2014

Practical Issues in GPRAM Development

Yin Li
University of Central Florida

PRACTICAL ISSUES IN GPRAM DEVELOPMENT

by

YIN LI
B.S. Northeast Petroleum University, 2011

for the degree of Master of Science
in the Department of Electrical Engineering
in the College of Engineering and Computer Sciences
at the University of Central Florida
Orlando, Florida

Fall Term
2013

Major Professor: Lei Wei

ABSTRACT

In this thesis, two parts of practical issues in the GPRAM system design are included. The first part is the coding part. The sum-product decoding algorithm of LDPC codes has been refined to fit for the GPRAM hardware implementation. As we all know, communication channel has noise. The noise in telecom system is different from that in GPRAM systems. So the noise should be handled well in the GPRAM design. A noise look-up table was created for FPGA and those noises in the table are quantized. The second part of the thesis is to convert perfect images in video stream to those similar to the coarse images in human vision. GPRAM is an animal like robot in which coarse images are needed more than the fine images in order for us to understand how to GPRAM progresses those images to generate as clear image as we experienced. We use three steps, Point Spread function, inserting Poisson Noise, and introducing Eye fixation movements to mimic the coarse images seen merely from our eyes at the retinal photo-receptor level, i.e., without any brain processing.

ACKNOWLEDGMENTS

I am deeply grateful to Dr. Lei Wei for his role in providing valuable guidance in the idea of GPRAM. His idea expands my vision in the coding theory. He is the one who guides me to think more of how our brains work and how to make a similar machine has the analogical functions as biological brain. I want to thank again for his continuous help in achieving my masters' degree.

Also, I wish to thank the following present members in our lab at the University of Central Florida who rendered many suggestions and supports in the preparation of my thesis: Bowen Dai and Huihui Li.

In addition, I wish to thank Wenjun Zhang, for his support in the writing of the thesis.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER ONE: INTRODUCTION

Over the last thousands of years, people have been wondering what is intelligence, how to define intelligence and how to build intelligent machines? Scientists and engineers are more and more interested in developing a system with intelligence similar to human behaviors. But what is intelligence? In (Gardner, 1983), Gardner describes the intelligences in eight categories: visual-Spatial intelligence, verbal-linguistic intelligence, Bodily-kinesthetic intelligence, logical-mathematical intelligence, interpersonal intelligence, musical intelligence, intra personal intelligence, naturalistic intelligence. In a more general point of view, intelligence is the ability to learn or to apply knowledge to deal with one's situation. Artificial intelligence (AI) is the intelligence of machines (Mackworth & Poole, 1988). No matter whether the machine will behave exactly like a human or not, developing an intelligent machine is important. That kind of machines should ultimately have the ability to communicate, learn and adapt, but at this stage, we need to search for a way to develop this type of machines, similar to what human has achieved in development of a computer.

Computers have been developed as a type of intelligent machine since 1946. With the help of computers, many computational tasks which the brain could not handle with, has been solved by them with great accuracy far beyond what human brain can do. As soon as people discovered computational machines, people started to think that they were building computational brains. However, after nearly 70 year's research and development, the progresses in artificial intelligent disappointed many researchers. Is brain a computational machine or something beyond? In fact in (Neumann, 1958), Neumann clearly highlighted the differences between the popular computer architecture and how human neural cells represent information.

Computer did in binary digital, but neural cells use mixture of digital and analog representations and computations. Today, the computer can run much accurately, faster, and precisely than human brain, but yet many common features in human brains (even existing in infant brain) cannot be duplicated by computer, for example, face recognition or simple commonly understandable gestures.

During the same time period, in 1948, Shannon started information theory in which he introduced a new and mathematical way to measure information for probabilistic events. Based on his (Shannon, 1948) definition of information and theory, he discovered physical limit for telecommunication systems which was later called Shannon limits. After that, thousands of researchers in telecommunication industry have been worked for nearly 60 years to develop telecommunication system (mainly error control coding mechanism) to achieve this limit (Lin & Costello, 2004). Activities started in 1950s as block codes, to convolutional codes in 1960s, trellis coded modulation (TCM) in 1970s and 80s. In 1993, turbo codes revolutionized the error control coding field (Berrou, Glavieux, & Thitimajshima, 1993) soon after that, rediscovery of low-density parity-check code made near Shannon limit decoding possible (MacKay & Neal, 1996). The key themes in Turbo and LDPC successes include (a) better understanding extrinsic information, how to pass it between decoders and how to use it to improve error rate; (b) sparse connection between codes and Tanner graph. During the course, people discovered the connection between turbo iterative decoding and neural network processing such as Pearl's belief algorithm described in (MacKay, McEliece, & Cheng, 1998) (Pearl, 1988).

Meanwhile in biological field, people discovered STDP (Spike-timing-dependent plasticity rules) which play essential roles to build sparsely connected networks (Yang &

Froemke, 2002b) (Yang & Poo, 2006a) and Sparse net in visual signal processing (Olshausen & Field, 1996a). The process of STDP is interesting. The input spike to a neuron may cause the post-synaptic neuron’s excitation only when the input spike has high probability to occur in the future. STDP rule discourage network which connected closely, otherwise, it encourage network which is sparsely connected. Sparse connected network can develop template filters closely matched to what we observed in human visual systems. These researches, combined with error control coding achievement, attracted Dr. Wei to propose a new theory called GPRAM machine (Wei, 2012a).

According to (Wei, 2012a), GPRAM is short for General Purpose Representation and Association Machine. For the general purpose, the machine is not optimal for one purpose, which means, it will perform well even for tasks not known before. Thus, GPRAM is different from the conventional machine. Each part of the conventional machine is designed precisely and fixed. Conventional machine could only perform a task after it gets trained and tested. GPRAM, as a general purpose machine, supposes to cover as many tasks as possible in the beginning. What’s more, GPRAM is a hierarchy system. When it imports a rule, it narrows down to a lower hierarchy which is less vague than the higher hierarchy. We hope GPRAM could move and make decision by itself even for some tasks we did not train it before. Due to the differences between the two systems discussed above, the way to build the GPRAM should be different. In GPRAM, iterative coding is used instead of the precise logic design. Also, in GPRAM, the noises in the system may be beneficial. They could behave very different compared with telecommunication systems.

To design and develop GPRAM machines, we cannot follow the traditional computational pathway, rather than we aim to develop a simple demonstration system to show their unique properties. Currently Dr. Wei's team is working on this direction and this thesis is part of the effort. During the demonstration of the system's implementation, Byron McMullen (Dr. Wei's PhD candidate) needs to implement the simulation of GPRAM system on the FPGA board. Similar to implement decoder for telecommunication industry, FPGA needs to implement the decoder in digital format. So the metric or probabilistic measurements of information need to be quantized. The decoding algorithm in the implementation is the Sum-Product algorithm for LDPC codes. In this method, implementation of sum function is necessary as well as the product function. Sum function is easy to implement in the FPGA design, but the product function is complex to achieve due to the hardware limitation. Thus we need to refine the traditional SPA algorithm to make it easier for its implementation. The messages updated between the variable nodes and check nodes are quantized to the integer and passed through the quantization function in each iteration. This is my first contribution in this thesis.

Unlike the decoder used in telecommunication system, we also need to find an effective way to quantize noise as well. The noise in FPGA should not be real numbers anymore and the only way is to generate the Gaussian noise table. By proper selection of digitized noises, we can closely emulate the effects of real noises in FPGA implementation. This forms the second contribution of this thesis, which is to provide insights in this aspect.

During constructing of demonstration system, we also need to find a simple way to process visual signals. In today's electronic devices industry, the video camera is more and more accurate, precision, and faster. Therefore, for our demonstration systems, the video streams we aimed to put into the GPRAM are similar to what captured from human eyes. It has been well documented that images presented at human retina has poorer quality than what we can image. The image suffers from Poisson noise, point spread, and deteriorated by head motion. Yet, the coarse quality of images may play an important role in hieratical structure in GPRAM system.

Three steps are needed to generate the coarse images. The first step is using the point spread function to perform the two dot stimulus blurred image. The second step is adding the Poisson noise in each pixel of the images. The third step is modeling the real cases which our eyeballs are moving consistently. We model the drift-like eye movement as a Gaussian random-walk function. By doing the above three steps, the coarse image will be generated and ready to be used in the GPRAM systems. My third contribution to this thesis includes the following two parts: (a) learn from biological visual signal processing. (b) Implement various stage of image processing to mimic one or several effects using MATLAB. This lays a foundation for other team members to continue the work in this direction to build up visual system for our GPRAM prototype.

This thesis is organized as follows. Chapter two is the literature review part. I summarize the background of error control coding, GPRAM and human visual system. Chapter three introduces the coding and decoding algorithm of LDPC codes because it is used in GPRAM system. In chapter four, you may find out how I quantized the metric information in the iterative decoding and also the noise. The results are also shown in this part. Chapter five is the background knowledge for bio-visual processing. Chapter six shows the simulation results of visual imperfectness images. Chapter seven is the conclusion and future works part.

CHAPTER TWO: LITERATURE REVIEW

2.1 Literature review of error control coding and quantization

Many engineers believed that the transmission was unreliable due to the presence of channel noise. However, Shannon thought the channel noise may not be the reason for unreliable transmission. In 1948, Shannon discovered that channel noise limit the transmission rate, not the accurate rate of the transmission (Shannon, 1948). Soon after the theory of Shannon limit, researchers started to find ways to achieve this limit. The first type of codes is linear block codes, which heavily relied on Algebraic mathematical theory to design codes, analyze codes, and perform decoding (Lin & Costello, 2004). Algebraic theory dramatically cut down its complexity of encoder and decoder. Also it provided insight to design best possible codes of these types. However, it can only deal with hard decision decoder, i.e., the decision statistics need to be converted to binary decision before performing decoding procedure. This quantization of decision statistics lead to loss of around 3 dB compared with the decoders which use soft decision decoders. Different from algebraic approach, Gallager in his PhD thesis proposed low density parity check code (Gallager, 1963a), which is randomly constructed sparsely connected parity check codes. Unfortunately, due to limitation of computational power in 1960s, his design has never been simulated out until 1996 of rediscovery of the beauty of LDPC codes (MacKay & Neal, 1996).

In 1955, Elias introduced convolution codes as an alternative to block codes (Elias, 1955). Convolution codes gained substantial attentions due to its capability to perform soft decision decoders. In 1963 Fano introduced the Fano algorithm which searches over a tree for best possible decision path (Fano, 1963). In 1967, Viterbi discovered Viterbi algorithm which is proved to be the optimal soft decision decoding algorithm for small constraint length (typically less than 10) convolutional codes (Viterbi, 1967). In 1971, Heller and Jacobs found out that if we quantized branch metric to only 3 bits, the loss of quantization is only about 0.2-0.3 dB (Heller & Jacobs, 1971). Therefore, we can approach optimally implement Viterbi algorithm in digital circuits. In 1974, BCJR algorithm was discovered and named after the four creators (Bahl, Cocke, Jelinek, & Raviv, 1974). This algorithm has a great influence on the turbo codes and LDPC codes. BCJR algorithm can deliver soft decision output, but unfortunately at that time, none knew how to use the soft decision. In (Forney, 1973a), Forney demonstrated that the Viterbi algorithm just required one way process, while BCJR algorithm requires almost about twice complexity due to forward and backward processing. Thus, over the next twenty years, BCJR algorithm has never found applications in real world due to its complexity.

