

# Real-Time Implementation of a Retinal Model

## Short end-of-first-year report

Garibaldi Pineda García

March 30, 2015

### Abstract

Converting frame-based video into spike-trains is a computationally intensive and time consuming task. There are very few equipments that can produce spikes from video and they are either in development or too expensive, thus every-day users are put behind a virtual wall, leaving them unable to experiment with visual input in their simulations or robotics applications. An efficient parallel implementation of a simple yet powerful retinal model would remove this wall by reducing the time it takes to compute a spike representation of a video frame. Furthermore if this is done on consumer graphics processing hardware, it will allow almost any user to generate their own spike-trains.

## 1 Introduction

In recent years neuro-morphic (i.e. one that mimics the brain) hardware has risen attention as a different way of computing. One key aspect is the high parallelism found in the infrastructure of the brain. Platforms such as *SpiNNaker* [5] emulate such parallelism; furthermore it does so while maintaining low power consumption. The *SpiNNaker* platform can also give neural simulations the flexibility of software models and keep them running in biological real-time.

Input to neural simulation has been limited, primarily, to pre-recorded or artificial data. This limits the use of optical input in real-time systems, e.g. robotics or security. Generating neuronal input (spikes) from video in real-time has been done either by using extremely low resolution video or using devices that do not rely on the classical frame-based camera paradigm.

In this work we present different strategies to achieve real-time encoding into spike trains of video with higher resolution (section 4) exploiting the parallelism that comes with image processing.

In most animals the eye is the organ that captures light so that it can be processed in the retina and sent further down the brain as multiple spike trains (section 2). The retina has been modelled using different approaches that go from the extremely detailed (i.e. cell-by-cell)[12] to the functional [8, 9]. We have chosen what we believe to be the best model for real-time video processing (section 3). This model provides a simple and elegant solution to encoding, though it suffers from redundancy issues.

To verify that our spike train sets are correct, we test them using a reconstruction procedure (section 5). We present ideas on how to improve our work with further development at the end of this paper (section 6).

## 2 A short introduction to the retina

Optical information is gathered by most animals through their eyes. The eye can be treated as a camera (Figure 1a): the cornea, iris, pupil and lens can be viewed as the mechanical lens found in commercial cameras. Light rays are bent and focused on the “film” or “sensor”, the *retina* in the eye.

Once the light enters the retina, it moves through several layers of neurons [3] and hits the *photoreceptors* (shown at the bottom of Fig. 1b). Light will elicit a chain reaction that has several steps, all occurring in the retina. A zone in the photoreceptor layer called the *Fovea*, within this region there is a small portion which is highly packed with *cones* [2], and has almost direct exposure to light, this region is known as the *Foveal Pit*. This is the zone where images are acquired at the highest resolution.

The final step of the processing in the retina is carried out by *ganglion cells* (top of Fig. 1b), this cells will emit spikes when certain conditions are met in the previous layers of the retina. The most common type of cells are the *Midget ganglion cells*. They are classified, depending on their input connectivity (dendritic

