A Real-time FPGA-based Vision System for a Bionic Eye

Horace Josh, Benedict Yong, Lindsay Kleeman
Monash Vision Group and Department of Electrical and Computer Systems Engineering, Monash University
Wellington Road, Clayton, Australia
{horace.josh, benedict.yong, lindsay.kleeman}@monash.edu.au

Abstract

Visual prosthesis simulators aim to provide a means of demonstrating, evaluating and improving the results of artificial vision devices. They transform the normal view of a camera scene into a form that represents the visual perceptions that would be caused by applying electrical stimulation to a part of the human visual pathway. These perceptions are called phosphenes and, in a cortical visual prosthesis, can be elicited through stimulation of the neurons in the visual cortex. This paper presents an FPGA implementation of a real-time vision system that simulates this phenomenon. The system is low-cost, mobile and consists of a CMOS camera, FPGA development board, and a head-mounted display. The methods of implementation discussed in this paper are applicable to other intelligent mobile sensor and machine vision applications where speed, low latency and low power consumption are important factors.

1 Introduction

A visual prosthesis is an implantable medical device that aims to restore vision to people who are blind. The principle behind visual prostheses comes from early physiological studies [Brindley and Lewin, 1968; Dobelle and Mladejovsky, 1974; Dobelle et al., 1976; Bak et al., 1990] conducted which showed that electrical stimulation of the human brain resulted in bright spots of light, known as 'phosphenes', occurring in the patient's visual field. Later studies [Humayun et al., 1996] have also shown that electrical stimulation of the retina similarly also elicits phosphenes.

The key part of a visual prosthesis is an array of electrodes, driven by specialised electronics, which is used to inject electrical current and stimulate neurons in certain sections of the visual pathway. The electrode array can produce a number of phosphenes, each corresponding to an individual electrode, and these phosphenes will form an 'image' in the patient's visual field. Thus, the resolution of the 'image' produced depends directly on the number of electrodes in the visual prosthesis.

The two most common types of visual prosthesis are retinal implants and cortical implants. Retinal implants electrically stimulate cells in the retina at the back of the eyeball, earlier in the visual pathway. Cortical implants electrically stimulate cells in the visual cortex, the part of the brain primarily responsible for vision and later in the visual pathway.

In 2010, the Australian Research Council (ARC) funded a Special Research Initiative (SRI) to develop a functional visual prosthesis. One of the two proposals accepted was from a Monash University led group of researchers now known as the Monash Vision Group (MVG) [Monash University, 2010]. The MVG aim to develop a visual prosthesis based on a cortical implant, and is anticipated to incorporate approximately 600 electrodes.

As research progresses on a visual prosthesis, there is also an increasing need for the simulation and visualisation of possible results from a visual prosthesis. Visual prosthesis simulators can be utilised as platforms for the testing of image processing algorithms and adjustment of parameters before clinical trials of actual visual prostheses. Visual prosthesis simulators' possible uses include psychophysics experimentation, where normally sighted subjects would wear the simulator and attempt to complete a range of tasks with the limited vision. The basic components of a simulator would include a video camera input, image processing that translates the input image into an image that represents what a blind patient with an actual visual prosthesis would 'see', and a display for the user to view.

Recently produced simulators [van Rheede et al., 2010; Fehervari et al., 2010; Srivastava et al., 2009; Chen et al., 2005] have largely been driven by external computers and hence have restriction on movement of the user, and they can also suffer from latency and frame rate limitations. The system produced here consists of a Field Programmable Gate Array (FPGA) based system. FPGAs contain a large number of reprogrammable logic gates, and offer highly parallelizable, low latency implementation that can be easily interfaced with external devices. In this case, the FPGA serves as the image processing unit. An external CMOS camera serves as the input and is interfaced with the FPGA development board. A head-mounted display is used to display the output processed image to the user.

As a mobile cortical bionic eye simulator, our system provides a suitable platform for research into both the psychophysical and image processing aspects of visual prostheses. The implementation discussed in this paper, however, also applies to intelligent mobile sensor and machine vision applications that require speed, low latency and low power consumption.

2 System Overview

We have put together the mobile system shown in Figure 1. It consists of a camera sensor for input, FPGA development board for processing and a pair of virtual reality goggles for output. An infra-red remote control allows for manipulation of all the implemented modes and parameters and optionally an external VGA monitor can be used for dual or alternate display of output.

The whole system is powered by a 12V lithium ion battery and the majority of components are fastened within a hardened plastic laptop casing. Figure 2 shows the layout of components within the casing. We have

Figure 1: Mobile simulator system (left), Infra-red remote control (top right), Hard casing for processing hardware (bottom right).

selected a Terasic DE2-115 FPGA development board for our system available from www.terasic.com. This board is ideal for our implementation as it has a low power Altera Cyclone IV FPGA chip with a large number of available logic elements and on-chip memory as well as many external peripherals and I/O pins that allow for connection to devices such as cameras and displays.