In 1981, NASA started to implement the Viterbi algorithm with 2^{15} states for deep space mission (Yuen & Vo, 1985). The performance of this system is about 2 dB away from Shannon limit (Dolinar, 1988). During the same time, Hagenauer and Hoeher proposed soft output Viterbi algorithm (SOVA) (Hagenauer & Hoeher, 1989), in which it not only takes soft-decision as input, but also generate soft output. In 1993 during Turbo coding invention, Berrou et al takes SOVA idea one step forward (Berrou, Glavieux, & Thitimajshima, 1993). Turbo codes are a new type of concatenated convolutional codes whose BER performance can achieve 0.7 dB away from Shannon limit. More importantly, Turbo codes define extrinsic information and then pass this information between two sub decoders to improve their reliability. The extrinsic information is based on soft decision output from decoder, which is called a feedback decoding method. Instead of SOVA, it uses BCJR algorithm to compute the extrinsic information. Although BCJR algorithm has twice the complexity for a given code constraint length, turbo codes use codes with small constraint lengths, so that its complexity is far less than conventional Viterbi algorithm which requires large constraint lengths.

Turbo codes excited the error control coding field. People started to search for new near optimal decoding algorithm. In 1996 Researchers rediscovered the Gallager's LDPC codes (MacKay & Neal, 1996). In 2001, Chung et al in (Chung & Forney, 2001) showed that LDPC codes can achieve 0.0045dB away from Shannon limit. During the progress, researchers developed analytical tools such as EXIT chart for turbo codes (Brink, 1999) and density evolution for LDPC codes. Using these tools, people discovered that as long as we construct codes with sparsely connected constraint, we can achieve near Shannon limit decoding (Richardson, 2001).

Turbo coding excitement did not stop at error control coding. Turbo codes as belief algorithm (Pearl J. , 1988) Graph theory has been a great implementation in many fields.

In (Wei, 2012a), Wei summarized the key lessons learned in error control coding are listed as follows:

1. Randomly constructed codes are good choices to achieve Shannon limit. (Gallager, Information Theory and Reliable Communication, 1968)
2. Due to the discovery of Tanner graphs (Tanner, 1981), near Shannon limit codes require to exclude small loops in Tanner graphs.
3. Message passing and updating between different sub-graphs provide a low complexity way to decode. (Berrou, Glavieux, & Thitimajshima, 1993)

4. Decoding can be implemented iteratively and result is not sensitive to noises and errors (Berrou, Glavieux, & Thitimajshima, 1993).
5. The discovery of connections between iterative decoding methods and Pearl's belief algorithm (Pearl, 1988) is significant in Bayesian network information processing (MacKay, McEliece, & Cheng, 1998).
6. With the help of low density graph and iterative decoding, many popular problems will be solved, for example, Fourier transforms and Kalman filtering (Kschischang, Frey, & Loeliger, 2001).
7. Having an abstract summary of how iterative decoding works into three steps: repetition, random shift, and non-linear operation (Forney, 2001b), Forney may bring us one step close to understand biological signal processing. The neurons could work the similar way as iterative decoding.

Is it possible unveiling the mystery on processing of a biological brain by the best known information transmission/reception mechanism? Dr. Wei not only paid a lot attention on long codes (Wei & Qi, 2000b), codes on graphs and large scale random ad-hoc networks (Wei, 2003c), but also reached to life science realm as it is the foundation of GPRAM theory.

2.2 Literature review of biological visual systems

Up to now, we do not know the architecture of biological brains. What makes matter worse is that over the last 50 years, our observations on human brains have been largely based on neuronal metabolism which consumes less than 5% of energy. In (Fox & Raichle, 2007), Fox and Raichle made the following statement. "Task-related increases in neuronal metabolism are usually small (<5%) when compared with this large resting energy consumption. Therefore, most of our knowledge about brain function comes from studying a minor component of total brain activity." We are not sure whether these approaches will lead to understand of basic principle of human intelligence and architecture of human brains. In a book review, "A neurocomputational jeremiad," appeared in the October 2009 issue of Nature Neuroscience, it pointed out that one of key ingredients missing in neuroscience researches is the architecture of bio-brains. Since 2003, Dr. Wei has been searching for biological lessons which may lead to new intelligent system design. Our team is following Dr. Wei's steps so that in this section, we will briefly review what in literature in biological sciences and mainly focus on bio-visual signal processing.

Between 2004 and 2007, Dr. Wei worked on human visual systems and he picked up a simple task called hyper-acuity capability. Hyper-acuity was coined by Westheimer (Westheimer, 1975) to describe a super-phenomenon that human visual systems can distinguish fine lines separated well beyond Nyquist rate. In fact, it will need a huge amount of templates for a tiny spatial of uncertainty to build a machine to achieve hyper-acuity at an accuracy of 1/10th to 1/30th of the diameter of photo-receptors (Klein & Levi, 1985). In (Wei, Levi, Li, & Klein, 2007), Dr. Wei and his collaborators at University of California at Berkeley showed in order to cover 9 by 9 arc min position uncertainty and speed uncertainty up to 2 degree/s, we need to construct 7776 templates if we follow ideal system design. This is only for a single stimulus with two simple dots. The human visual system can handle various hyper-acuity tasks under very dynamic environments, yet its performance closely approach to various ideal observers, which each is optimized to a particular setting. It seems the visual system has used a super approximated template which is approximated as the superposition of many ideal templates. It may tune toward a particular template smoothly and slowly when it is exposed to some artificially designed stimuli intensively.

During Dr. Wei’s visit to UC Berkeley, he also encountered two other research teams, one led by Dr. Yang who is working on STDP rules (Yang & Froemke, 2002b) and other led by Olshausen who is working in the area of sparse coding (Olshausen & Field, 1996a) (Olshausen & Field, 1997b). These works significantly opened his field of horizon.

People may wonder how does neurons connected and functioned together. Can we build a system which has the function of human brain? STDP rule was discovered by (Levy & Steward, 1983) when they tried to find out the time difference on the pre and postsynaptic actions on plasticity. Dr. Yang and M.M. Poo’s paper (Yang & Poo, 2006a) on neuromuscular helps us have a better understanding of the synaptic activity in a long term depression. Dr. Yang’s team has been working on observing the how will the cat’s brain acting when it sees human faces by fixing the cat’s head. The videos got from the cat’s brain have given light on Dr. Wei’s idea of visual part in GPRAM system.

Each item which is called neuron in sparse code is encoded by a set of neurons. This set is the subset of the original neuron before encoding, which supply a hierarchy structure in the coding process (Rehn & Sommer, 2007). Sparse coding has been greatly used in the computer vision field. In computer vision, the input patterns are huge. Thus, sparse coding will help in finding a small number of in use patterns and begin processing. Sparse coding provides us a state structure which is effective in the computer vision processing. In Olshausen’s neural coding in (Olshausen & Field, 1997b), the details of images will be resemble together in the eyes’ receptors, which will reduce the complexity in bio field. The idea of sparse coding has been used in the design of GPRAM. The input includes many information, using the idea of sparse coding, it will narrow down to a subset (Wei, 2012a), and then make decision.

In (Wei, 2012a), Dr. Wei summarized lessons learnt from biological system as follows:

[Blurring a dot in order to see it clear] The basic principle of human eye achieving hyper-acuity is to blur on one little dot stimulus. Knowing that, probably improving quality is not the only way achieving high resolution images. Take a step further, our eyes are continue moving even we sit still, so coarse images may be more suitable and will take less efforts and time. Same idea, although precision approach provides the optimal solution, it rolls out other smart ways which may not the optimal solution in this particular part, but maybe better and efficient result in all. Therefore, versatile or blurred approaches provide as many solutions as possible which may contain a better result regards to the whole system.

[Over-complete and sparse coding] In (Olshausen & Field, 1997b), Olshausen and Field's sparse coding idea lead us to understand how to use over-complete templates to achieve sparse neural activities.

[STDP rule (Yang & Froemke, 2002b)] STDP rule will be applied in the GPRAM system in the future, because it prevents the loops in the structure of coding and apply the posterior information calculation.

Overall, blurred or coarse images may perform better than the fine or high resolution images in the implementation of GPRAM system for the following reasons. First, at the initial levels, low quality images could be sufficient to recognize many different objects or deal with tasks at initial levels. Then, it will move to higher levels only when it is necessary. Second, if our cortex uses this coarse image to work out a result far better than sharp one, then its focus does not need to be improved. Third, prepared in blurred could easily handle head and eye movements. Forth, coarse image does not need a lot computation power to process them. Fifth, fine images have similar low resolution counterpart, so these can be grouped together to form hierarchical structures. Sixth, we just take random sampling points out of images, so coarse images will perform as good as fine ones.

2.3 Literature review of GPRAM systems

Based on lesson learnt in error control coding and biological visual system, Dr. Wei proposed GPRAM system in (Wei, 2012a). GPRAM is general purpose intelligent system which may lead to systems which could perform functions similar to a bio-brain. GPRAM could move and make decision by itself even for some tasks we haven't trained yet. GPRAM has hierarchy structure. At the highest hierarchy, it does rough and quick estimates on how to process a task. When adding a rule into the system, it becomes more special on processing the task and finally completing the task in a lower hierarchy. Conventional machine was designed precisely and functionally. It only could perform a task after it gets trained and tested. This is the main difference between GRPARM and conventional machine. The design of GPRAM system includes two procedures (Wei, 2012a):

Each part of the conventional machine is designed precisely and fixed. Conventional machine could only perform a task after it gets trained and tested. GPRAM, as a general purpose machine, supposes to cover as many tasks as possible in the beginning. What's more, GPRAM is a hierarchy system. When it imports a rule, it narrows down to a lower hierarchy which is less vague than the higher hierarchy.

The first part is group design. At this part, GPRAM collects as much information as possible. Then set up the aim of the machine. After that, a specific rule which narrows the scope of GPRAM system into a subset was added into the system. After these three steps, GPRAM will contain less uncertainty compare to the beginning. The second part is individual specification. According to different tasks, select different learning methods to implement the task by using existing technology. The design procedures of GPRAM are different from traditional machine. In tradition machine, how it will act to achieve one function is determined. Whereas in GPRAM, different approaches are being used which work similar as human brain. Imagine this scenario, if a person goes to a city and encounter with heavy traffic on his way, he may pick another route to that city next time. Just like that, GPRAM will act as human brain which is learning from its past experience. It will be not achievable by traditional machine.

