FPGA-ACCELERATED PRE-ATTENTIVE SEGMENTATION IN PRIMARY VISUAL CORTEX

Christos-Savvas Bouganis, Peter Y. K. Cheung

Electrical and Electronic Engineering
Imperial College London
London, U.K.
email: {ccb98, p.cheung}@imperial.ac.uk

Li Zhaoping

Department of Psychology
University College London
London, U.K.
email: z.li@ucl.ac.uk

ABSTRACT

Visual attention systems inspired by the behavior of neural architectures have attracted the attention of many researchers in the computer vision field. Of special interest is the model proposed by Li where the bottom-up saliency features of an image are detected through a mechanism that simulates the operation of the primary visual cortex (V1). Beyond its biological nature, the specific model is also of interest because it performs texture segmentation and contour enhancement using the same circuitry. The main drawback of the proposed model is its computational complexity, making it time consuming to simulate the model in software to, e.g., explore the model parameters, and also limits its applicability in real-time scenarios. In this work, we explore the inherent parallelism that exists in the model and propose a flexible hardware architecture that can accelerate the model. Moreover, the flexibility of the proposed architecture to adapt to similar models of the brain is of significant concern. Performance evaluation shows that the proposed architecture gives results close to the software model, achieving at the same time a speed up of one order of magnitude.

1. INTRODUCTION

The human visual system has the ability to capture a vast amount of information but only a fraction of this reaches higher level of processing in the human brain. This is due to its remarkable ability to focus its attention only to parts of the image that are more ”interesting”. The low-level features of the image that attract the human eye have been the focus of many researchers in recent years, and many computational models have been proposed [1, 2, 3]. Many researchers from different fields have focused on employing these bottom-up attention models for applications that are related to humans. Such applications can be found in the field of communications, where the non-salient parts of the image can be compressed more heavily than the salient parts [4], in computer graphics applications where high-fidelity selective rendering can be achieved based on the output of the saliency model [5], and in 3D reconstruction using a pair of cameras, where the most salient points between the two images have to identified and the correspondence among them has to be established [2], to name a few.

A number of pre-attentive saliency models can be found in the literature. Itti *et. al.* [1] suggest a visual attention system that is inspired by the neural architecture of the early primate visual system. Their system combines information using three distinct channels; the color channel, the intensity channel, and the orientation channel. In order to process the input image in multiple scales, the proposed system creates nine spatial scales using dyadic Gaussian pyramids. These channels are further processed by a set of linear ”center-surround” operations, similar in operation to the visual receptive fields. That is, the output of a neuron is more sensitive in the center of its region, where stimuli from the surrounding regions inhibit the neural response. They argue that this type of architecture is well suited for prediction of regions that stand out from the neighboring regions. The transformed channels are then processed through a normalization operator, which is responsible to find the regions in a channel that exhibit higher levels of ”activity” than the average ”activity” level in the channel. The final channels are combined together and a winner-take-all (WTA) process is applied. In order the model to be able to ”focus” in different regions in the image, an ”inhibition of return” mechanism has been inserted to the model. This mechanism resets the activity of the previous WTA neurons allowing other regions to ”pop-out”. Such a mechanism has been also demonstrated in human visual psychophysics [1]. Experiments have shown that the above model can predict a number of ”pop-out” phenomena but its feed-forward feature-extraction architecture limits its performance. Moreover, due to non existence of recurrent mechanism, the model can

[1]The authors would like to acknowledge the help of Dr. Roberta Piroddi for contributing part of the software.

[2]The work was supported by the UK Research Council under the Basic Technology Research Programme ”Reverse Engineering Human Visual Processes” GR/R87642/02, and by the EPSRC (EP/C549481/1) grant.

not predict phenomena like contour-completion or closure.

Kadir and Brady [3] proposed a model where they argue that a saliency map algorithm should be based on visual saliency, scale selection and content description factors. They base the visual saliency part of their algorithm in the definition given by Gilles [6], where visual saliency is defined in terms of local signal complexity or unpredictability. The Shannon entropy of local attributes is used as a measure of local saliency. They also suggest that a multi-scale representation of the input image should be used, which allows the signal to be analyzed in different scales. However the appropriate scale for each point has to be estimated.

