurs it also specifies its direction.

sufficient for symmetrical Gabor functions which are decaying similarly in both dimensions.

Here, the direction of motion was always perpendicular to the Gabor's orientation, so for a given orientation there are only two possible moving directions. The selectivity for edge size can be optimized for various shapes and is set by the Gabor's wavenumber (which is the inverse of the sinusoidal wave length – its frequency).

### 2.4. The neural ENGineering objects simulator

We used Neural ENGineering Objects simulator (Nengo) [16] to simulate our model, as it efficiently facilitates neuromorphic implementation. Here we utilized Nengo 2.0, which was implemented with Python and supports GPU acceleration and a graphical user interface. Particularly, we used the following Nengo's objects: The *Ensemble* object to specify visual encoding and to implement neuronal oscillators; the *Node* object to introduce spikes-encoded frames and to evaluate the Gabor functions; and the *Connection* object to describe connections between ensembles, particularly, the feedback connections that are required for oscillatory behavior. In Nengo, the neural simulator is decoupled from model creation, allowing *separation of concerns*. We used Nengo's *Model* object to simulate our model. Elaborated description of Nengo is available in [16].

### 2.5. GPU computing

Graphics Processing Units (GPUs) are powerful, low-cost computing resources, which are available in most modern workstations, and are programmed using the OpenCL framework [22]. Nengo was implemented to support GPU implementation utilizing PyOpenCL [23]. It therefore permits to parallelize computations over GPUs, accelerating the model simulation and approaching real-time. The OpenCL-based Nengo simulator can simulate large neuronal models at least an order of magnitude faster than with a CPU-based hardware. Particularly, it was shown that, a Radeon 7970 GPU performs 500-dimensional circular convolution with about half a million neurons faster than real time, and 50 times faster than with CPU-based hardware. In the 50-dimensional case, the Radeon 7970 GPU is 200 times faster. Here, we used NVIDIA TITAN Xp to accelerate the model. It is based on NVIDIA's pascal architecture, has 3840 CUDA cores operated at 1.5 GHz and a 12 GB memory with a bandwidth of 547 GB/s.

### 2.6. Dynamic visual sensor

We used three types of visual modalities as inputs: animated patterns, regular frame-based camera and a Dynamic Visual Sensor (DVS). Particularly, we used a $128 \times 128$ pixels DVS, developed by the Neuroscientific System Theory group at the Technical University of Munich. This DVS is an asynchronously address-event camera, responding to temporal contrast by generating a spike address, which represents a quantized change of log intensity at a particular location. While the camera's protocol discriminates between positive and negative changes in contract, here they were treated the same. It delivers time-stamped address events to the server via USB at a 1 $\mu$s level resolution, has a dynamic range of 120 dB, a contrast threshold mismatch of 2.1%, a bandwidth of 3 kHz and a power consumption of only 23 mW. A detailed description of the sensor is available in [24]. Details regarding the emulator and the generation of the animated patterns are given in the results section.

## 3. Results

### 3.1. System architecture

Our system is comprised of three main parts: (1) an asynchronous event based visual stimulus; (2) a neuromorphic motion detection algorithm, which was developed using the Neural Engineering Framework and deployed over a GPU; and (3) a results visualization module. Inputs to our algorithm are not restricted to a certain format but rather can arrive either via a commercial spike-based camera, or via a frame-based camera or computer animation which are translated to spike sequences by our emulator. In this work, we developed an emulator that converts visual input arriving from either an off-the-shelf camera or an animated pattern to spiking activity. System schematics is given in Fig. 1.

Most spiking cameras, as well as the spike-generating emulator we developed, communicate spikes using the AER protocol, in which events are represented by pixel coordinates. For example, if a change in the acquired illumination in a pixel indexed 23 at time stamp 2 (frame) is detected, the string '23, 2′ is delivered (Fig. 2, A, top). Our emulator is parametrized with a frame rate (FPS) and number of pixels. As the number of pixels increase, the more accurate and sharper the resulted image would be (Fig. 2, A, bottom). For example, to emulate the DVS spiking camera (described in the method section), we would parameterize our emulator with 16,384 pixels, and $10^6$ FPS. In a case where our input modality is 30 FPS (e.g. standard camera), our module interpolates the acquired frames to achieve the desired frame rate.

Our system includes a framework for the generation of visual drifting grating, a stimulus often used to measure the directional tuning of neurons [25]. Given a desired FPS and grating parameters (spatial frequency, temporal frequency, orientation and direction), our framework generates an animated video, which can be introduced to the spike-generating emulator. We used this framework extensively in this work, generating and evaluating grating patterns moving in different directions and speeds and of various frequencies. Our emulator logs the generated events in a .dat file,

Please cite this article as: E.E. Tsur and M. Rivlin-Etzion, Neuromorphic implementation of motion detection using oscillation interference, Neurocomputing, https://doi.org/10.1016/j.neucom.2019.09.072