After a brief understanding of GPRAM design, Dr. Wei and his team clarified the method for a GPRAM prototype by adopting LDPC codes in (Li, Dai, Schultz, & Wei, 2013). They used coding mechanism to build the system instead of the traditional logic design. At first, Hamming (7, 4) code was used at first to present the connections in GPRAM. Due to short length of Hamming (7, 4) code, the tasks it performed were limited. Therefore, longer codes, LDPC codes were tried in the simulations. It has been conclude that the longer the code word is, the more difficult for the system to learn. In the future, the learning algorithm should be improved for applying long codes. It also indicates that, bio-implications need to be developed.

To learn one task each time may not be enough for an intelligent system. In (Li & Wei, 2013c), they carry on performing multi-tasks learning method. This paper continues on the exploration of LDPC codes to establish GPRAM system. Good codes were used to improve the performance of GPRAM. Also, progressive learning was introduced in the paper to improve the function of multi-tasks learning. The current error control coding knowledge may not enough for supporting the GPRAM design, to move one step further, the knowledge of it needs to be extended.

The primary goal is to achieve an animal-like-robot, which can perform simple function for demonstration. In (Li, Dai, Schultz, & Wei, 2013), different length of code words have been simulated, good code word has picked. Thus, based on (Li, Dai, Schultz, & Wei, 2013), Byron McMullen (Dr. Wei's PhD candidate) is working on converting the simulation method into FPGA. Sensors s_1 and s_2 are inputs in (Li, Dai, Schultz, & Wei, 2013), in the FPGA design, wireless camera and microphone will substitute two input sensors. By doing that, outputs are images and sounds, which would be more directly to convince people.

As Mr. McMullen has been working on implementation the GPRAM system on FPGA, some troubles have encountered. First of all, how to implement decoder in digital format? Secondly, how to generate noises in FPGA to mimic noise effects in real systems? Thirdly, how to deal with simple image input? In his initial trials, he simply selected 4 bit quantization to approximate decoding computation and 4 bits to quantize noises, which lead to huge loss in implementation as we will see in the later section of this thesis. He also implemented very simple visual image process. My thesis took these challenges and evaluated how to near optimally achieve decoding and mimic human visual images.

CHAPTER THREE: QUANTIZATION IN LDPC DECODING AND NOISE

In this chapter, we will first review error control coding in telecommunication system design in Section 3.1. Then, we highlight application of LDPC coding in GPRAM prototype design in Section 3.2. After that, we illustrate what has been done in Mr. McMullen's implementation and its performance in Section 3.3. We then present our work on digital implementation of LDPC decoding in Section 3.4. We further present our work on digitalizing noises in Section 3.5. Finally, we conclude this chapter.

3.1 Error Control Coding in Telecommunication systems

Coding theory is an important skill in the telecommunication. It is a research field which helps us to understand how information was transmitted between transmitter and receiver through an unreliable channel. By using coding and decoding technique, the error-free transmission over an unreliable channel is practicable.

The following figure shows the block diagram of digital communication system. At the transmitter side, the encoder adds redundant bits to create a code word and will be used for error correction. Also, with the help of encoder, the rate of the code will be limited by channel capacity. Modulator transforms the code word into a format suitable for the channel, like the analog carrier signal. Then pass the modulated signal through a noisy channel. The demodulator attempts to recover the correct channel symbol in the presence of channel noise. Decoder then uses the redundant bits to detect or correct the bit errors and made hard or soft decision to quantize the receive bit to be "0" or "1" and received by the receiver.

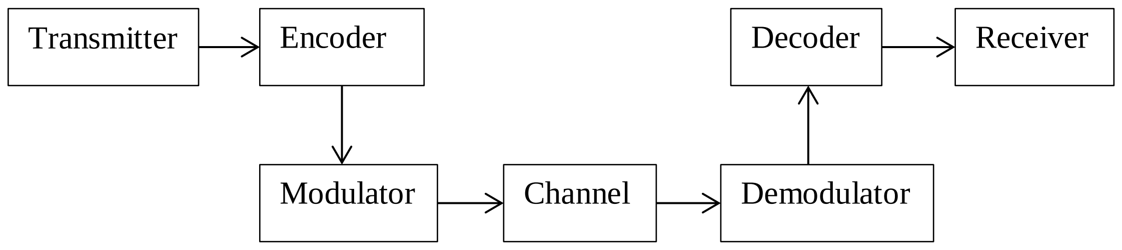


Figure 1 Block Diagram of Communication System

3.1.1 Description of Several Types of Error Control Coding

3.1.1.1 Linear Block Codes

The information transmitted in block codes is formatted into binary form, and grouped into blocks of k information bits. Messages are represented as $M=2^k$ and mapped to a longer block of n bits, which n>k, called code-words. The code-words have systematic structure, as shown in following figure. The structure has two parts, the checking part and the message part.

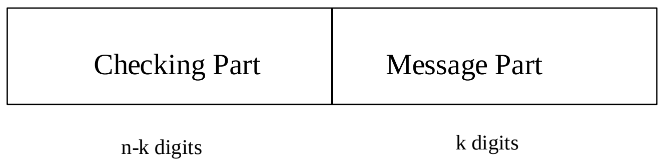


Figure 2 Structure of Codewords

The code rate denotes as R=k/n, which is ratio of information bits represented in the messages. Generator matrix (G matrix) is used to generate all possible codewords. G matrix could be represented by the following structure: $G=[I_k \mid P]$, where I_k is an identity matrix which is k by k. P has the dimension of k*(n-k).

Parity check matrix is derived from the G matrix or vice versa. The Parity check of the above G matrix is the following: $H= [P^T \mid I_{n-k}]$. G matrix and parity check matrix should satisfy the following equation: $G\ H^T=0$. Several examples of block codes will be introduced in the following:

3.1.1.2 Repetition Code

Repetition Code is one of the easiest ways of implementing the coding theory just by repeating the transmitted symbol n times. The code rate is 1/n. For example, a repetition code with n equals to 3. The code words are (000) or (111), i.e., if the message is 0, we select code word (000); and if the message is 1, we select code word (111). Thus, if we receive code words as (001) or (110), then we may be regarded as an error event and using simple decoding rule, we can recover from this one bit error event. If there are error events more than 1 bit, then we cannot recover.

3.1.1.3 Hamming Code

Hamming code is also a kind of block codes, discovered by Richard Hamming in 1950. To illustrate more properties of block code, we use (7, 4) Hamming code as an example. The code rate is 4/7. (7, 4) Hamming code has 16 possible code words, seen in Fig. 3. In each 7-bits code word, contains 4-bits message data (the first 4 bits) and 3-bits check codes (the last 3 bits).

In Fig. 3, we also include generator matrix (**G**) and parity check matrix (**H**). Let **m** be a 4 by 16 message matrix, which includes all 16 tuples from 0000 to 1111**,** seen the first 4 bits in code word. We can generate all code words (**C**) by **C**=**mG**. Also, we can check code words by using $\mathbf{CH}^T=\mathbf{0}$, that is, any valid code word will satisfy parity check matrix, where $\mathbf{H}^T$ is the transpose of **H**.

$$\begin{bmatrix} 0\,0\,0\,0 & 0\,0\,0 \\ 0\,0\,0\,1 & 0\,1\,1 \\ 0\,0\,1\,0 & 1\,1\,0 \\ 0\,0\,1\,1 & 1\,0\,1 \\ 0\,1\,0\,0 & 1\,1\,1 \\ 0\,1\,0\,1 & 1\,0\,0 \\ 0\,1\,1\,0 & 0\,0\,1 \\ 0\,1\,1\,1 & 0\,1\,0 \\ 1\,0\,0\,0 & 1\,0\,1 \\ 1\,0\,0\,1 & 1\,1\,0 \\ 1\,0\,1\,0 & 0\,1\,1 \\ 1\,0\,1\,1 & 0\,0\,0 \\ 1\,1\,0\,0 & 0\,1\,0 \\ 1\,1\,0\,1 & 0\,0\,1 \\ 1\,1\,1\,0 & 1\,0\,0 \\ 1\,1\,1\,1 & 1\,1\,1 \end{bmatrix} \qquad G = \begin{pmatrix} 1\,0\,0\,0 & 1\,1\,0 \\ 0\,1\,0\,0 & 1\,0\,1 \\ 0\,0\,1\,0 & 0\,1\,1 \\ 0\,0\,0\,1 & 1\,1\,1 \end{pmatrix} \; H = \begin{pmatrix} 1\,1\,0\,1\,1\,0\,0 \\ 1\,0\,1\,1\,0\,1\,0 \\ 0\,1\,1\,1\,0\,0\,1 \end{pmatrix}$$

Figure 3 (7, 4) Hamming code words, generator matrix (G) and parity check matrix (H)

The Hamming distance is very important in the coding theory. Two equal length code words were needed to calculate the Hamming distance. The number of differences of positions at the corresponding symbols is the Hamming distance. For example, the distance of (1101110) and (1110111) is 3. The minimum distance usually denotes as d_{min} is the minimum Hamming distance between two code words. The Hamming weight of a code word is the number of non-zero positions. For example, the Hamming weight of (0011100) is 3. The (7, 4) Hamming code can detect 2 errors, can correct 1 error.

3.1.1.4 LDPC codes

LDPC code was developed by Gallager in 1963 (Gallager, Low-Density Parity-Check Codes, 1963), which is a linear block code. LDPC codes are represented by a sparse parity-check matrix (**H**). That is, the parity-check matrix **H** contains fewer 1's than the amount of 0's, and has a fixed number of 1's per row and also per column. As for an (n, k) LDPC code, n denotes the variable nodes, n-k denotes the check nodes connected to the variable nodes. To encode the LDPC code, the parity-check matrix **H** should be rearranged by Gauss-elimination into $[\mathbf{P}^T \mid \mathbf{I}_{n\text{-}k}]$. After get the modified matrix, generator matrix **G** could be obtained by using $\mathbf{G}\cdot\mathbf{H}^T=0$. Then the codewords are generated from $\mathbf{C}= \mathbf{G}^{\mathbf{T}}\boldsymbol{\cdot}\mathbf{m}$, where m denotes the messages vector. More details of LDPC codes will be introduced in section 3.1.2.

3.1.1.5 Convolutional Codes

Convolution code was developed in 1956. It is another important technique in coding theory. The encoder does not encode the information in block form; it has memory. It uses the both previous and present information to format the code-words. The previous information is stored in shift register. Each encoded block does not depend only on the corresponding message at the same time unit, but also depends on previous information blocks which remembered in the register. The following figure shows the simple block diagram of a convolutional encoder.

