SOFTWARE AND HARDWARE IMPLEMENTATION OF THE NEUROMORPHIC LGN BASED IMAGE PROCESSING AND FEATURE EXTRACTION

Anuar Dorzhigulov, Bachelor of Engineering

Submitted in fulfillment of the requirements
for the degree of Master of Science

School of Engineering
Department of Electrical and Electronic Engineering
Nazarbayev University

53 Kabanbay batyr Avenue,
Astana, Kazakhstan, 010000

December 9

Abstract

The processing of the graphical data is popular methodology of obtaining important information. However, there is a major drawback: it typically requires large computational resources. The human brain is an excellent example of the efficient image processing hardware, due to the fact the biological visual system can allow to easily and quickly obtain the information of the world around, such as object identification and movement detection.

In particular, there is one element of the biological visual system that has unique functionality in image processing, which is lateral geniculate nucleus (LGN). Ganglion cells, which are terminated in LGN, have high sensitivity to the image spatial intensity difference. This cell feature is used for pre-processing of the visual data before being modulated and relayed to the main processing module, visual cortex. As a result, the processing load on cortex is reduced, due to the pre-processing of the data.

The aim of the project is to develop the algorithm for visual features extraction, such as edge detection, based on the structure and properties similar to the LGN, with a possibility of the hardware model implementation.

Preliminary results show the edge detection property of the proposed method. Moreover, the measured performance is comparable to other popular edge detection techniques, even exceeding expectations to small extent in the noisy environment.

Acknowledgments

This thesis took about a year to be finished. Throughout this time there were persons who gave me an valuable assistance in preparing and finishing this project. Foremost, I would like to express my gratitude to professor Alex James for helpful supervision and guidance. His continuous support inspired unexperienced engineering student to take serious interest in the field of the electronic engineering, and continue this strive for knowledge on graduate level. He motivated me to study the opportunities of the neuromorphic circuits in the modern electronics. His experience in this field was a tool that helped to shape the project on all stages, from the preliminary research to the thesis writing. It could be hard to consider any other person to be fit to supervise such project. In addition, his supervision was helpful to summarize this research in a form of conference paper submission to ISCAS 2017 titled "Coarse to fine difference edge detection with binary neural firing model".

Next, I would like to thank Yerlan Berbaliyev, for his assistance with the project development, especially hardware design. His empirical expertise in electronics was a decisive factor in finalizing the hardware design of the proposed circuitry. Moreover, his patience was invaluable in the final stages of the project development, when the simulations and design validations took place.

In addition, I am grateful that professor Amin Zollanvari had an opportunity to review the thesis. His comments and proposals was necessary to prepare the final version of this work.

I am grateful that I had an opportunity to do a research with them.

Finally, the Nazarbayev University provided an exceptional academic environment for extensive study and research. Without it, two years spent in this institution will not be so fruitful.

Contents

List of Figures

Chapter 1 – Introduction

The basic image can be considered as a graphical source of the raw data, so there is a need for tools which will assist in processing through this pile of raw data to obtain some information that can be considered relevant or interesting. Today, these tools are called image processing algorithms, and the interest in obtaining graphic based data is increasing. Movement detection, object recognition, classification and tracking, visuals based machine learning and robotics, statistical analysis of the video feed. These are just some examples of the activities, where image processing is heavily involved. As a result, the image processing is the field that draws the attention of the modern researchers. Moreover, there is growing interest from different fields of industry as well: from the medical imaging in medicine, to the optimized surveillance systems in security.

Graphical information has a significant drawback, which lies in the fact that it is considered as rather heavy source of data. Single picture taken by the modern even not high-end smartphone can occupy up to several MB of memory on hard drive, even if some basic compression techniques are involved, and what if high quality video stream at 100+ FPS from a remote quadcopter in the oil field is required to be processed in order to locate the leak? As a result, there are some limitation that comes from this major drawback of graphically represented information and image processing. Conventional algorithms require great computational resources to analyze raw graphical data, especially if it is required to be accurate, fast, large scaled and in real time. In addition, the popularization

of mobile wireless platforms, such as smartphones and small UAVs, have risen the additional concerns about the energy consumption.

In order to address mentioned above issues, the major bottlenecks for improvements in efficiency and performance should be identified. Firstly, most popular modern image processing techniques heavily rely on the software based solutions, still being based on the conventional, widely available, universal hardware. Secondly, there is high dependency on memory, required for storage and processing of images. Lastly, is the ability to perform within mobile platform main limitations - energy and space.

First obvious question is if specialized hardware based, memory independent, compact and energy efficient solution for simple and complex image processing solution already exists. It does, but not in conventional form. It is our brain. As a result, the main interest of this work is to identify the key features of the brain visual system, that can be used to develop neuromorphic system for image processing with desired qualities and performance.

Chapter 2 – Structure of the human visual system

2.1 Biological visual system overview

The first step in the analysis of the brain-like image processing, is to identify the core elements of the biological visual system.

The human visual system consist of 3 main elements [2]:

- Eye retina
- Lateral geniculate nucleus (LGN)
- Visual cortex of the brain

2.1.1 Retina

The eye retina acts as the very first step of the biological image processing. The visual signal, which is light, terminated on retina, where multiple photo sensor cells are located to capture light signal. The response of the cells is graded electrical signal, based on the photochemical reaction of Na^+ ions inside. The produced signal then drives the bipolar cells into excitation state, which will later trigger the response of the ganglion cell. There are two types of the bipolar and ganglion cell allocation: ON center/OFF surround or OFF center/ON surround in concentric fashion, with circular central region and respective surroundings. Each produces a dedicated part of receptive field for each ganglion cell. The