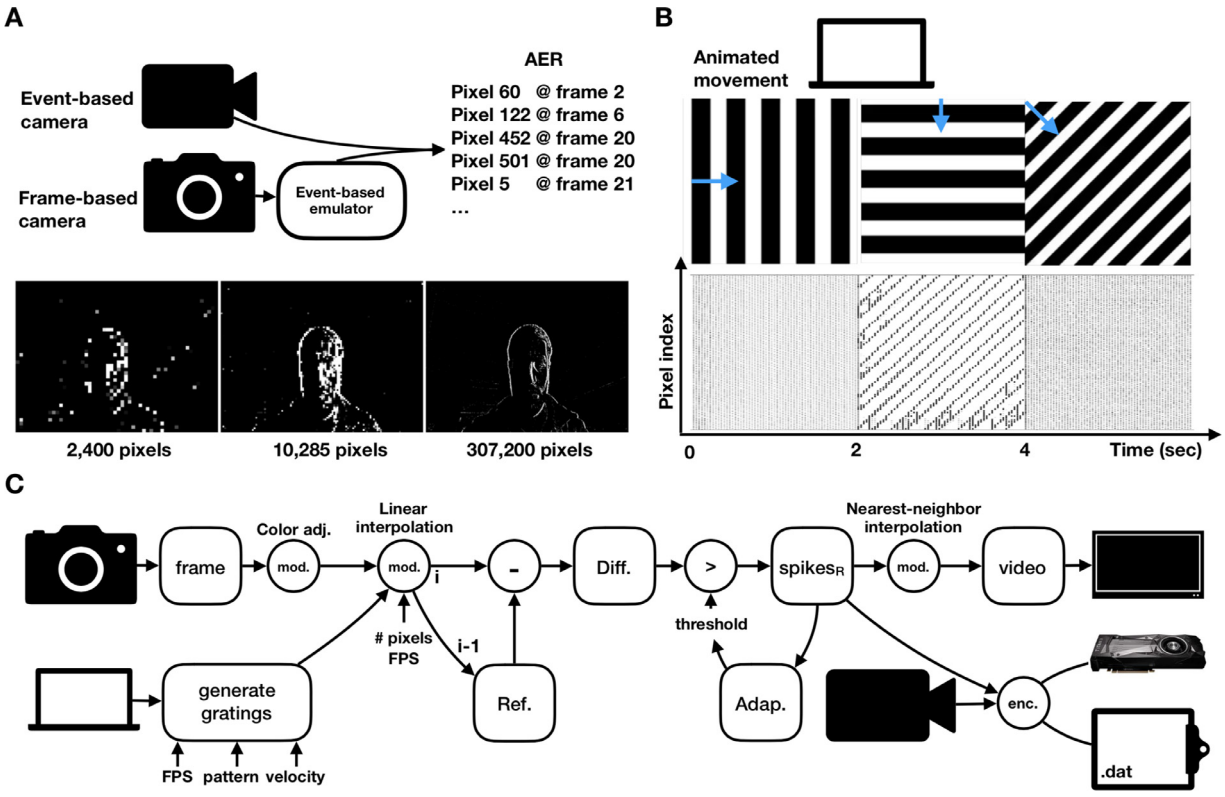



**Fig. 2.** Visual stimuli. (A) Stimuli is encoded by a combination of pixel location and time-stamp (constituting Address-Event Representation (AER)) and can be generated by either a commercial event-based camera or an event-based emulator. The emulator can produce AER data in varying rates and resolutions (number of pixels) from a standard frame-based camera as well as from animated patterns of motion. (B) Examples of different grating patterns, drifting horizontally, vertically and diagonally for two seconds. This animation is introduced to the emulator, which generates spikes in response, depicted in the pixel raster plots below. (C) Emulator schematic. Round boxes are data objects and circles are operations. Frames are sequentially modified for histogram equalization and converted to gray color scale. Giving the required resolution in space (# pixels) and time (Frames Per Seconds (FPS)), data is interpolated. Each frame *i* is pixel-wise subtracted from its predecessor *i-1* (Ref.), producing a differentiated frame (Diff.). Each pixel in the Diff frame is compared to a threshold value. If the threshold was exceeded, a spike is generated ($spikes_R$). The threshold can be adapted to generate responses with varying sensitivities to changes in luminance. $Spikes_R$ can be interpolated in space to fit a video feed or encoded (enc.) according to the AER protocol. Encoded data are logged and sent for analysis.

which can be an input to our algorithm for evaluation or can later be used to generate pixels raster plots (Fig. 2, B).

Our emulator was designed with multiple computational and representational states, supporting frame-based camera or animated patterns. Schematics is given in Fig. 2, C. If the visual source is a standard frame-based camera, frames are sequentially undergoing histogram equalization and gray scale color conversion and introduced to a modification kernel, where data is discretized given a number of pixels. Animated patterns are directly introduced to the discretizing kernel. To generate the spiking pattern, sequential frames are subtracted, and if the difference exceeds a certain threshold, a spike is generated. Threshold value is automatically adapted to support emulating cameras with varying sensitivities. Spikes are logged, time stamped and introduced to our GPU-powered neuromorphic algorithm for detection of motion direction. At the same time, spikes are interpolated and visualized on a computer. The emulator was primarily developed using the OpenCV package [26].

### 3.2. The oscillation interference model

As was shown in the *methods* Section 2.2, a 2D oscillatory pattern in which the values of $x_1$ and $x_2$ alternates at a rate $r$ can be defined using Eq. (5), and a dumped 2D oscillatory pattern can be defined by introducing $\lambda$ as a dumping factor Eq. (6)). These oscillatory patterns can be characterized with NEF (*methods 2.1*). Particularly, Eqs. (5) and ((6) can be implemented in NEF to resolve the connection weighted matrix $w$, optimized as g(x) (Eq. (4)).

Such a circuit comprises of a stimulation node, which introduces an initial stimulus and recurrently connected neuron ensemble, which calculates Eq. (6). In Fig. 3, A we defined a stimulation node, which introduces a signal at $t = 0$ which induce the oscillator to oscillate indefinitely - all neurons oscillate at rate $r$, with each neuron's preferred direction was locked to a different phase of the oscillator (see supplementary video $m_1$).

In Fig. 3, B we defined the same oscillating architecture as in Fig. 3, A, with the exception of applying a dumping factor of 5. After signal induction the oscillator begins it descend toward the origin. However, we can introduce a second orthogonal stimulus, at the exact moment when the oscillator arrives to a point where $x_1$ is zeroed. The oscillator will bounce back into an oscillating mode, soon to be dumped back to the origin. In Fig. 3, C, we created the same neuronal architecture as in Fig. 3, B, where we introduce a second orthogonal stimulus of at $t = 0.3$ s. The network behavior depends on the number of neurons in the ensemble and the dump factor $\lambda$. As we use less neurons for signal representation, the oscillatory path is gaining biases toward randomized locations. For $\lambda > 1$ the oscillator dumps to 0, for $\lambda < 1$ it converges to lower amplitude oscillations, and time to convergence increases as $\lambda$ decreases (Fig. 3, D). Below, we define a dumped oscillator, which oscillates as long as movement is detected at a specific orientation. Based on the dumped oscillators, we design a 3-dimensional array of motion-sensitive oscillators which are direction selective, i.e., each oscillator encodes motion in one specific direction by preserving oscillatory behavior only when exposed to motion in this specific direction (termed the preferred direction).

