Revista de Informática Teórica e Aplicada - RITA - ISSN 2175-2745
Vol. 26, Num. 3 (2019) 26-39

RESEARCH ARTICLE

A Model for the Diffusive Filling-In Algorithm Operating in Spike Mode

Um Modelo para o Algoritmo do Completamento e Preenchimento em Modo Pulsado

Genildo Nonato Santos[1]*, José Gabriel Rodriguez Carneiro Gomes[2]

Abstract: To run cortical circuit simulations in spike mode, i.e. taking into account the neural representation of information in terms of sequences of electrical pulses (also known as spikes), the use of customized hardware, which is specific for this purpose, is recommended. Simulations using more traditional hardware can be prohibitive. In this context, theoretical predictions are important for customized hardware design. For example, theoretical predictions lead to an adequate neuron model choice. To make such theoretical predictions, the cortical circuit simulations are carried out in amplitude mode. Differently from the spike mode, in amplitude mode information is represented by sequences of scalar values that describe neural input and output spike rates. In this paper, it was proposed amplitude and spike mode simulations of a cortical algorithm, namely the diffusive filling-in algorithm, to investigate whether predictions based on the amplitude-mode results approximate well the behavior of the customized hardware (spike mode results). The diffusive filling-in algorithm was chosen because it is simple enough for spike-mode simulation in a conventional computer, but the proposed amplitude-mode prediction method is the same for more complex algorithms or circuits. We provide a highly realistic comparison between amplitude-mode and spike-mode in the diffusive filling-in case, which suggests that the amplitude mode is reliable for theoretical predictions useful for customized hardware design for cortical circuit simulation. The goal of this paper is not to bring closure to these discussions but to suggest a way of avoiding possible issues that could compromise the success of the customized device design.
Keywords: diffusive filling-in — visual system — silicon retina — spike encoding

Resumo: Para realizar simulações de circuitos corticais em modo pulsado, considerando a representação de informação por meio de sequências de pulsos elétricos (também são conhecidos por spikes), o uso de um hardware dedicado, que são específicos para esse propósito, é recomendado. Simulações usando hardware mais tradicional podem ser proibitivas. Nesse contexto, predições teóricas são importantes na fase de projetos desse hardware. Por exemplo, tais predições permitem a escolha de um modelo de neurônio mais adequado. Para fazer tais predições teóricas, as simulações dos circuitos corticais são conduzidas em modo de amplitude. Diferentemente do spike mode, em modo de amplitude a informação é representada por sequências de valores escalares que descrevem as taxas de pulsos nas entradas e saídas neuronais. Neste trabalho, foi proposto simulações em ambos os modos de um algoritmo cortical, nomeado de completamento e preenchimento, para investigar se as predições baseadas nos resultados das simulações em modo de amplitude aproximam bem o comportamento do hardware customizado (resultados das simulações em modo pulsado). Esse algoritmo foi escolhido por ser simples o bastante para permitir simulação em modo pulsado em um computador com hardware convencional, considerando que o proposto modo em amplitude pode ser usado em casos mais complexos. Nós provemos uma comparação altamente realista entre ambos os modos para o caso do algoritmo de completamento e preenchimento que sugere que o amplitude mode é capaz de gerar predições teóricas suficientes para projetos de hardware específico para simulações de circuitos corticais em modo pulsado. O objetivo deste trabalho não é o de encerrar as discussões sobre esse assunto mas sugerir formas de evitar possíveis problemas que possam comprometer o sucesso do projeto de customização do dispositivo.
Palavras-Chave: completamento e preenchimento— sistema visual — retinas de silício — codificação por pulsos

[1] *Gestão da Produção Industrial, IFRJ, Brazil, genildo.santos@ifrj.edu.br*
[2] *Engenharia Elétrica, UFRJ, Brazil, gabriel@pads.ufrj.br*
***Corresponding author**: genildo.santos@ifrj.edu.br
DOI: https://doi.org/10.22456/2175-2745.86439 • **Received:** 04/09/2018 • **Accepted:** 11/08/2019

1. Introduction

Studying cortical circuits is important to understand the brain, but it also leads to inspiration for the development of efficient signal processing systems [1], [2]. The computational burden associated with cortical circuit simulation using conventional hardware becomes prohibitive, as the simulations start to address realistic neural circuits [3]. Spike neuron models are non-smoothing and highly non-linear functions [4], and therefore it is computationally expensive in computer simulations using more traditional methods. For these reasons, very realistic simulations of the large-neural networks, which are composed of spike neurons, could become prohibitive (in a traditional computer). For example, spike-mode simulations of cortical circuits maybe 9,000 times slower than the real-time behavior of the biological cortical circuits [5]. The development of customized hardware (neuromorphic), specific for cortical circuit simulation, has become popular among research centers and academic institutions [6]. The neuromorphic hardware is a customized semiconductor device which is implemented by millions of spike neurons (analog or digital microelectronic circuits that mimic the behavior of the model) [7]. Parallel and distributed processing characteristics of the neuromorphic hardware allow running simulations of cortical circuits in real-time. To design neuromorphic hardware, one must study either the behavior of the cortical circuit itself or of cortical circuits in general. A design like this is not generic for all types of the cortical circuit since neuron model parameters need to be drastically changed according to the simulated circuit. If one can analyze a cortical circuit before it is implemented in hardware, then the analysis may provide insight into the hardware implementation details. However, it is not simple to perform this analysis. There are some ways to conduct this preliminary cortical circuit analysis. Cortical circuit simulation may be carried out using high-performance computers (some models) [8], using *amplitude mode* [9],[10] or neglecting the preliminary cortical circuit analysis [6]. Supercomputers, FPGA (Field Programmable Gate Arrays) or GPU (Graphic Processor Units) grids are examples of the high-performance computers. In an amplitude-mode simulation, neural spike rates, also denoted as neural firing rates, are represented by scalar numbers organized into conventional discrete sequences. It allows that functions related to the neuron model become smoother and thus computationally less expensive. Therefore, an accessible starting point for cortical analysis is thus amplitude-mode simulation. However, by representing neural signals by conventional discrete sequences containing neural firing rates, the amplitude-mode simulation approach does not capture important features of neural dynamics such as the influence of spike timing on feedback loops, the influence of spike timing on the linear combination among the inputs of a neuron, the definition of an exact moment when a neuron fires (as an internally established threshold is reached), the exact behavior of a neuron after it has started to fire, and the influence of the spike shape on the signal processing results. It is therefore not obvious that these predictions, which are made based on amplitude-mode, to be realistic. There is a gap in the literature concerning academic studies deal with how realistic these predictions are. In this paper, we perform the implementation of a well-known cortical algorithm, the diffusive filling-in (DFI) [11], using the amplitude-mode approach and the spike-mode approach to assess the extent to which the obtained results are similar. A comparison between modes permits evaluating the error associated with predictions that were made in amplitude-mode. Thus, more accurate estimates about the expected behavior of the customized hardware can be achieved. And these estimates can help to identify problematic features of the design. A bio-inspired imager was built, in an electrical-simulation level, to generate the input stimulus for the implemented cortical algorithm model. Main electrical parasitic effects, which certainly would be present on the real device, were considered in this electrical simulations. It has provided a more realistic estimate. Section 2 briefly describes fundamental concepts that are needed for understanding cortical circuit behavior. Section 3 describes the DFI algorithm. This algorithm is involved in image reconstruction at the visual cortex. The DFI behavior (in conventional hardware implementation) has been previously assessed in spike-mode [9] and amplitude-mode [11] and, for that reason, we chose it for a comparison between spike-mode and amplitude-mode implementation of cortical circuits. The system-level numerical simulation results regarding amplitude-mode and spike-mode DFI implementations are presented in Section 4, and the conclusions of this work are presented in Section 5.

2. Visual System

In this section, we briefly describe the retina behavior, the signals that are generated by the retina, and the way these signals are processed at the early stages of the visual cortex. For that purpose, we start with a working model for the retina, including details about its input and output signals. Visual cortex signal processing is briefly mentioned and, after that, two neural models, namely amplitude-mode and spike-mode, are introduced. To show how these neural models are used for signal processing, an example is presented in Figs. 3a and 3b.