Apart from these models, which are more popular in the computer vision community, there are some models in Cognitive Psychology field that aim to model the human visual saliency process. One of these models has been proposed by Li [7]. Li suggests a model for pre-attentive segmentation, which is based only on the intensity information, and it is implementable by V1 mechanisms[1]. One of the advantages of the proposed model is that it performs texture segmentation using the same circuit that is used for contour enhancement. This model is of special interest because it addresses the pre-attentive visual search problem using a biological plausible model of V1.

The computational complexity of the Li's model prohibits any simulation of the model in software to explore the model parameters using real images or its application in real-time scenarios. However, its inherent parallel structure makes it a good candidate for hardware acceleration. The aim of this work is to accelerate the proposed model using FPGA technology allowing a certain degree of flexibility to the algorithm for further parametrization and adaptation to similar biological models.

In this paper we propose a flexible parameterized architecture for implementing the V1 model in FPGA achieving at the same time a considerable acceleration. The architecture is based on defining a "complete" set of operators, which are fully pipelined and parameterizable, that can describe the V1 mechanisms. The main advantage lies in the simplicity of extending the current set of operators, making the design able to be used by other similar algorithms.

The paper is organized as follows: In section 2, a detailed description of the V1 model is given. Section 3 describes the details of the proposed architecture where section 4 evaluates the performance of the design and compares it against the software version. Finally, section 5 concludes the paper.

[1]The primary visual cortex (V1) is the first cortical visual area. It receives information directly from the lateral geniculate nucleus (LGN) of the thalamus, which receives information directly from the retina, and sends output to superior colliculus to direct eye movements and to visual area V2 for other processes.

2. V1 MODEL

The V1 model proposed by Li [7], contains models of neurons tuned to a specific orientation and spatial location. Each unit (i, θ) has it center at location i and is tuned to orientation θ. Interneuron connections allow neurons to interact with each other. These connections tend to link together neurons that are tuned to similar orientations, giving specific contextual influences including co-linear facilitation and iso-orientation suppression [7]. Moreover, the model includes an activity normalization process which is responsible for orientation unspecific suppression. The response output of each neuron depends on the stimuli to the neuron itself, but also on the surrounding neurons through the interneuron connections.

In more details, at each location i, there are twelve neuron pairs. Each pair (i, θ) consists of an excitatory neuron and an inhibitory neuron, whose potentials are noted by x and y respectively, interconnected with each other and are tuned to an orientation θ. The input to the model is an edge map of the image. In the current implementation of the V1 model, twelve discrete possible orientations are selected, thus $\theta = 0^0, 15^0, ..., 165^0$.

Based on experimental data, interneuron connections $J_{i\theta,j\theta'}$ model excitatory monosynaptic connections, and link co-linear neurons, where the interneuron connections $W_{i\theta,j\theta'}$ model inhibitory disynaptic connections and link non co-linear neurons tuned to similar orientations.

The model uses two first order differential equations to describe the membrane potentials x and y as follows:

$$\begin{aligned} \dot{x_{i\theta}} &= -\alpha_x x_{i\theta} - \sum_{\Delta\theta} \psi(\Delta\theta) g_y(y_{i,\theta+\Delta\theta}) + J_o g_x(x_{i\theta}) \\ &\quad + \sum_{j \neq i, \theta'} J_{i\theta,j\theta'} g_x(x_{j\theta'}) + I_{i\theta} + I_o \end{aligned} \quad (1)$$

$$\dot{y_{i\theta}} = -\alpha_y y_{i\theta} + g_x(x_{i\theta}) + \sum_{j \neq i, \theta'} W_{i\theta,j\theta'} g_x(x_{j\theta'}) + I_c \quad (2)$$

where $\alpha_x x_{i\theta}$ and $\alpha_y y_{i\theta}$ model the decay to resting potential, $g_x(x)$ and $g_y(y)$ are sigmoid-like functions modelling cells' firing rates in response to the membrane potentials given in (3) and (4), $\psi(\Delta\theta)$ is the spread of inhibition (5), $J_o g_x(x_{i\theta})$ is self-excitation, I_c models the noise in the system, and I_o models the local normalization of activities (6). $I_{i\theta}$ corresponds to the input of the model and is a function of the magnitude of the edge at point i and its orientation. Temporal averages of $g_x(x_{i\theta})$ are used as the model's output.