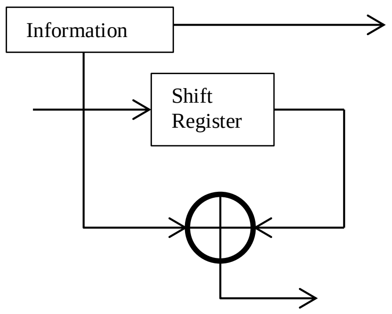


Figure 4 Block Diagram of Convolutional Encoder

One previous bit, together with the newly input data bit, determines the output of coded bit. The other encoded bit should be the present bit information. According to apply different code rate, the number of shift registers was various. Convolutional encoders various however they will fall into two general categories: feed-forward and feedback. In each category, convolutional encoders can be different with data rate. A feed-forward with rate ½ will be illustrated in the following figure.

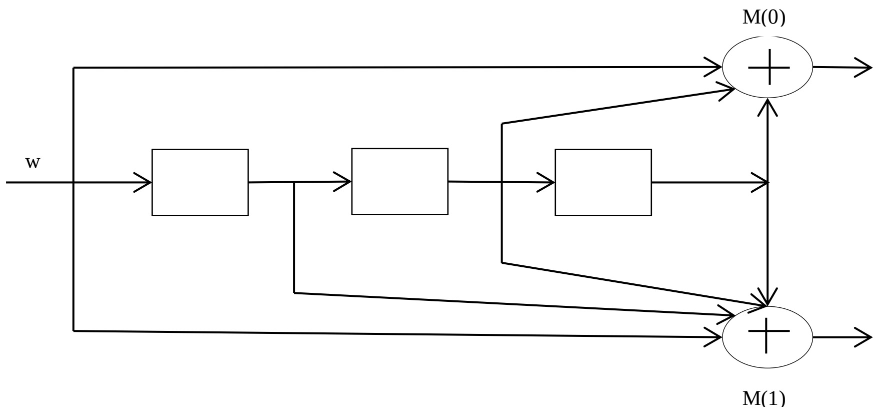


Figure 5 Convolutional Encoder

This encoder consists of three registers and two modulo-2 adders M (0), M (1). Information sequences w= (w_0, w_1, w_2, w_3…) enter this linear encoder one bit a time. One information bit enters in this system, and two encoded bits come out and form into new coded sequences. Time sequence unit or called generator sequence for adder M (0) should be g (0) = (1 0 1 1), for adder M (1) should be g (1) = (1 1 1 1). Then, the encoding equations should be easy to get:

$$\mathrm{M}(0) = w \otimes g(0) \tag{1}$$

$$\mathrm{M}(1) = w \otimes g(1) \tag{2}$$

The output sequence could obtain by implement the following equation:

$$M_i^j = \sum_{i=0}^{3} w_{k-i} g_i^j = w_k g_0^j + w_{k-1} g_1^j + w_{k-2} g_2^j + w_{k-3} g_3^j, j = 0,1 \tag{3}$$

where k is the length of the information sequence.

For example, for the information sequence w= (1 0 1 1 1), then

$$\mathrm{M}\ (0) = (1\ 0\ 1\ 1\ 1) \otimes (1\ 0\ 1\ 1) = (1\ 0\ 0\ 0\ 0\ 0\ 0\ 1) \tag{4}$$

$$\mathrm{M}\ (1) = (1\ 0\ 1\ 1\ 1) \otimes (1\ 1\ 1\ 1) = (1\ 1\ 0\ 1\ 1\ 1\ 0\ 1) \tag{5}$$

Thus, the code-word is

$$\mathrm{M} = (1\ 1, 0\ 1, 0\ 0, 0\ 1, 0\ 1, 0\ 1, 0\ 0, 11) \tag{6}$$

3.1.1.6 Turbo Codes

The technique of Turbo coding was introduced by (Berrou, Glavieux, & Thitimajshima, 1993) in 1993. This technique concatenates two convolution codes; two encoders are arranged in parallel. The information is encoded two times. The following figure shows the block diagram of turbo encoder. The information pass through the first encoder to do the first time encoding, before it pass through the second encoder, a random interleaving block is needed. In this procedure, the output sequences which passing by two encoders should be independently. Information passing by two encoders will reduce the code rate. However, puncture block is designed to improve the rate of the code.

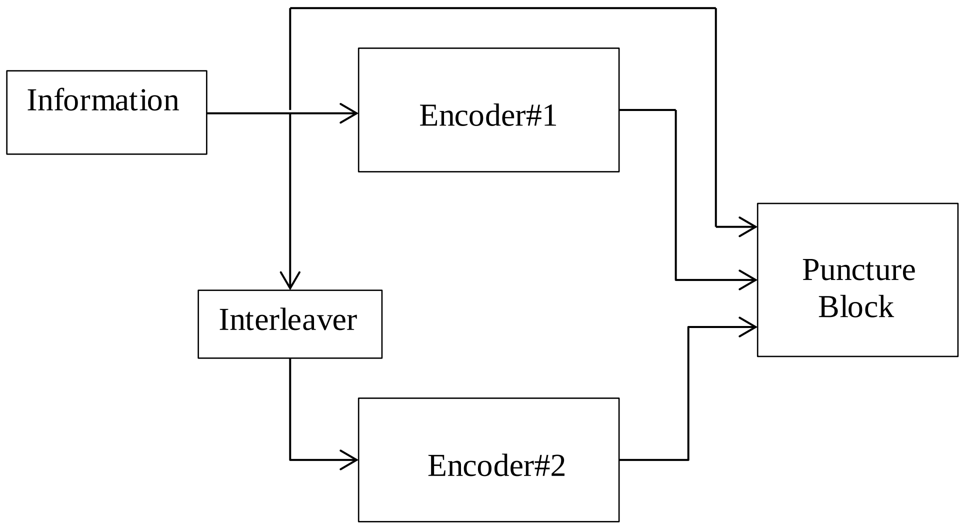


Figure 6 Block Diagram of Turbo Encoder

3.1.2 Description of Several Decoding Procedures

3.1.2.1 Maximum A Posterior (MAP) and Maximum-likelihood (ML) Decoding

Maximum-likelihood (ML) decoding is to maximize the probability that received noisy code-words **r** given the condition code-words **c** were sent, i.e., max P(r | c). Maximum a posterior (MAP) decoding is to maximize the probability that probability that a bit c_i was sent given the condition **r** is received, i.e., max P(c | r). According to Bayes theorem,

$$maxP(c|r) = max\frac{P(r|c)P(c)}{P(r)} = maxP(r|c)\cdot P(c) \qquad (7)$$

When codewords are equal-probable, $P(c_m)=1/m$, equation (7) can be rewrite into

$$maxP(c|r) = maxP(r|c) \qquad (8)$$

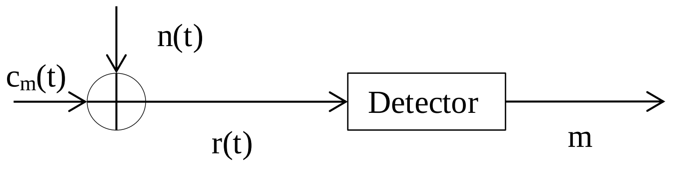


Figure 7 Flow chart shows message decoding process

For AWGN channels, we have

$$\mathrm{maxP(r|c) = maxlogP(r|c)}$$

$$= \max[-\frac{N}{2}\log(\pi N_0) - \frac{1}{N_0}\sum_{k=1}^{N}(r_k - c_{mk})^2] = \min\sum_{k=1}^{N}(r_k - c_{mk})^2$$

If we have equal-probability, MAP decoder is the same as ML decoder. But for a given code trellis, MAP method is more complexity than ML method and MAP method needs to estimate noise variance.

3.1.2.2 Decoding method of Linear Block Codes

Syndrome decoding method has higher efficiency compare to ML decoding especially for linear codes. Consider a simple (7, 4) Hamming code.

$$G = \begin{pmatrix} 1\,0\,0\,0 & 1\,1\,0 \\ 0\,1\,0\,0 & 1\,0\,1 \\ 0\,0\,1\,0 & 0\,1\,1 \\ 0\,0\,0\,1 & 1\,1\,1 \end{pmatrix} H = \begin{pmatrix} 1\,1\,0\,1\,1\,0\,0 \\ 1\,0\,1\,1\,0\,1\,0 \\ 0\,1\,1\,1\,0\,0\,1 \end{pmatrix}$$

From the property of parity matrix, for any codewords c, $H \cdot c^T = 0$. Thus, if the received codewords have no errors, $H \cdot r^T = 0$. The decoded message should be the received k bits. If $H \cdot r^T$ is not equal to zero, errors exist in the received codewords. $H \cdot r^T = H \cdot (c + e)^T = 0 + H \cdot e^T$. Let's assume there is at most only one bit error and error is the first k elements which e is 1. For example, if e= 0 1 0 0…0, then the error is the second bit. The above equation $s = r \cdot H^T$ which used to detect whether or not is called the syndrome of the received signal r. Syndrome s is the summation of the received parity signal (r_0, r_1 r_2 …r_{n-k-1}) and parity check signal (r_{n-k}, r_{n-k+1}, …, r_{n-1}).

Most of hard decision decoders are based on efficient computation syndromes and find the error location based on syndromes. There are also various methods to perform soft decoding of block codes. The readers can find these algorithms in (Lin & Costello, 2004).

3.1.2.3 The Viterbi Algorithm for soft decoding convolution codes

Viterbi algorithm was introduced in (Viterbi, 1967) as a decoding method for convolution codes by performing maximum likelihood decoding. After the convolutional encoder, the bit block can be modulated into BPSK signals and to be transmitted through an AWGN channel. Modulated signal is represented in the following equation:

$$s_i = \sqrt{E_b R}(2v_i - 1) \qquad (9)$$

where E_b is the energy per bit, R is the code rate, v is the bit stream.

The received signal should be the modulated signal plus Gaussian noise with zero mean and varience $\frac{N_0}{2}$, which is the following equation $r_i = s_i + n_i$. Viterbi decoding algorithm usually includes three steps:

Step1: branch metric calculation. Branch metric is used to calculate the distances between received signals and transmitted signals. The branch metric for Viterbi algorithm is

$$w(v_i) = -\log P(r_i \mid v_i) \qquad (10)$$

which is always greater or equal to zero. Beginning at time unit t=1 in the following figure, separately calculate the partial metric of each path which entering in the state. Keep records of the path and metric for each state.

Step 2: Increase the time unit by one, which means at time t=2, calculate the branch metric for all paths is by adding the branch metric entering that state together with survivor path at t-1 time unit. Keep records of the path and metric for each state.

Step 3: Keep repeating step 2 until reach the final state. Trace back the survivor path.

Trellis diagram is used to implement this algorithm. Viterbi algorithm is finding the maximum likelihood path through the trellis. Black nodes stand for the states in time sequence.