Please cite this article as: E.E. Tsur and M. Rivlin-Etzion, Neuromorphic implementation of motion detection using oscillation interference, Neurocomputing, https://doi.org/10.1016/j.neucom.2019.09.072

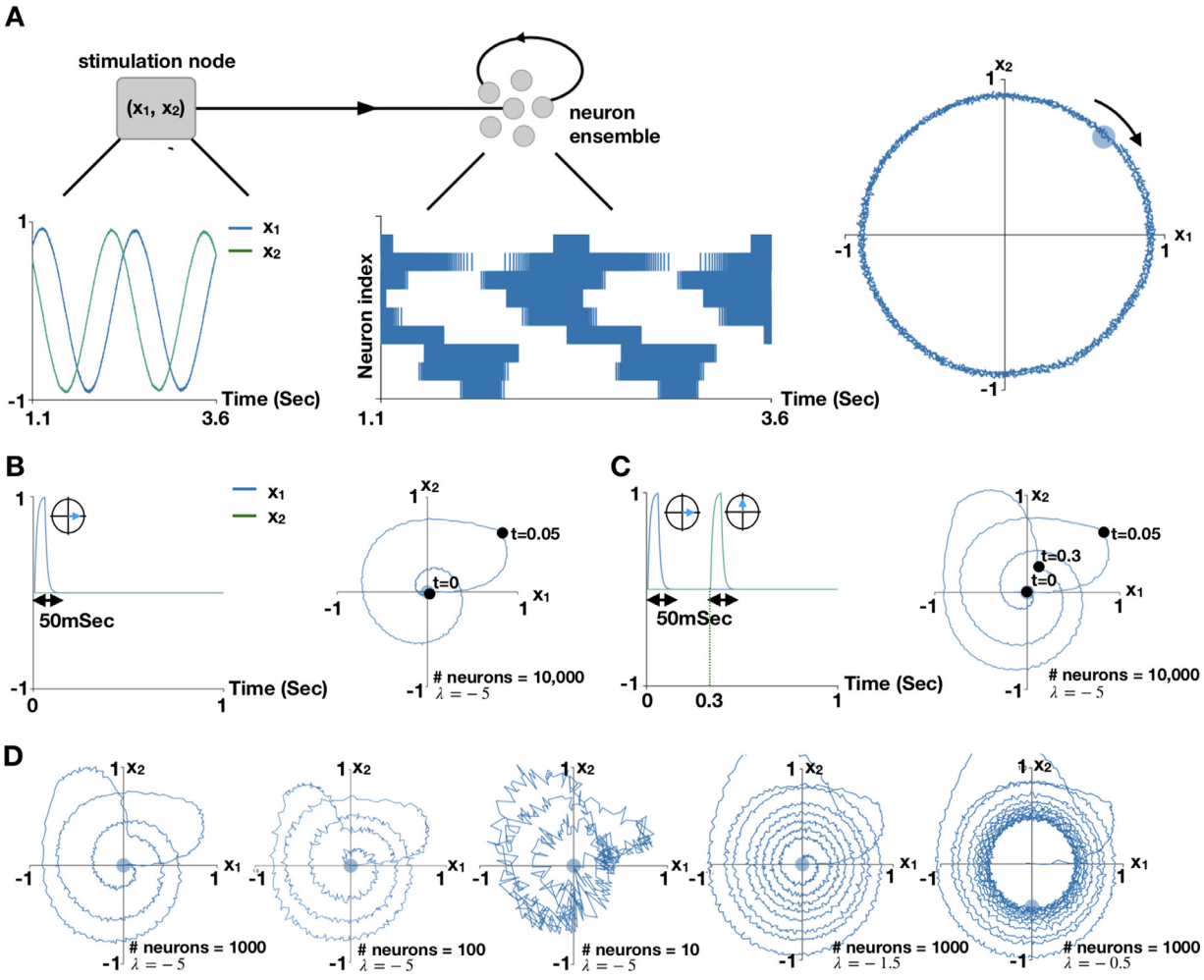



**Fig. 3.** NEF-based neuro-oscillators. (A) oscillatory behavior can be generated using self-feeded neuron ensemble which was optimized via NEF to oscillate the value of two state variables ($x_1$, $x_2$). Oscillation is induced from a stimulation node, which introduces a signal of [0, 1] (thus, driving $x_1$) for 100 ms. A dumping factor of 0 allows the oscillator to permanently oscillate. The stimulation node is connected to an ensemble of 10,000 neurons via a slow synapse ($\tau = 100$ ms) (tuning curves are distributed randomly). (B) Same oscillator as in A, induced for 50 ms in $t = 0$. A dumping factor of 5 quickly ceases the oscillations. (C) Same damped oscillator as in B, introduced with a second stimulus of [0, 1] in $t = 0.3$ s, inducing the oscillator again. (D) Same induced damped oscillator as in C, with different number of neurons (# neurons) used to build the neuron ensemble, as well as different damping factors ($\lambda$).

### 3.3. Algorithm design

As was described above (methods 2.3), motion in the visual field can be represented using Gabor functions of different parameters. Seven Gabor functions with equally-spaced orientations (denoted by $\theta$) along the unit circle are shown in Fig. 4, A top. Here, we used each Gabor function as a spatial filter for selectively driving an oscillatory neuron when a visual pattern matches its precise orientation. A stationary edge will activate the appropriate oscillator, but it will quickly dump as long as it is not being induced at the appropriate time and direction. In order to obtain motion detection, each Gabor function was phase shifted at multiple phases, so a moving pattern will match a series of Gabor functions, and will sequentially activate a series of oscillatory neurons. Phase shifted Gabor functions at $\theta = 0$ are shown in Fig. 4, A bottom. In our implementation, we have assigned for each given orientation 80 phase shifted Gabor functions equally-spaced along the unit circle. In accordance, we defined each neuronal ensemble with 80 encoders aligned with the Gabor function.

When exposed to horizontal grating orientation, two oscillators are induced into continuous oscillatory behavior: the one which was initialized with $\theta = 0$ and the one which was initialized with $\theta = \pi$. All other oscillators are not induced into action (Fig. 4, B shows 6 out of 8 oscillators). In order to distinguish between the two directions of motion, we take advantage of the fact that the two activated oscillators are driven in different directions (clockwise/counter clockwise), as a result of the sequential activation of the neuronal ensemble according to the phase parameter. Using a neuromorphic differentiation circuit, we discriminate between movement in clockwise and counter clockwise directions (Fig. 4, C). The differentiator circuit is comprised from 3 ensembles ($dx_1$, $dx_2$, *theta*). The $dx_1$ and $dx_2$ differentiators are connected to $x_1$ *and* $x_2$ respectively and are defined with two types of connections: fast synapses ($\tau = 10$ ms) and slow ($\tau = 20$ ms) synapses. Connections of each ensemble are multiplied, where the result of $x_2$ is scaled by $-1$. Both ensembles are joined together at the theta ensemble, resulting in a positive value when the direction of the oscillation is clockwise, a negative value when it is counter clockwise, and zero when no oscillation is apparent. The *theta* ensemble was specified with 50 neurons and randomly distributed encoders. This implementation produces eight direction selective oscillators, each encoding one specific direction of motion.