$$g_x(x) = \begin{cases} 0 & \text{if } x < P_6 \\ x - P_6 & \text{if } P_6 \leq x \leq P_6 + 1 \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

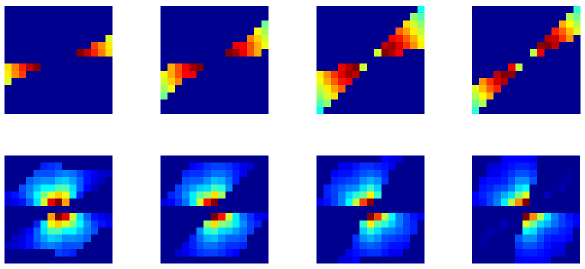


Fig. 1. Top row illustrates four $J_{i,\theta,0,0}$ for $\theta = 0, 15^0, 30^0, 45^0$ out of the twelve possible excitation masks. Bottom row illustrates four $W_{i,\theta,0,0}$ for $\theta = 0, 15^0, 30^0, 45^0$ of the inhibition masks.

$$g_y(y) = \begin{cases} 0 & \text{if } y < 0 \\ P_7 y & \text{if } 0 \le y \le P_8 \\ P_9 y + (P_7 - P_9)P_8 & \text{otherwise} \end{cases} \tag{4}$$

$$\psi(\theta) = \begin{cases} 1 & \text{if } \theta = 0^0 \\ P_2 & \text{if } |\theta| = 15^0 \\ P_3 & \text{if } |\theta| = 30^0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$I_o = P_4 - P_5 \left(\frac{\sum_{j \in S_i} \sum_{\theta'} g_x(x_{j\theta'})}{12} \right)^2 \tag{6}$$

A small subset of the excitation $J_{i\theta,j\theta'}$ and inhibition $W_{i\theta,j\theta'}$ kernels is illustrated in Figure 1. For more details about the formulae of J and W, and the value of parameters P_i the reader is referred to [7].

3. PROPOSED DESIGN

The proposed design architecture addresses the two important issues of the above problem, flexibility in the design to accommodate new pre-attentive saliency algorithms and achieving an adequate performance.

The system design is divided into two parts. The first part is the edge detection and is performed in software. The second part, the more computational expensive, is the V1 saliency computation and is implemented in hardware.

In more detail, the design flow is as follows. A camera captures the input image and transmits the data to a PC. Initially, a median filter of 3×3 is applied to the image to reduce the noise. Then, the gradient of the image is calculated (edge detection step). The resulting image is transmitted to the FPGA through a USB 2.0. The FPGA simulates the differential equations (1) and (2) for a number of iterations, and returns the resulting saliency map to the PC for display.