Branch metric stands for the information transmission. Labeled number on each branch metric stands for the length of received data between the four states. The Viterbi algorithm shows in the following:

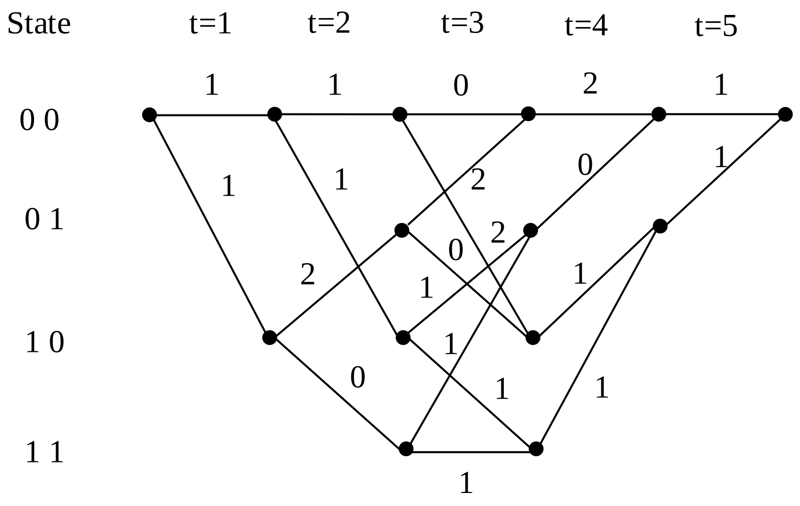


Figure 8 Viterbi Decoder

Step 1: Initialization

At time unit 0, the survivor path should start from zero.

Step 2: Recursion

Compute the shortest path of each state by adding the previous time branch metric. Store the survivor path and delete other paths.

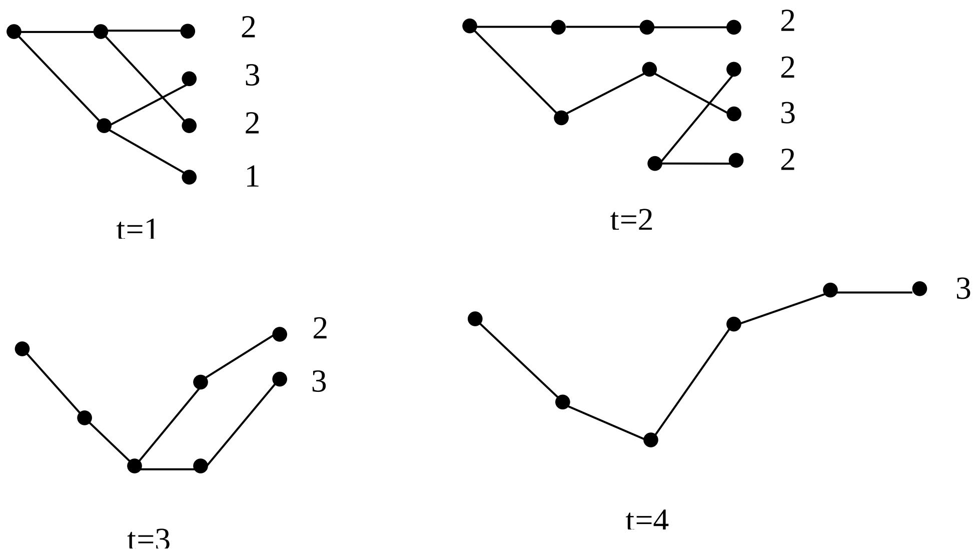


Figure 9 Decoding Procedure of VA algorithm

From the figure above, trace back the final path, we can get the decoded codes.

3.1.2.4 BCJR Algorithm for soft decoding convolutional codes

BCJR algorithm can be applied to block codes as well as convolutional codes. This algorithm is complicated than VA algorithm, so it doesn't bring much attention until turbo codes discovered. BCJR is also known as Maximum a posteriori probability (MAP) decoding. The input information should be the log-likelihood ratio (LLR) value, which we will discuss more details in the LDPC decoding algorithm. After the processing of the decoder, a hard decision will be made. If the LLR value is less than 0, the decoder will estimate the sent bit equals to -1. If LLR value is larger or equal to 0, decoder will count it as +1.

3.1.2.5 Sum-Product Algorithm for decoding LDPC block codes

LDPC code can be decoded in various ways. In this paper, we only focus on the sum-product algorithm. It is known that the sum-product algorithm (SPA) has the better decoding performance than other decoding algorithms for LDPC codes.

It is an algorithm which uses iteration to update the soft information between check nodes and variable nodes. The implementation of SPA decoding is based on the computation of a posteriori probabilities, its log-likelihood ratio (LLR). Then, the LLR for each code bit is given by the following equation.

Step 1: Initialization

In the initialization step, each variable node may get the posterior probability at the receiver, which is the information from the transmitter over AWGN channel. The soft information

$\lambda_{n\to m}$ denotes the message transmitted from the nth variable node to mth check node.

$$\lambda_{n\to m}(C_i) = \text{LLR}\,(C_i) = \frac{2y}{\sigma^2} \qquad (11)$$

where y denotes the received signal over AWGN channel, which is the real number.

$\Lambda_{m\to n}$ denotes the message transmitted from mth check node to nth variable node.

$$\Lambda_{m\to n}(C_i) = 0 \qquad (12)$$

The mth check node will accumulate the LLR information getting from the other check nodes.

Step 2(1): Check node update

From the knowledge of the message passing in the last section, for each check node m, we need to calculate the accumulate values in the $n\in N\,(m)$, shows in the following:

$$\Lambda_{m\rightarrow n}(C_i) = 2\tanh^{-1}\prod_{\mathrm{n}'\in \mathrm{N(m)}\backslash \mathrm{n}}\tanh\left(\frac{1}{2}(\lambda_{n'\rightarrow m})\right) \qquad (13)$$

Step 2(2): Variable node update

For each variable node *n*, sum up all the information from the check nodes, which in the set m∈M (*n*) and also the information of the variable node itself.

$$\lambda_{\mathrm{n}\rightarrow \mathrm{m}}(C_i) = \mathrm{L}(C_i) + \sum_{\mathrm{m}'\in \mathrm{M(n)}\backslash \mathrm{m}}\Lambda_{\mathrm{m}'\rightarrow \mathrm{n}}(C_i) \qquad (14)$$

For each iteration, compute:

$$\lambda_{\mathrm{n}}(C_i) = \mathrm{L}(C_i) + \sum_{\mathrm{m}\in \mathrm{M(n)}}\Lambda_{\mathrm{m}\rightarrow \mathrm{n}}(C_i) \qquad (15)$$

Step 3: Make decision

Quantize C_n such that

$$C_n=0 \text{ if } \lambda_n(C_i)<0, \text{ and } C_n=1 \text{ if } \lambda_n(C_i)>=0 \qquad (16)$$

3.1.3 Quantization Issues in Implementation of Decoding Algorithms

3.1.3.1 Quantization of Viterbi Algorithm

In order to design Viterbi decoder, quantization of state metrics is needed. In (Heller & Jacobs, 1971), quantization method was presented. Viterbi decoder used log-likelihood in each path calculation. Instead of that, the metric was quantized into 2-, 4-, and 8 -levels. The metric used digital symbol, which shows in the following figure.

8-level:	0	1	2	3	4	5	6	7
0	7	6	5	4	3	2	1	0
1	0	1	2	3	4	5	6	7

Figure 10 8-level quantization code metrics

After the convolutional encoder, the encoded bits are sent into an AWGN channel, after BPSK modulation, all bits are mapped into +1 or -1. The noise is zero mean, independent, and double-sided power density. At +1 side, combine with the above figure, branch metric was quantized from 0 to 7. At -1 side, branch metric was quantized from 7 to 0. The branch metric has +1 and -1 part. Thus, yielding to the following equation $\log\frac{P(r|v=0)}{P(r|\mathrm{v}=1)}$, in which, the r should be the value from 0 to 7.

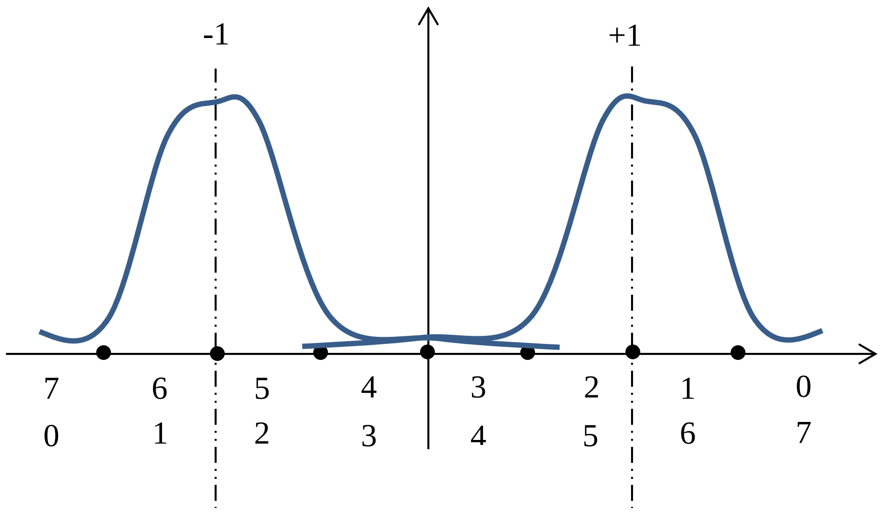


Figure 11 Quantization range

State metric in Viterbi algorithm could also be quantized. One method is rescaling approach introduced by Hekstra in (Hekstra, 1989). This implementation is to subtract the minimum metric, which is c_k, from a metric $m_s(k)$ continuously.

In traditional method, path metric from state s at time t-1 to a state m at time t denotes as $b_{sm}(k)$, information transition could be shown in following equation:

$$m_{sm}(k) = m_s(k-1) + b_{sm}(k) \qquad (17)$$

After several accumulations, overflow problem may occur. To avoid this problem, rescaling approach will apply without sacrifice correctness. In equation 17, addition operator could be change to modulo 2^c operator (for c bits). Based on the Two's complement arithmetic in (Hekstra, 1989), equation 17 should be modified as follows:

$$m_{sm}(k) = m_s(k-1) + b_{sm}(k) - \left(m_{s'}(k-1) + b_{s'm}(k)\right) \text{mod } 2^c \quad (18)$$

where in equation 18, replacing state accumulation by residuals $m_{sm}(k)$ mod 2^c. By using this method, no overflow problem will occur. This rescaling method will use one more bit than traditional accumulation method, however due to high probability of errors will occur when we met metric overflow, we would rather agree on adding one bit data.

3.1.3.2 Quantization of Iterative Viterbi Algorithm

IVA decoding procedure will be discussed in this section. Branch metric equation shows in Section 3.1.1. Decoding steps will show clearly in the IVA flow diagram.