As can be seen in Figure 2, the analogue video input and VGA output ports on the FPGA board have been extended to ports accessible from the outside of the casing. These ports provide power and signal connections to and from the camera, virtual reality goggles, external monitor, as well as the IR receiver sensor which is connected to an I/O pin on the board.

The camera sensor utilised for our system is a Sparkfun Electronics CM-26N/P CMOS camera with analogue signal output. Its small physical size, simple 3 wire power/signal connection, capability for longer cable lengths and low cost made it a good choice. Image data is captured by the camera with a resolution of 640x480 pixels at a frame rate of 60Hz. The data is passed onto the FPGA board where various processing takes place, before the data is finally sent to the virtual reality goggles for display. We use a Vuzix iWear VR920 head-mounted display for display of the output image data. This pair of virtual

Figure 2: Component layout within casing (top), external port connections for camera, displays and IR receiver (bottom).

reality goggles provides a 640x480 display resolution via VGA input and so was suitable for our system. The goggles are powered via a USB connection, which is provided by the FPGA development board.

The infra-red remote control (Figure 1) was included in the DE2-115 package and provided a simple way of acquiring control input from a user. In our system, all of the implemented functions can be toggled and configured using the IR remote.

3 System Implementation

Our system is implemented using Verilog hardware description language. Figure 3 shows a block diagram of the complete implementation of the system. Each block in the diagram represents one or more Verilog modules within our system.

Image data is received from the camera as an NTSC analogue signal. This is converted to a YUV 4:2:2 signal and stored in an SDRAM provided on the FPGA development board. The stored data is retrieved from the SDRAM and converted to RGB data which is passed to the processing modules we have created one pixel at a time. After leaving the processing modules, the data is passed to the VGA controller which handles the signalling for the VGA output port. The IR Receiver block interprets the remote control signals from the IR receiver sensor and generates a function code which is sent to a control block that interfaces with all the processing modules in order to toggle modes or configure parameters.

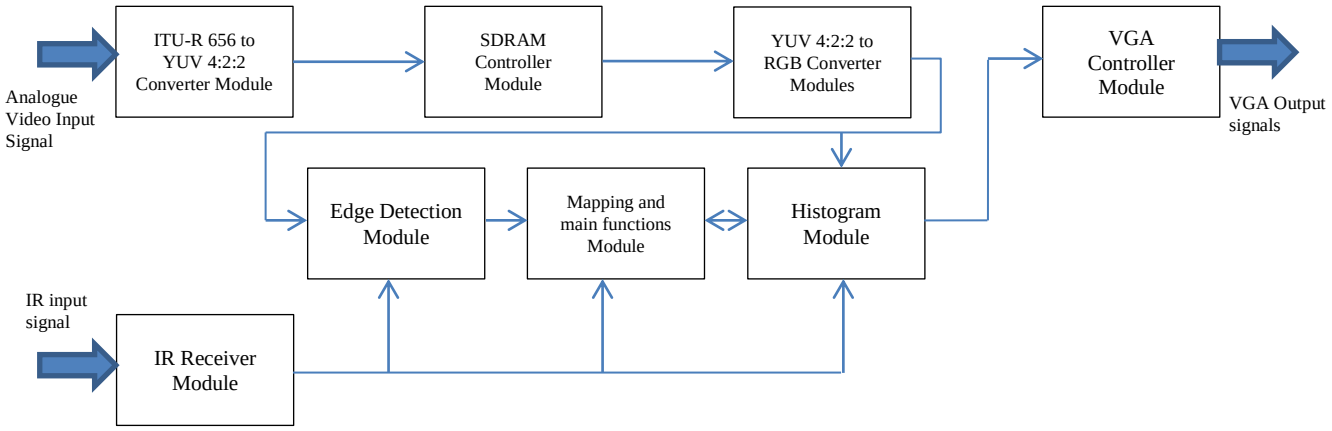

Figure 3: Block diagram of our Verilog implementation.

3.1 Visuotopic Mapping

It has been proven in early research [Schwartz, 1977; Wandell et al., 2007] that a link exists between points of stimulation on the visual cortex and specific phosphene locations in the visual field. This finding along with subsequent research [Horton and Hoyt, 1991; Duncan and Boynton, 2003] which support it have shown that a non-linear map or transfer function is inferable (visuotopic map).