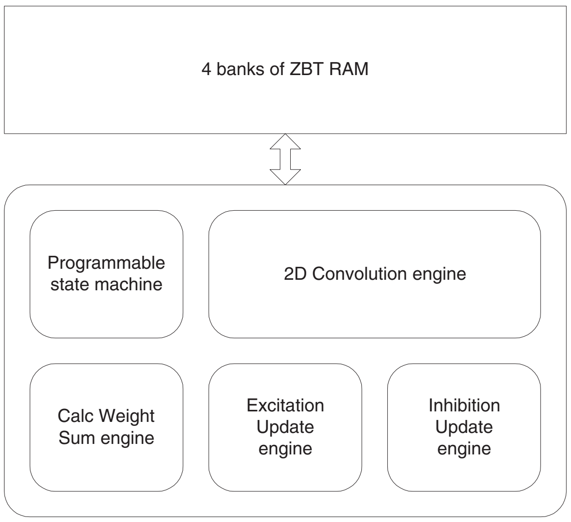


Fig. 2. Top level design

The hardware design of the system is based on a processor-memory type design, where the system comprises a set of operators that are applied in a pipelined fashion to the whole image and the results are stored back to the RAMs. The communication between different operators is performed through the available RAMs. Thus, a flexible design is achieved, where the sequence of operations can be altered or even new operators/engines can be added in order new algorithms to be accommodated by the system.

The target board is the RC300 from Celoxica. It is fitted with a XC2V6000, and comprises 4 banks of ZBT RAM. Each bank can store up to 8MB. The proposed design takes into account the above restrictions.

3.1. Top-level architecture

Investigating the two core equations of the system, four main building blocks can be identified. The first block is a convolution engine module with the feature to perform a 2D convolution with any set of coefficients. The excitation and the inhibition updates are performed by two separate engines, in order for the design to be able to perform the update operations in parallel. An engine that produces the weighted sum of different images is also included in the set of operators. The main difference between this engine and the rest is that input and output data are stored in the same RAM. The final engine is a programmable state machine which is responsible to control the other engines. A top-level diagram of the system is illustrated in Figure 2.

3.1.1. Convolution engine

The convolution engine comprises the main engine of the design. According to the algorithm's specifications, synaptic connections should extend up to 7 neurons, implying that a convolution core with a 15×15 mask is required. The convolution core has the ability to be programmed to any set of coefficients and can read/write data from/to any RAM of the system. The input to the system can be transformed by either of the sigmoid-like functions $g_x(x)$ or $g_y(y)$, allowing maximum flexibility to the user. Moreover, the engine has the ability to add data to the result of the convolution and store the final result to a memory. This is very useful in the present application for the implementation of the double sums as in $\sum_{j \neq i, \theta'} J_{i\theta, j\theta'} g_x(x_{j\theta'})$ and $\sum_{j \neq i, \theta'} W_{i\theta, j\theta'} g_x(x_{j\theta'})$.

In general, the convolution engine can apply the following computation in a pipelined form providing one output per clock cycle:

$$O = K \otimes g.(D_1) + D_2 \tag{7}$$

where $\otimes$ represents a two-dimensional convolution, K represents the convolution kernel, and D_1 and D_2 represent the data. $g.(\cdot)$ represents a sigmoid-like function.

3.1.2. Update Excitation and Inhibition engines

The update excitation and inhibition engines are responsible for updating the excitation and inhibition potentials of each neuron of the system. They can be programmed to read any RAM of the system up to three in parallel, weight the data by programmable coefficients, and store the results in the fourth memory. Due to the limited number of memories in the RC300 system, two passes are required in order to update the excitation and inhibition potentials of the neurons.

3.1.3. Calc_Weight_Sum engine

The Calc_Weight_Sum engine can be programmed to perform a weighted sum of images that are placed in the same RAM, and store the result in the same RAM. The weights and the starting addresses of the images, along with the store address are downloaded to the engine. The existence of such an engine is necessary due to limited available memory bandwidth. In the current algorithm, the engine is used to calculate the weighted sum $\sum_{\Delta\theta} \psi(\Delta\theta) g_y(y_{i,\theta+\Delta\theta})$.

3.1.4. Programmable state machine

The control of the system is performed through a programmable state machine. The user can download commands to the system which are executed by the state machine. These commands configure the different engines of the system. The overhead for programming the engines of the system during run-time is overcome by the pipeline fashion that they work.

