Real-Time Implementation of a Retinal Model

Short end-of-first-year report

Garibaldi Pineda García
Advanced Processing Technologies Group
School of Computer Science
University of Manchester, U.K.

May 13, 2015

Acknowledgements

This research is funded by the National Council of Science and Technology (CONACyT) and the Secretariat of Public Education (SEP) of México.

Contents

1 Abstract

Converting frame-based video into spike-trains is a computationally intensive and time consuming task. There are very few equipment 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.

2 Introduction

In recent years neuromorphic (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* [9] 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 to, primarily, 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. Event-based visual sensors [6, 5] are a great step towards biologically inspired vision, though they might have limited availability.

In this work we present different strategies to achieve real-time encoding into spike trains of video with higher resolution (section 7.3) 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 7.1). The retina has been modelled using different approaches that go from the extremely detailed (i.e. cell-by-cell)[16] to the functional [12, 13]. We have chosen what we believe to be the best model for real-time video processing (section 7.2). This model provides a simple and elegant solution to encoding, though it suffers from redundancy issues, which are alleviated using further processing.

To verify that our spike train sets are correct, we test them using a reconstruction procedure (section 7.6). We present conclusions of the work so far and ideas on how to improve our work with further development at the end of this report (section 9).

3 Research Aims and Contribution

Develop computer vision algorithms using SpiNNaker.
Model stages of biological vision using SpiNNaker hardware.
Robotics, security, transport applications.

4 Potential Impact

Biologically inspired real-time vision systems in robotics, security, transport, etc.

5 Previous Work

Simple image encoding using Poisson spiking.
One of the most accurate retinal models is [16].
First biologically inspired silicon retina by Mead and Mahowald [7]
Real-time, high temporal resolution hardware based artificial retinas in [5, 6]

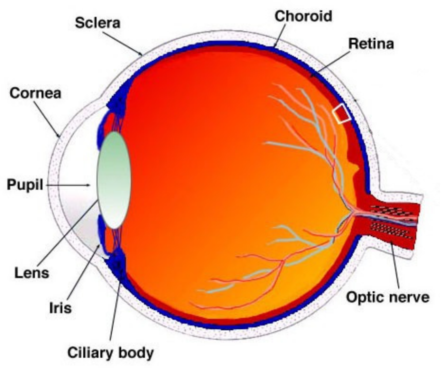
6 Proposed Methodology

Study background neural/biology vision.
Implement models in SpiNNaker.
Develop new models.

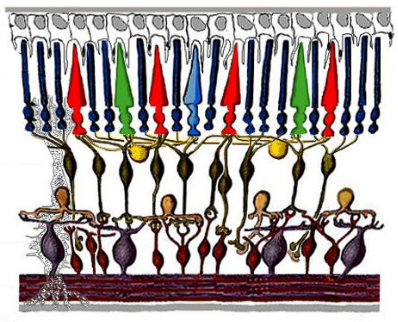
7 Project Progress to Date

7.1 A short introduction to the retina

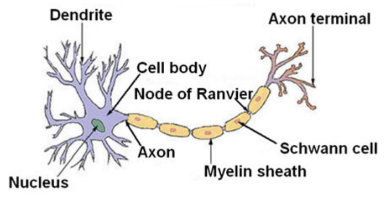
Optical information is gathered by most animals through their eyes. The eye can be treated as a camera (Figure 1a): the cornea, iris, pupil and lens can be viewed as the mechanical lens found in commercial cameras. Light rays are bent and focused on the "*film*" or "*sensor*", the *retina* in the eye.



(a) Eye schematics



(b) Retina



(c) Neuron

Figure 1: Anatomy of the (human) eye

Once the light enters the retina, it moves through several layers of neurons [4] and hits the *photoreceptors* (shown at the top of Fig. 1b [1]). Light will elicit a chain reaction that has several steps, all occurring in the retina. On the central regions of the retinal layer there's a zone called the *Fovea*. The Fovea includes a small portion which is highly packed with *cones* [3], 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* (bottom 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 tree, left side of Fig. 1c), as *ON-centre* or *OFF-centre* [11, 3]. 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.

7.2 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 [11, 12], 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 [15].