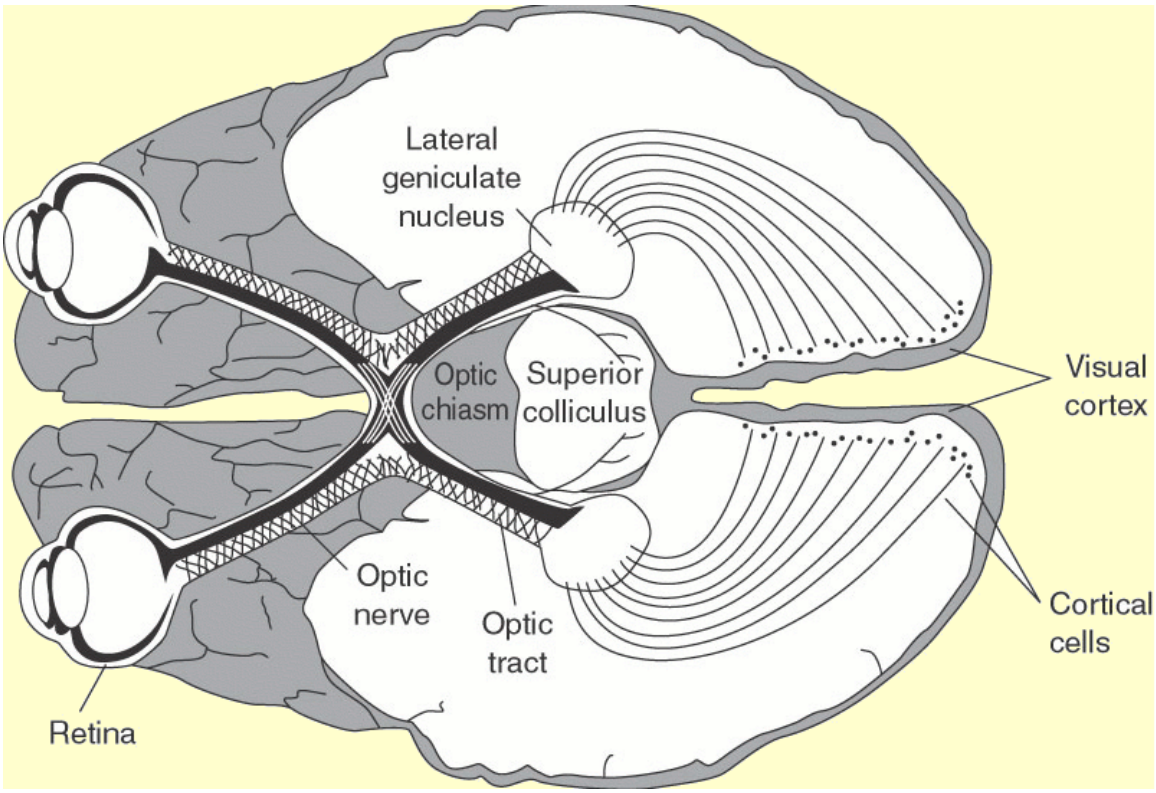


Figure 2.1: The structure of the human biological visual system. There are more elements, besides Retina, LGN and cortex, however, only mentioned ones take role in the signal capture and processing. Adopted from [1]

excitation of the ON region will increase the ganglion response, while excitation of the OFF region will decrease it. If both region saturated by light equally, the produced response will be close to baseline. As a results, the ganglions are not effective in detecting the uniform light distribution, but sensitive to the light difference, or contrast [3–7].

2.1.2 LGN

LGN located between the retina and cortex. In addition, it acts as the preliminary tool of vision coordination and synchronization. There are six layers of LGN, three for each eye. Each layer responds to the specific area of the receptive field in retina. In addition, there are similar two groups of cells, which are responsible for the color or contrast sensitivity. Based on the response, the ganglions fire

spike with variable frequency or probability, based on the cell excitation [8–12].

2.1.3 Visual cortex

The final part of the biological visual processing occurs in the back part of the brain, the visual cortex. It consists of six primary regions. The V1 region is the final node for all visual routes to terminate, as a results, primary information is interpreted and processed there, such as edges, colors, objects and orientations. Other regions is used to process and interpret the parts of the incoming information in further details, for example main color processing occurs in the V2 area of the visual cortex. With assistance of the non-visual aspects, such as memory, prediction, the cortex is the source of the complex visual perception [13–18].

2.2 LGN based image preprocessing

As it can be seen, from the previous section, the eye retina acts as a light signal detection device, similar to the CMOS photo sensor of modern digital cameras, while the cortex acts as the processing unit, such as digital signal processors (DSP). Meanwhile, the LGN does not have similar functioning elements in the conventional image processing hardware. The LGN is responsible not only for relaying signals between retina and cortex, but to perform preprocessing of the visual signals. As a result, the cortex does not process just the raw data captured by the retina photo sensors, but some preselected regions of the interest, based on the contrast, movement and color sensitivity of the special ganglion cells, shared by retina and LGN. This property has been suggested after multiple experiments on mammalian brain, such as cat and macaque [19–21].

The implementation of the LGN-like element into the modern image processing circuitry and hardware can be beneficial [22].The amount of the memory required for storage and processing is significantly reduced, due to the preselected areas of interest from the raw data [23, 24].

It can be achieved, if the basic structure of the LGN can be implemented as hardware. It requires the hardware that can segment the fractions of the general receptive field into the ON/OFF regions and produce the spike sequence response based on the difference between two regions of excitation and inhibition.

Chapter 3 – Methodology

The main goal of this project is to develop LGN-inspired edge detection algorithm that can utilize contrast sensitivity similar to ganglion cells. The core concept is to produce the output depending on the balance between excitation and inhibition of each individual cell.

3.1 Algorithm

The generalized pseudocode is presented in Algorithm 1. The entire process consists of four major steps: 2D convolution, thresholding, windowing, and subtraction.

3.1.1 Convolution

The convolution is core operation to implement the cell output excitation and inhibition, similar to LGN.

The input image in grayscale is represented by a two-dimensional matrix with floating point numbers ranging between 0 and 1. In the first stage, this input matrix is processed with 2D convolution filter with kernels of different sizes and values. This filter generates a matrix labeled C. Each element of C matrix is generated based on the difference of average saturation of excitation and inhibition zones as given in Eq. 3.1, where E_n is the value of the excitation pixel, I_m is the value of inhibition pixel, n is the total number of the excitation pixels, m is the total number of the inhibition pixels, and M is the maximal voltage output of the photosensor.

Algorithm 1 Edge detector algorithm

procedure $Edgedetector(I(N, M))$ ▷ I input image matrix, size (N,M), elements range: 0..1, type: float

for $i \leftarrow 1, N_f$ **do** ▷ N_f: number of filters

Create $K(N_i, N_i)$ ▷ $Create\ filter\ matrix\ of\ size\ N_i$ ▷ $N_i : i-th\ odd\ number > 1$

Fill $K\ with\ random\{1\ or\ -1\}$ based on predefined probability P

$H(i) \leftarrow K$

for all $h \in H(N_f)$ **do**

$C \leftarrow \frac{|h*I|}{|\sum h|}$ ▷ convolution operation

for $j \leftarrow 1, Q$ **do** ▷ $Q : number\ of\ intensity\ levels$

for all $c \in C(N, M)$ **do** ▷ thresholding operation

if $c \geqslant j * (1/Q)$ **then**

$I_{lvl}(j, c) \leftarrow 1$

else

$I_{lvl}(j, c) \leftarrow 0$

$N_{windows} \leftarrow Q/W$ ▷ $N_{windows} : number\ of\ windows$ ▷ $W : window\ size$

for $m \leftarrow 1, N_{windows}$ **do**

$W_n(m) \leftarrow \sum_{n=Wm}^{W(m+1)} I_{lvl}(n)$

$W_{out}(i) \leftarrow W_n$ ▷ $W_{out} : result\ of\ 2D-convolution, thresholding\ and\ windowing$

Initialize $F(N, M)$ ▷ set all elements to zero

for $n \leftarrow 1, N_f$ **do** ▷ subtraction operation

for $m \leftarrow n + 1, N_f$ **do**

$F = F + |W_{out}(n) - W_{out}(m)|$

return F

$$C = \frac{|\frac{1}{n}\sum_{1}^{n} E_n - \frac{1}{m}\sum_{1}^{m} I_m|}{M} \tag{3.1}$$