2.1 Retina Computational Model

We use the model introduced in [12] and [13]. It takes into account three signal processing features of the retina: spatial filtering, rectification, and conversion into a time-domain representation. The model input is a still image I_p and the model output is a pair of images, v_{On} and v_{Off}. The output images v_{On} and v_{Off} correspond to bandpass filtering in space and time domains applied to I_p. Our description considers four layers of the retina: photoreceptors, outer plexiform layer (horizontal cells), inner plexiform layer, and ganglion cells. Bandpass filtering in space and time domains: the interaction between photoreceptors and horizontal cells at the outer plexiform layer (OPL) of the retina was modeled in [14] by equations that describe electrical signal propagation in a net-

work composed by electrical cables. These networks have the space domain and time domain filtering property that is taken into account by the model. The model presented in [14] e and [13] is more adequate for a silicon implementation based on analog CMOS transistors. It is indicated in:

$$\begin{cases} I_c = I_p + \lambda_c \cdot \frac{\partial^2 I_c}{\partial x^2} - \tau_c \cdot \frac{\partial I_c}{\partial t} - k_{ch} \cdot I_h, \\ I_h = \lambda_h \cdot \frac{\partial^2 I_h}{\partial x^2} - \tau_h \cdot \frac{\partial I_h}{\partial t} + k_{hc} \cdot I_c, \end{cases} \quad (1)$$

The I symbol indicates electrical current that changes continuously as a function of position x and time t. Photoreceptor network activity is represented by I_c, where the subscript stands for a kind of photoreceptor known as *cone*. The OPL input is represented by I_p. The scalar constant λ_c describes the electrical conductivity of spatial connections between neighboring photoreceptors, and the scalar time constant τ_c represents the speed in photoreceptor response. Similarly, the symbols I_h, λ_h and τ_h represent analogous characteristics in the horizontal cell layer. The k_{ch} and K_{hc} constant gains indicate, respectively, amplification applied to variations of I_c at the horizontal cell network, and amplification applied to variations of I_h at the photoreceptor network. If the time-domain filtering is ignored, which means that only still images are used as retina inputs, then the operation implemented by Eq. (1) may be approximately represented by a *difference-of-gaussians* (DoG) operation [15]. If an input image I_p contains a single edge separating two regions, and the pixel values within each region are constant, then the retinal spatial passband filtering creates an output image that is mostly composed of zero values. Only pixels that are close to the edge (i.e. the separation boundary between the two regions) will have nonzero values. Positive pixel values are obtained at the side of the boundary corresponding to the higher input values, and negative pixel values are obtained at the other side. After the visual signal is processed by the OPL, the output I_c goes to the inner plexiform layer input.

Rectification: the positive and negative values in the I_c image are separated into exclusive channels, denoted as *on* channel for the positive values, and *off* channel for the negative values, for transmission into the subsequent visual system stages. This is a biologically efficient signal representation because it allows for negative signal representation without dynamic range loss, and it also saves energy by not have spike transmission in the zero-signal case [12]. To represent positive-valued pixels, we define the v_{On} image, and to represent negative-valued pixels, we define the v_{Off} image. At the inner plexiform layer of the biological retina, this operation is implemented by *bipolar cells* (not to be confused with *bipoles* or *bipole groups*, which will be described in Section 3). This operation is denoted as *rectification*. Besides rectification, the inner plexiform layer also filters high temporal frequency content out from the v_{On} and v_{Off} images, and prepares that content for transmission along specialized channels [15]. Since we only consider still input images I_p, we do not take the time-domain highpass filtering operation into account in this work. Next, the inner plexiform layer output signals v_{On} and v_{Off} are processed by the ganglion cells. As described next, the ganglion cells provide suitable time-domain encoding for the v_{On} and v_{Off} information.

Time-domain representation: ganglion cells convert the image representation from an amplitude-based representation into a time-domain, spike-mode representation. Instead of having a pixel value continuously represented in time (such as in the case of I_c, for example), at a ganglion cell output we have a neural spike sequence representing the pixel value. Pixel is individually encoded by spike sequences so that the time interval between two spikes (also denoted as inter-spike interval) is a function of their corresponding amplitude-mode pixel values. The ganglion cells form the last signal processing stage at the retina. The optical nerve takes the retina outputs to the visual cortex.

2.2 Neuron Models: Spike and Amplitude Mode

Several models for the biological neuron have been proposed in the literature. In the *integrate-and-fire* (IF) model [3], [4], for example, the neuron inputs are described by N continuous-time spike sequences $v_i(t)$, $i = 1, \ldots, K$. At the neuron, each input spike sequence is multiplied by a constant synaptic weight g_i, $i = 1, \ldots, K$. The K resulting spike sequences at added according to $s(t) = \sum_i (v_i(t,i) \cdot g(i))$, and the summation result is integrated according to $v(t) = \int s(t) \cdot dt$. Whenever the $v(t)$ integral value goes above an arbitrary threshold v_{th}, a spike is generated at the neuron output $v_o(t)$ and the $v(t)$ value is reset to zero. The integrate-and-fire model is not suitable for the implementation of signal processing operations containing positive feedback loops, which is the DFI algorithm case, because the direct path from the neuron input to the neuron output may contribute to the instability of the loop. To overcome this limitation, the integrate-and-fire model is enhanced by also taking into account a *refractory period*. The refractory period is a time interval during which, immediately after having fired, the neuron cannot fire again, even if $v(t)$ reaches the pre-established threshold. The integrate-and-fire model with a refractory period is useful for the implementation of signal processing operations containing positive feedback loops [3], [9].

Spike Mode: In the present work, we will use the *leaky integrate-and-fire* (LIF) model [4], which corresponds to the basic integrate-and-fire model improved by an adaptation mechanism that causes an effect similar to the effect obtained with the refractory period. The leaky integrate-and-fire model is similar to the basic integrate-and-fire model, except that the integrating variable $v(t)$ is leaky. In the LIF model, a neuron that does not receive any input (or, more generally, that receives an amount of input that is below the amount of leakage) will have its integration variable value reduced until it reaches a minimum value v_L. In [4] was applied the Euler integration method to a differential equation system including

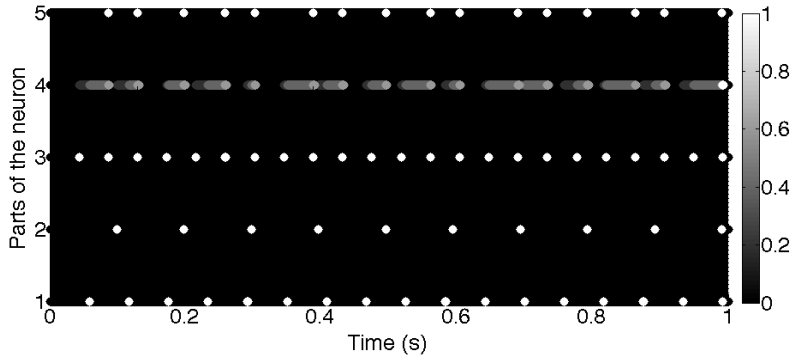


Figure 1. Illustration of the LIF model described by Eq. (2). At the vertical axis, labels 1, 2, and 3 indicate signals $v_i(n)$, $i = 1,2,3$. The label 4 indicates $v(n)$, and the label 5 indicates $v_o(n)$. In this example, we used $A = 0.9$, $B = 1$, $C = 0.1$, $D = 1$, $E = 0.2$, we have $v_{th} = 1$, $v_H = 1$, and $v_L = 0$.

leakage on $v(t)$, and was obtained the result shown in Eq. (2):

$$\begin{cases} v(n+1) = A \cdot v(n) + B \cdot v_i(n) - C \cdot v_a(n), \\ v_a(n+1) = D \cdot v_o(n) - E \cdot v_a(n), \\ if\ v(n) > v_{th},\ then\ v_o(n) \leftarrow v_H,\ and\ v(n) \leftarrow v_L, \end{cases} \tag{2}$$

In Eq. (2), the A, B, C, D and E positive constants must be adjusted according to the available data set. To obtain the results provided in Section 4, we used $A = 0.9$, $B = 1$, $C = 0.1$, $D = 1$, and $E = 0.2$. There are two discrete-time state variables ($v(n)$ and $v_a(n)$), where $v_a(n)$ causes adaptive leakage, which will lead to an effect similar to the one obtained with the refractory period. The neuron inputs are $v_i(n)$, $i = 1,\ldots,N$, and the neuron output is $v_o(n)$. An example of the model in Eq. (2) is shown in Fig. 1, for $v_{th} = 1$, $v_H = 1$, and $v_L = 0$.) The dotted line in Fig. 2 shows the output firing rate of a biological neuron located at the V2 layer of the visual cortex [16], plotted as a function of the firing rate of its single input. This input firing rate corresponds to the normalized contrast (0.5 contrast correspond to an input firing rate approximately equal to 120 spikes/sec) of an image to which the visual system of an animal was exposed. To perform the fitting of this relationship between the output firing rate and the input firing rate, we used the IF model and the LIF model. One can clearly note the difference between the IF and LIF behavior as the input firing rate increases, which suggests that the LIF model can capture the limitation in output firing rate that takes place in a biological neuron, as the neuron input firing rate increases.