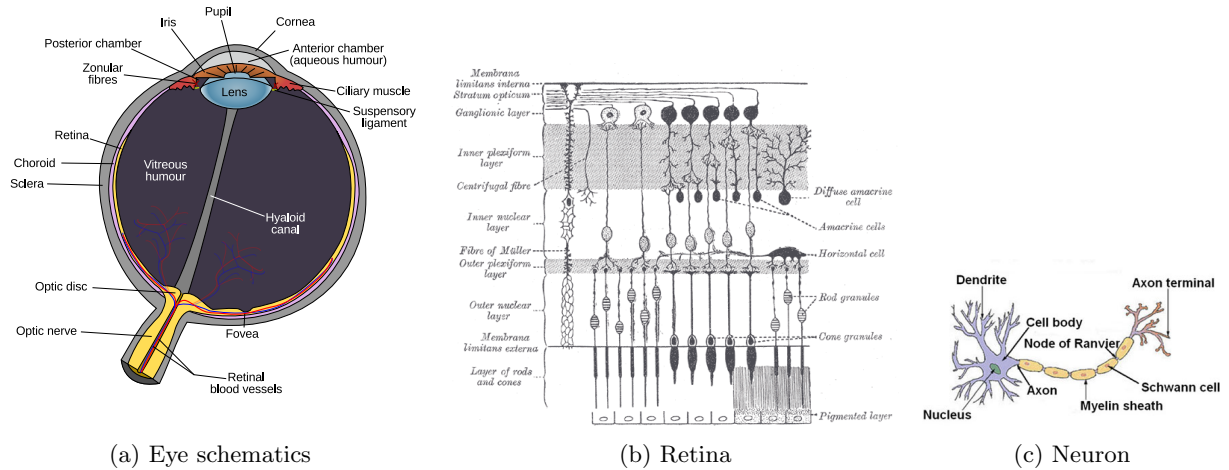


Figure 1: Anatomy of the (human) eye <sup>1</sup>

tree, left side of Fig. 1c), as *ON-centre* or *OFF-centre* [7, 2]. What this means is that, an ON-centre cell will emit a spike when the input in the “centre” region is stimulated but the surrounding area is not so much. The inverse case is true for OFF-centre type cells. The retina also contains another type of ganglion cells whose dendritic trees span at much longer distances, thus sampling wider portions of the image in the eye. This last type of cells are called *Parasol ganglion cells* and come in both ON-centre and OFF-centre variants.

### 3 The foveal pit model

A functional model is one that treats a system as a black box and tries to “map” observed inputs to their respective outputs by means of a set of mathematical expressions. The model created by Sen and Furber in [7, 8], consists of several instances of four ganglion cells (midget and parasol, with ON and OFF centres) which transform the pixels in the received image into rank-ordered spike trains (Figure 2). This model is based on previous work by Van Rullen and Thorpe in [11].

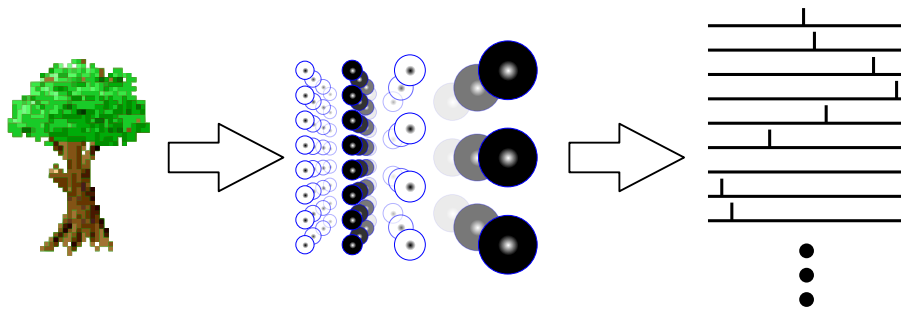


Figure 2: Schematic of the foveal pit model

#### 3.1 Neural codes

There are many ways of encoding information using spikes. The most common one is rate-based, in which the average count of spikes fired by a neuron encodes a value. A different approach is to take the exact timing of spikes when they were generated as a means to transmit information. The latter has great encoding capabilities since there are many times in which a spike can occur. Rate-based codes tend to have limited representational power because the same spike count can happen in different ways.

<sup>1</sup>Images where taken from wikipedia.org and wikimedia.org, with reuse licenses

A simpler way of encoding spikes is to not really pretend to know exactly when they were actually emitted, but to just take into account the order in which they happened; this is referred as a rank-ordered encoding. This code has the advantage of being able to transmit more information than rate-based ones [7, 9, 1], yet maintain a simple way of interpretation. Furthermore, rank-ordered encoding has shown to provide enough information for reconstruction in about the highest 20 to 30% spikes.

### 3.2 Mathematical model

As discussed in section 2, the Foveal pit region of the retina is the highest resolution zone in the retina. This is taken into account by simulating one midget cell per pixel and one parasol cell about every 7 pixels.

Each ganglion cell is characterized by a Difference of Gaussian (DoG), described in equation 1.

$$DoG_w(x, y) = \pm \frac{1}{2\pi\sigma_{w,c}^2} e^{-\frac{(x^2+y^2)}{2\sigma_{w,c}^2}} \mp \frac{1}{2\pi\sigma_{w,s}^2} e^{-\frac{(x^2+y^2)}{2\sigma_{w,s}^2}} \quad (1)$$

where  $\sigma_{w,c}$  and  $\sigma_{w,s}$  are the standard deviation for the centre and surround components of the DoG at scale  $w$  (cell type). The signs will be  $(-, +)$  if the ganglion cell is OFF-centre;  $(+, -)$  if it is ON-centre.