For motion detection we create an array of direction selective oscillators, each continuously oscillates only when a motion is detected at a specific direction. This produces a point process, se-

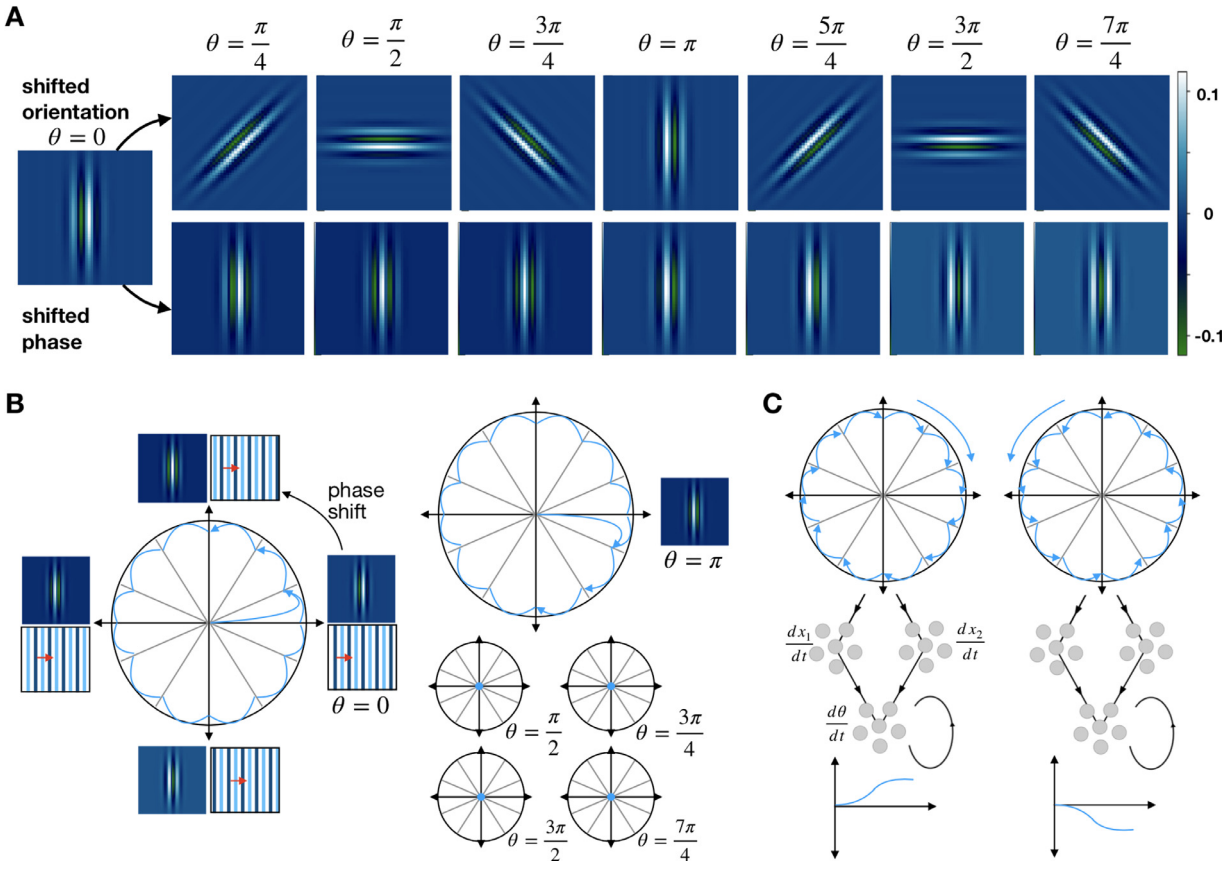



**Fig. 4.** Inducing direction selective oscillations with Gabor functions-mediated inductions. (A) visualization of a vertically aligned Gabor function ($\theta = 0$), which was shifted in orientation (top images) and phase (bottom images) in $\pi/4$ intervals. (B) Induction of Gabor functions results in a continues oscillation in response to one specific orientation. Example of 6 oscillators, each defined with a Gabor function in one specific orientation, which was phase shifted and distributed along the unit circle to match a moving stimulus edge. For a given horizontal visual stimuli, two out of the six oscillators oscillate ($\theta = 0$, $\theta = \pi$) – each in different direction (counter clockwise, clockwise). (C) Discrimination between oscillation direction is established using a differentiation neuronal circuit, resulting in positive or negative value for clockwise or counter clockwise oscillation respectably. The $dx_1$ and $dx_2$ differentiators were connected to $x_1$ *and* $x_2$, respectively, with fast ($\tau = 10$ ms) and slow ($\tau = 20$ ms) synapses and a scaling factor of $-10$ and 10, respectably. Result of $x_2$ was scaled by $-1$.

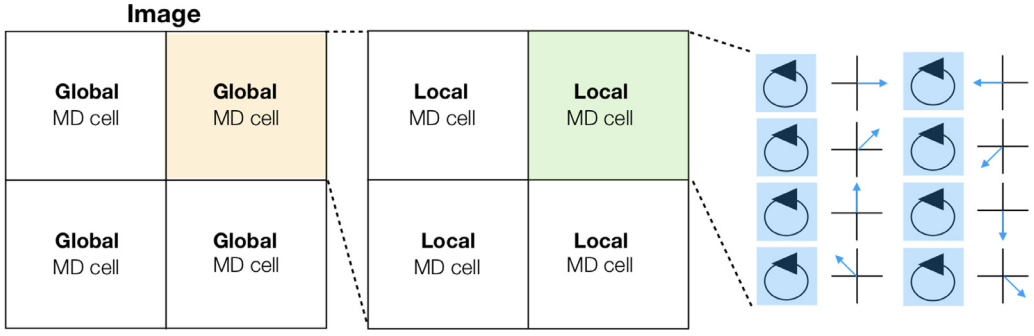



**Fig. 5.** Tiled architecture of the direction sensitive MD cells. Each oscillator has its own preferred direction. Every 8 oscillators constitute a local motion detector (MD) cell. Each 4 neighboring local MD cells comprise a global MD cell, which are evenly distributed over an image.

lective to local motion. To scale this architecture up, we defined each 8 oscillators (or point process) with different preferred directions as a local motion detector (MD) cell – which detects motion in a specific location. Each 4 neighboring local MD cells comprise a global MD cell (Fig. 5). MD cells can be distributed over a grid in varying resolutions according to requirements and available resources (neuron capacity). Each MD cell sums the contributions of each group of direction selective oscillators, multiplying their total activity with the relevant direction vector, thus, representing a global notion of directionality. Our system enables automatic distribution of Gabor functions across the image, according to the number of MD cells defined by the developer.