Schwartz [1977] indicated in his work that an approximation to this visuotopic mapping is via a 'log-polar' representation. A number of mathematical models have been developed to represent this mapping. The most well-known are the Monopole model [Schwartz, 1977; Polimeni et al, 2006; Schira et al, 2010], the Wedge-Dipole model [Balasubramanian et al, 2002; Polimeni et al, 2006], and the Double-Sech model [Schira et al, 2007; Schira et al, 2010].

Assuming an implant placed in the primary visual cortex (V1), closer to the foveal region, we have chosen to implement the monopole model in our system. This model is mathematically simpler, yet still reasonably accurate. A simple explanation of the Monopole model equations is given in [Fehervari et al., 2010]. Due to the shape and structure of the brain, there are physical limitations on electrode placement. These limitations will result in perceivable 'gaps' or areas without phosphene elicitation in the visual field of a patient. This has been taken into account in our system implementation and is quite evident in the diagrams shown in the results section.

Our implementation of the described mapping involves the use of a number of lookup tables (LUTs) and innovative sequential logic that utilises the information provided by these lookup tables. In order to transform the input frame data into the low resolution visuotopically mapped frame, the incoming frame pixels need to be sampled, averaged and stored, ready for display during the next output frame. Figure 4 shows a block diagram layout of the mapping implementation (excluding intensity transformations, Gaussian profile phosphene modelling, and dead electrode simulation).

The most resource-expensive part of this implementation (and of the whole system) is the v-map lookup table. This LUT stores a 10-bit number for each of its 230400 elements. Each of these elements corresponds to one pixel within the center 480x480 square of pixels in the input frame. The numbers stored in the elements correspond to the index numbers of phosphene that a particular pixel belongs to, ranging from phosphene 1 to phosphene 648. A value of zero is given to elements that

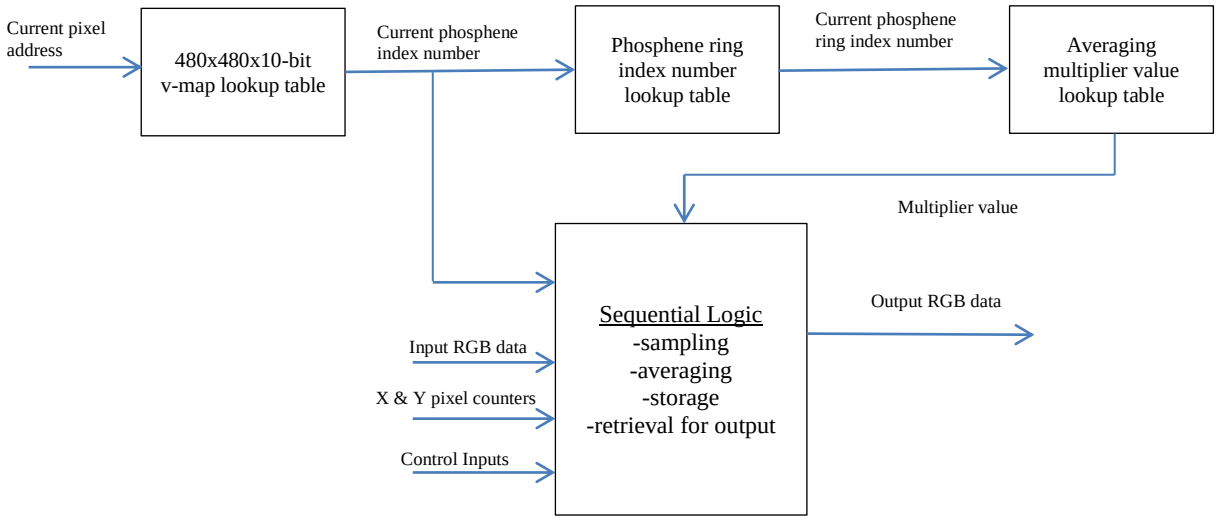

Figure 4: Mapping implementation (excluding intensity transformations, Gaussian phosphene modelling and dead electrode simulation).

correspond to background pixels (pixels that do not belong to a phosphene and will appear black in the output frame). MATLAB scripts were written and used to generate the arrangement of phosphenes and subsequently produce a memory initialisation file for the v-map lookup table. This innovative method of defining the mapping can be extended to almost any low resolution mapping with a simple redefinition of the LUT using MATLAB. This implementation could ideally be utilised by a system that requires low latency vision mapping.