Amplitude Mode: Because of the large computational effort required for the simulation of circuits composed by a large number of spike-mode neurons modeled by Eq. (2), the neuron model is replaced by a simplified model, denoted as *amplitude-mode* model, in which scalar values represent firing rates. The amplitude-mode neuron model does not have effective spikes at its inputs or at its output, but it describes nevertheless the firing rate at the neuron inputs or at its output, and it is able to handle approximate computations in that way. The neuron firing rate is defined as the summation of the number of spikes generated by the neuron within a time interval of t, divided by t itself. Not explicitly using spikes reduces the computational burden, which makes a difference between viable simulations, and simulations with prohibitive computational cost [9]. Eq. (3) shows an amplitude-mode model corresponding to the spike-mode model that was presented in Eq. (2). In Eq. (2), we also have A = 0.9 and B = 1.

$$\begin{cases} v(n+1) = A \cdot v(n) + B \cdot v_i(n), \\ v_o(n) = [\tanh(v(n) - v_{th})]^+, \end{cases} \tag{3}$$

At the visual cortex, biological circuits process visual information according to operations that are performed individually by the neurons and according to the connections established among those neurons. In the spike-mode simulation, the individual operations are implemented by Eq. (2).

Each connection corresponds to a synaptic weight placed between one neuron output and one input of another neuron. In the remainder of this paper, we will compare spike-mode and amplitude-mode implementations of edge detection operations on still images. Spike-mode implementations have the structure shown in Fig. 3a, and amplitude-mode implementations have the structure shown in Fig. 3b. In both block diagrams, $I(x,y)$ is an input image, v_{th} denotes a thresholding operation, and $g(x,y)$ is a set of synaptic weights. In spike-mode implementations the value of v_{th} is 1 and amplitude-mode implementation is 0.1. The synaptic weights describe a highpass filter such as $g(x,y) = [-1\ -1\ -1;\ -1\ 8\ -1;\ -1\ -1\ -1]$. In this work, the small blocks with input $v_i(n)$ and output $v_o(n)$ at the lower right corner of Figs. 3a and 3b are denoted as *neuron activation processing stages*, are these stages are indicated by an N symbol according to equations such as $v_o(n) = N[v_i(n), v_{th}]$. In the cases of Figs. 3a and 3b, we have $v_o(n) = N_p[v_i(n), v_{th}]$ and $v_o(n) = N_a[v_i(n), v_{th}]$, respectively.

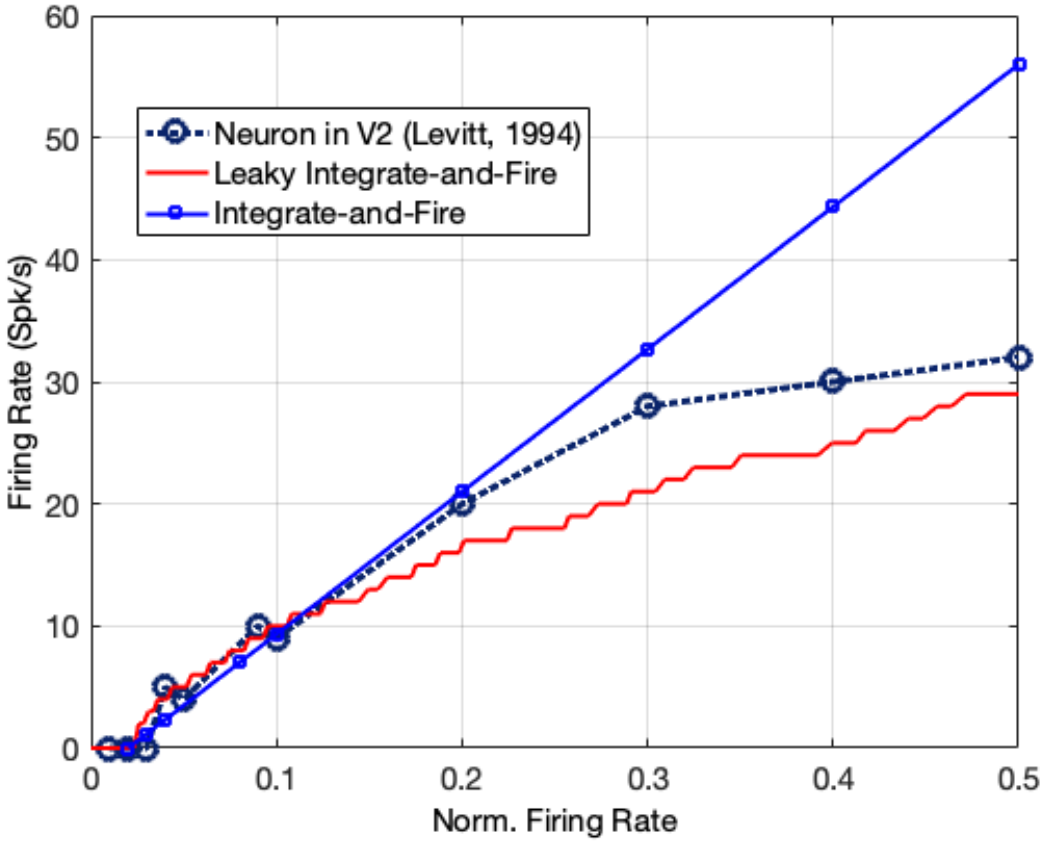


Figure 2. Output firing rate of a biological neuron (dotted line) as a function of an input firing rate corresponding to normalized contrast [16], and approximations of that function based on the integrate-and-fire model (solid line) and on the leaky integrate-and-fire model (line with square markers).

3. Diffusive Filling-In (DFI)

Diffusive filling-in is a signal processing operation that is executed by the visual cortex in order to completely fill-in finite and fully connected regions. Each region is fully surrounded by a closed contour. The block diagram of the DFI algorithm is shown in Fig. 4. The diffusive fill-in operation inputs are the still images v_{On} and v_{Off}, which are coming from the retina, and the output is v_{DFI}. In the v_{DFI}, fully connected regions inside closed contours are filled in.

Because of the space-domain filtering that takes place in the retina according to Eq. (1), the v_{On} and v_{Off} images contain contours. The contours may be closed imperfectly. The DFI algorithm uses the partial contour information available in v_{On} and v_{Off} in order to prevent that only the correct regions are filled in. In the visual system, a border between a closed region and its surroundings is described by two contours (inner and outer), which are not transmitted by the same visual channel. As a consequence, information fill-in may begin with the v_{On} contour and finish off with (or be blocked by) the v_{Off} contour, or the opposite may take place. In the remainder of this paper, we will use v_{On} as a reference for fill-in, and we will use v_{Off} as the blocking signal. Because of the retina biological structure, not all image samples are transmitted to the cortex via v_{On} and v_{Off}. Information is lost at specific pixel positions and the corresponding original sample values are replaced by zero. The artifacts created by the missing samples in v_{Off} translate into discontinuities along imperfect contours that would otherwise be fully closed. With missing samples, v_{Off} is obviously unable to block the fill-in operation over v_{On}. The contours in v_{Off} must be properly completed. To work with properly closed contours in v_{Off}, the DFI algorithm has a *contour completing* stage, which is the first stage in a set of two stages that are simultaneously executed: - In the first stage (contour completing), the v_{Off} image is processed so that zero-valued pixels located on gaps among neighboring parts of a to-be-completed contour take larger values, which are typically closer to the original pixel values. To achieve that result, neural activity is propagated across neighboring pixels according to properties, such as direction and offset, which are common to the neighboring pixels. The image v_{BCS} is the result of the signal processing operation over the input image v_{Off} that occurs in this stage.

- In the second stage (filling-in), Regions defined by v_{On} are filled-in by neural activity spreading among neighboring