### 3.4. Model specifications

Our model is comprised of a hierarchical layers-based structure of interacting components. At the bottom layer we defined 2-dimensional 8 neuronal ensembles, or the direction selective oscillators. Each oscillator comprised of 80 neurons, which were specified to encode 80 normalized vectors with similar orientation and different phases equally distributed along the unit circle. All neurons in the ensemble were defined as LIF neurons with 100 Hz maximal firing rate, and 0.1 intercepts (which correspond to the inflection point of each neuron tuning curve). Our input spikes (initiates by the spiking camera/emulator giving a change in luminance)

Please cite this article as: E.E. Tsur and M. Rivlin-Etzion, Neuromorphic implementation of motion detection using oscillation interference, Neurocomputing, https://doi.org/10.1016/j.neucom.2019.09.072

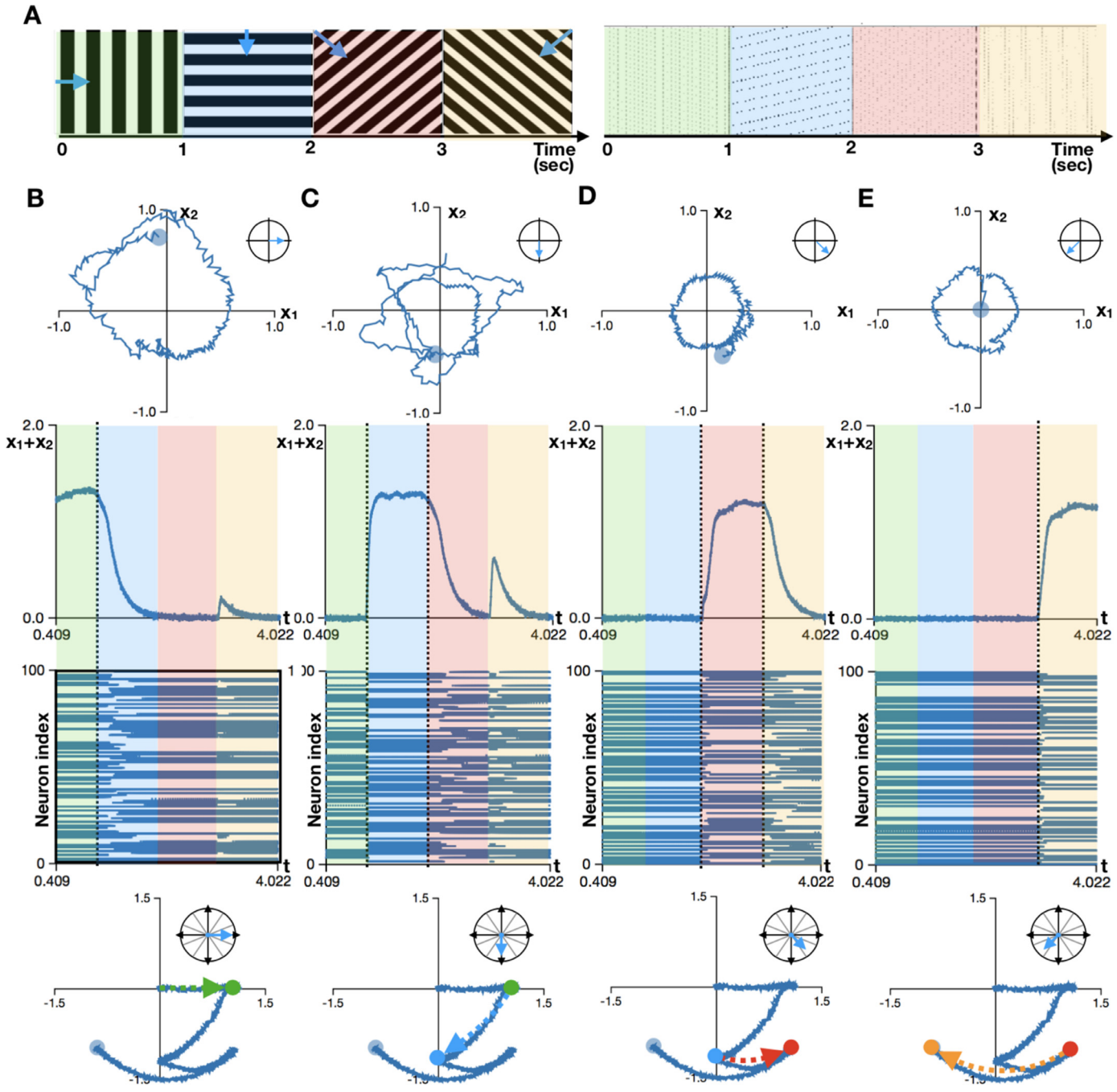



**Fig. 6.** Global MD cell response to drifting grating. (A) Stimuli used for the evaluation, where drifting direction is changing every second. Directions are color coded: green: East, blue: South, red: South-East, orange: South-West. Responses for each stimulus is shown in (B–E) respectively. Responses were recorder from a single global MD cell, which was placed at the middle of the field of view. Direction selectivity of the shown oscillators were respectively tuned to [0, 1], [1, 0], [1, 1] and [−1, −1]. For each direction, the oscillation dynamics as a state chart and as a traced sum are presented at the figure's top. At the bottom, the neuro-oscillator raster plot as well as the response of the local MD cell are shown.

were multiplied by the relevant Gabor function, scaled down by 0.2 and connected to their corresponding neuronal ensemble. Each ensemble was connected to itself for the generation of oscillatory behavior, through slow synapses ($\tau = 100$ ms). Each oscillator was specified with: $\lambda = -8$ and $r = 2\pi$. Discriminating left and right oscillations was performed via a differentiating neural circuit.