3.1.2 Thresholding

After convolution operation, each element of C is segmented into the Q number of intensity levels I_{lvl}. It can be done by utilizing thresholding operation, where $1 \leqslant q < Q$, eq. 3.2.

$$I_{lvl}(q) = \begin{cases} 1, & \text{for } \frac{q-1}{Q} < C \\ 0, & otherwise \end{cases} \tag{3.2}$$

3.1.3 Windowing

After performing thresholding operation, the intensity levels are accumulated within some specific windows of size W, where $1 \leqslant n < \frac{Q}{W}$, as shown in Eq. 3.3. The output is a window vector $W_{out}(n)$.

$$W_{out}(n) = \sum_{q=n(W-1)}^{nW} I_{lvl}(q) \tag{3.3}$$

3.1.4 Subtraction

In the 2D convolution stage, N_f kernels with random binary values and with different sizes were convoluted with the input image which generated N_f number of different window vectors $W_{out}(n)$. Then, these output matrices are subtracted from each other, and the subtraction results are added to generate feature vector

F, as given in Eq. 3.4. Feature vector involves a set of masks with specific features of the original image such as edges, object textures, and background data.

$$F = \sum_{n=1}^{N_f-1} \sum_{m=n+1}^{N_f} |W_{out,n} - W_{out,m}| \tag{3.4}$$

3.2 Hardware implementation

After successful MATLAB simulation, the algorithm was translated in a circuit and simulated using SPICE. The overall circuit can also be divided into four major circuit blocks: (1) convolution, (2) thresholding, (3) windowing, and (4) subtraction.

3.2.1 Convolution circuit

The circuit schematic for performing two-dimensional convolution operation is shown in Fig. 3.1. Switches route input signals from pixels to either positive or negative terminal of the amplifier, thus, deciding whether it is an inhibitory or excitatory pixel. These switches are controlled by values of the kernel that are mapped to a specific input pixel according to 2D convolution theory. Summing amplifier circuit was used to perform summation and subtraction operations. Therefore, this circuit outputs convolution filtered signals.

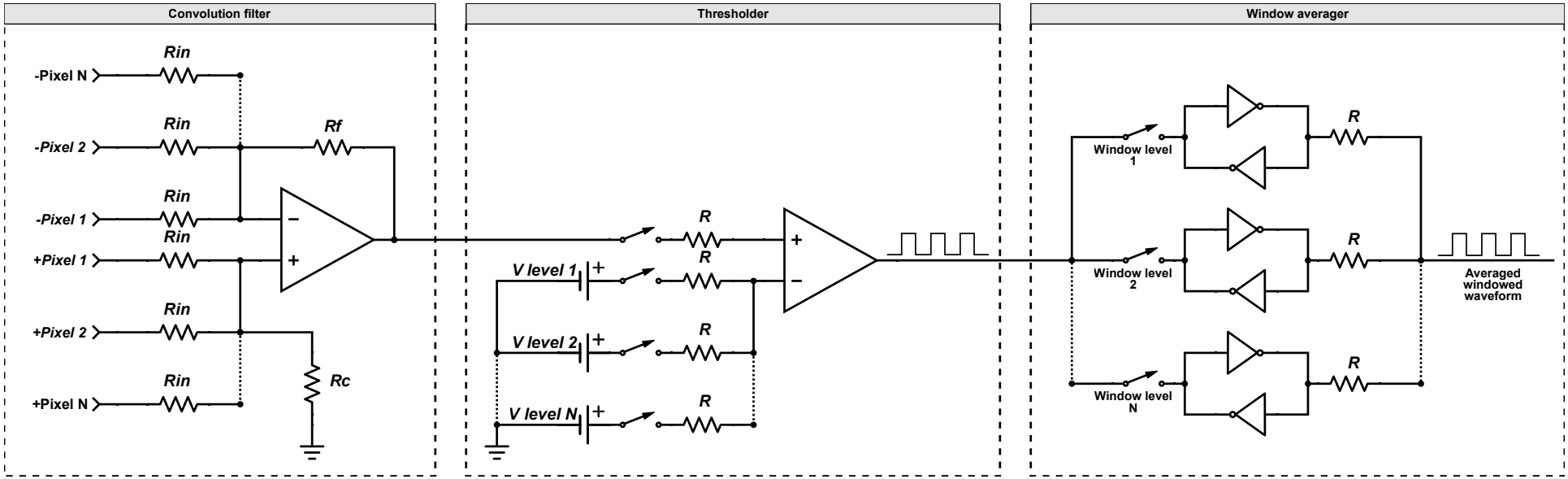


Figure 3.1: The pixel processing circuit consisting of the convolution operator that implements the difference between the excitatory and inhibition pixels, a threshold circuit that relates the amplitude of the pixel with a number of pulses generated and an average window operation that takes the average within a given set of the time window.

The number of such convolution circuits is as large as the number of input pixels. That is, each convolution circuit performs convolution operation for each pixel individually. For example, suppose there are 3x3 input image and 3x3 filter. Then, there will be nine convolution circuits with nine outputs for each element of C. Each of these circuits needs to be mapped with a specific filter kernel h, according to two-dimensional convolution algorithm. Therefore, there is a convolution matrix consisting of 9 convolution blocks that accept 18 inputs both from the 3x3 image and 3x3 filter kernel.

3.2.2 Thresholding

After passing the input signal through different convolution filters, it needs to be averaged by dividing into a maximum value of the output. The total number of outputs from convolution procedure is calculated by multiplying a total number of input pixels to the number of convolution kernels. The obtained convolution outputs will be further separated into multiple intensity levels. Figure 3.1 shows the threshold circuit that will divide input from convolution filter into Q number of intensity levels and the output is serially encoded in time. All of the switches are synchronized such that operational amplifier compares each intensity level with the input signal at a particular instant of time and outputs a high level if it exceeds the intensity level. If there are input from 9 input pixels convoluted with three different kernels, then there will be in total 27 thresholding circuits.

3.2.3 Windowing

Windowing circuit shown Fig. 3.1 takes outputs from the thresholding circuit as its input and accumulates it in a window with specified size W. Therefore, there are W levels of cross-coupled double inverters that memorize previous state. The output waveforms from these W levels sum up at the end to generate a windowed waveform.

3.2.4 Subtraction

After undergoing threshold, window and filtering with multiple kernels, output waveform needs to pass through subtraction process that can be modeled by N_f subtractors and a single adder using operational amplifiers, as shown in Fig. 3.2. Therefore, N_f waveforms that were filtered with N_f different kernels are subtracted from each other and subtraction results are added as described in Eq.3.4. Finally, the output waveform represents specific image features such as edges, background detail, foreground texture detail, and other features depending on parameters such as kernel values, window size and number of intensity levels.