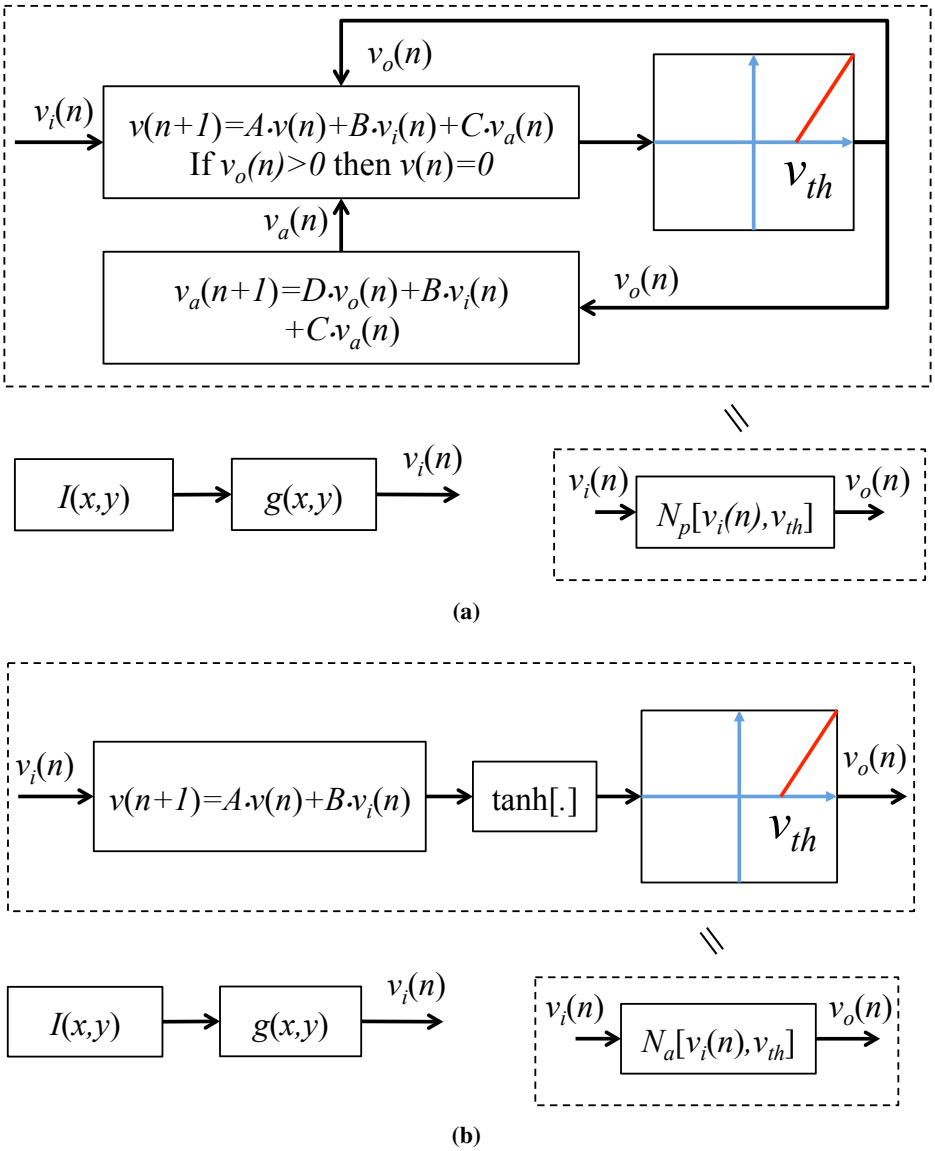


Figure 3. Spike-mode (a) and amplitude-mode (b) implementations of an edge-detection example. The input image $v_i(n)$ is obtained from the space-domain convolution between an input image $I(x,y)$ and a highpass filter $g(x,y)$ (also denoted as a set of synaptic weights). In the spike-mode implementation (a), the input image $v_i(n)$ is applied to difference equations according to the LIF model in Eq. (1). In the amplitude-mode implementation (b), the difference equations correspond to the IF model in Eq. (3)

pixels. For the v_{On} signal, it is expected that only pixels located on the closed contours will have nonzero values, and therefore the starting points for neural activity spreading are precisely those located on the estimated inner side of the closed contours. In the v_{FFI} image in Fig. 4, neural activity spreads among neighboring pixels along all directions having connected neighbors. To prevent neural activity from spreading indefinitely, the v_{Off} information is used. The expression *second stage* refers to the fact that the cortical layer in charge of the filling-in task is, in the visual cortex structure, hierarchically superior to the layer in charge of contour completing. Both layers process signals simultaneously. By completing and filling-in, the visual system recovers original low-frequency information that had been lost because of defects in the visual system or because of the retina space-domain passband filtering [17], [11]. Low-frequency information recovery is important because it significantly improves the perception of the faces of solid objects or, equivalently, the inner part of flat objects. More details about contour completing and filling-in will be provided next, in Section 3.1.

3.1 DFI Details

The $v_{DFI}(x,y)$ is obtained from $v_{DFI}(x,y) = N[v_{On}(x,y) + v_{FFI}(x,y), v_{th1}]$, as shown in Fig. 4. The initial value of $v_{FFI}(x,y)$ is zero, so the initial $v_{DFI}(x,y)$ contains approxi-

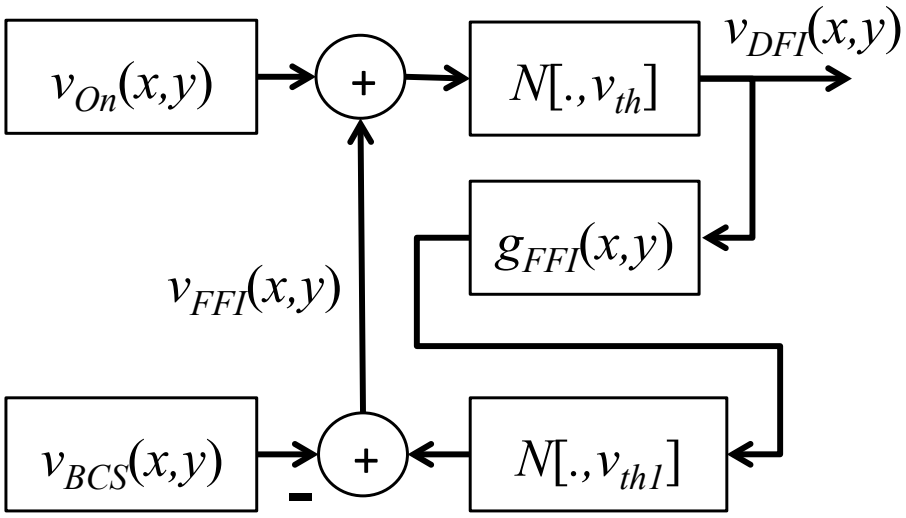


Figure 4. Diffusive filling-in algorithm block diagram (either spike-mode or amplitude-mode). To generate $v_{DFI}(x,y)$ from $v_{On}(x,y)$, $v_{FFI}(x,y)$ and v_{th}, we use one of the neuron activation processing stages indicated at the lower right corners of Figs. 3(a) or 3(b). Spatial convolution with $g_{FFI}(x,y)$ spreads neural activity from one pixel to its neighbors. To obtain feedback via a neural pathway, an additional neural activation processing stage prepares the convolution result for feedback, prior to subtraction with the boundary contour system (BCS, more details in Section 3.1) output signal $v_{BCS}(x,y)$. In amplitude mode, we have $v_{th} = v_{th1} = 0.01$. In spike mode, we have $v_{th} = v_{th1} = 1$. More details are provided in the text.

mately the same contours that are present in $v_{On}(x,y)$. After that, a space-domain convolution between $v_{DFI}(x,y)$ and $g_{FFI}(x,y)$ is computed. We use $g_{FFI} = [010;101;010]$. This convolution copies the value from one pixel at the $v_{DFI}(x,y)$ image to its four immediate neighbors (up, down, left and right), which represents the fact that contour information is spread among neighboring pixels. To represent the convolution result in terms of a neural signal (either amplitude-mode or spike-mode), we execute the operation $N[g_{FFI}(x,y) * v_{DFI}(x,y)]$, which corresponds to one of the neuron activation processing stages indicated at the lower right corners of Figs. 3(a) or 3(b). Positive feedback allows the active pixels in $v_{On}(x,y) + N[v_{DFI}(x,y) * g_{FFI}(x,y), v_{th1}] - v_{BCS}(x,y)$ to reinforce the corresponding pixel values in $v_{DFI}(x,y)$, as long as these active pixel values are large enough (i.e. larger than v_{th}). After undergoing these reinforcement operations, the $v_{DFI}(x,y)$ image contains its own original contours, and also pixels that are neighbors to this original contour. As the algorithm proceeds, all contours get thicker. If the filling process was not interrupted (based on the boundary contour system information, in this work), then the image would be entirely covered with active pixels. As we have stated before, we assume that the v_{On} information is used for filling-in, whereas the v_{Off} information is used to prevent neural activity from spreading beyond the closed contour surrounding the activity region. To interrupt the filling-in process, the v_{Off} image might replace the v_{BCS} image in being subtracted, pixel-by-pixel, from the image that is fed back in order to generate v_{FFI}. Such subtraction would generate negative values at the corresponding pixel positions on the corresponding contour in v_{Off} and, inhibited by those negative values at those positions, the neural activity would stop spreading. However, a biological visual system has natural anomalies present throughout the retina signal processing stages. These anomalies are usually created by the blind spot, and by blood vessels that partially cover photoreceptors and prevent them from receiving complete light input [18]. Because of these anomalies, some v_{Off} positions correspond to incorrect retinal readings. The v_{Off} signal may contain inactive pixel positions where activity should have been detected. The contour defined by v_{Off} may, in this case, be open, but it must be closed in order to block neural activity from spreading from within the contour to its outer side. We assume that the v_{Off} signal always contains at least one closed contour that was originally present in an image captured by the retina, although the contour may be defective or incomplete. The *boundary completion* operation is therefore required for keeping the contour closed. As it closes the contour (or more than one contour), the boundary completion operation recovers missing samples in v_{Off} (samples that were unavailable because of the blind spot, or samples that were corrupted because of any other natural defect of the visual system). The BCS (boundary contour system) is detailed in Fig. 5. It operates on v_{Off} and generates, from it, a new version of v_{Off} (in which all contours are closed), which is the v_{BCS} output image. As we mentioned before, the v_{BCS} output image will be used to interrupt the filling-in process. To recover missing/defective samples in v_{Off}, the BCS uses three operations: directional filtering, spatial competition, and bipole grouping.