### 3.5. Motion detection

As a proof of concept, we used our stimulator engine to produce a movie of drifting grating in different directions (Fig. 6). Our emulator was utilized to produce DVS-like spiking pattern. In this illustration, a single global MD cell was defined at the middle of the field of view. During the first second, the grating pattern moved right, inducing the Gabor oscillators, which were tuned toward [0, 1] to oscillate continually (Fig. 6, B top). The values of both dimensions are summed, generating a stable step response, which indicates motion detection at this particular direction. Traced value, as well as a raster plot of the spiking ensemble are shown in Fig. 6, B middle. The response of the MD cell is shown in Fig. 6, B bottom, indicated by the green arrow. The MD cell sums the response of all oscillators, deviating motion at the [1,

Please cite this article as: E.E. Tsur and M. Rivlin-Etzion, Neuromorphic implementation of motion detection using oscillation interference, Neurocomputing, https://doi.org/10.1016/j.neucom.2019.09.072

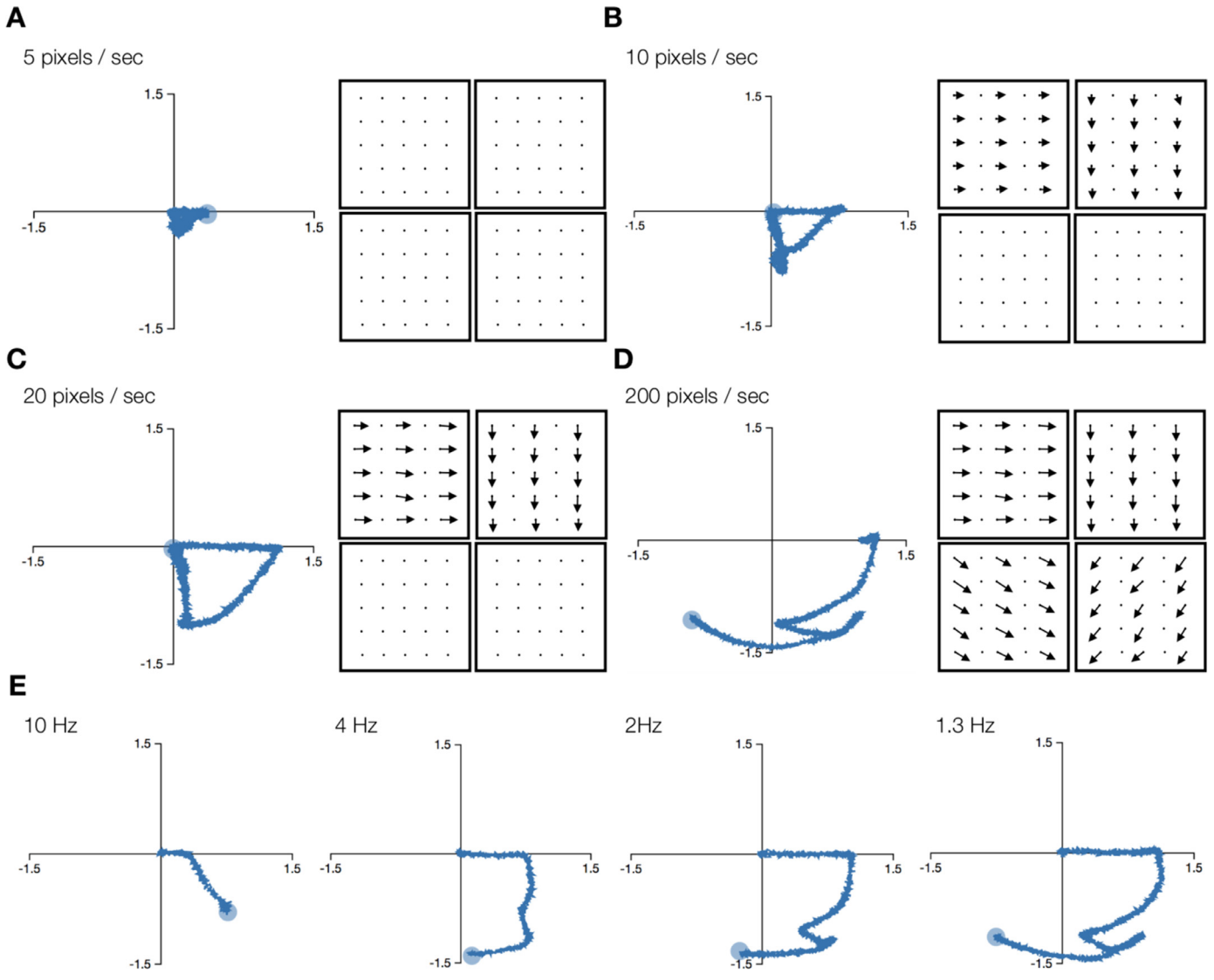



**Fig. 7.** Global MD cell response to drifting grating. $5 \times 5$ global MD cells were evenly distributed across the image. Same visual stimuli were used here as in Fig. 4, only in varying velocities and frequencies. (A–D) pattern drifting times are 5, 10, 20, and 200 pixels/s respectively, gained by moving 2.5%, 5%, 10%, 100% of the gratings image width in 1 s intervals. In each panel, the state space of the middle global MD cell is shown on the left, and an arrow field, were each dot/arrow signifies the detected motion direction of one localized global MD cell. (E) Applying the same stimuli in different pattern switching frequencies ranging from 10 to 1.3 Hz, gained by moving 10%, 25%, 50%, 75% of the grating image width in 0.1, 0.25, 0.5 and 0.75 s, while preserving drifting velocity of 200 pixels/s.

0] direction. Once the movie switches to a different stimulus, such as horizontal lines moving downwards, the oscillator tuned toward [1, 0] dumps to [0, 0] (as it is indicated by its value trace), and the oscillator tuned toward [0, −1] starts oscillating. In response, the MD cell shifts its detected motion value toward [0, −1] as is reflected in Fig. 6, C bottom, indicated by the blue arrow. The process is similar for the diagonally moving bars (Fig. 6, D–E). Note that when stimulus is switching to diagonally moving bars, the vertically aligned oscillators start to briefly oscillate. This is due to their response to the vertically moving component of the motion. However, their oscillatory behavior is quickly dumped, as the oscillatory induction is not persisted at their tuned direction. This has little effect on the MD motion detection, as is exemplified in Fig. 6, D–E, bottom. MD cell dynamics are shown in supplementary video m2. This shows that neuromorphic oscillators can be utilized to detect the direction of a moving visual stimulus.

We have elaborated this proof of concept to a $5 \times 5$ grid of MD cells, which were evenly distributed throughout the field of view. We utilized it to evaluate the speed range and frequency tolerance of our method. Two main factors in our framework limit the range of velocities for which our proposed method is relevant for: the oscillation speed (r) and the Gabor wavenumber ($\kappa$). Here, we tuned r and $\kappa$ to maximized at a speed of 200 pixels/sec. First, we used four different speeds for the visual stimuli, from motion over 2.5% of the image width in 1 s intervals (5 pixels/s) for the low velocity motion, via 5% and 10% of the image width (10 and 20 pixels/s respectively), to motion over 100% of the image width (200 pixels/s) for the high velocity motion. When velocity was at 5 pixels/s, the MD cell did not perceive movement. When velocity increased to 10 pixels/s the MD cell perceived horizontal and vertical movements with a normalized amplitude of 0.75. When velocity increased to 20 pixels/s the MD cell was able to detect horizontal and vertical movements with an amplitude of 1.25. When velocity increased to 200 pixels/s, diagonal movements were also perceived (Fig. 7, A–D). We further increased the stimulus speed to 2000 pixels/s and were able to capture movement in all directions, demonstrating very high range of tolerance to velocity. For each velocity we traced the value of each MD cell ensemble at the relevant directions. Re-

Please cite this article as: E.E. Tsur and M. Rivlin-Etzion, Neuromorphic implementation of motion detection using oscillation interference, Neurocomputing, https://doi.org/10.1016/j.neucom.2019.09.072

sults are shown in supplementary figure S1. It is apparent that the voltage traces are stabilized when the pattern velocity matches the designed oscillators. Results showing the voltage traces as well as a raster plot for each relevant ensemble are given in supplementary figure S2. This shows that our proposed algorithm is robust and can encode the direction of motion over a broad range of velocities.