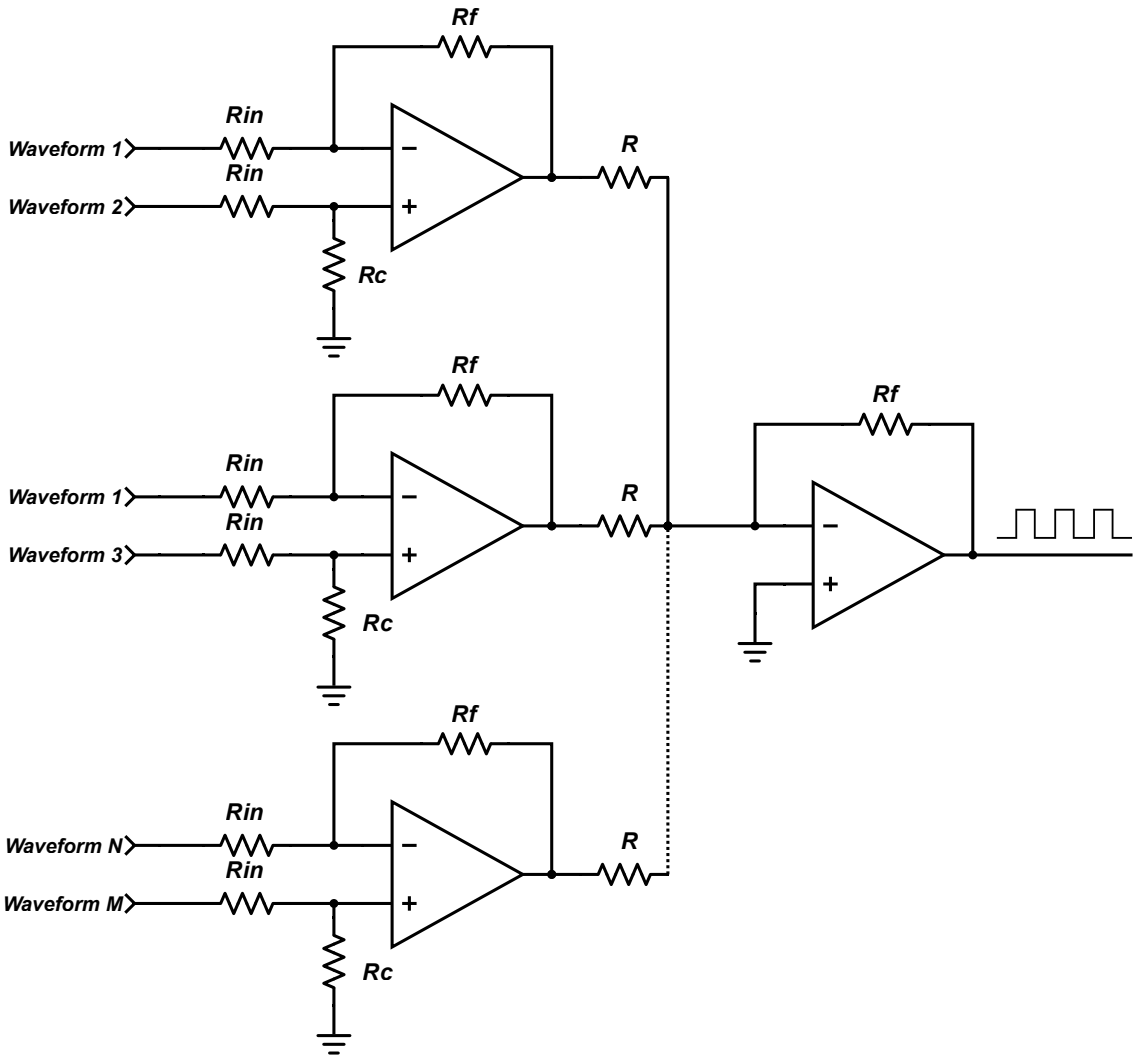


Figure 3.2: The subtractor circuit that calculates the difference between the feature outputs from differently sized filters.

Chapter 4 – Simulation results.

4.1 Software

The proposed algorithm is simulated in MATLAB using an image of size 256 × 256 pixels. The randomly generated filters of four fixed sizes (from 3×3 up to 9×9) have been used in the simulations. The results are shown in Fig. 4.1. Each row corresponds to the filter of certain size, while the column represents the intensity level increment. For this simulation, a total of 98 intensity levels and window size of 7 levels have been randomly selected. These images are used to develop feature vectors, based on the difference between filtered outputs. Examples can be seen in Fig. 4.2. Results show that the proposed method can be used to extract edge features from the original image.

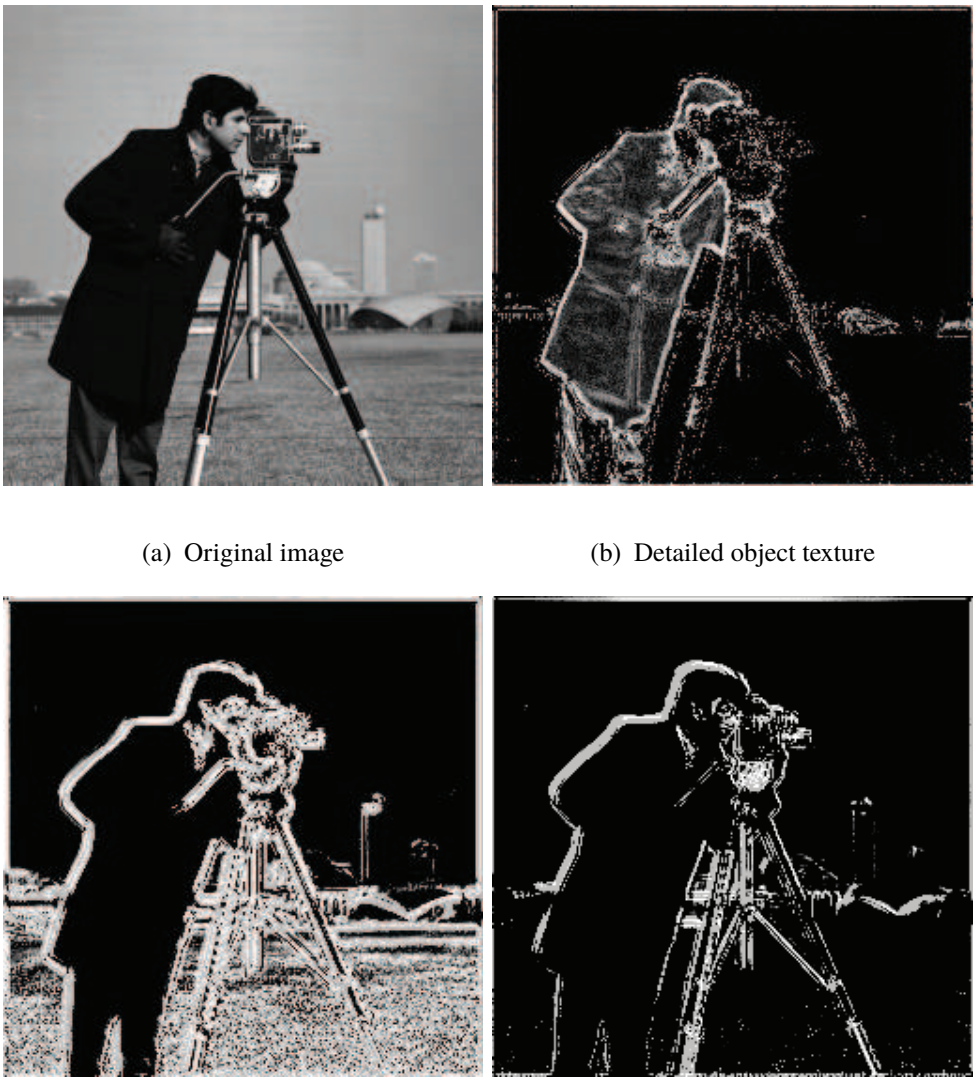
(a) 3×3 filter

(b) 5×5 filter

(c) 7×7 filter

(d) 9×9 filter

Figure 4.1: The output of the filter operations on (a) 3×3, (b) 5×5, (c) 7×7 and (d) 9×9 filter kernels.