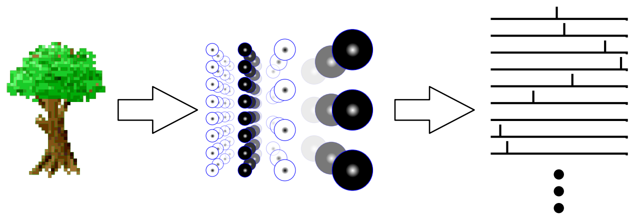


Figure 2: Schematic of the foveal pit model

7.2.1 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 due to different inputs.

A simpler way of encoding spikes is to not really pretend to know exactly when they where 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 [11, 13, 2], 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.

7.2.2 Mathematical model

As discussed in section 7.1, 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.

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

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 a cell is carried by a discrete convolution (Eq. 2) of a DoG over the input image.

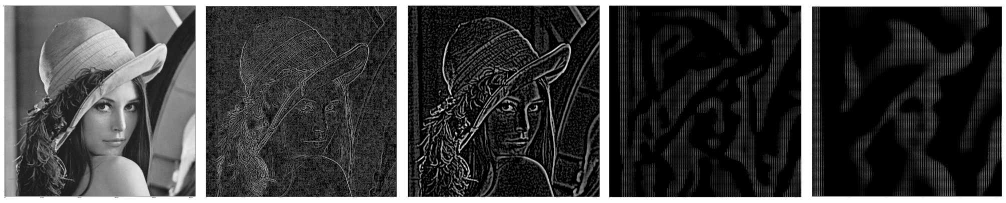
$$C(x,y,w) = \sum_i \sum_j \left(I(i+x, j+y) \cdot DoG_w(i,j) \right) \tag{2}$$

This will provide a set of coefficients C for every scale w. The authors in [15] refer to this as a wavelet-like transformation. The value of the coefficient will mean how soon does the neuron fire. If all $c_{i,w}$ are sorted according to their value, we shall posses a rank-ordered spikes set.

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

Table 1: Simulation parameters for ganglion cells

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



(a) Original image (b) Midget OFF-centre (c) Midget ON-centre (d) Parasol OFF-centre (e) Parasol ON-centre

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

7.3 Parallel implementation

The retinal model described in section 7.2 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 [8].

7.4 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 [17, 10].

Algorithm 7.1: NAÏVE CONVOLUTION(*image* I, *kernel* K)

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

7.5 Coding optimization

The first aspect to consider in optimizing code 7.1 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 7.1 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.

7.5.1 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).

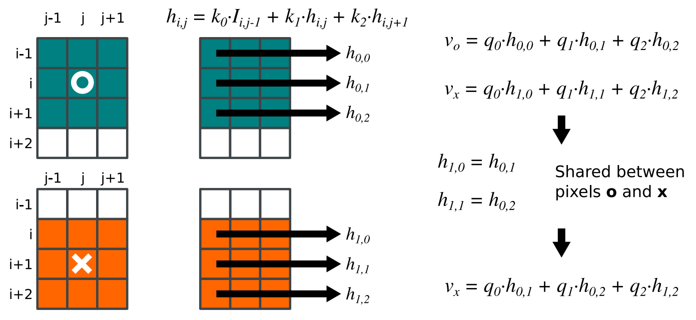
$$C(x,y,w) = \sum_i \sum_j \left(\left[\pm \frac{1}{2\pi\sigma_{w,c}^2} e^{\frac{-(i^2+j^2)}{2\sigma_{w,c}^2}} \mp \frac{1}{2\pi\sigma_{w,s}^2} e^{\frac{-(i^2+j^2)}{2\sigma_{w,s}^2}} \right] I(i+x, j+y) \right) \tag{3}$$

$$= \pm \left[\frac{1}{2\pi\sigma_{w,c}^2} \sum_i e^{\frac{-i^2}{2\sigma_{w,c}^2}} \sum_j e^{\frac{-j^2}{2\sigma_{w,c}^2}} I(i+x,j+y) \right]_c \mp \left[\frac{1}{2\pi\sigma_{w,s}^2} \sum_i e^{\frac{-i^2}{2\sigma_{w,s}^2}} \sum_j e^{\frac{-j^2)}{2\sigma_{w,s}^2}} I(i+x,j+y) \right]_s \tag{4}$$

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 $O(N + M)$.