The simulation of the cell is carried by a discrete convolution of equation 1 over the input image. We can think of each cell being characterized by a convolution kernel.

$$C(x, y, w) = \sum_i \sum_j (I(i + x, j + y) \cdot DoG_w(i, j)) \quad (2)$$

This will provide a set of coefficients  $C$  for every scale  $w$ . The authors in [11] refer to this as a wavelet-like transformation. From this sets we'll take only the positive ( $C(x, y, w) > 0$ ) to signify a neuron at location  $(x, y)$  fired a spike. The value of the coefficient will mean how fast does the neuron fire, i.e. the largest value's neuron is the first to spike. If the coefficients are sorted according to their value, we shall possess a rank-ordered spikes set.

Cells are parametrized according to table 1. After applying discrete convolutions to a test image Fig. 4a, the results can be seen in Figs. 3b, 3c 4b and 4c.

Table 1: Simulation parameters for ganglion cells

Cell type	Matrix size	Centre std. dev. ( $\sigma_c$ )	Surround std. dev. ( $\sigma_s$ )	Sampling resolution
Midget OFF-centre	$3 \times 3$	0.8	$6.7 \times \sigma_c$	col: 1, row: 1
Midget ON-centre	$11 \times 11$	1.04	$6.7 \times \sigma_c$	col: 1, row: 1
Parasol OFF-centre	$61 \times 61$	8	$4.8 \times \sigma_c$	col: 5, row: 3
Parasol ON-centre	$243 \times 243$	10.4	$4.8 \times \sigma_c$	col: 5, row: 3

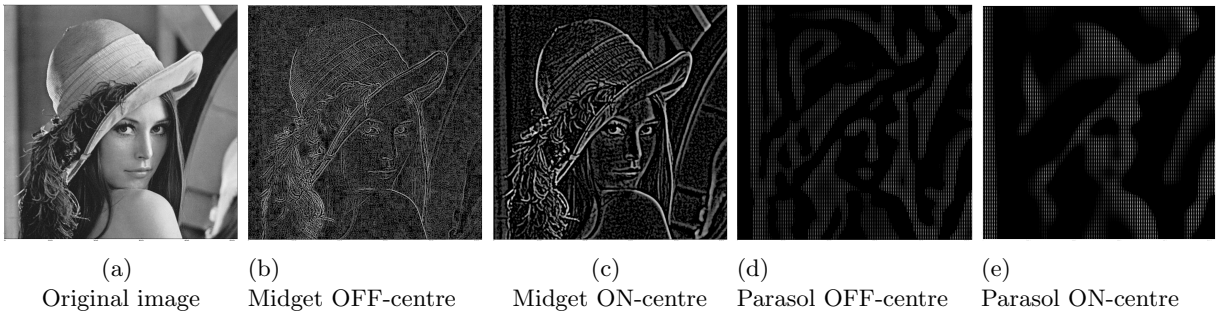


Figure 3: Results of simulating ganglion cells (convoluted images, enhanced for better contrast)

## 4 Parallel implementation

The retinal model described in section 3 is parallel in nature it involves changing each pixel locally. Convolution kernels are computed and stored in memory prior to any convolution procedure. OpenCL is used since the same code can be used in multiple Operating Systems (OS) and hardware targets [4].

### 4.1 Naïve approach

The easiest way of implementing the ganglion cell simulation is to code equation 2 into OpenCL and do the same operation for every pixel in the image (code). This is a rather inefficient way of performing convolution on images [13, 6].

**Algorithm 4.1:** NAÏVE CONVOLUTION(*image*  $I$ , *kernel*  $K$ )

```

for each pixel  $p \in I$ 
  do (in parallel):  $x \leftarrow \text{row}(p)$ ,  $y \leftarrow \text{column}(p)$ ,  $\text{sum} \leftarrow 0$ ,  $k \leftarrow 0$ 
    for  $i \leftarrow x - \text{width}(K)/2$  to  $x + \text{width}(K)/2$ 
      for  $j \leftarrow y - \text{width}(K)/2$  to  $y + \text{width}(K)/2$ 
         $\text{sum} \leftarrow \text{value}(I, i, j) * \text{value}(K, k)$ ,  $k \leftarrow k + 1$ 
    return ( $\text{convolved\_image} \leftarrow \text{pixel}(\text{sum}, x, y)$ )