Directional Filtering: At the directional filtering stage, an

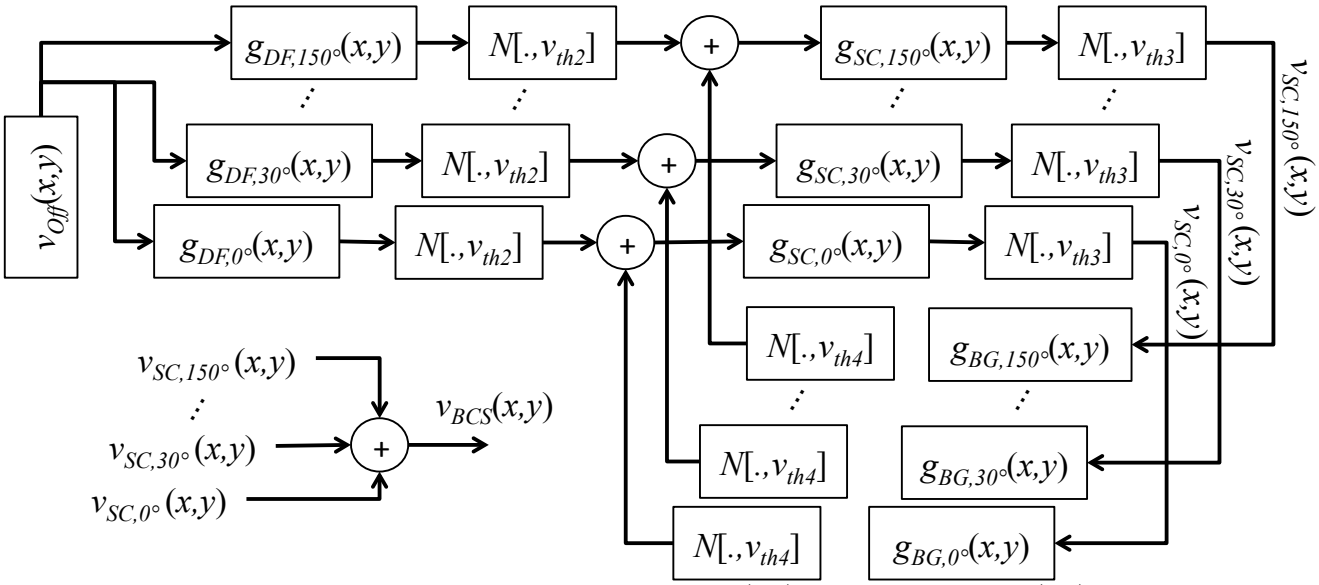


Figure 5. Boundary contour system block diagram. The input is $v_{Off}(x,y)$ and the output is $v_{BCS}(x,y)$. All operations are simultaneously performed along 0^o, 30^o, 60^o, 90^o, 120^o, and 150^o orientation channels. Convolution with DF (directional) filters followed by comparison against threshold v_{th2} tends to keep contour segments along particular orientations. Convolution with SC (spatial competition) filters followed by v_{th3} thresholding removes residual (weak) segments that may remain after previous filtering operations, also taking into account reinforced segments along the correct orientations. Convolution with BG (bipole grouping) filters connects contour segments that should be merged into the same original contour, although they may have been separated by missing samples or directional filtering. The BG outputs reinforce the SC results. The BCS output is obtained by adding all SC channels.

image described by input signal vOff is filtered by a Gabor filter bank [19]. Each Gabor filter $g_{FD,\theta}(x,y)$ corresponds to a specific direction θ_i, $i = 1,2,\ldots,N$, where N is the number of discrete directions taken into account by the visual system. In this work, we use $N = 6$, for the multiples of 30 degrees from 0 to 150 degrees. All Gabor filters are on the same scale. This operation generates one output image for each Gabor filter. In every output image, objects or closed contours will have their edges emphasized along the direction associated with the Gabor filter that generated that very image. Pixel values below an arbitrary threshold (v_{th2}) are set to zero. $v_{th2} = 1$ in spike-mode or $v_{th2} = 0.01$ in amplitude-mode are used in these implementations. This helps in eliminating residual segments from other contours not having a direction related to the associated Gabor filter, but a few residual segments remain at the output image after the thresholding. The directional filtering procedure is defined by Eq. (4).

$$v_{FD,\theta}(x,y) = \begin{cases} v_{Off}(x,y) * g_{\theta}(x,y), \\ \quad \text{if } v_{Off}(x,y) * g_{DF,\theta}(x,y) > v_{th2} \\ 0, \text{ otherwise.} \end{cases} \quad (4)$$

Spatial Competition: To further reduce the presence of residual contours, every image coming out from the thresholding after directional filtering is convolved with the filter $g_{SC} = [-1\ -1\ -1;\ -1\ 8\ -1;\ -1\ -1\ -1]$ and every pixel from the resulting image is compared with an arbitrary threshold v_{th2}. If the pixel value is lower than v_{th3}, then it is set to zero. Otherwise, the convolution result is kept. To describe a biological counterpart for this operation [17], we point out that the convolution and thresholding cascade association approximately corresponds to *spatial competition* among the center pixel value (times 8) and the summation of its eight neighbors. If the center pixel wins the competition, then its value is kept. Otherwise, its value is set to zero, which corresponds to information loss. Spatial competition thus leads to a simple implementation of a well-known edge detector, which is composed of a difference-of-Gaussians filtering operation followed by thresholding, $v_{th3} = 1$ in spike-mode or $v_{th3} = 0.01$ in amplitude-mode. As a result of spatial competition, weaker (residual) contours are eliminated, whereas stronger contours are emphasized. The spatial competition procedure is defined by Eq. (5), which also takes into account nonzero feedback $v_{BG,\theta}(x,y)$ from a subsequent operation denoted as BG, which stands for "Bipole Grouping". The bipole grouping operation will be defined next, but Eq. (5) may be understood at this point as having $v_{BG,\theta}(x,y) = 0$ for the description that was presented.

$$v_{SC,\theta}(x,y) = \begin{cases} (v_{FD,\theta}(x,y) + v_{BG,\theta}(x,y)) * \\ g_{SC}(x,y), \\ \quad \text{if } (v_{FD,\theta}(x,y) + v_{BG,\theta}(x,y)) * \\ \quad g_{SC}(x,y) > v_{th3}, \\ 0, \text{ otherwise.} \end{cases} \quad (5)$$

Bipole Grouping: Similarly to what happened inside the

directional filtering and spatial competition building blocks, the bipole grouping building block also processes images individually according to their orientation (dominant direction). A *horizontal* image (i.e. an image associated with a horizontal Gabor filter) is filtered along the horizontal direction, a 30-degree image is filtered by a 30-degree Gabor filter, and so on for all directions that are described by the Gabor filter bank. We may focus on the horizontal direction and so describe the image shifts to the left or to the right. The explanation for other directions is similar. At first, the horizontal image is shifted twice: one pixel to the left, and one pixel to the right. As the image content corresponds primarily to horizontal edges, adding the shifted images tends to connect horizontal segments which were originally disconnected because of defects in the visual system. In other words, the missing samples in the vOff signal are recovered. In the one-pixel shift cases (left and right), the summation is equivalent to convolution between the input image and the [0 0 0; 1 0 1; 0 0 0] filter. We typically need larger shifts to the left and to the right, because the defects create gaps larger than one pixel between segments that must be merged. Adding images that were shifted (left and right) by more than one pixel is equivalent to the convolution with filters such as [0 0 0 0 0; 0 0 0 0 0; 1 0 0 0 1; 0 0 0 0 0; 0 0 0 0 0] and so on. To keep only the summation results with a larger intensity, the convolution and summation results are compared (pixel-by-pixel) with an arbitrary threshold (v_{th4}). $v_{th2} = 2$ in spike-mode or $v_{th2} = 0.015$ in amplitude-mode are used in these implementations. Typically, a pixel located midway between segments to be merged has magnitude larger than that of a pixel close to a single segment. Image shifts (to the left and to the right, in the horizontal direction case) create nonzero values for pixels which, by not being between segments to be merged, should otherwise have their values equal to zero. Pixels whose values are below the threshold will, therefore, be set to zero. The bipole grouping procedure is defined by Eq. (6).

$$v_{BG,\theta}(x,y) = \begin{cases} v_{SC,theta}(x,y) * g_{SC}(x,y), \\ \quad \text{if } v_{SC,theta}(x,y) * g_{SC,\theta}(x,y) > v_{th4}, \\ 0, \text{otherwise.} \end{cases} \quad (6)$$

Shifts to the left or to the right (or similar shifts along other orientations) have a biological counterpart: in natural systems, specific structures (neural circuits involving long-range connections [20]) known as *bipoles* [21], [22] implement those shifts. Bipoles are neural circuits composed by a large number of neurons, which can together generate a weighted response that takes into account two input connection types: short-range connections (with smaller weight) and long-range connections (with larger weight). Since long-range connections are taken into account, these neural circuits may generate nonzero output even if its hierarchically and immediately previous inputs are equal to zero. In the implementation of the filters inside the Bipole Grouping building block, for example, the summation of several images shifted by many pixels along opposite directions corresponds to the same operation that is implemented by natural bipoles located at the visual cortex V2 layer. By grouping (adding by means of bipoles) oriented filtering results obtained at different vOff signal locations, the visual system becomes able to merge disconnected parts belonging to the same segment, if those parts were originally perceived as disconnected.