7.5.2 Tiled convolution

The final step is an optimization published by Advanced Micro Devices (AMD) described in [14]. 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 process. First a horizontal convolution step that creates a set of coefficients ($h_{y,x}$ in the middle of Fig. 4). Most h coefficients are shared by immediate vertical neighbours (e.g. pixels $\circ$ and $\times$ in Fig. 4), thus it is desired to reuse them instead of re-computing them. This way, every core in the GPU will compute 2 pixels in order to efficiently compute a



j-1 j j+1
i-1
i
i+1
i+2
$h_{i,j} = k_0 \cdot I_{i,j-1} + k_1 \cdot h_{i,j} + k_2 \cdot h_{i,j+1}$
$h_{0,0}$
$h_{0,1}$
$h_{0,2}$
$v_o = q_0 \cdot h_{0,0} + q_1 \cdot h_{0,1} + q_2 \cdot h_{0,2}$
$v_x = q_0 \cdot h_{1,0} + q_1 \cdot h_{1,1} + q_2 \cdot h_{1,2}$
$h_{1,0} = h_{0,1}$
$h_{1,1} = h_{0,2}$
Shared between pixels **o** and **x**
j-1 j j+1
i-1
i
i+1
i+2
$h_{1,0}$
$h_{1,1}$
$h_{1,2}$
$v_x = q_0 \cdot h_{0,1} + q_1 \cdot h_{0,2} + q_2 \cdot h_{1,2}$

Figure 4: Tiled convolution flow

2D convolution; plus most of this operations are done using registers which is the fastest memory available to the cores.

7.5.3 Comparison

The Naïve approach seems to be much faster than any other attempts for the smallest kernel (3×3), this is simply because a single DoG kernel would require 9 operations while a separated DoG requires 12. As the size of the kernels increases tiled convolution has the lead performance-wise.

Table 2: Time comparison of different convolution algorithms.

Algorithm	Midget Off-centre	Midget On-centre	Parasol Off-centre	Parasol On-centre
Naïve	0.000932 s	0.003150 s	0.058797 s	N/A[1]
Separated	0.002953 s	0.005550 s	0.017261 s	0.047226 s
Tiled	0.001947 s	0.002722 s		

7.6 Lateral inhibition

Redundancy is present in biology to ensure robustness of the system. Nature has developed systems with little data transmission redundancy. A way to reduce redundancy is through lateral inhibition.

7.6.1 Filter overlap correction algorithm (FoCal)

Neighbouring pixels may contain very similar information; this redundancy is dealt in the eye, prior to ganglion cell spiking, with a mechanism known as lateral inhibition. In [12], Sen and Furber present an algorithm that simulates this behaviour, this process is done after all ganglion cells have been simulated. We shall treat the biggest value as the one to spike first, then adjust spatially near pixels (in every resulting image) with the correlation of the kernels in different images (Algorithm 7.2). This process is repeated until we run out of pixels with positive values and list P is full of corrected coefficients.

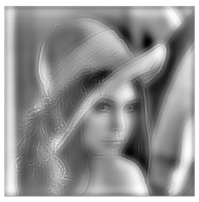
Algorithm 7.2: FOCAL(*coefficients S*, *correlations K*)

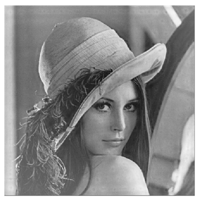
```
$P \leftarrow \emptyset$
while $S \neq \emptyset$
  do $index \leftarrow max(S),\ value \leftarrow val(S, index)$
     $remove(index, S)$
     $append(P, index, value)$
     for each $c \in K \neq 0$
          for each $p \in near\_pixels(index, S)$
               $S(p) \leftarrow p - c * value$
return $(P)$
```

7.6.2 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 (Fig. 5a). If no lateral inhibition is simulated, the resulting picture is similar to the original but has redundant information all over the place as seen in Fig. 5b. After correcting for redundancy the resulting reconstruction is much better, in fact with just 30% of the spikes we can obtain almost a 100 % of the visual information[12].