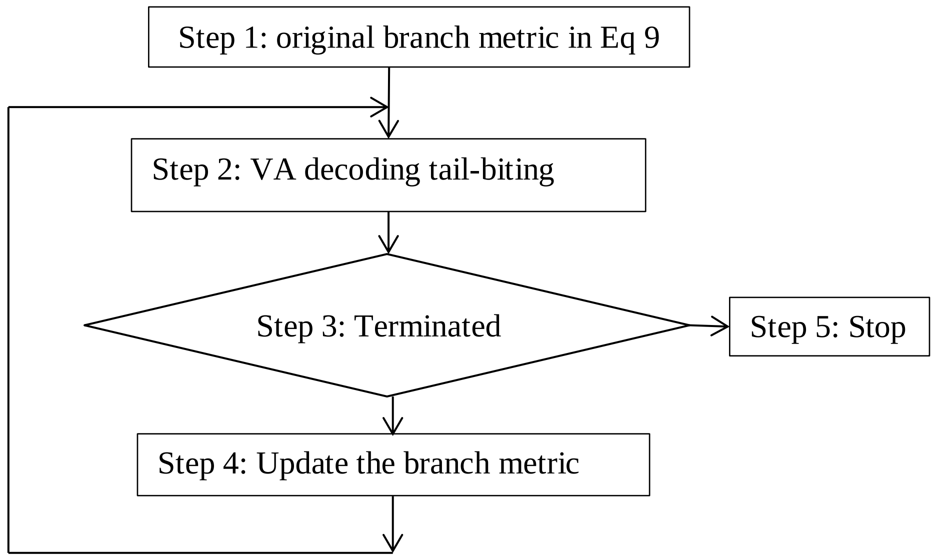

Step 1: original branch metric in Eq 9
Step 2: VA decoding tail-biting
Step 3: Terminated
Step 5: Stop
Step 4: Update the branch metric

Figure 12 Flow chart of VA decoding

In Step one, compute the branch metric using equation 9 and apply to traditional Viterbi algorithm in step two. Iteration begins in this step. Iteration will be terminated if the decoded codeword is valid or the number of iterations steps reaches the maximum value. Otherwise, it will update the branch metric and step back to Step 2.

In IVA branch metric will be modified as the following equation:

$$w^U(v_i) = \mathrm{w}(v_i) + (-1)^c \lambda w(v_A) \tag{19}$$

where w^U is the updated metric, v_A denotes the bit used in its branch metric. $\mathrm{w}(v_i)$ denotes the metric of v_i. The quantization table can be simplified as follows:

Originalw:	0	1	2	3	4	5	6	7
λw	0	0	0	0	0	1	1	1

Figure 13 Quantization Level

As shown in table 2, the branch metric is quantized into 3-bit, the original branch metric w is 7, however, the metric is scaled down to be either 0 or 1. λ=1/4 in this table.

State metrics will constantly update in the iterative decoding. Each branch metric could be quantized into 3 bit value (8 quantization levels) as shown in section 3.1.4. Branch metric will keep gathering the non-negative symbol value form the branch metrics. Thus, state metric will facing the overflow problem in the iterative VA. State metric normally has 7-bit or 8 bit precision. In (Wei, 2004), two quantization methods were implemented to solve the overflow problem. One method is state metric rescaling, which all state metrics subtract the smallest state metric after each decoding step. If 3 bits quantization is conducted, the value of branch metric per transmission bit should be no larger than 7. For any ½ rate code with the number of registers in convolutional encoder m_c could be 2, 3, 4, 5, 6, 7, 8, and $m_c \cdot n_c \cdot B$=28, 42, 56, 70, 84, 98, 112. Thus, the state metric quantization should be 5 bits, 6 bits, 6 bits, 7 bits, 7 bits, 7 bits, 7 bits, This method do not need to change the precision of branch and state metric in the decoding procedure. Another method in(Wei, 2004) is the two's complement arithmetic approach. The number of bits in state metric has limitation, which is

$$c \geq log_2[(m_c + 1)n_c B + 1] + 1 \qquad (20)$$

If we also select 3 bit quantization as in state metric rescaling, the state metric quantization should be 7 bits, 7 bits, 8 bits, 8 bits, 8 bits, 8 bits, 8 bits. Hence, the precision of state metrics only need to increase for 1 bit, however the improvement of the metrics are only needed for some cases.

3.1.3.3 Quantization of Sum-Product Algorithm

The core operation for the check node update is the calulate the by log-likelihood function. By using two times Jacobian algorithm, the function should be revised to a linear function and easy to design a look-up table in the decoding procedure.

$$\mathrm{LLR}(U * V) = log\frac{1+e^{\mathrm{LLR}(U)+\mathrm{LLR}(V)}}{e^{\mathrm{LLR}(U)+\mathrm{LLR}(V)}} \tag{21}$$

$$= \log\left[1 + e^{\mathrm{LLR}(U)+\mathrm{LLR}(V)}\right] - log\left[e^{\mathrm{LLR}(U)+\mathrm{LLR}(V)}\right]$$

By using Jacobian logarithm twice, equation (21) should be modify into the following:

$$\max[0, LLR(U) + LLR(V)] + log\left(1 + e^{-|LLR(U)+LLR(V)|}\right) - \max[LLR(U), LLR(V)] - log\left(1 + e^{-|LLR(U)+LLR(V)|}\right)$$

$$= sign[LLR(U)]sign[LLR(V)] \cdot min[|LLR(U)|, |LLR(V)|] + log[1 + e^{-|LLR(U)+LLR(V)|}] - log[1 + e^{-|LLR(U)-LLR(V)|}]$$

The above two log terms could be regard as the following funcion:

$$g(x) = log\left(1 + e^{-|x|}\right) \tag{22}$$

The quantization range shows in the following figure. In the hardware implementation, a look up table is generated by the guidence of the following figure.

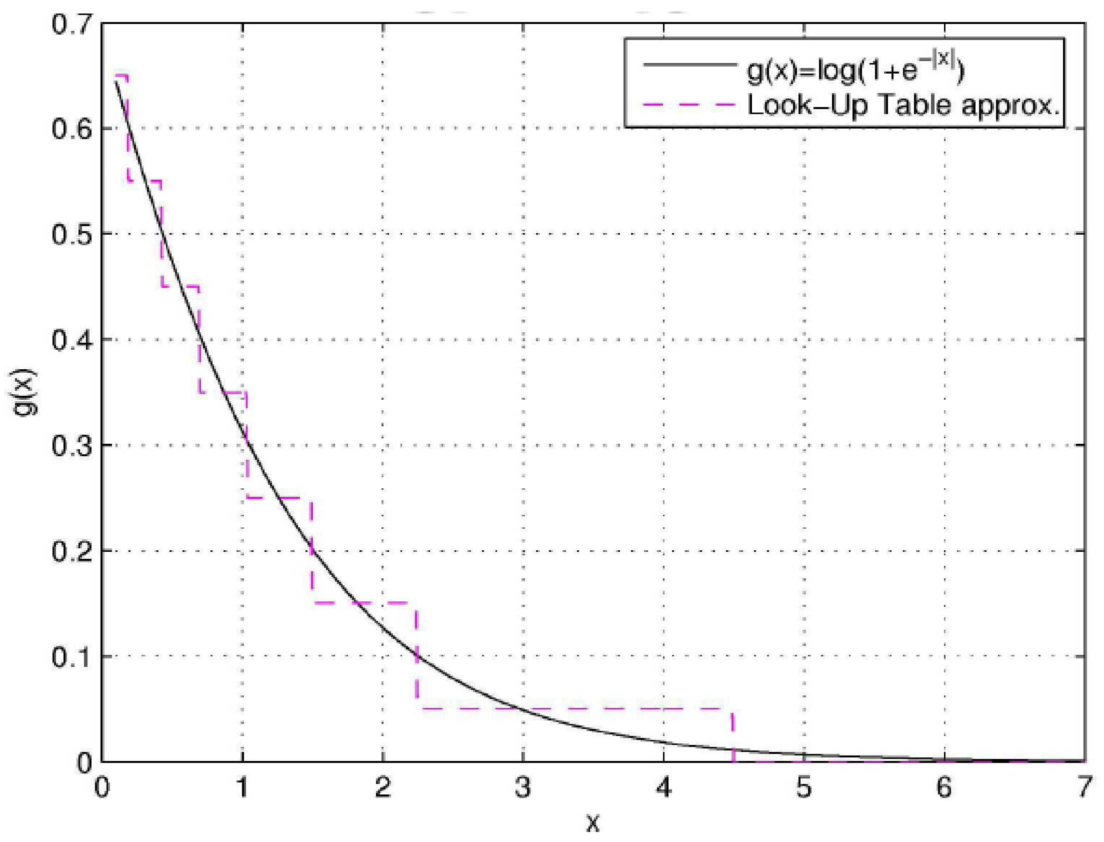


Figure 14 LUT design

The decoding implementation should be achieved by the serial implementation and parallel implementation. Serial implementation can be view as the forward and backward implementation. This method has a limitation, it can deal with two nodes each time, so in require of high throughput, and parallel implementation will have a better performance. Tree topology can do fast check nodes update in (Hu, 2001).

3.2 LDPC Decoding in GPRAM System

3.2.1 Overview a Simple GPRAM Prototype

The prototype of a simple GPRAM system shows in the following block diagram. The system contains some sensor inputs (s_1, s_2), input/output actions (a_1, a_2) and the output action a_3. The result of action a3 could be achieved from using LDPC coding theory instead of the traditional logic design in a machine. Usually, in a conventional system design, the goal is to achieve the optimal solution. For example, given all inputs s_1, s_2, a_1, a_2, we can get the result of output of a_3. However, in GPRAM, a_1, a_2 are no long be the input only, they could also be outputs of a machine. This somehow self-explains the differences between the conventional machine and the GRPAM.

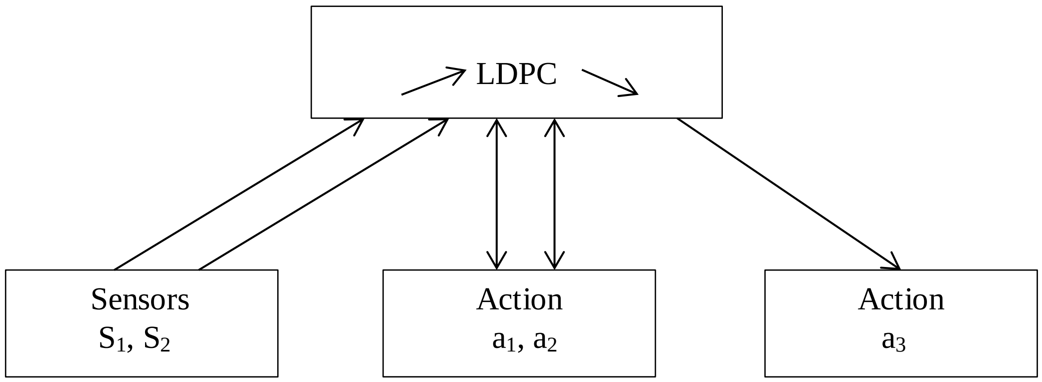


Figure 15 Prototype of GPRAM

3.2.2 Decoding process in GPRAM

In (Li, Dai, Schultz, & Wei, 2013), a simple example shows the how GPRAM system work using (7, 4) Hamming code. In the following figure, v_1-v_7 are variable nodes, C_1-C_3 are check nodes. The links represent information exchange in each iteration. Sum-product operation is used to update messages between check nodes and variable nodes. As introduced in the block diagram above, sensors (s_1, s_2) and actions (a_1, a_2, a_3) are desired to connect with variables nodes. The initial value of variable node j should be the log-likelihood ratio

$$u_j^l = \frac{2y_j^l}{\sigma_a^2} \tag{23}$$

in which, *l* denotes the number of iteration, in this example, *l*=30, j denotes the variable node, y should be the signal plus noise, as it is the initial value, y is the noise with zero mean, variance one. For iterative decoding, we have:

$$v_{ji}^l = \begin{cases} u_j^l & l=0 \\ u_j^l + \sum_{\substack{k=1 \\ k\neq j}}^{d_y} \frac{v_{ik}^l}{2} & l>0 \end{cases} \tag{24}$$

In the above equation, v_{ji} denotes the LLR which transfer from variable node to check node, u_{ij} denotes the LLR transfer from check node to variable node. After each iteration, a hard decision was made:

$$x_j^l = \begin{cases} 1, & v_j^l > 0 \\ 0, & v_j^l < 0 \end{cases} \tag{25}$$

3.2.3 Connection methods in GPRAM

After knowing the basic method of decoding method, we need to focus on how GPRAM sets connections.