Space Competition using Nonzero Feedback from Bipole Grouping: As Eq. (5) has already defined, the bipole grouping output contains information that must be used as feedback to reinforce edge detection based on spatial competition. The bipole grouping stage generates an output signal $v_{BG,\theta}$ in which originally missing samples were recovered at the contour positions to which they belong. This output signal thus contains enhanced contours that must be used, in feedback mode, to reinforce the spatial competition block input. To obtain that effect, all bipole grouping output channels $v_{BG,\theta}$ are added to their respective $v_{SC,\theta}(x,y)$ at the spatial competition input, so that the spatial competition that was previously described actually operates on those combined channels.

4. Results and Discussion

We implemented the DFI algorithm using both the spike-mode and amplitude-mode neuron models, which were described by Eqs. (2) and (3), and numerical simulation results are presented in this section. In both cases, the DFI input images are described by signals v_{On} and v_{Off}, which were estimated from an electrical model of the retina. This electrical model corresponds to an impulse response estimated from electrical simulations of a CMOS imager (the retina), which is shown at the bottom part of Fig. 6. There are no new relevant features in the design of the imager and details about implementation can be found in [12]. The original input image is shown at the top left part of Fig. 6. To generate a DFI input image, we compute the convolution between this original image and the retina impulse response. At any given time instant, positive values from the convolution result are represented exclusively by v_{On} (top center part of the figure), and the negative values are represented exclusively by v_{Off} (top right part of the figure). We use either Eq. (2) or Eq. (3), so that v_{Off} and v_{On} represent neural signals (spike-mode or amplitude-mode). Because of the convolution with a retina impulse response that was obtained from electrical simulations, the DFI inputs (spike or amplitude-mode) are more realistic than the inputs that would be obtained by difference-of-Gaussians filtering of the original image. The retina impulse response approach generates pixel values with a larger dynamic range, which leads to more variation among spike sequences, in the spike-mode simulation. Additional differences between spike-mode and amplitude-mode simulations are observed, in the case of DFI input signals that are more realistic [9]. The simulated DFI results are shown in Fig. 7. The spike-mode results have larger variations at pixel positions that are close to the v_{BCS} contours, which may be interpreted as slightly noisier results. Except for that, the amplitude-mode and spike-mode results

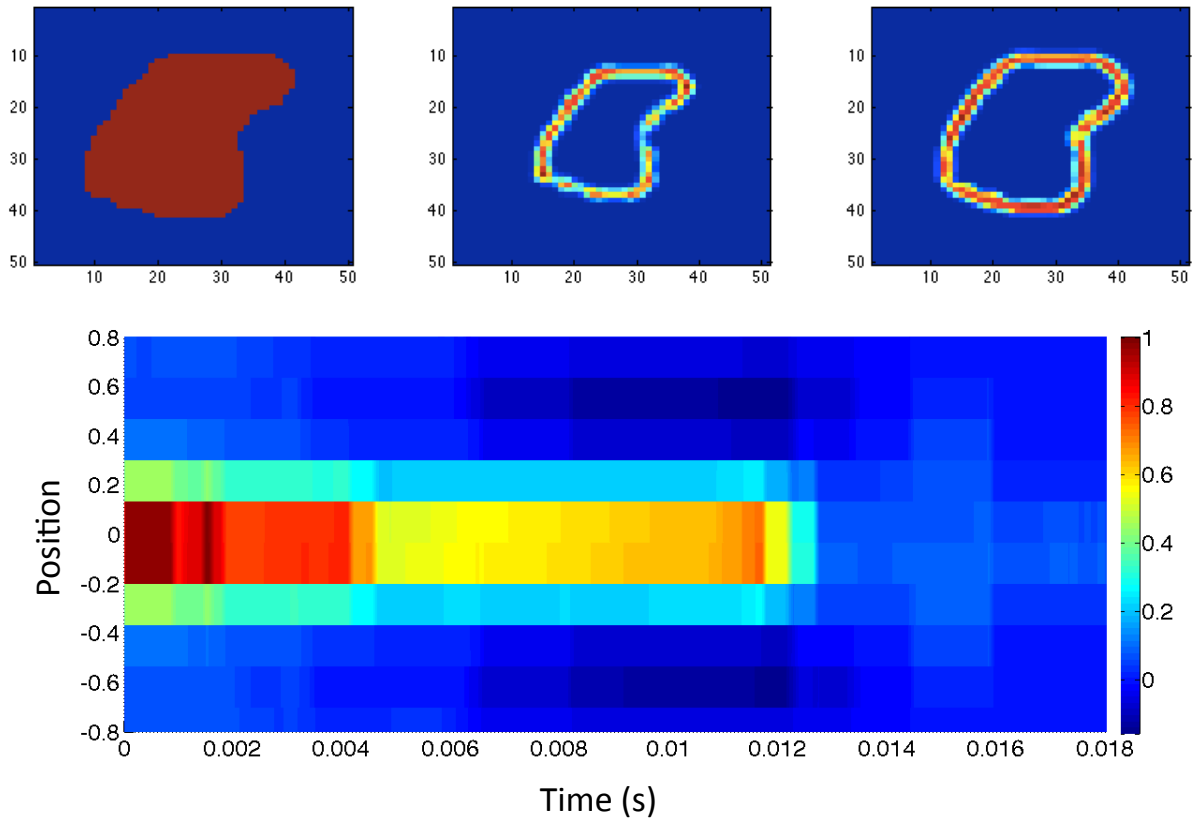


Figure 6. The original image, from which the DFI input is generated, is shown at the top left part of the figure. To generate a DFI input image, the original image is filtered by the imager (retina). The retina impulse response in space and time domains is shown at the bottom part of the figure. For simplicity, only one space dimension is shown in this figure. The impulse response along the other spatial dimension is the same. The retina output corresponds to the convolution between the original image and this impulse response. At any given time instant, the convolution result contains positive values and negative values. The positive values are represented exclusively by v_{On} (top center) and the negative values are represented exclusively by v_{Off} (top right). To obtain spike-mode v_{On} and v_{Off} signals, Eq. (2) is used. Otherwise, Eq. (3) is used.

are similar.

In Figs. 8a and 8b, we show histograms of DFI pixel values on the v_{On} and v_{Off} channels v_{On} values on the positive part of each plot and v_{Off} on the negative part) under two different circumstances: DFI image obtained by convolution of the input image with the retina impulse response estimated from electrical simulations, and DFI image obtained by difference-of-Gaussian filtering applied to the input image. This emphasizes the fact that the convolution with an impulse response estimated from an electrical simulation leads to larger diversity in the DFI input values. This larger diversity leads to larger variations in the spike-mode results, which are nevertheless well-approximated by the amplitude-mode results. In Fig. 9, we show the mean squared error (MSE) between the spike-mode and amplitude-mode DFI outputs. After approximately 50 ms, the amplitude-mode DFI output has converged to the spike-mode DFI output with a small error.

Acknowledgements

This work was supported by Brazilian research funding agencies CAPES, CNPq, IFRJ and FAPERJ.