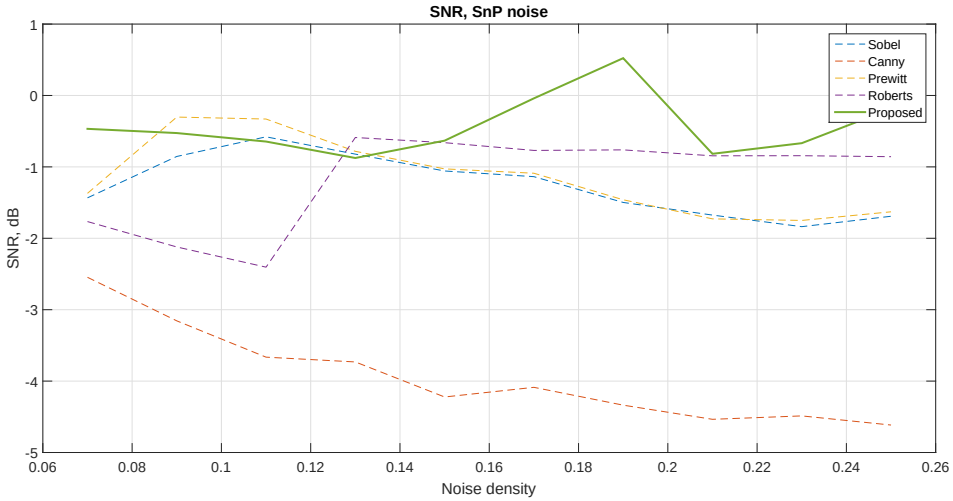


(a) Original image (b) Detailed object texture

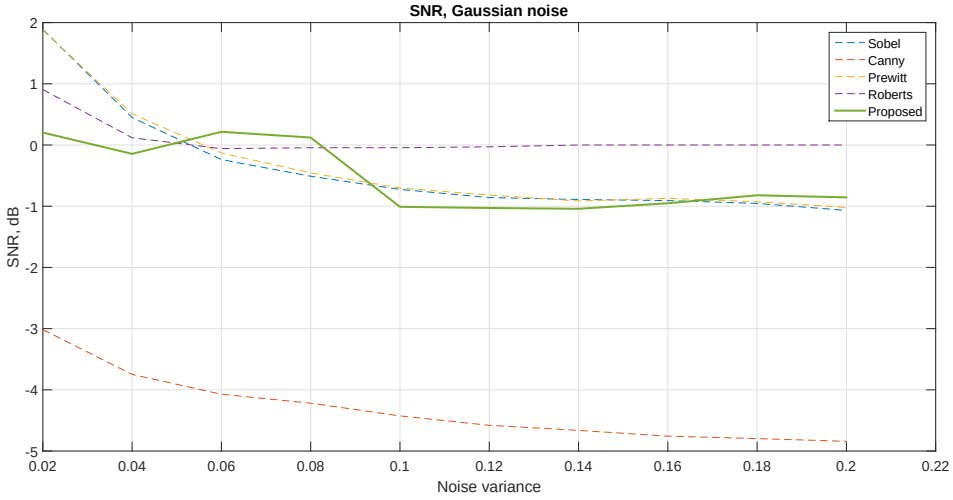
(c) Background information (d) Edge detection

Figure 4.2: The original image (a) when applied to the proposed algorithm results in the detection of edge features that can be used to analyse (b) object texture, (b) background details and (c) object edges.

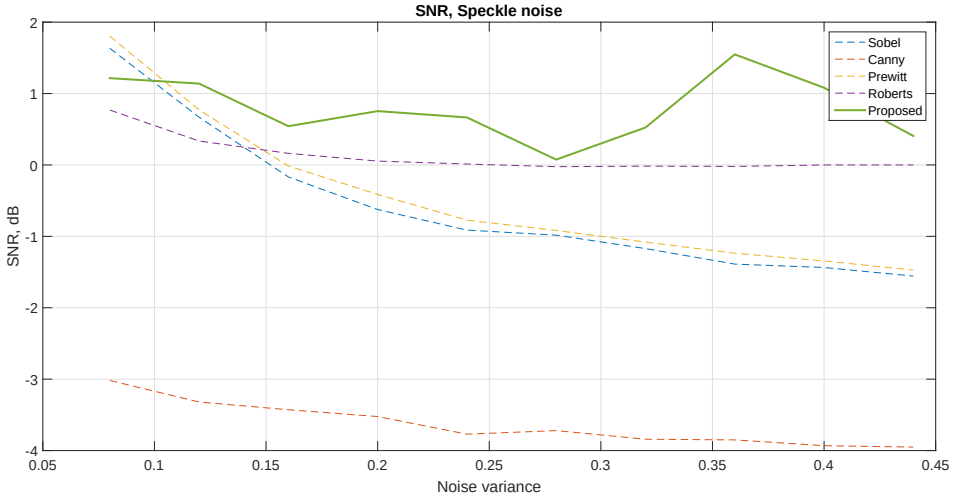
Noise tolerance has been examined, it is based on SNR calculation of the proposed method with comparison to other popular edge detection techniques, such as Canny, Sobel, Prewitt and Roberts. The noise analysis was done using the image shown in Fig. 4.2 (a) and repeated using image with added noise of different configurations and nature, Fig. 4.3.



(a) Salt and Pepper noise



(b) Gaussian noise



(c) Speckle noise

Figure 4.3: The results of the SNR analysis of the proposed methodology for different noise configurations: Salt and Pepper (a), Gaussian (b), Speckle (c)

4.2 Hardware

The proposed methodology has been simulated in SPICE for a single cell. The circuit was configured to use 3×3 input image matrix (randomized from 0 to 1), a total of 12 intensity levels and a window size of 3. The performance of the circuit simulations from SPICE were verified by comparing with that from MATLAB simulations. The results in Fig. 4.1 and Fig 4.2. are replicated by exporting the simulations over a different set of pixel values in the original image.

The temperature sensitivity of the hardware has been tested in the SPICE, Fig. 4.4.