(a)
Original image


(b)
100% raw spikes


(c)
100% *corrected* spikes

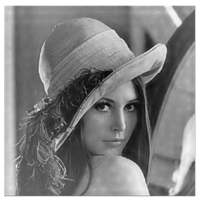

(d)
30% of *corrected* spikes

Figure 5: Results of reconstruction procedure

8 Thesis Outline

A draft thesis outline is presented in Appendix A

9 Conclusions and further work

Obtained further knowledge about basic anatomy of the eye from a functional approach. Importance of mutual inhibition to enable high efficiency computing in the brain and robustness of neural structure.

Real-time, although with low temporal resolution, is achievable with common GPU and the right combination of mathematics and engineering. Memory reads and writes in a GPU is extremely important; it might be one of the biggest constraints of the presented algorithms.

We established a timing mechanism to emit spikes from a rank-ordered source. Possible solutions for a faster mutual inhibition algorithm might be to do it in-line as we send spikes to neuromorphic hardware; or let the neural simulation deal with mutual inhibition.

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

References

[1] *Anatomical retina imagery source. Modified to fit needs.* 2015. URL: http://webvision.med.utah.edu/.

[2] 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. URL: http://www.sciencedirect.com/science/article/pii/S0303264798000501.

[3] 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/.

[4] 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/.

[5] J.A. Leñero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco. "A Five-Decade Dynamic-Range Ambient-Light-Independent Calibrated Signed-Spatial-Contrast AER Retina With 0.1-ms Latency and Optional Time-to-First-Spike Mode". In: *Circuits and Systems I: Regular Papers, IEEE Transactions on* 57.10 (2010), pp. 2632–2643. ISSN: 1549-8328. DOI: 10.1109/TCSI.2010.2046971.

[6] P. Lichtsteiner, C. Posch, and T. Delbruck. "A 128 x 128 120 dB 15 us Latency Asynchronous Temporal Contrast Vision Sensor". In: *Solid-State Circuits, IEEE Journal of* 43.2 (2008), pp. 566–576. ISSN: 0018-9200. DOI: 10.1109/JSSC.2007.914337.

[7] Carver A. Mead and M.A. Mahowald. "A silicon model of early visual processing". In: *Neural Networks* 1.1 (1988), pp. 91 –97. ISSN: 0893-6080. DOI: http://dx.doi.org/10.1016/0893-6080(88)90024-X. URL: http://www.sciencedirect.com/science/article/pii/089360808890024X.

[8] A. Munshi et al. *OpenCL Programming Guide.* OpenGL. Pearson Education, 2011. ISBN: 9780132594554.

[9] 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.

[10] Khairi Reda. *A study of OpenCL image convolution optimization.* 2012. URL: https://www.evl.uic.edu/kreda/gpu/image-convolution/.

[11] Basabdatta Sen. "Information Recovery From Rank-Order Encoded Images". Doctor of Philosophy Thesis. Faculty of Engineering and Physical Sciences, University of Manchester, 2008.

[12] Basabdatta 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.

[13] 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. URL: http://www.sciencedirect.com/science/article/pii/S0893608001000831.

[14] *Tiled Convolution: Fast Image Filtering.* 2015. URL: http://developer.amd.com/resources/documentation-articles/articles-whitepapers/tiled-convolution-fast-image-filtering/.

[15] 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.

[16] 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.

[17] www.cmsoft.com.br. *Case study: High performance convolution using OpenCL __local memory.* 2015. URL: http://www.cmsoft.com.br/opencl-tutorial/.

A Thesis table of contents

- List of Figures
- List of Tables
- Abbreviations
- Abstract
- Declaration
- Copyright
- Acknowledgements
- Chapter 1. Introduction.
- Chapter 2. Background.
- Chapter 3. Methodology.
- Chapter 4. Results.
- Chapter 5. Conclusions and Further Work.
- References.

B Project Plan

First year	Learn background concepts (biological and computationally, AI, etc.), strengthen research proposal. Familiarize with the SpiNNaker platform. Create data structures and an initial mapping of biological visual circuity. Start implementation on the SpiNNaker environment.
Second year	Develop computer vision algorithms for the SpiNNaker.
Third year	Complete implementation and test it on different tasks that current computer vision offers solutions. Start to write thesis dissertation
Fourth year	Finish thesis dissertation