3.2. Determining signal word-length

In order to determine the necessary word-length of the signals at the different parts of the design each engine is examined alone. For the convolution engine, it has been assumed that the possible set of coefficients have magnitude less or equal to one. The convolution is a linear operator, thus the maximum value at the output of the convolution engine can be calculated as in (8), where C_i is the i^{th} coefficient of the convolution mask and O_{max} is the maximum value at the output of the mask. The maximum achievable value is 225, thus 9 bits for the integer part are enough to describe the output of the convolution engine.

$$O_{max} = \sum_{i=1}^{225} g_x(x_i) C_i \leq \sum_{i=1}^{225} |C_i| \leq 225 \tag{8}$$

Simulations with different images were performed in order to determine the necessary number of bits for the fractional part of the output of the convolution engine. A number of 7 bits was enough for the hardware version of the algorithm in order to give results visually close to the ones from the floating point version of the algorithm.

The word-lengths of the rest of the signals were determined using simulations. The variables were fixed to 20 bits, where 13 of them are used for the integer part.

3.3. Memory management

The memory management of the system is shown in Figure 3. The proposed configuration was selected in order to optimize the available memory bandwidth and complete one iteration of the algorithm in the least possible time. Of special interest is the fact that the code that is executed by the programmable state machine is copied in the initialization phase to an on-chip RAM, decoupling the control unit from the availability of the on-board RAMs.

The design has to keep in memory 24 maps of the neurons, where 12 of them keep track of the excitation and the other 12 keep track of the inhibition potentials. By distributing the excitation and the inhibition maps to different memory banks, the design can process them in parallel, utilizing the available memory bandwidth from the RC300 board. The rest two RAMs are used by the architecture to store intermediate results.

3.4. Job scheduling

By examining the two core equations (1) and (2), can be concluded that 216 convolutions between the exhibitory/inhibitory maps and a 15×15 mask are required. Due to the existence of a single convolution engine, these convolutions have to be performed sequential. The optional feature of adding another vector to the output of the convolution is applied in the current application for the implementation of the double

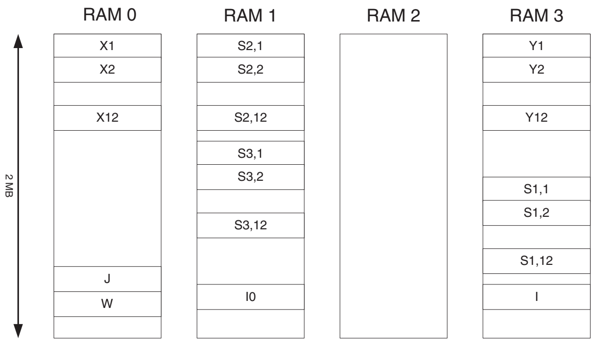


Fig. 3. Memory management. The excitation maps Xi and the inhibition maps Yi of the neurons, with $i = 1...12$ are stored to different memories to permit parallel accesses. The $S2i$ entries correspond to $\sum_{j\neq i,\theta'} J_{i\theta,j\theta'} g_x(x_{j\theta'})$ for the twelve different values of θ'. The $S3i$ entries correspond to $\sum_{j\neq i,\theta'} W_{i\theta,j\theta'} g_x(x_{j\theta'})$ where the $S1i$ entries correspond to $\sum_{\Delta\theta} \psi(\Delta\theta) g_y(y_{i,\theta+\Delta\theta})$.

sums. This implies that three memory banks are utilized by the engine. The calculation of the $\sum_{\Delta\theta} \psi(\Delta\theta) g_y(y_{i,\theta+\Delta\theta})$ is performed in parallel with the calculations of $\sum_{j\neq i,\theta'} J_{i\theta,j\theta'} g_x(x_{j\theta'})$ and $\sum_{j\neq i,\theta'} W_{i\theta,j\theta'} g_x(x_{j\theta'})$.

Figure 4 illustrates the scheduling of the jobs along with the memory bandwidth utilization for estimation of the V1 saliency map for one iteration of the algorithm. It is clear that most of the time the algorithm uses 100% of the available memory bandwidth, making that the limiting factor of its performance.