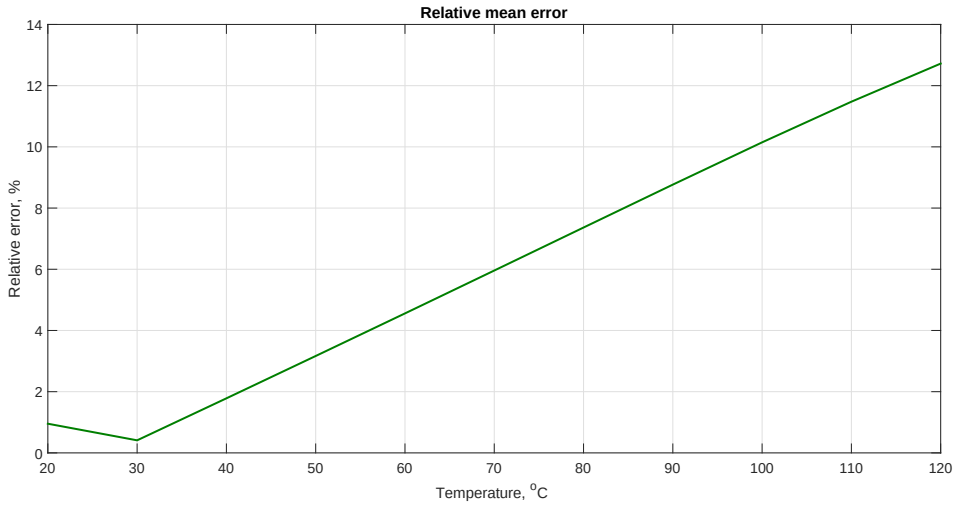


Figure 4.4: The temperature performance of the proposed hardware design, where the performance at the room temperature of 27 ^{o}C has been tested as the reference operating condition.

Chapter 5 – Discussion

Proposed methodology can be considered successful, based on several project research results. First of all, as it can be seen from the Fig. 4.1, the edge detection property is present. In addition to the object edges, some extra features can be extracted from the image, such as detailed background or the texture.

The proposed methodology seems to be competitive with conventional edge detection algorithms in the noisy environment, even slightly surpassing them in some cases.

Currently, the methodology was examined without any optimization in system configurations, such as intensity resolution (number of the intensity levels), window size, cell size, and probability of input binary weighting distribution.

The proposed hardware design has been simulated in the SPICE environment. Simulation was done for small scale cell, and both software and hardware simulation show similar result for identical conditions. Moreover, the preliminary sensitivity study shows that circuitry can operate with negligible error, within up to 5% resistance tolerance margin. Additional hardware testing suggests stable performance for temperatures under 120^{o}C, with peak error not exceeding 14%

The next step of the hardware development is to obtain the raw metrics of the large scale performance of the circuitry, with image input greater than 256×256. However, the instabilities, significant delays or other major problems of the system scaling is not expected, due to the parallel nature of the design. However, the issues of energy consumption and circuit size still are required to be addressed. One solution could be to consider the memristors to be used as a resistive circuit

elements, instead of the conventional resistors. Preliminary design based on 741 op-amp topology already requires around one hundred resistive elements for a single pixels, so the memristors could greatly boost the circuit performance in terms of power and size. Moreover, further hardware optimization could be considered, for example, the mentioned 741 topology is reliable, but dated, so something more compact and efficient could be taken into account.

Chapter 6 – Conclusion

The bio-inspired LGN based algorithm for feature extraction has been proposed. According to the preliminary results analysis, the proposed methodology is successful in edge detection, and competitive compared to some popular edge detection techniques in noisy environment. Moreover, developed methodology is ready to be implemented as a hardware. There are already some performance analysis available, which are suggesting robust performance. However, it should be noted that all obtained results have been obtained for non optimized configurations, for both hardware and software, so there is a possibility to obtain a superior performance, there is a possibility of the further design improvements, such as implementation of the memristors as a resistive circuit elements.

The next step in the project design is to test the scalability of the proposed design, even if it is highly parallel, the issue of the power and size should still be addressed. It is important to observe the performance of the circuit with large scale input. Next, is to observe the real time performance, in other words if it is will be feasible to use the proposed hardware for the continuous video stream processing. In this case, additional performance parameters should be taken into account, such as working FPS.

Finally, the possibilities of the integration of the fast processing neuromorphic semi analog hardware into already existing digital circuits. As it can be mentioned before, the CMOS photosensor can be used as alternative to the retina, while DSP circuits can substitute the cortex processing, so the proposed LGN-inspired hardware can be used to implement full functionality of the human visual system as a single image processing device, being compact, energy effi-

cient, reliable and fast.

Bibliography

[1] P. A. van der Helm, "Cognitive architecture of perceptual organization: From neurons to gnosons," *Cognitive processing*, vol. 13, no. 1, pp. 13–40, 2012.

[2] *Basic and Clinical Science Course (BCSC), Section 05: Neuro-Ophthalmology*. American Academy of Ophthalmology, 2016.

[3] Z. Pei and Q. Qiao, "An approximate retina model with cascade structures," in *2010 Sixth International Conference on Natural Computation*, vol. 4, Aug 2010, pp. 2009–2012.

[4] K. Palczewski, A. S. Polans, W. Baehr, and J. B. Ames, "Ca 2+-binding proteins in the retina: structure, function, and the etiology of human visual diseases," *Bioessays*, vol. 22, no. 4, pp. 337–350, 2000.

[5] A. Hughes, "The topography of vision in mammals of contrasting life style: comparative optics and retinal organisation," in *The visual system in vertebrates*. Springer, 1977, pp. 613–756.

[6] E. Rasch, H. Swift, A. Riesen, and K. L. Chow, "Altered structure and composition of retinal cells in dark-reared mammals," *Experimental cell research*, vol. 25, no. 2, pp. 348–363, 1961.

[7] R. H. Masland, "The fundamental plan of the retina," *Nature neuroscience*, vol. 4, no. 9, pp. 877–886, 2001.

[8] B. Cleland, M. Dubin, and W. Levick, "Sustained and transient neurones in

the cat's retina and lateral geniculate nucleus," *The Journal of Physiology*, vol. 217, no. 2, p. 473, 1971.

[9] A. M. Derrington, J. Krauskopf, and P. Lennie, "Chromatic mechanisms in lateral geniculate nucleus of macaque." *The Journal of Physiology*, vol. 357, no. 1, pp. 241–265, 1984.

[10] A. Derrington and P. Lennie, "Spatial and temporal contrast sensitivities of neurones in lateral geniculate nucleus of macaque." *The Journal of Physiology*, vol. 357, p. 219, 1984.

[11] S. Sherman and C. Koch, "The control of retinogeniculate transmission in the mammalian lateral geniculate nucleus," *Experimental Brain Research*, vol. 63, no. 1, pp. 1–20, 1986.

[12] E. Kaplan and R. Shapley, "X and y cells in the lateral geniculate nucleus of macaque monkeys." *The Journal of Physiology*, vol. 330, no. 1, pp. 125–143, 1982.