```

### 4.2 Coding optimization

The first aspect to consider in optimizing code 4.1 is that not every pixel in the image will have valid data to compute a convolution, thus we'll discard them. Without having to worry about valid or invalid pixels, one can easily unroll the two inner **for** loops in code 4.1 and free the processors from those operations. The next thing to notice is that both the image ( $I$ ) and convolution kernel ( $K$ ) may be stored in local (*faster*, shared by a set of processors) memory instead of global (*slower*, shared by *all* cores). Changing data access from global to local memory brings a significant speed-up.

### 4.3 Separability

The kernel that represents the DoG is not separable, i.e. it may not be computed by the multiplication of a column and a row vector. Nonetheless the matrices that, when subtracted, form a DoG are, in fact, separable (eq. 4).

$$\begin{aligned}
 C(x, y, w) &= \sum_i \sum_j \left( \left[ \pm \frac{1}{2\pi\sigma_{w,c}^2} e^{\frac{-(i^2+j^2)}{2\sigma_{w,c}^2}} \mp \frac{1}{2\pi\sigma_{w,s}^2} e^{\frac{-(i^2+j^2)}{2\sigma_{w,s}^2}} \right] I(i+x, j+y) \right) \\
 &= \pm \left[ \frac{1}{2\pi\sigma_{w,c}^2} \sum_i e^{\frac{-i^2}{2\sigma_{w,c}^2}} \sum_j e^{\frac{-j^2}{2\sigma_{w,c}^2}} I(i+x, j+y) \right] \mp \left[ \frac{1}{2\pi\sigma_{w,s}^2} \sum_i e^{\frac{-i^2}{2\sigma_{w,s}^2}} \sum_j e^{\frac{-j^2}{2\sigma_{w,s}^2}} I(i+x, j+y) \right]_s
 \end{aligned} \tag{3}$$

We take advantage of this to perform a *separable convolution*; the first step is to convolve the image with a horizontal kernel, next we take the resulting image and convolve it vertically. This reduces the computation and memory requirements from  $O(N * M)$  to  $O(N + M)$ .

### 4.4 Tiled convolution

The final step is an optimization published by Advanced Micro Devices (AMD) described in [10]. They take advantage from using separable kernels and reusing operations to get the most out of resources. We can see separable convolution as a two step addition process. First we multiply and add horizontally (image times kernel), this leaves us with a set  $H = \{h_1, h_2, \dots, h_n\}$  which represent the horizontally convoluted image. We then proceed to multiply and add (convoluted image times kernel) again to compute the final result. The key aspect of this optimization technique is that adjacent elements share most computations; that is, if we want to calculate the convolution value for the southern neighbour of the current pixel, we only need to perform one extra horizontal multiply-and-add and a vertical one.

## 4.5 Comparison

STILL WORK IN PROGRESS (Got new hardware so I have to redo statistics)

# 5 Testing spikes via reconstruction

## 5.1 Filter overlap correction algorithm (FoCal)

Neighbouring pixels may contain very similar information, this redundancy is dealt in the eye with a mechanism known as lateral inhibition. When a ganglion cell spikes, it will inhibit its neighbours from spiking for some time. In [7] Sen and Furber developed an algorithm that simulates this behaviour, the only caveat is that this process is done after all spikes have been fired.

For every pixel set (resulting images, sec. 3) and for a determined number of times, we will search for the maximum value and adjust its neighbours' value according to the correlation of their convolution kernels. The correlation represents the overlap occurred in the simulation of the ganglion cells. (STIL WORKING ON THIS)

## 5.2 Reconstruction procedure

Spikes are obtained using a *quasi-orthogonal* transformation, thus reconstruction is a matter of applying the same operation to every spike generated and to add them up to obtain an approximate version of the original image ().



Figure 4: Results of reconstruction procedure

# 6 Conclusions and further work

Sampling pattern is key for a good representation, thus subdivide image into smaller blocks to have a better reconstruction results. Real-time with low temporal resolution is achievable with common GPU and the right combination of mathematics and engineering. We established a timing mechanism to emit spikes.