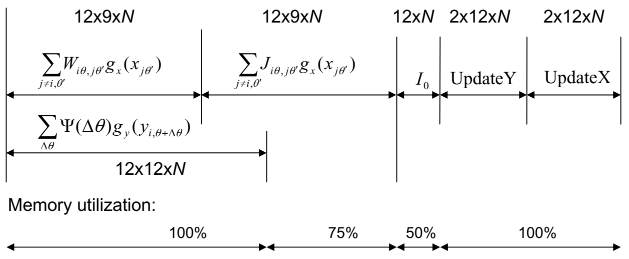
4. PERFORMANCE EVALUATION

The system was implemented on an RC300 board fitted with a XC2V6000 FPGA. The Handle-C language from Celoxica was used due to fast prototyping. The Xilinx tools were used for placement and routing.

4.1. Area and speed

The required area and the achieved speed of the design are illustrated in Table 1 along with a comparison to the software model of V1. An optimized software version of the algorithm was implemented for comparison and testing reasons. The software version of the algorithm requires 3.14 seconds for one iteration for a 256×256 input image, running in a Pentium 4 at 3.2GHz. In general, for the case of 256×256 images, around 25 iterations are required for the convergence of the algorithm. This corresponds to 78.5 seconds for a full run of the algorithm.

The hardware model can perform one iteration of the algorithm in 0.31 seconds. This corresponds to 7.8 seconds



12x9xN
12x9xN
12xN
2x12xN
2x12xN
$\sum_{j\neq i,\theta'} W_{i\theta,j\theta'} g_x(x_{j\theta'})$
$\sum_{j\neq i,\theta'} J_{i\theta,j\theta'} g_x(x_{j\theta'})$
I_0
UpdateY
UpdateX
$\sum_{\Delta\theta} \Psi(\Delta\theta) g_y(y_{i,\theta+\Delta\theta})$
12x12xN
Memory utilization:
100%
75%
50%
100%

Fig. 4. Proposed job scheduling. N denotes the number of pixels in the image. Most of the time, the memory utilization is 100% making that the limiting factor for the performance of the algorithm. It should be noted that three out of the twelve excitation/inhibition masks have zero coefficients and thus they are not calculated by the architecture.

	HW	SW
Frequency	60MHz	3.2GHz
Area	23,206 slices	-
Sec/iter (256x256)	0.31	3.14

Table 1. Hardware versus software implementation of V1. For the SW, a Pentium 4 is used. The HW version is implemented in a XC2V6000. It should be noted that all the available embedded multipliers (144) have been used, where 22 out of the 144 block RAMs have been used.

for a full run of the algorithm, assuming 25 iterations and a 256×256 input image. Thus, the achieved speed up of the hardware version over the software version is around ten times. Although this does not permit a real-time performance of the algorithm when is executed in hardware, allows the researchers to investigate faster the impact of the parameters of the algorithm to the final saliency map.

The proposed architecture requires $23,206$ slices out of the available $33,792$ slices in the XC2V6000 FPGA. It should be noted that even that the required area is only 68% of the available area in the FPGA, the memory bandwidth utilization is almost 100%, imposing a bound to any further optimization. A different memory management strategy could achieve a better packing of the data in the memories, but this would lead to a less general design since any change in the word-length of the membrane potential variables would change the memory map, and consequently, some parts of the design.

4.2. Accuracy

The accuracy of the hardware was investigated using two test images, the lenna image and the baboon image. Figure 5 illustrates the input image (lenna), the output of the floating point software version, the output of the hardware, and

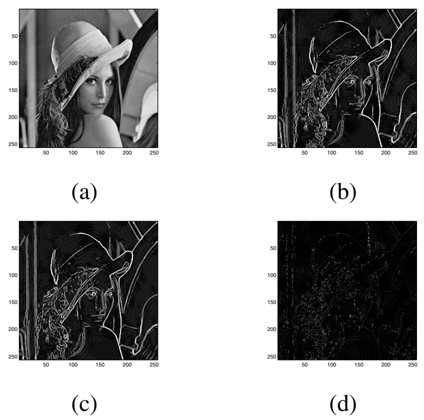


Fig. 5. Lenna image. (a) Input image, (b) output of the software model (25 iterations), (c) output of the hardware model (25 iterations), and (d) absolute error between (b) and (c). The intensities are scaled in the region [0,255], and the mean absolute error is 9.15.