[13] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[14] C. M. Gray, P. König, A. K. Engel, W. Singer *et al.*, "Oscillatory responses in cat visual cortex exhibit inter-columnar synchronization which reflects global stimulus properties," *Nature*, vol. 338, no. 6213, pp. 334–337, 1989.

[15] J. Moran and R. Desimone, "Selective attention gates visual processing in

the extrastriate cortex," *Frontiers in cognitive neuroscience*, vol. 229, pp. 342–345, 1985.

[16] R. L. De Valois, D. G. Albrecht, and L. G. Thorell, "Spatial frequency selectivity of cells in macaque visual cortex," *Vision research*, vol. 22, no. 5, pp. 545–559, 1982.

[17] C. M. Gray and W. Singer, "Stimulus-specific neuronal oscillations in orientation columns of cat visual cortex," *Proceedings of the National Academy of Sciences*, vol. 86, no. 5, pp. 1698–1702, 1989.

[18] R. L. De Valois, E. W. Yund, and N. Hepler, "The orientation and direction selectivity of cells in macaque visual cortex," *Vision research*, vol. 22, no. 5, pp. 531–544, 1982.

[19] D. H. O'Connor, M. M. Fukui, M. A. Pinsk, and S. Kastner, "Attention modulates responses in the human lateral geniculate nucleus," *Nature neuroscience*, vol. 5, no. 11, pp. 1203–1209, 2002.

[20] V. Perry, R. Oehler, and A. Cowey, "Retinal ganglion cells that project to the dorsal lateral geniculate nucleus in the macaque monkey," *Neuroscience*, vol. 12, no. 4, pp. 1101–1123, 1984.

[21] M. J. Friedlander, C. Lin, L. Stanford, and S. M. Sherman, "Morphology of functionally identified neurons in lateral geniculate nucleus of the cat." *Journal of Neurophysiology*, vol. 46, no. 1, pp. 80–129, 1981.

[22] L. C. Kiong and T. J. Peng, "Quantum bio-inspired invariant object recog-

nition model on system-on-a-chip (soc)," in *2008 IEEE Conference on Robotics, Automation and Mechatronics*, Sept 2008, pp. 433–438.

[23] C. Tan, S. Lallee, and G. Orchard, "Benchmarking neuromorphic vision: lessons learnt from computer vision," *Frontiers in Neuroscience*, vol. 9, p. 374, 2015. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fnins.2015.00374

[24] G. Orchard and R. Etienne-Cummings, "Bioinspired visual motion estimation," *CoRR*, vol. abs/1511.00096, 2015. [Online]. Available: http://arxiv.org/abs/1511.00096

Appendix

6.1 MATLAB

Biofilter:

```
function out2=biofilter(I,h)
mg=abs(imfilter(I,h))/abs(sum(sum(h)));

out0=[];
for i=1:98
    out=mg>=.01*i;
    kout=out(:);
out0(i,:)=kout;
end
out2=[];
for i=1:7:91
    outc=sum(out0(1*i:7+i,:),1);
    out1=reshape(outc,size(mg));
    out2=[out2,out1];
end
```

Algorythm:

```
clear all;
I = mat2gray(imread('cameraman.tif'));

count=1;
% out=[];
for i=3:2:9
h=rand(i);
h=double(h>mean2(h));
h(h==0)=-1;
out2=biofilter(I,h);
a=size(out2);
% out(count,:)=out2(:)';
out{count,1}=out2;
out3(count,:)=out2(:);
count=count+1;
end

something=cell2mat(out);
% figure;
% imshow(something(1:256,:));
% figure;
% imshow(something(256*1 + 1:256*2,:));
% figure;
% imshow(something(256*2 + 1:256*3,:));
```

```
25 % figure;
26 % imshow(something(256*3 + 1:256*4,:));
27
28 arr=[1:1:count-1];
29 for i=1:count-2
30     ab=(arr>=i).*arr;
31     ab(ab==0)=[];
32     ab=uint8(ab');
33     zigbee=out3(ab,:);
34     a1=repmat((out3(i,:)), length(ab),1);
35     b1=(zigbee);
36     final{i,1}=abs(a1-b1);
37 end
38 lastfinal=cell2mat(final);
39 figure;
40 imshow(mat2gray(reshape(sum(lastfinal),a)));
41 show = sum(lastfinal);
```