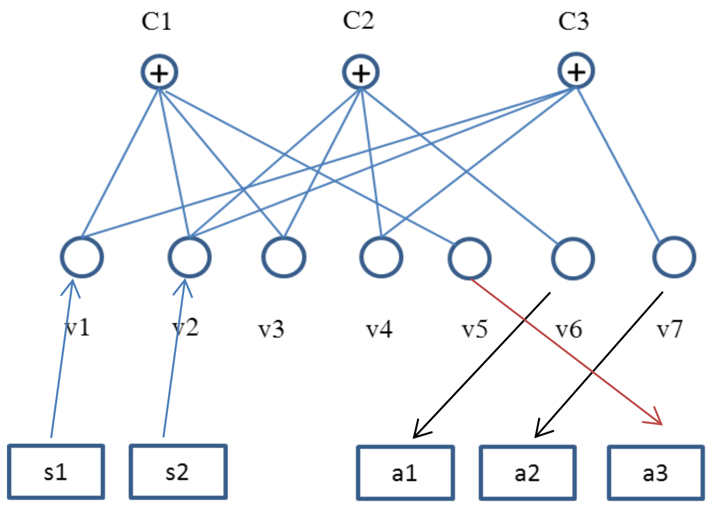


Figure 16 GPRAM connecting procedure

Step 1: For s_1=1 and s_2=1, no connection has been set up. According to equation 20, for j=1 to 7, x_1^1 to x_7^1 were known. After thirty iterations, a table contains 30 rows, and each row are the values from x_1 to x_7. Then we need to process numbers in the table, count the number of "1" in each column, and pick two columns which have largest number of "1", these two columns are connected with s_1 and s_2 separately.

Step 2: Make connections for a_1 and a_2. First, all variable nodes are set to zero and keep the connection set up in Step one. Run thirty iterations again as in Step one, as s1 and s2 are fixed, we counted other five uncounted columns. A1 is connected to the column which has the most "0", and a2 is connected to the column which has the most "1".

Step 3: Repeat setting all variable nodes to zero and keep existed connections. In order to let a3 connected, we introduce two tasks together this time. Each task runs 10 iterations, after thirty iterations, a3 is connected to largest number equal to x_j^l.

Thus, all connections are completed. There is one more thing needs to mention, as we introduced above, decoding method in GPRAM is different from that in telecommunication. In telecommunication, noises should be the same in the whole decoding procedure. However, in GPRAM, noises could change randomly in the end of each iteration, which is due to the fact that the signal is continuously coming into system.

3.3 Problems in Existing Implementation

The goal is to design an animal-like robot to let people have an intuitive view of what GPRAM is like. Thus, computer simulation is not enough for that. We need to build up the GPRAM machine. However, hardware implementation may different from the computer simulation, FPGA design needs a simply way to do the decoding procedure, it couldn't deal with real numbers which will add complexity to the hardware design. Thus, quantization is needed and look up table should be designed. Mr. Mcmullen did some decoding work in hardware design using FPGA.

Mr. Mcmullen derived sign-min algorithm as decoding method. The core operation is like the following:

$$\mathrm{LLR}(\mathrm{u}\otimes\mathrm{v}) \approx sign[\mathrm{LLR}(\mathrm{u})]\mathrm{sign}[\mathrm{LLR}(\mathrm{v})]\cdot\min[|\mathrm{LLR}(\mathrm{u})|,|\mathrm{LLR}(\mathrm{v})|] \qquad (26)$$

However, this method will lead to a 0.4 dB performance loss than sum-product algorithm in (Hu & Eleftheriou, 2001). The simulation result shows in the following figure.

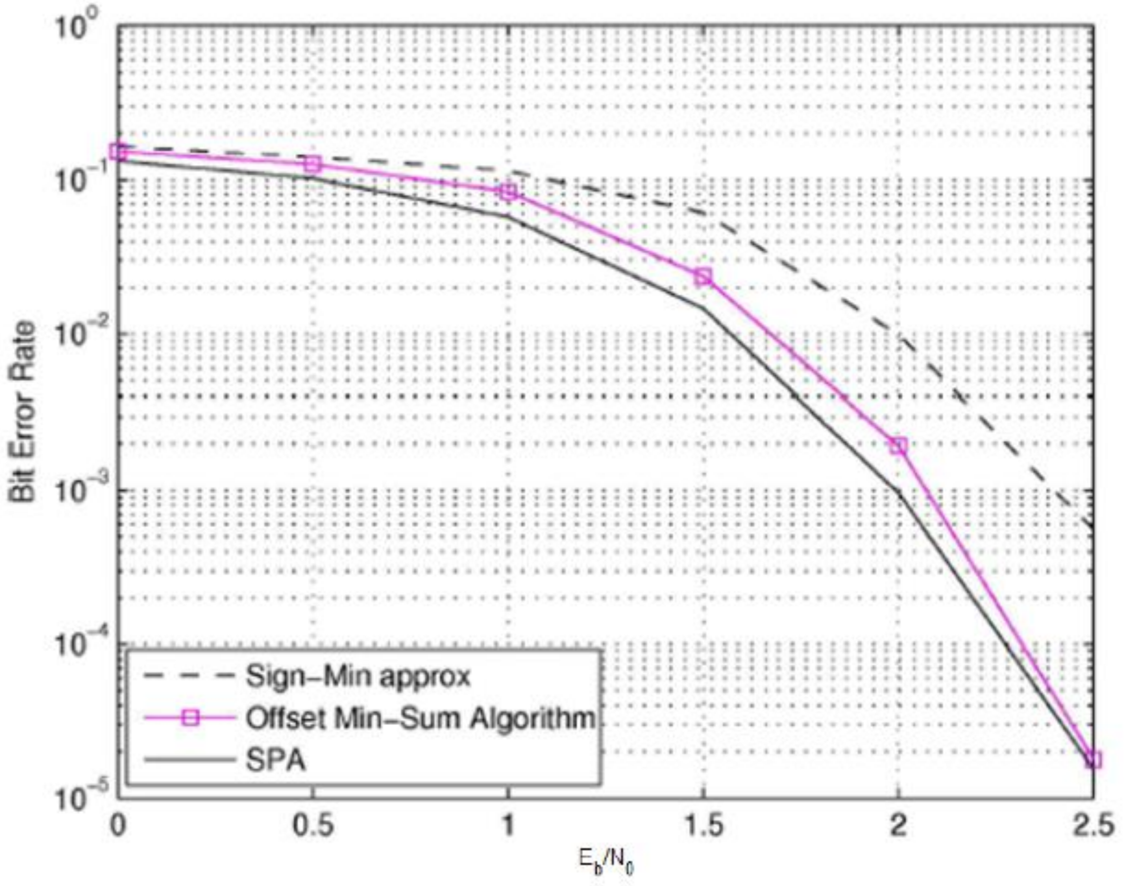


Figure 17 BER as a function of E_b/N_0 as a method of Sign-Min algorithm

lut input	lut output
0	101010
1	11110
10	10101
11	1110
100	1000
101	101
110	11
111	0
others	0

Figure 18 LUT 3-6

lut input		lut output		lut input		lut output	
0	000000	6	110	32	100000	0	000
1	000001	6	110	33	100001	0	000
2	000010	6	110	34	100010	0	000
3	000011	6	110	35	100011	0	000
4	000100	5	101	36	100100	0	000
5	000101	5	101	37	100101	0	000
6	000110	4	100	38	100110	0	000
7	000111	4	100	39	100111	0	000
8	001000	4	100	40	101000	0	000
9	001001	3	011	41	101001	0	000
10	001010	3	011	42	101010	0	000
11	001011	3	011	43	101011	0	000
12	001100	3	011	44	101100	0	000
13	001101	3	011	45	101101	0	000
14	001110	3	011	46	101110	0	000
15	001111	2	010	47	101111	0	000
16	010000	2	010	48	110000	0	000
17	010001	2	010	49	110001	0	000
18	010010	2	010	50	110010	0	000
19	010011	2	010	51	110011	0	000
20	010100	2	010	52	110100	0	000
21	010101	2	010	53	110101	0	000
22	010110	1	001	54	110110	0	000
23	010111	1	001	55	110111	0	000
24	011000	1	001	56	111000	0	000
25	011001	1	001	57	111001	0	000
26	011010	1	001	58	111010	0	000
27	011011	1	001	59	111011	0	000
28	011100	1	001	60	111100	0	000
29	011101	1	001	61	111101	0	000
30	011110	1	001	62	111110	0	000
31	011111	0	000	63	111111	0	000

Figure 19 LUT 6-3

He made a LLR look up table in his design. Two units were built in his design, one is check node unit, and the other is variable node unit. In the check node unit, the input of doing LLR operation was quantized into 3 bits. When doing the LLR algorithm, to improve the performance, 3 bits were transformed into 6 bits. After finish all the sum operation, they were transformed back to 3 bits to save the memory. As for the noise, he randomly generated a set of Gaussian noise and use Matlab to generate a LUT. The result is not very well. So I did the simulation to check how bad the performance is. By adding two features together, performance loss is more than 3dB in the following figure. The decoding result is almost half right half wrong.