With the knowledge of the phosphene number of the current pixel, sampling is made possible. The sampling method we employ in our system is an averaging sampler. For each phosphene, the values of all of its pixels are summed in a storage register that corresponds to that phosphene. After all phosphenes have been sampled in this way, the stored sums are averaged. Since division logic is quite expensive in hardware, our system uses a multiply and shift implementation to approximate the divide. This involves predefining a fraction that has a divisor which is a power of base 2. The numerator of the fraction will be the rounded result of dividing the number of pixels by the base 2 power. Averaging is then achieved by multiplying the stored sum by the numerator and bit shifting to the left by an amount equal to the power of the divisor. Since not all phosphenes are the same size, a lookup table is required to store the multiplier and divisor amounts for each phosphene. In our implementation, we have taken advantage of the fact that phosphenes within the same ring are all the same size. This allows for a cheaper implementation, albeit with two separate LUTs (ring number and averaging multiplier). We have also chosen to use the same divisor amount for all phosphenes to further simplify the implementation.

After averaging the sampled phosphene pixels, the result is stored in one of 648 10-bit storage registers. Each register corresponds to one phosphene and its contents will be accessed and displayed in the next output frame.

Display pixel data is provided by simply retrieving the value stored in the register that corresponds to the current pixel's phosphene number (provided by the v-map LUT). As discussed in later sections, a slightly more complex method is required when implementing Gaussian phosphene modelling, intensity transformations, and dead electrode simulation. The display pixel data is passed on to a subsequent VGA controller module which handles the interface to the physical VGA port.

3.2 Intensity Transformations

Early studies have shown that some level of intensity modulation of phosphene elicitation is possible [Brindley and Lewin, 1968]. Our system models this attribute through thresholding of greyscale intensity to a limited number of discrete levels.

Thresholding in our system is performed within the sequential logic block that appears in Figure 4. Before a frame's averaged pixel data is stored in display storage registers, its greyscale intensity is transformed according to a model with a highly reduced number of discrete levels. In the case of our system, this can be 2, 4 or 8 discrete levels. This transformation is achieved with a Verilog case statement based on the greyscale intensity value of the current pixel data that is to be stored. The full range of possible greyscale values was evenly divided in order to define thresholds. The case statement sets the final intensity value according to the highest threshold the greyscale intensity value has exceeded. The transformed intensity is then stored into its appropriate display storage register.

A key problem that arises with this implementation is high frequency oscillation of phosphene intensity value about its closest threshold. A very noticeable flicker will occur when the image is displayed as the difference in intensity of two adjacent levels is rather large. To overcome this we have implemented a simple hysteresis feature into the case statement. Each threshold is further divided into two 'paired thresholds', the original plus a predefined hysteresis value and the original minus the hysteresis value. The revised case statement checks for which two adjacent (non-paired) thresholds the greyscale intensity lies between and sets the final intensity accordingly. If the greyscale intensity value falls between two paired thresholds, then the value currently stored in the display storage register is unchanged.

3.3 Gaussian Phosphene Modelling

Literature shows that electrode stimulation of the visual cortex produces a bright spot of light in the patient's visual field [Brindley and Lewin, 1968]. A common approach to model approximately the look of a phosphene is with a 2D Gaussian masking function [Chen et al., 2009].

In order to implement this method on FPGA, the intensity of each pixel within a phosphene needs to be weighted according to its Euclidian distance from the center of the phosphene. A set of discrete weightings that follow a Gaussian curve was generated for each different size of phosphene. A number of lookup tables were generated using MATLAB scripts. Figure 5 outlines the use of these tables in our implementation.

In order to get the Euclidian distance of a particular pixel, we require the coordinates of the phosphene's center. The two LUTs (cent_pix_X and cent_pix_Y) achieve this, given that the phosphene number of the current pixel is available. The euc_dist LUT then provides a Euclidian distance value given the center pixel and current pixel coordinates. Taking as inputs the Euclidian distance and current ring number, the Gaussian_weights LUT then provides the weightings. To simplify the logic, a similar multiply and shift method was used for applying the weightings to pixels. This method is carried out in the sequential logic block on pixels that are to be output for display.

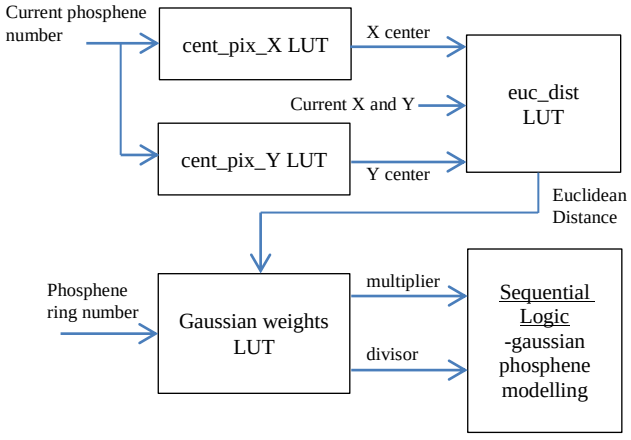

Figure 5: Gaussian phosphene modelling implementation.