5. Conclusions

In this paper was performed spike-mode and amplitude-mode simulations (Figure 7) of a signal processing algorithm that implements a cortical mechanism known as DFI. To compare (visually) the results, image sequences showing the temporal evolution of boundary completion and filling-in stages (parts of the DFI algorithm) in spike mode and amplitude mode were presented. In both simulations, it was used as input a non-ideal stimulus (Figure 6). This non-ideal stimulus, which was estimated from electrical simulations of a retina model, highlights the differences, in a comparative analysis, between both approaches. The richness in details of the retina model impulse response in comparison of the difference-of-Gaussian model impulse response becomes apparent as Figures 8a and 8b are analyzed. The DFI outputs

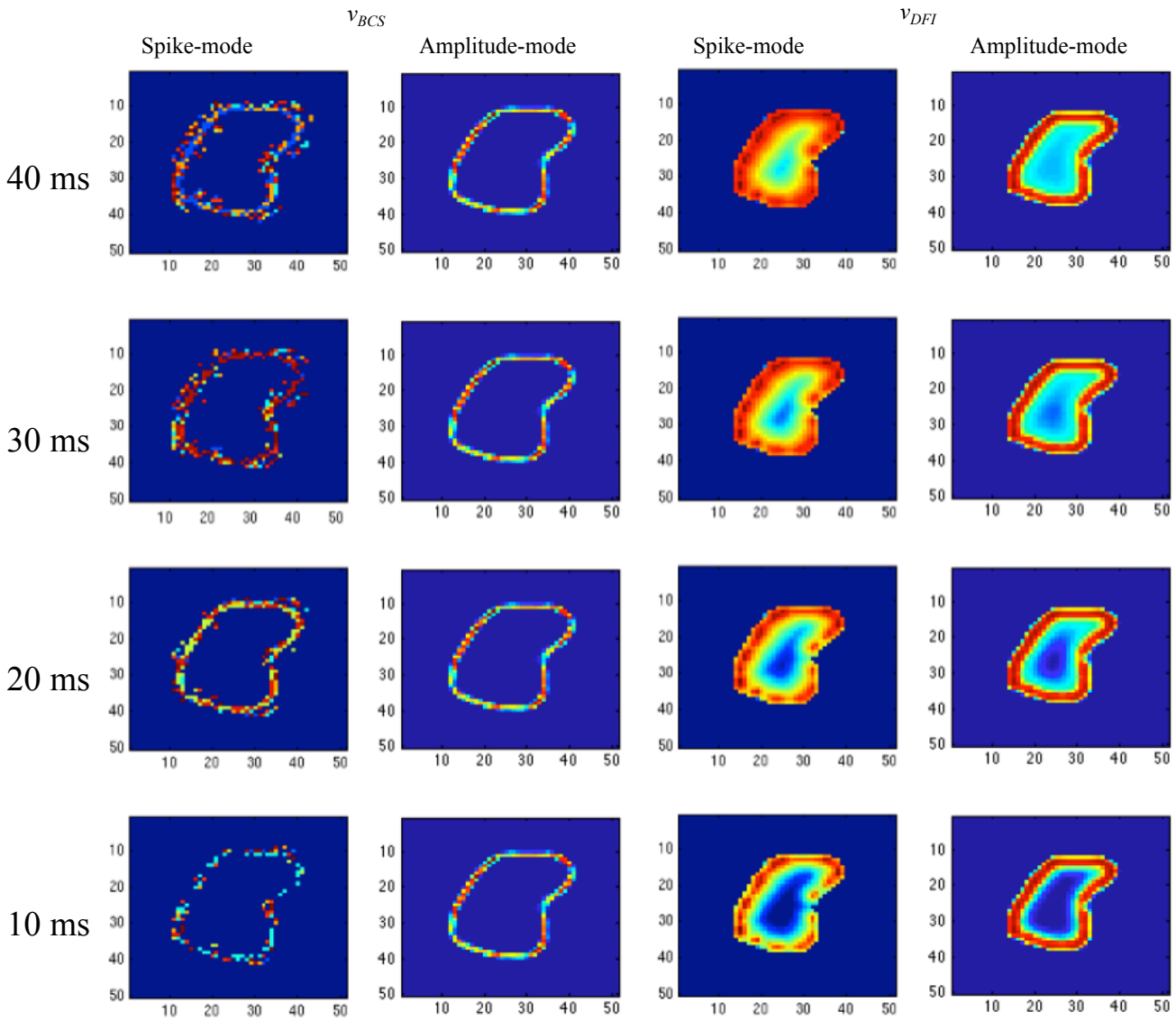
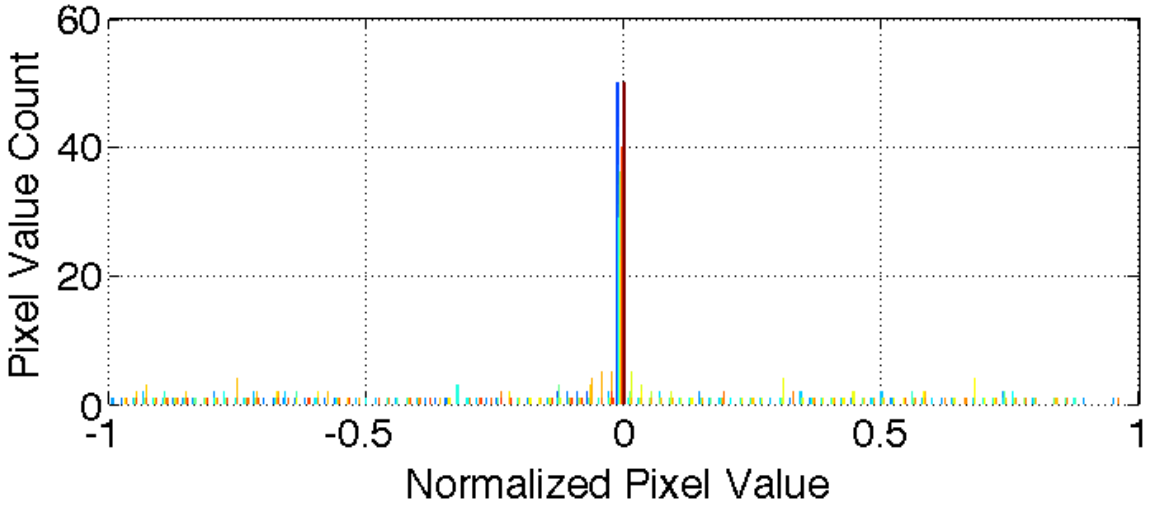
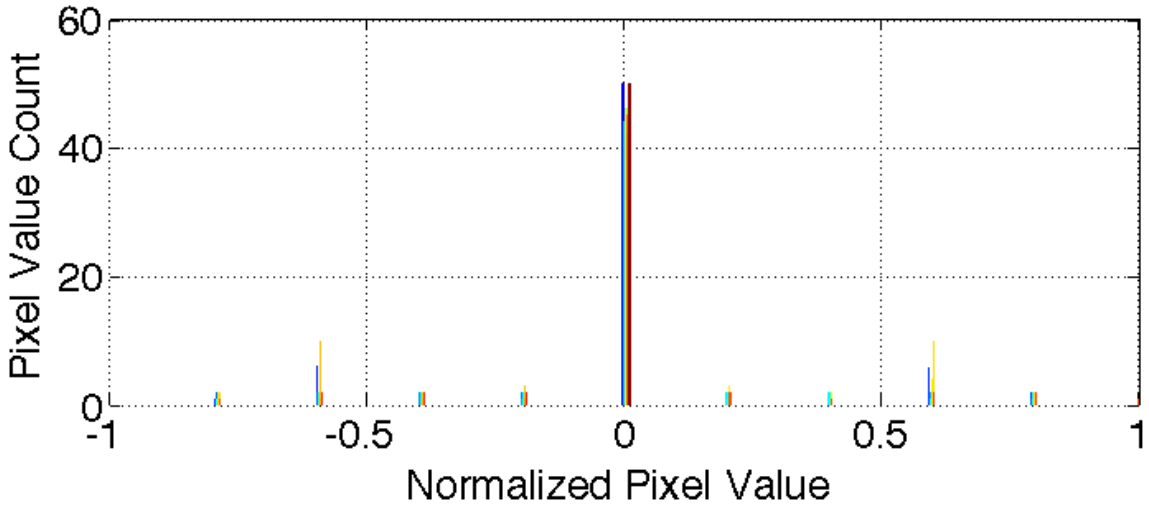


Figure 7. Filling-in and boundary completion results at four different time instants (from the bottom to the top: 10 ms, 20 ms, 30 ms, and 40 ms). The two leftmost columns show v_{BCS} results, and the two rightmost columns show v_{DFI} results. The first and third columns (from left to right) are obtained from spike-mode neuron models. The second and fourth column are obtained from amplitude-mode neuron models.

(v_{DFI}) in both modes were compared and, although the spike-mode results seem slightly noisier, both simulation results are similar, which suggests that amplitude-mode simulations may be used to theoretically predict the behavior of hardware dedicated to cortical circuit simulation. This comparison is important to know the best performance (minimum error) that can achieve when amplitude-mode simulations are used to predict spike-mode system behavior. The most important contribution of this work was showing that the minimum error related to amplitude-mode predictions is about 20 percent (steady-state response). However, the minimum error in transient response can overcome the 30 percent (Figure 9). A way to improve this result would be to introduce main electrical parasitic effects, which influence the behavior of dedicated hardware, in the spike-mode simulation, but, such simulation may become prohibitive in non-dedicated hardware.