Possible solutions for a faster mutual inhibition algorithm might be to: do it in-line as we send spikes to neuro-morphic hardware; design low-power/low-cost hardware to perform the task; think of a different and fully orthogonal basis set as an equivalent to the DoG.

To reduce the power consumption/hardware requirements for mobile applications the best way to go might be change the sampling pattern/resolution so not all image is perceived in high resolution.

## References

- [1] Jacques Gautrais and Simon Thorpe. “Rate coding versus temporal order coding: a theoretical approach”. In: *Biosystems* 48.1–3 (1998), pp. 57–65. ISSN: 0303-2647. DOI: [http://dx.doi.org/10.1016/S0303-2647\(98\)00050-1](http://dx.doi.org/10.1016/S0303-2647(98)00050-1). URL: <http://www.sciencedirect.com/science/article/pii/S0303264798000501>.
- [2] Helga Kolb. *Midget Pathways of the Primate Retina Underlie Resolution and Red Green Color Opponency*. Ed. by Moran Eye Center. Webvision. 2015. URL: <http://webvision.med.utah.edu/book/part-iii-retinal-circuits/midget-pathways-of-the-primate-retina-underly-resolution/>.
- [3] Helga Kolb. *Simple Anatomy of the Retina*. Ed. by Moran Eye Center. Webvision. 2015. URL: <http://webvision.med.utah.edu/book/part-i-foundations/simple-anatomy-of-the-retina/>.
- [4] A. Munshi et al. *OpenCL Programming Guide*. OpenGL. Pearson Education, 2011. ISBN: 9780132594554.
- [5] Alexander D. Rast et al. “Scalable Event-driven Native Parallel Processing: The SpiNNaker Neuromimetic System”. In: *Proceedings of the 7th ACM International Conference on Computing Frontiers*. CF ’10. Bertinoro, Italy: ACM, 2010, pp. 21–30. ISBN: 978-1-4503-0044-5. DOI: 10.1145/1787275.1787279. URL: <http://doi.acm.org/10.1145/1787275.1787279>.
- [6] Khairi Reda. *A study of OpenCL image convolution optimization*. 2012. URL: <https://www.evl.uic.edu/kreda/gpu/image-convolution/>.
- [7] Basabaddatta Sen. “Information Recovery From Rank-Order Encoded Images”. Doctor of Philosophy Thesis. Faculty of Engineering and Physical Sciences, University of Manchester, 2008.
- [8] Basabaddatta Sen and Steve Furber. “Evaluating Rank-order Code Performance Using a Biologically-derived Retinal Model”. In: *Proceedings of the 2009 International Joint Conference on Neural Networks*. IJCNN’09. Atlanta, Georgia, USA: IEEE Press, 2009, pp. 1835–1842. ISBN: 978-1-4244-3549-4. URL: <http://dl.acm.org/citation.cfm?id=1704175.1704440>.
- [9] Simon Thorpe, Arnaud Delorme, and Rufin Van Rullen. “Spike-based strategies for rapid processing”. In: *Neural Networks* 14.6–7 (2001), pp. 715–725. ISSN: 0893-6080. DOI: [http://dx.doi.org/10.1016/S0893-6080\(01\)00083-1](http://dx.doi.org/10.1016/S0893-6080(01)00083-1). URL: <http://www.sciencedirect.com/science/article/pii/S0893608001000831>.
- [10] *Tiled Convolution: Fast Image Filtering*. 2015. URL: <http://developer.amd.com/resources/documentation-articles/articles-whitepapers/tiled-convolution-fast-image-filtering/>.
- [11] Rufin Van Rullen and Simon J Thorpe. “Rate Coding Versus Temporal Order Coding: What the Retinal Ganglion Cells Tell the Visual Cortex”. In: *Neural computation* 13.6 (2001), pp. 1255–1283.
- [12] Adrien Wohrer and Pierre Kornprobst. “Virtual Retina: A biological retina model and simulator, with contrast gain control”. English. In: *Journal of Computational Neuroscience* 26.2 (2009), pp. 219–249. ISSN: 0929-5313. DOI: 10.1007/s10827-008-0108-4. URL: <http://dx.doi.org/10.1007/s10827-008-0108-4>.
- [13] www.cmsoft.com.br. *Case study: High performance convolution using OpenCL \_\_local memory*. 2015. URL: <http://www.cmsoft.com.br/openc1-tutorial/>.