3.4 Dead Electrode Simulation

It is very likely that after implantation of a visual prosthesis there could be electrodes that do not function or are not able to elicit phosphenes in the patient's visual field. To simulate this we have implemented a dead electrode or phosphene dropout feature.

A simple lookup table was used to implement this feature. Using MATLAB, 3 pairs of 648 normally distributed random binary numbers were generated to simulate 10%, 25%, and 50% of electrode failure. A MATLAB script was used to create a Verilog lookup table module (rand_masks) that would provide these random binary values as output. Taking as input the current phosphene number and selected dropout mode, the rand_masks LUT provides to the sequential logic block a binary value that would enable (binary 1) or disable (binary 0) the current phosphene from being displayed. Disabled phosphenes have a stored intensity of zero.

3.5 Frame Rate Control

It is anticipated that initial implementations of visual prosthesis will be limited in terms of temporal resolution. It is not yet known exactly what effective frame rate will be achievable by a cortical bionic eye implant. For this reason we have decided to simulate varying frame rates in our system.

The frame rate in our system is dependent on the rate at which data is stored into the display storage registers. Therefore, reducing the frame rate is simply a matter of limiting how often the data in these registers is updated. Our implementation keeps track of the number of frames that have passed in a frame counter register and updates the display storage at periods that correspond to the current selected frame rate. The frame rates achievable by the system are 1, 2, 3, 4, 8, 10, 15, 30 and 60 frames per second.

3.6 Histogram Assisted Binary Threshold Selection

A static threshold for binary greyscale conversion can be a problem in environments where lighting variation is severe. Although our system's camera employs a built-in automatic gain controller, it is sometimes not enough to compensate for such lighting variations. In order to mitigate this problem, information about the range and spread of pixel data within a frame is required. This can be achieved through the generation of a histogram containing pixel intensity data.

Since frame data in our system is only available one pixel at a time, implementation of a histogram was performed sequentially over the period of a single frame. Storage registers were used as histogram 'bins'. There are 20 bins, each of which store an 18-bit count of the pixels that fall within the bin's allocated intensity range. The intensity range of pixels in our system is 0 to 1023, therefore each histogram bin has its own allocated intensity range of 51 levels, apart from the final bin which has a range of 55 levels. Only pixels within the center 480x480 square of a frame are used to make up the histogram. For each of these pixels, the value in the appropriate bin storage register is incremented.

Once the histogram is constructed the bin with the maximum count is determined through the use of a conditional logic block. After this bin is found, an iterative approach is used to perform a discrete integral of the surrounding bins. Adjacent bin values are added one at a time and alternating from left to right of the bin with the max count. This method is performed over the remaining clock cycles in the frame until a predefined threshold is reached. At this point the bins with the highest and lowest valued intensity ranges in the integral are known and a median value can be calculated. The median value is used as the binary threshold.

3.7 Edge Detection

Extracting edge information from an image can often be a useful way to show salient information in a scene. When used as the input to a visual prosthesis simulator, the edge result represents the main structural information, which in an uncluttered environment, can be a bit more intuitive to the user.

A study done by [Dowling et al., 2004] compared the Sobel and Canny edge detection methods [Lee et al., 1987; Canny, 1986] with a visual prosthesis simulator in order to determine which would give more useful mobility information. The results showed no significant difference between the two methods, and concluded that the Sobel operator was more suited for a visual prosthesis system due to its low computational cost. Guided by these results we have chosen to implement Sobel edge detection.

In order to implement Sobel edge detection simultaneous access of a 3x3 window of pixels is required. In our system this meant that pixels needed to be buffered in a shift register. A 1920x10-bit shift register was used. This shift register buffers exactly three lines of pixels and has three taps or access points to the end pixels in each line. These end pixels are shifted into two consecutive sets of 10-bit registers, thus creating access to a 3x3 window of pixels from a frame. At each clock cycle a new pixel is inserted into the shift register, and the three pixels at the taps are shifted through the sets of 10-bit registers. Performing this method is equivalent to shifting the 3x3 window over every possible location in the frame in a sequential fashion. A discrete convolution of the Sobel edge operators is performed with the 9 available pixels (3 from taps and 6 from 10-bit registers) at each clock cycle and the result is stored in a temporary register. This value is then checked against a threshold in order to set the output edge image pixel values.

3.8 Infra-Red Remote Control

All functions within our system implementation can be controlled by an infra-red remote control. Signals from the remote control are interpreted by the IR Receiver module and a unique code is generated as output. This code is used as input to our IR control combinational logic block. The logic is implemented as a large Verilog case statement that toggles or modifies a variety of control registers. These control registers are forwarded to all of the modules and sub-modules that they control. In each of these modules there are subsequent combinational and sequential logic conditional statements that alter the modules outputs according to these control signals.