Next, we tested out framework ability to detect motion with varying spatial frequencies. First, we created a stimulus where each pattern of movement travels 10% of the image width in 0.1 s (10 Hz). Spatial frequency was then increased to 25% of the image width in 0.25 s (4 Hz), to 50% of the image width in 0.5 s (2 Hz), and finally to 75% of the image width in 0.75 s (1.3 Hz). Note that in all cases, velocity is preserved at 200 pixels/s. When frequency was at 10 Hz, the MD cells were not able to arrive at the correct dynamics. At 4 Hz, vertical and horizontal movements were captured, and at lower frequencies the entire dynamics in all directions were captured (Fig. 7, E). Voltage traces for each frequency were also captured and are shown in supplementary figure S3.

## 4. Discussion

Motion detection is fundamental for visual perception in both biological and computer vision. Established approaches for motion detection traditionally rely on frame-based visual modality, where each frame is sequentially analyzed. Such approaches include Heeger approach for motion estimation using spatio-temporal filters [27], as well as differential, matching, energy-based, and phase-based methods, reviewed by Barron and colleagues [28]. Another implementation by Aziz and colleagues, proposed to use the oscillatory interference model for motion detection [29]. However, all of the above were applied to frame-based cameras which discretely acquire the visual information as frames at a predetermined rate, conveying information from the entire field of view, regardless if it was changed or not. This results in increase in transmission power dissipation, channel bandwidth requirements, memory capacity and postprocessing power demands [12].

The rise of neuromorphic architectures, which provide both high computational capabilities and low power density, are of increasing importance, especially for visual applications. Therefore, developing approaches for motion perception in neuromorphic circuits are fundamental to the field. Neuromorphic architectures (as biology) has no concept of a frame. Currently, most neuromorphic event-based vision sensors communicate changes in luminance - they are comprised of an array of silicon neurons, each generates a spike in response to a change in luminance. Current state of the art approaches for neuromorphic motion detection are based on either latency encoding selectivity or spatiotemporal offsets between excitation and inhibition, tuned for directionality. For example, Abdul-Kreem and colleagues convolved biphasic slow and fast temporal filters with the luminance input signal to derive first order temporal derivative of the visual scene and retrieve motion estimation [30]. A similar approach was taken by Brosch and colleagues, which also implemented their solution over IBM's TrueNorth neuromorphic chip [31]. Giuliono and colleagues implemented an architecture in which excitatory and inhibitory post-synaptic current pulses where spatially arranged to generate a signal for a preferred direction [32]. They implemented their solution over FLANN neuromorphic chip. These methods, while proved useful for different applications, have limited velocity sensitivity and are hard to optimize. Moreover, they lack the implementation generality of the Neural Engineering Framework, which becomes an important neural compiler for neuromorphic circuits, such as the BrainDrop and Intel's Loihi chips.

Here we proposed an approach for motion detection as well as identification of the motion direction, which can be easily deployed on neuromorphic hardware using NEF. This method has a wide range of velocity sensitivity (up to 10 times the optimal specification) and does not heavily relay on the neuronal spatial organization and synaptic time constants. More importantly, once neuronal weights are optimized via NEF, it does not rely on real-time convolution with spatio-temporal filters, contributing to its usability and reliability. This current implementation is however, relaying on a large number of neurons, which were used to accurately represent the dynamics and values in the system. The accuracy costs in long run times of the simulation, despite having a GPU as an accelerator, and the run time is expected to improve over neuromorphic machines. For example, using the pre-described method in Section 2.5, run time was ~25x real-time, while in neuromorphic architecture, the expected run time would be ~1x real-time.

By integrating asynchronous cameras, NEF and event-driven algorithms, we have implemented a neuromorphic architecture for motion perception. Our neuromorphic interference oscillation-based motion detection method provides an important step in the growing efforts to use neuromorphic implementations and minimize energy exploitation while preserving computational quality.

## Declaration of Competing Interest

None.

## CRediT authorship contribution statement

**Elishai Ezra Tsur:** Conceptualization, Data curation, Visualization, Writing - original draft, Writing - review & editing. **Michal Rivlin-Etzion:** Conceptualization, Data curation, Visualization, Writing - original draft, Writing - review & editing.

## Acknowledgments

This work was initiated during a research summer program at the center for theoretical neuroscience, at the University of Waterloo. The authors would like to thank the faculty and students of the center for the discussions, and particularly to Dr. Terrence C. Stewart and Prof. Chris Eliasmith.

## Funding

This work was supported by research grants from the I-CORE (51/11), the Minerva foundation, the ISF foundation (1396/15), and the European Research Council (ERC-StG 757732), and by Dr. and Mrs. Alan Leshner; the Lubin-Schupf Fund for Women in Science; the Charles and David Wolfson Charitable Trust; and Ms. Lois Pope. E. E. T. was supported by the Dean of Faculty postdoctoral fellowship at Weizmann Institute of Science. M.R.-E. is incumbent of the Sara Lee Schupf Family Chair. The Titan Xp used in this work was generously granted by the NVIDIA Corporation.

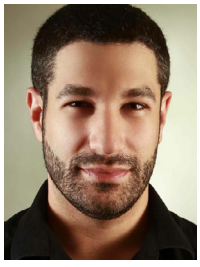
## Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.neucom.2019.09.072.

## References

[1] G. Johansson, Visual perception of biological motion and a model for its analysis, Percept. Psychophys. 14 (2) (1973) 201–211.
[2] A. Borst, T. Euler, Seeing things in motion: models, circuits, and mechanisms, Neuron 71 (6) (2011) 976–994.
[3] S. Kumar, P. Singhal, V.N. Krovi, Computer-vision-based decision support in surgical robotics, IEEE Des. Test 32 (5) (2015) 89–97.
[4] N. Sawasaki, M. Nakao, Y. Yamamoto, K. Okabayashi, Embedded vision system for mobile robot navigation, IEEE International Conference on Robotics and Automation (ICRA 2006), 2006.