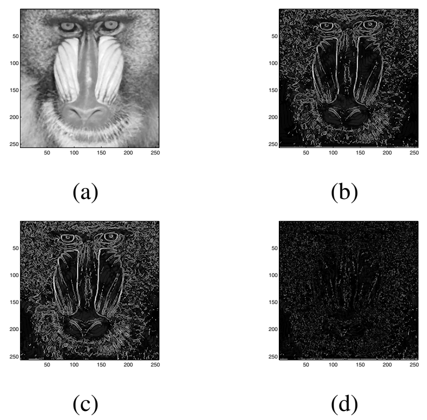


Fig. 6. Baboon image. (a) Input image, (b) output of the software model (25 iterations), (c) output of the hardware model (25 iterations), and (d) absolute error between (b) and (c). The intensities are scaled in the region [0,255], and the mean absolute error is 17.67.

the absolute error between the two output images. The images have been scaled up to 255 levels of intensity, and the mean absolute error achieved is 9.15. Focusing on the output images of the algorithm, the bright regions of the images correspond to areas that attract the attention, where the dark regions do not. It is clear that these regions are concentrated around the edges of the image. However, some edges give higher responses than others due to their interaction with other edges in their regions.

Figure 6 illustrates similar results for the test case of the baboon's image. The same scaling as in the lenna image has been applied and the mean absolute error is 17.67. The above test images, lead to the conclusion that the fixed point hardware version of the algorithm gives very similar results to the floating point software version and it can be used by the researchers to investigate the V1 mechanism in more detail, in a shorter time.

5. CONCLUSION

The present work focuses on the acceleration of the computational model for the primary visual cortex presented by Li [7]. The model is of special interest because of its biological nature and it ability to perform texture segmentation and contour enhancement using the same circuitry. The proposed architecture is based on a processor-memory type model which provides the required flexibility to adapt to similar saliency models, achieving at the same time significant acceleration. The evaluation of the proposed architecture illustrates that the hardware model can achieve similar results to the software version, achieving at the same time a speed up factor of one order of magnitude. Future work will involve the applicability of the proposed architecture to similar models like the model for border ownership detection from intra-cortical interaction in visual area V2 [8] by adding appropriate engines/operations to the presented architecture.

6. REFERENCES

[1] L. Itti, C. Koch, and E. Niebur, "A Model of Salinecy-Based Visual Attention for Rapid Scene Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, Nov. 1998.

[2] L. Zhaoping, "Pre-attentive segmentation and correspondence in stereo," *Philosophical Transactions of The Royal Society, Biological Sciences*, vol. 357, no. 1428, pp. 1877–1883, 2002.

[3] T. Kadir and M. Brady, "Saliency, Scale and Image Description," *International Journal of Computer Vision*, vol. 45, no. 2, pp. 83–105, 2001.

[4] T. E. Slowe and I. Marsic, "Saliency-Based Visual Representation for Compression," in *IEEE International Conference on Image Processing*, Oct. 1997, pp. 554–557.

[5] P. Longhurst, K. Debattista, and A. Chalmers, "A GPU based Saliency Map for High-Fidelity Selective Rendering," in *AFRIGRAPH 2006 4th International Conference on Computer Graphics, Virtual Reality, Visualization and Interaction in Africa, ACM SIGGRAPH*, Jan. 2006, pp. 21–29.

[6] S. Gilles, "Robust description and matching of images," Ph.D. Thesis, University of Oxford, 1998.

[7] Z. Li, "Visual segmentation by contextual influences via intracortical interactions in primary visual cortex," *Network: Computation in Neural Systems*, vol. 10, pp. 187–212, 1999.

[8] L. Zhaoping, "Border Ownership from Intracortical Interactions in Visual Area V2," *NEURON*, vol. 47, pp. 143–153, 2005.