4 Results and Discussion

Table 1 shows the logic and memory utilisation of the processing modules we have created in our system. It was found that the total system logic element utilisation was rather minimal, using just 18% of the FPGA's logic resources. On the other hand the on-chip memory usage

was rather high at 70%. This was as expected as the main LUT used for the visuotopic mapping implementation was very large. It can be seen in the table that the histogram module uses almost half of the total logic utilisation of the system. This is because of the conditional logic that is used to determine the bin with the highest count. Using a smaller number of bins, or implementing an iterative approach would possibly decrease the logic utilisation for this module substantially, however both of these alternatives have drawbacks in terms of accuracy and latency.

Table 1: Logic and memory utilisation of created modules

Module	Logic Cells (% of FPGA logic cells)	Logic Registers (% of FPGA logic registers)	Memory bits (% of FPGA on-chip memory)
Whole system	20061 (17.5%)	6236 (6%)	2774535 (70%)
Edge detection	314 (0.3%)	146 (0.1%)	19140 (0.5%)
Mapping and main functions	4738 (4%)	1224 (1%)	2633571 (66%)
Histogram	8258 (7%)	812 (0.7%)	0 (0%)

Figures 6 to 9 show some of the visual results of the each of the modes implemented in the system. It is worth noting that the results look much more intuitive when wearing the system. The system was also found to be more useful when viewing a non-static scene or when the camera sensor was being moved around (via head movements that are registered to the viewpoint by the user's ego motion sensors). This can be said to be due to the ability of the viewer to build up a mental model of the scene as more slightly differing temporal information is provided to them. Refer to the video submitted along with this paper for a better indication of the systems operation.

The latency of the system (with no frame rate limitations enabled) is roughly one frame or about 17ms. The majority of this latency is accrued by the mapping implementation which waits until the end of a frame to update the display storage registers. Updating these registers earlier would result in a 'tearing' effect in the output frame, as half the frame data would not be completely synchronised.

Prolonged testing of the system showed that battery life was roughly 4 hours from full charge. Preliminary psychophysical testing hinted that certain tasks benefited from 4-level thresholding as opposed to binary (2-level), and that a frame rate of 4 frames per second was the lowest at which users could adequately navigate through a cluttered environment. Edge detection proved useful in identifying objects in a non-cluttered, low-contrast environment.

4.1 Limitations

In an actual cortical visual prosthesis, static electrode placement on the visual cortex means that the patient would not be able to scan over sections of the elicited phosphene pattern with movements of their eyes. In other words, phosphenes are locked to the gaze of the patient unlike our system where the user is able to focus on different parts of the displayed pattern with eye movements. Simulating this attribute would require an active eye-tracker and movement of the phosphene pattern along with the eyes.

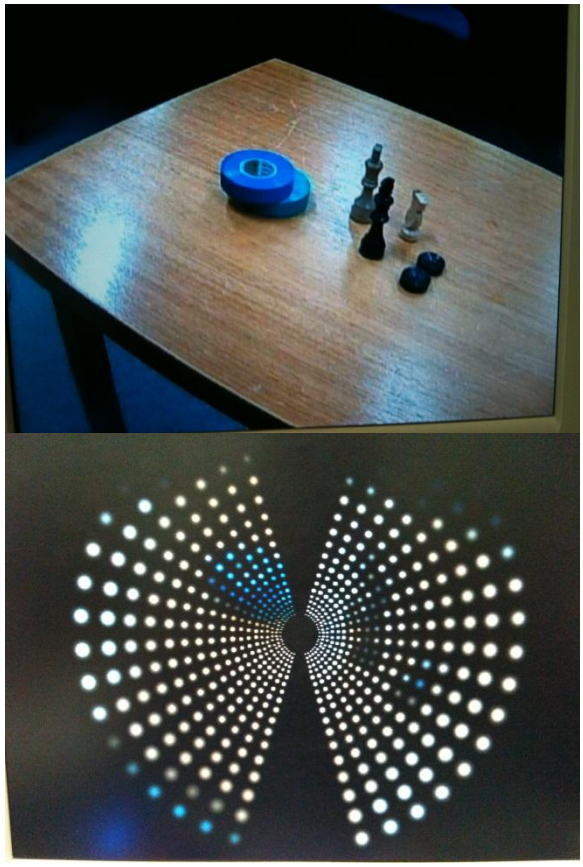

Figure 6: Visuotopic mapping (no intensity transformations)