[5] S. Saha, S.S. Bhattacharyya, Design methodology for embedded computer vision systems, in: Embedded Computer Vision, London, Springer, 2009, pp. 27–47.
[6] K. Boahen, Neuromorphic microchips, Sci. Am. 292 (5) (2005) 56–63.
[7] K. Boahen, A neuromorph's prospectus, Comput. Sci. Eng. 19 (2) (2017) 14–28.
[8] C. Mead, Neuromorphic electronic systems, Proc. IEEE 78 (10) (1990) 1629–1636.
[9] G. Indiveri, R. Douglas, Neuromorphic vision sensors, Science 288 (5469) (2000) 1189–1190.
[10] M.A. Mahowald, C. Mead, The silicon retina, Sci. Am. 264 (5) (1991) 76–82.
[11] K.A. Zaghloul, K. Boahen, A silicon retina that reproduces signals in the optic nerve, J. Neural Eng. 3 (4) (2006) 257.
[12] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, T. Delbruck, Retinomorphic event-based vision sensors: bioinspired cameras with spiking output, Proc. IEEE 102 (10) (2014) 1470–1484.
[13] A. Amir, P. Datta, W.P. Risk, A.S. Cassidy, J.A. Kusnitz, S.K. Esser, A. Andreopoulos, T.M. Wong, M. Flickner, R. Alvarez-Icaza, E. McQuinn, B. Shaw, N. Pass, D.S. Modha, Cognitive computing programming paradigm: a corelet language for composing networks of neurosynaptic cores, The 2013 International Joint Conference on Neural Networks (IJCNN), 2013.
[14] T.C. Stewart, C. Eliasmith, Large-scale synthesis of functional spiking neural circuits, Proc. IEEE 102 (5) (2014) 881–898.
[15] N. Burgess, C. Barry, J. O'keefe, An oscillatory interference model of grid cell firing, Hippocampus 17 (9) (2007) 801–812.
[16] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T.C. Stewart, D. Rasmussen, X. Choo, A. Voelker, C. Eliasmith., Nengo: a python tool for building large-scale functional brain models, Front. Neuroinform. 7 (48) (2014).
[17] A. Neckar, T.C.S.B.V. Benjamin, K. Boahen, " optimizing an analog neuron circuit design for nonlinear function approximation, IEEE International Symposium on Circuits and Systems (ISCAS), 2018.
[18] M. Davies, N. Srinivasa, T.-.H. Lin, G. Chinya, Y. Cao, S.H. Choday, G. Dimou, et al., Loihi: a neuromorphic manycore processor with on-chip learning, IEEE Micro 38 (1) (2018) 82–99.
[19] C. Eliasmith, C.H. Anderson, Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems, MIT press, 2004.
[20] C. Eliasmith, A unified approach to building and controlling spiking attractor networks, Neural Comput. 17 (6) (2005) 1276–1314.
[21] J.P. Jones, L.A. Palmer, An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex, J. Neurophysiol. 58 (6) (1987) 1233–1258.
[22] A. Munshi, The opencl specification, in: Hot Chips 21 Symposium, 2009, pp. 1–314.
[23] A. Klöckner, N. Pinto, Y. Lee, B. Catanzaro, P. Ivanov, A. Fasih, PyCUDA and pyopencl: a scripting-based approach to gpu run-time code generation, Parallel Comput. 38 (3) (2012) 157–174.
[24] J. Conradt, R. Berner, M. Cook, T. Delbruck, An embedded AER dynamic vision sensor for low-latency pole balancing, in: IEEE 12th International Conference in Computer Vision Workshops (ICCV Workshops), 2009, pp. 780–785.
[25] M. Rivlin-Etzion, W. Wei, M.B. Feller, Visual stimulation reverses the directional preference of direction-selective retinal ganglion cells, Neuron 76 (3) (2012) 518–525.
[26] G. Bradski, A. Kaehler, OpenCV, Dr. Dobb's J. Softw. Tools 3 (2000).
[27] D. Heeger, Optical flow using spatiotemporal filters, Int. J. Comput. Vis. vol. 1 (4) (1988) 279–302.
[28] J.L. Barron, D.J. Fleet, S.S. Beauchemin, Performance of optical flow techniques, Int. J. Comput. Vis. 12 (1) (1994) 43–77.
[29] A. Hurzook, O. Trujillo, C. Eliasmith, Visual motion processing and perceptual decision making, in: Proceedings of the Annual Meeting of the Cognitive Science Society, 2013.
[30] L.Issa Abdul-Kreem, H. Neumann, Neural mechanisms of cortical motion computation based on a neuromorphic sensory system, PLoS One 10 (11) (2015) e0142488.
[31] T.A.H.N. Brosch, Event-based optical flow on neuromorphic hardware, in: Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies, 2016.
[32] M. Giulioni, X. Lagorce, F. Galluppi, R.B. Benosman, Event-based computation of motion flow on a neuromorphic analog neural platform, Front. Neurosci. 10 (2016) 35.



**Elishai Ezra Tsur.** Elishai was a Postdoctoral Research Fellow at the Weizmann Institute of Science for Computational Neuroscience at Dr. Michal Rivlin lab and he is currently heading the Neuro-Biomorphic Engineering Lab (NBEL-lab.com) at the Jerusalem College of Technology; Dr. Ezra Tsur holds degrees in Life sciences (B.Sc, Open University of Israel), Philosophy and History (B.A, Open University of Israel), Computer Science (M.Sc, Interdisciplinary Center, Herzelia), and Bioengineering (M.Sc, PhD, Hebrew University of Jerusalem). His main research include neuromorphic vision processing and neuro-robotics.

**Michal Rivlin Etzion.** Michal Rivlin graduated with a BS in Mathematics and Computer Science from the Hebrew University of Jerusalem in 2001. She completed her PhD in the Interdisciplinary Center for Neural Computation at the Hebrew University in 2009. Dr. Rivlin was a postdoctoral fellow in the Department of Molecular and Cell Biology at the University of California at Berkeley from 2009 until joining the faculty of the Department of Neurobiology at the Weizmann Institute of Science in September 2013. Her main research interests are neuronal computations in the visual system, their mechanisms, and how they are transferred along the visual pathway.

Please cite this article as: E.E. Tsur and M. Rivlin-Etzion, Neuromorphic implementation of motion detection using oscillation interference, Neurocomputing, https://doi.org/10.1016/j.neucom.2019.09.072