6.2 Hardware

Spice netlist:

```
R10 N007 N003 {RX_10} tol=1
R2 N003 in2 {RX_2} tol=1
R3 N003 in3 {RX_3} tol=1
R4 N003 in4 {RX_4} tol=1
R5 N012 in5 {RX_5} tol=1
R6 N012 in6 {RX_6} tol=1
R7 N012 in7 {RX_7} tol=1
R8 N012 in8 {RX_8} tol=1
R9 N012 in9 {RX_9} tol=1
R1 N003 in1 {RX_1} tol=1
V1 V+ 0 5
V2 V- 0 -5
V3 in1 0 0.356
V4 in2 0 0.144
V5 in3 0 0.851
V6 in4 0 0.338
V7 in5 0 0.275
V8 in6 0 0.006
V9 in7 0 0.802
V10 in8 0 0.497
V11 in9 0 0.538
XX1 N007 N011 absolute
```

```
V12 Clock_1 0 PULSE(0 1 6m 0.05m 0.05m 8m 100m 1)
V13 V1 0 0.08
V14 V2 0 0.16
V15 V3 0 0.24
V16 V4 0 0.32
V17 V5 0 0.4
S1 N001 V1 Clock_1 0 MYSW
S2 N001 V2 Clock_2 0 MYSW
S3 N001 V3 Clock_3 0 MYSW
S4 N001 V4 Clock_4 0 MYSW
S5 N009 N011 Clock_G 0 MYSW
R12 N005 N009 10k tol=1
R13 N002 N001 10k tol=1
S6 N001 V5 Clock_5 0 MYSW
V18 Clock_2 0 PULSE(0 1 26m 0.05m 0.05m 8m 100m 1)
V19 Clock_3 0 PULSE(0 1 46m 0.05m 0.05m 8m 100m 1)
V20 Clock_4 0 PULSE(0 1 66m 0.05m 0.05m 8m 100m 1)
V21 Clock_5 0 PULSE(0 1 86m 0.05m 0.05m 8m 100m 1)
V24 Clock_G 0 PULSE(0 1 6m 0.05m 0.05m 8m 20m 12)
V22 V6 0 0.48
V23 V7 0 0.56
V25 V8 0 0.64
V26 V9 0 0.72
V27 V10 0 0.8
V28 V11 0 0.88
```

```
48 V29 V12 0 0.96
49 V30 Clock_6 0 PULSE(0 1 106m 0.05m 0.05m 8m 100m 1)
50 V31 Clock_7 0 PULSE(0 1 126m 0.05m 0.05m 8m 100m 1)
51 V32 Clock_8 0 PULSE(0 1 146m 0.05m 0.05m 8m 100m 1)
52 V33 Clock_9 0 PULSE(0 1 166m 0.05m 0.05m 8m 100m 1)
53 V34 Clock_10 0 PULSE(0 1 186m 0.05m 0.05m 8m 100m 1)
54 V35 Clock_11 0 PULSE(0 1 206m 0.05m 0.05m 8m 100m 1)
55 V36 Clock_12 0 PULSE(0 1 226m 0.05m 0.05m 8m 100m 1)
56 S7 N001 V6 Clock_6 0 MYSW
57 S8 N001 V7 Clock_7 0 MYSW
58 S9 N001 V8 Clock_8 0 MYSW
59 S10 N001 V9 Clock_9 0 MYSW
60 S11 N001 V10 Clock_10 0 MYSW
61 S12 N001 V11 Clock_11 0 MYSW
62 S13 N001 V12 Clock_12 0 MYSW
63 S14 N006 N004 Clock_1_av 0 MYSW
64 R14 N013 N006 10k tol=1
65 XX2 N006 N008 inverter
66 XX3 N008 N006 inverter
67 S15 N010 N004 Clock_2_av 0 MYSW
68 R15 N013 N010 10k tol=1
69 XX4 N010 N014 inverter
70 XX5 N014 N010 inverter
71 S16 N015 N004 Clock_3_av 0 MYSW
72 R16 N013 N015 10k tol=1
```

```
XX6 N015 N016 inverter
XX7 N016 N015 inverter
V37 Clock_1_av 0 PULSE(0 1 6m 0.05m 0.05m 8m 60m 4)
V38 Clock_2_av 0 PULSE(0 1 26m 0.05m 0.05m 8m 60m 4)
V39 Clock_3_av 0 PULSE(0 1 46m 0.05m 0.05m 8m 60m 4)
V40 Clock_4_av 0 PULSE(0 1 46m 0.05m 0.05m 8m 60m 4)
S17 Waveform_out N013 Clock_4_av 0 MYSW
S18 Waveform_out 0 v111 Clock_4_av MYSW
V41 v111 0 1
XU1 N012 N003 V+ V- N007 LM741
XU2 N005 N002 V+ V- N004 LM741

* block symbol definitions
.subckt absolute in out
R1 out N001 20k tol=1
R2 N001 N003 10k tol=1
D1 N003 N004 D
R3 N003 N002 20k tol=1
D2 N004 N002 D
R4 N002 in 20k tol=1
R5 N001 in 20k tol=1
V1 v+ 0 5
V2 v- 0 -5
XU1 0 N002 v+ v- N004 LT1007
XU2 0 N001 v+ v- out LT1007
```

```
98  .ends absolute
99
100 .subckt inverter Vin Vout
101 M1 Vout Vin 0 0 NMOS l=1.5u w=40u
102 M2 Vdd Vin Vout Vdd PMOS l=1.5u w=85u
103 V1 Vdd 0 5
104 .MODEL NMOS NMOS LEVEL=3 PHI=0.7 TOX=9.5E-09 XJ=0.2U
        TPG=1
105 + VTO=0.7 DELTA=8.8E-01 LD=5E-08 KP=1.56E-04
106 + UO=420 THETA=2.3E-01 RSH=2.0E+00 GAMMA=0.62
107 + NSUB=1.40E+17 NFS=7.20E+11 VMAX=1.8E+05 ETA=2.125E
      -02
108 + KAPPA=1E-01 CGDO=3.0E-10 CGSO=3.0E-10
109 + CGBO=4.5E-10 CJ=5.50E-04 MJ=0.6 CJSW=3E-10
110 + MJSW=0.35 PB=1.1
111 .MODEL PMOS PMOS LEVEL=3 PHI=0.7 TOX=9.5E-09 XJ=0.2U
        TPG=-1
112 + VTO=-0.95 DELTA=2.5E-01 LD=7E-08 KP=4.8E-05
113 + UO=130 THETA=2.0E-01 RSH=2.5E+00 GAMMA=0.52
114 + NSUB=1.0E+17 NFS=6.50E+11 VMAX=3.0E+05 ETA=2.5E-02
115 + KAPPA=8.0E+00 CGDO=3.5E-10 CGSO=3.5E-10
116 + CGBO=4.5E-10 CJ=9.50E-04 MJ=0.5 CJSW=2E-10
117 + MJSW=0.25 PB=1
118 .ends inverter
119
```

```
120 .model D D
121 .lib C:\Program Files (x86)\LTC\LTspiceIV\lib\cmp\
      standard.dio
122 .model NMOS NMOS
123 .model PMOS PMOS
124 .lib C:\Program Files (x86)\LTC\LTspiceIV\lib\cmp\
      standard.mos
125 .tran 0 250m 0 1m
126 .model MYSW SW(Ron=1 Roff=100g Vt=.5 Vh=-.4)
127 .step param X list 1 2 3 4 5 6 7 8 9
128 .param RX_1 = {table(X, 1, 8000, 2, 8500, 3, 9000,
      4, 9500, 5, 10000, 6, 10500, 7, 11000, 8, 11500,
      9, 12000)}
129 .param RX_2 = {table(X, 1, 8000, 2, 8500, 3, 9000,
      4, 9500, 5, 10000, 6, 10500, 7, 11000, 8, 11500,
      9, 12000)}
130 .param RX_3 = {table(X, 1, 8000, 2, 8500, 3, 9000,
      4, 9500, 5, 10000, 6, 10500, 7, 11000, 8, 11500,
      9, 12000)}
131 .param RX_4 = {table(X, 1, 8000, 2, 8500, 3, 9000,
      4, 9500, 5, 10000, 6, 10500, 7, 11000, 8, 11500,
      9, 12000)}
132 .param RX_5 = {table(X, 1, 8000, 2, 8500, 3, 9000,
      4, 9500, 5, 10000, 6, 10500, 7, 11000, 8, 11500,
      9, 12000)}
```

```
133 .param RX_6 = {table(X, 1, 8000, 2, 8500, 3, 9000,
      4, 9500, 5, 10000, 6, 10500, 7, 11000, 8, 11500,
      9, 12000)}
134 .param RX_7 = {table(X, 1, 8000, 2, 8500, 3, 9000,
      4, 9500, 5, 10000, 6, 10500, 7, 11000, 8, 11500,
      9, 12000)}
135 .param RX_8 = {table(X, 1, 8000, 2, 8500, 3, 9000,
      4, 9500, 5, 10000, 6, 10500, 7, 11000, 8, 11500,
      9, 12000)}
136 .param RX_9 = {table(X, 1, 8000, 2, 8500, 3, 9000,
      4, 9500, 5, 10000, 6, 10500, 7, 11000, 8, 11500,
      9, 12000)}
137 .param RX_10 = {table(X, 1, 24000, 2, 25500, 3,
      27000, 4, 28500, 5, 30000, 6, 31500, 7, 33000, 8,
      34500, 9, 36000)}
138 .lib BEE215\LM741.lib
139 .lib LTC.lib
140 .backanno
141 .end
```