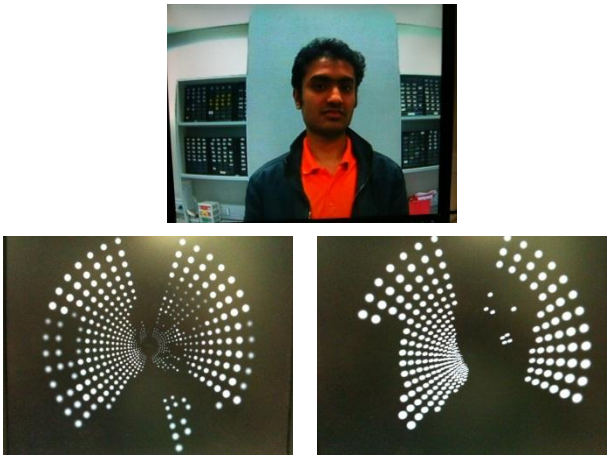

Figure 7: Intensity transformations: full resolution image (top), 4 level (bottom left), binary (bottom right).

Electrode placement within the visual cortex will also be significantly varied from patient to patient and will need to be placed carefully to avoid blood vessels in the brain. This will mean that the pattern of phosphenes will not appear as neat and 'mathematically' located as is the case with our system.

It was found that the histogram assisted binary threshold selection implementation was not as effective as expected. This is thought to be due to the automatic gain control that is built in to the camera sensor. Despite having the histogram feature disabled, the automatic gain of the camera would often still pose problems to the system as certain lighting conditions would cause oscillation through the limits of the camera sensor gain.

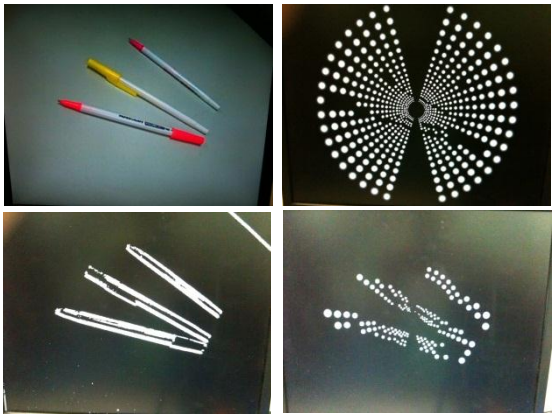

Figure 8: Edge detection: full resolution (top left), output without edge detection (top right), extracted edges (bottom left), output with edge detection (bottom right).

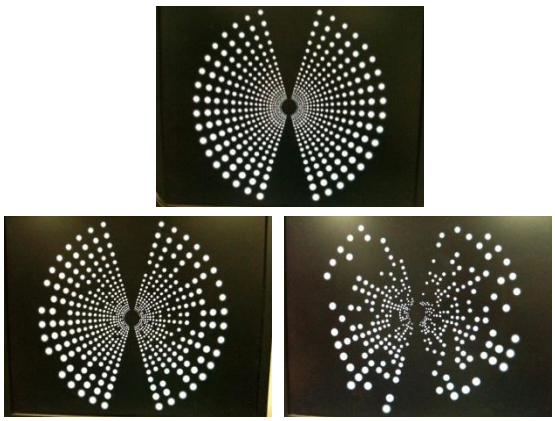

Figure 9: Dead electrode simulation: 0% (top), 10% (bottom left), 50% (bottom right).

5 Conclusion

In this paper we have presented a real-time, FPGA-based vision system that simulates bionic vision. The system's main application is simulation of a bionic vision system, however its implementation would benefit and find use in a variety of different intelligent mobile sensor and machine vision applications. The system has very low latency and would suit applications that require high frame rates. The ability to generate different low resolution mappings by simply modifying the mapping lookup table means that a variety of different mappings can be investigated very easily. This may prove useful when researching mobile robotic vision applications with resolution limitations.

Acknowledgements

Monash Vision Group is funded through the Australian Research Council Research in Bionic Vision Science and Technology Initiative (SRI 1000006). The authors would like to thank Grey Innovation for assisting with industrial design of parts of the system. The authors would also like to thank members of the Monash Vision Group that shared their valuable opinions and advice.

References

[Bak et al., 1990] Bak, M., Girvin, J. P., Hambrecht, F. T., Kufts, C. V., Loeb, G. E., Schmidt, E. M., 1990. Visual sensations produced by intracortical microstimulation of the human occipital cortex. *Medical & Biological Engineering & Computing*, vol. 28, pp. 257-259.

[Balasubramanian et al., 2002] Balasubramanian, M., Polimeni, J. R., Schwartz, E. L., 2002. The v1-v2-v3 complex: quasiconformal dipole maps in primate striate and extra-striate cortex. *Neural Networks*, vol. 15, iss.10, pp1157-1163.

[Brindley and Lewin, 1968] Brindley, G. S., Lewin, W. S., 1968. The sensations produced by electrical stimulation of the visual cortex. *Journal of Physiology*, vol. 196, pp. 479-493.