(a)



(b)

Figure 8. Histogram of DFI input values (v_{On} on the right part of the plots and v_{Off} on the left part of the plots) under two circumstances: (a) DFI input values obtained from convolution between an original input image and the retina impulse response estimated from electrical simulations, and (b) DFI input values obtained from difference-of-Gaussians filtering applied to an original input image.

Author Contributions

Genildo Nonato Santos: Spike-mode and amplitude-mode simulations of a signal processing algorithm that implements a cortical mechanism known as DFI using a non-ideal stimulus (retina model), which was estimated from electrical simulations of a retina model in a comparative analysis.

José Gabriel Rodriguez Carneiro Gomes: Spike-mode and amplitude-mode simulations of a signal processing algorithm that implements a cortical mechanism known as DFI using an ideal stimulus (DoG model), translation, and the result analysis.

References

[1] CHOUDHARY, S. et al. Artificial neural networks and machine learning – icann 2012: 22nd international conference on artificial neural networks, lausanne, switzerland, september 11-14, 2012, proceedings, part i. In: ____. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. cap. Silicon Neurons That Compute, p. 121–128. Disponível em: ⟨http://dx.doi.org/10.1007/978-3-642-33269-2_16⟩.

[2] XUE, Y. Recent development in analog computation: a brief overview. *Analog Integrated Circuits and Signal Processing*, v. 86, n. 2, p. 181–187, 2016. Disponível em: ⟨http://dx.doi.org/10.1007/s10470-015-0668-y⟩.

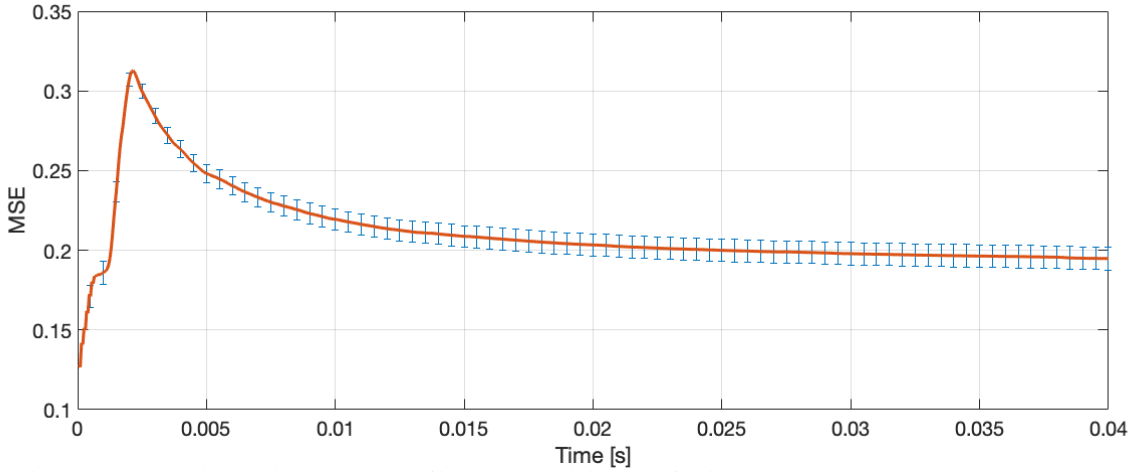


Figure 9. The mean squared error between the spike-mode and the amplitude-mode DFI outputs (v_{DFI}). Such v_{DFI} signal is a matrix (50x50) at each discrete time point.

[3] MEROLLA, P. et al. A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm. In: *Custom Integrated Circuits Conference (CICC), 2011 IEEE*. [S.l.: s.n.], 2011. p. 1–4.

[4] IZHIKEVICH, E. Which model to use for cortical spiking neurons? *Neural Networks, IEEE Transactions on*, v. 15, n. 5, p. 1063–1070, Sept 2004.

[5] BENJAMIN, B. V. et al. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, v. 102, n. 5, p. 699–716, 2014. Disponível em: ⟨http://dblp.uni-trier.de/db/journals/pieee/pieee102.html#BenjaminGMCCBAAMB14⟩.

[6] CASSIDY, A. S.; GEORGIOU, J.; ANDREOU, A. G. Design of silicon brains in the nano-cmos era: Spiking neurons, learning synapses and neural architecture optimization. *Neural Networks*, v. 45, n. 1, p. 4 – 26, 2013. Neuromorphic Engineering: From Neural Systems to Brain-Like Engineered Systems. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0893608013001597⟩.

[7] Frenkel, C. et al. A 0.086-mm^2 12.7-pj/sop 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm cmos. *IEEE Transactions on Biomedical Circuits and Systems*, v. 13, n. 1, p. 145–158, Feb 2019.

[8] KUNKEL, S. et al. Spiking network simulation code for petascale computers. *Frontiers in Neuroinformatics*, v. 8, n. 78, 2014. Disponível em: ⟨http://www.frontiersin.org/neuroinformatics/10.3389/fninf.2014.00078/abstract⟩.

[9] CAO, Y.; GROSSBERG, S. Stereopsis and 3d surface perception by spiking neurons in laminar cortical circuits: A method for converting neural rate models into spiking models. *Neural Networks*, v. 26, p. 75 – 98, 2012. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0893608011002723⟩.

[10] SANTOS, G. N.; GOMES, J. G. R. C. Implementation of a biologically inspired diffusive filling-in algorithm for focal-plane image processing applications. In: *Proc, BRICCCI and CBIC, Recife, Brazil, Conference on Computational Intelligence*. [S.l.: s.n.], 2013.

[11] GROSSBERG, S.; KUHLMANN, L.; MINGOLLA, E. A neural model of 3d shape-from-texture: Multiple-scale filtering, boundary grouping, and surface filling-in. *Vision Research*, v. 47, n. 5, p. 634 – 672, 2007. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0042698906004950⟩.

[12] BOAHEN, K. A. *Retinomorphic Vision Systems: Reverse Engineering the Vertebrate Retina*. Tese (Ph.D. thesis) — California Institute of Technology, Pasadena, CA, 1997.

[13] ZAGHLOUL, K. A.; BOAHEN, K. A silicon retina that reproduces signals in the optic nerve. *Journal of Neural Engineering*, v. 3, n. 4, p. 257, 2006. Disponível em: ⟨http://stacks.iop.org/1741-2552/3/i=4/a=002⟩.

[14] CHEN, E. P.; FREEMAN, A. W. A model for spatiotemporal frequency responses in the x cell pathway of the cat's retina. *Vision Research*, v. 29, n. 3, p. 271 – 291, 1989. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/004269898990076X⟩.

[15] MARR, D. *Vision : a computational investigation into the human representation and processing of visual information*. San Francisco: W.H. Freeman, 1982. Disponível em: ⟨http://opac.inria.fr/record=b1079861⟩.

[16] LEVITT, J. B.; KIPER, D. C.; MOVSHON, J. A. Receptive fields and functional architecture of macaque v2. *Journal of Neurophysiology*, American Physiological Society, v. 71, n. 6, p. 2517–2542, 1994. Disponível em: ⟨http://jn.physiology.org/content/71/6/2517⟩.

[17] GOVE, A.; GROSSBERG, S.; MINGOLLA, E. Brightness perception, illusory contours, and corticogeniculate

feedback. *Visual Neuroscience*, v. 12, p. 1027–1052, 11 1995. Disponível em: ⟨http://journals.cambridge.org/article_S0952523800006702⟩.

[18] YAZDANBAKHSH, A.; GROSSBERG, S. Fast synchronization of perceptual grouping in laminar visual cortical circuits. *Neural Networks*, v. 17, n. 5-6, p. 707–718, 2004. Disponível em: ⟨http://dx.doi.org/10.1016/j.neunet.2004.06.005⟩.

[19] SEO, J.; SHNEIDERMAN, B. A rank-by-feature framework for interactive exploration of multidimensional data. *Information VisualizationInformation Visualization*, v. 4, p. 96 – 113, 2005/06/20/ 2005. Disponível em: ⟨http://ivi.sagepub.com/content/4/2/96⟩.

[20] COHEN, M. A.; GROSSBERG, S. Neural dynamics of brightness perception: Features, boundaries, diffusion, and resonance. *Perception & Psychophysics*, v. 36, n. 5, p. 428–456. Disponível em: ⟨http://dx.doi.org/10.3758/BF03207497⟩.

[21] BOSKING, W. H. et al. Orientation Selectivity and the Arrangement of Horizontal Connections in Tree Shrew Striate Cortex. *The Journal of Neuroscience*, Society for Neuroscience, v. 17, n. 6, p. 2112–2127, mar. 1997. Disponível em: ⟨http://www.jneurosci.org/content/17/6/2112.abstract⟩.

[22] AZZI, J. a. C. B. et al. Precise visuotopic organization of the blind spot representation in primate V1. *Journal of Neurophysiology*, American Physiological Society, v. 113, n. 10, p. 3588–3599, jun. 2015. Disponível em: ⟨http://dx.doi.org/10.1152/jn.00418.2014⟩.