[Canny, 1986] Canny, J., 1986. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, pp. 679-698.

[Chen et al., 2005] Chen, S. C., Hallum, L. E., Lovell, N. H., Suaning, G. J., 2005. Visual acuity measurement of prosthetic vision: a virtual-reality simulation study. *Journal of Neural Engineering*, vol. 2, pp. S135-S145.

[Chen et al., 2009] Chen, S. C., Suaning, G. J., Morley, J. W., Lovell, N. H., 2009. Simulating prosthetic vision: i. visual models of phosphenes. *Vision Research*, vol. 49, pp. 1493-1506.

[Dobelle and Mladejovsky, 1974] Dobelle, W. H., Mladejovsky, M. G., 1974. Phosphenes produced by electrical stimulation of human occipital cortex, and their application to the development of a prosthesis for the blind. *Journal of Physiology*, vol. 243, pp. 553-576.

[Dobelle et al., 1976] Dobelle, W. H., Mladejovsky, M. G., Evans, J. R., Roberts, T. S., Girvin, J. P., 1976. 'Braille' reading by a blind volunteer by visual cortex stimulation. *Nature*, vol. 259, pp. 111-112.

[Dowling et al., 2004] Dowling, J. A., Maeder, A. J., Boles, W., 2004. Mobility enhancement and assessment for a visual prosthesis. *Proceedings of SPIE Medical Imaging 2004: Physiology, Function, and Structure from Medical Images*, vol. 5369, pp. 780-791.

[Duncan and Boynton, 2003] Duncan, R. O., Boynton, G. M., 2003. Cortical magnification within human primary visual cortex correlates with acuity thresholds. *Neuron*, vol. 38, pp. 659-671.

[Fehervari et al., 2010] Fehervari, T., Matsuoka, M., Okuno, H., Yagi, T., 2010. Real-time simulation of phosphene images evoked by electrical stimulation of the visual cortex. *Neural Information Processing*, vol. 6443, pp. 171-178.

[Horton and Hoyt, 1991] Horton, J. C., Hoyt, W. F., 1991. The representation of the visual field in human striate cortex: a revision of the classic holmes map. *Archives of Ophthalmology*, vol. 109, pp. 816-824.

[Humayun et al., 1996] Humayun, M. S., de Juan, E., Dagnelie, G., Greenberg, R. J., Propst, R. H., Phillips, D. H., 1996. Visual perception elicited by electrical stimulation of retina in blind humans. *Archives of Opthalmology*, vol. 114, pp. 40-46.

[Lee et al., 1987] Lee, J. S. J., Haralick, R. M., Shapiro, L. G., 1987. Morphologic Edge Detection. *IEEE Journal of Robotics and Automation*, vol. 3, pp. 142-156.

[Monash University, 2010] Monash University, 2010. Monash vision direct to brain bionic eye. Viewed 11th July, 2011, <http://monash.edu.au/bioniceye>.

[Polimeni et al., 2006] Polimeni, J. R., Balasubramanian,

M., Schwartz, E. L., 2006. Multi-area visuotopic map complexes in macaque striate and extra-striate cortex. *Vision Research*, vol. 46, pp. 3336-3359.

[Schira et al, 2007] Schira, M. M., Wade, A. R., Tyler, C. W., 2007. Two-dimensional mapping of the central and parafoveal visual field to human visual cortex. *Journal of Neurophysiology*, vol. 97, pp. 4284-4295.

[Schira et al, 2010] Schira, M. M., Tyler, C. W., Spehar, B., Breakspear, M., 2010. Modeling magnification and anisotropy in the primate foveal confluence. *PLoS Computational Biology*, vol. 6, iss.1, pp. 1-10.

[Schwartz, 1977] Schwartz, 1977. Spatial mapping in the primate sensory projection: analytic structure and relevance to perception. *Biological Cybernetics*, vol. 25, pp. 181-194.

[Srivastava et al., 2009] Srivastava, N. R., Troyk, P. R., Dagnelie, G., 2009. Detection, eye-hand coordination and virtual mobility performance in simulated vision for a cortical visual prosthesis device. *Journal of Neural Engineering*, vol. 6, pp 1-14.

[van Rheede et al., 2010] Van Rheede, J. J., Kennard, C., Hicks, S. L., 2010. Simulating prosthetic vision: optimizing the information content of a limited visual display. *Journal of Vision*, 10(14):32, pp. 1-15.

[Wandell et al., 2007] Wandell, B. A., Dumoulin, S. O., Brewer, A. A., 2007. Visual field maps in human cortex: review. *Neuron*, vol. 56, pp. 366-383.