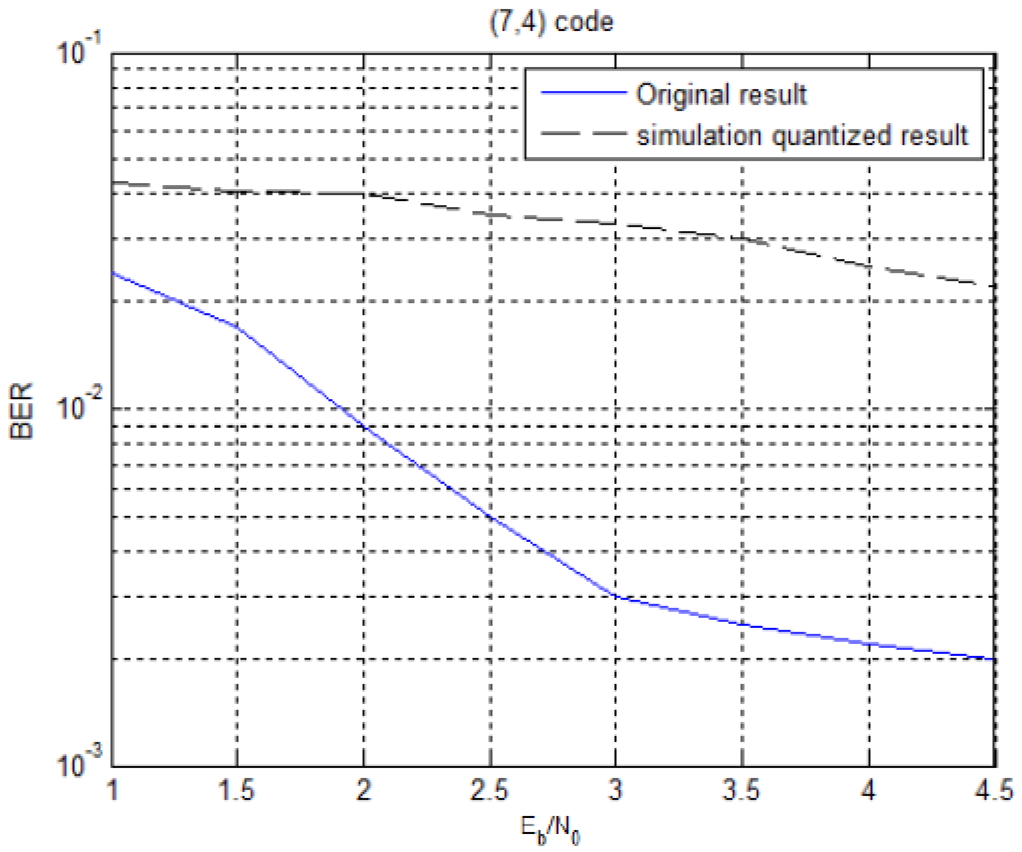


Figure 20 (7, 4) Code Simulation in Bryon's method

The simulation result is not good in the above figure. One reason is sign-min algorithm is approached in Bryon's decoding. 0.4dB loss occurred using sign-min algorithm compare to the Sum-product algorithm. Another reason is for the LUT table Bryon created. Errors occur due to the 3 bit to 6 bit and also 6 bit to 3 bit message exchange.

We couldn't bear that much loss in the design of GPRAM. In this circumstance, changing the quantization method is necessary. Also, the LUT is also needed to be change. With setting proper LUTs for check node update and also noise, we hope the performance within 0.2dB and even 0dB loss. This is my major work in the thesis.

In the following sections, quantization of LDPC decoding and noises will be introduced.

3.4 Quantization of LDPC decoding

According to the hardware implementation, the check-node updates part has the most computationally complexity due to the "tanh" rule. In this circumstance, we need to implement a quantized sum product algorithm to fit for the hardware implementation. Also, messages update between the check nodes and variable nodes are real numbers which need to be quantized to integers.

In Step 2(1): Check node update:

$$\Lambda_{m\rightarrow n}(C_i) = 2tanh^{-1} \prod_{n'\in N(m)\backslash n} tanh\left(\frac{1}{2}(\lambda_{n'\rightarrow m})\right) \qquad (27)$$

This equation needs to make a change.

The $\lambda_{n\to m}$ may be regard as two parts. First is the symbol part, which could be achieved by using the sign function. Second part is the absolute value of the message. The equation can be rewrite into the following:

$$\Lambda_{\mathrm{m}\to\mathrm{n}}(C_i) = 2\tan^{-1}\prod_{\mathrm{n}'\in\mathrm{N}(\mathrm{m})\backslash\mathrm{n}} \tanh\left(\frac{1}{2}\left\{\mathrm{sign}\left(\lambda_{\mathrm{n}'\to\mathrm{m}}(C_i)\right)\right\}\left|\lambda_{\mathrm{n}'\to\mathrm{m}}(C_i)\right|\right)$$

$$= \prod\nolimits_{\mathrm{n}'\in\mathrm{N}(\mathrm{m})\backslash\mathrm{n}}\left\{\mathrm{sign}\left(\lambda_{\mathrm{n}'\to\mathrm{m}}(C_i)\right)\right\}2\tan^{-1}\log^{-1}\left\{\log\left(\prod\nolimits_{\mathrm{n}'\in\mathrm{N}(\mathrm{m})\backslash\mathrm{n}}\tanh\left[\frac{\left|\lambda_{\mathrm{n}'\to\mathrm{m}}(C_i)\right|}{2}\right]\right)\right\}$$

$$= \prod\nolimits_{\mathrm{n}'\in\mathrm{N}(\mathrm{m})\backslash\mathrm{n}}\left\{\mathrm{sign}\left(\lambda_{\mathrm{n}'\to\mathrm{m}}(C_i)\right)\right\}2\tan^{-1}\log^{-1}\left\{\log\left(\prod\nolimits_{\mathrm{n}'\in\mathrm{N}(\mathrm{m})\backslash\mathrm{n}}\tanh\left[\frac{\left|\lambda_{\mathrm{n}'\to\mathrm{m}}(C_i)\right|}{2}\right]\right)\right\} \quad (28)$$

where, $\emptyset(x)$ is the quantized function.

$$\emptyset(x) = log\frac{e^x+1}{e^x-1} = \emptyset^{-1}(x) \quad (29)$$

Using Matlab, $\emptyset(x)$ was shown in the following figure.

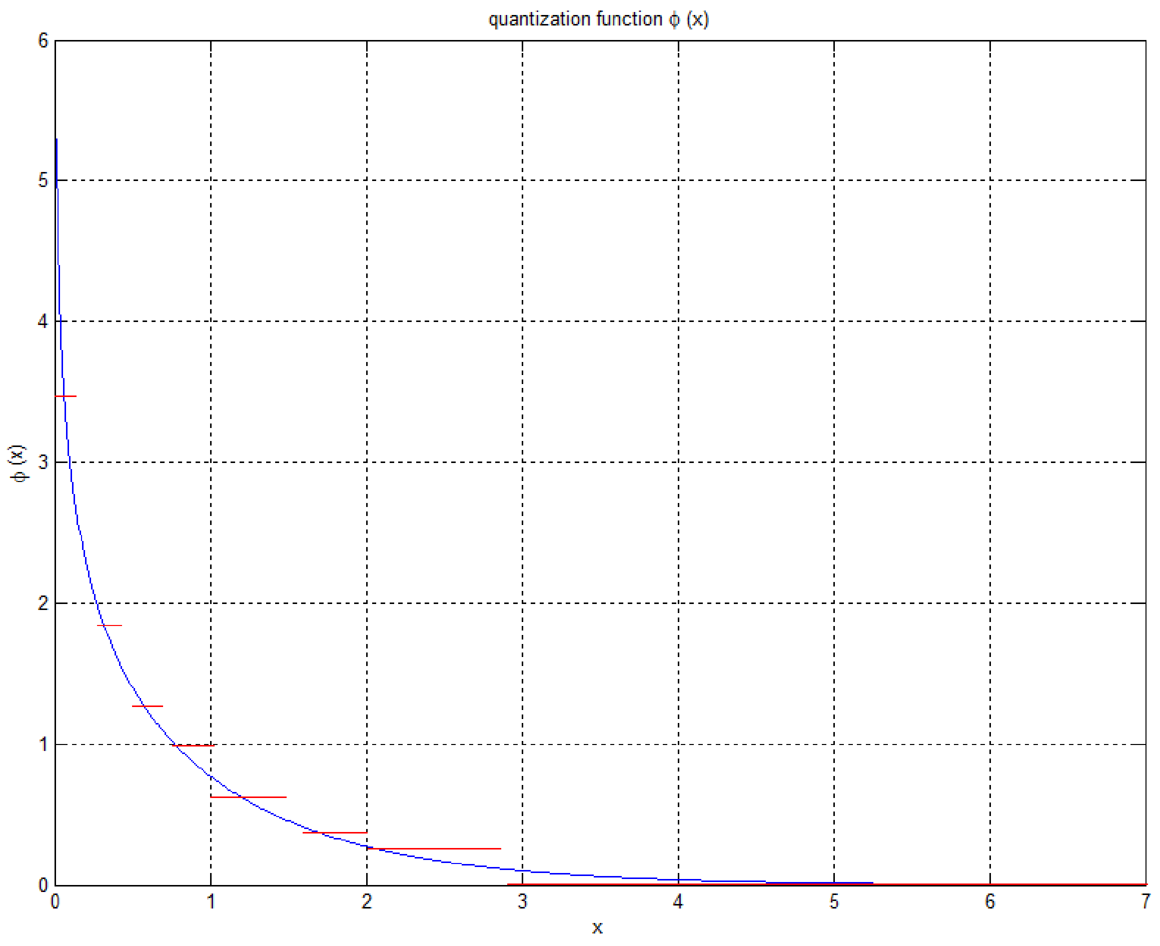


Figure 21 Quantization function

LLR	**Q-LLR**
<0.25	3.5
[0.25,0.5)	1.875
[0.5,0.75)	1.25
[0.75,1)	1
[1,1.5)	0.625
[1.5,2)	0.375
[2,2.875)	0.25
>=2.875	0

Figure 22 Quantization into eight levels

The quantized range was set as the above figure. All the LLR real number value was quantized into Q-LLR integer value. The LLR values have been divided into two parts, one bit is for sign operation, and other 3 bits are the absolute values. In the hardware implementation, the sign operation can be implemented as the XOR switch. The other 3 bits can use a look up table to map the value of ∅ (x). According to the equation 11, sum up all the values of ∅ (x), re-mapping to the LLR value.

Variable Node Update and the decision part should remain the same. After doing the quantized decoding algorithm, the Bit Error Rate (BER) performance has at most 0.3dB loss. Noise Quantization will help us bring back the loss in the GPRAM design.

Quantization level could be designed as you need. 4-bit quantization have better performance in figure 21, but only a little better than 3-bit quantization. If we take hardware limitation into consideration, it will need more memory to process the 4-bit information. Also, from figure 17, value in x axis changes little in range [3, 7]. So there is no need that we divide x-axis into so many levels. Results will not influence that much if we count 0 in that range. Performance of 3-bit is only 0.3 dB worse than the original performance. If we change to 2-bit quantization, performance is figure 21 is much worse than original one. Result is not acceptable in 2-bit quantization. Quantization in 3 bit is more desirable compare to 2-bit and 4-bit quantization. Thus, in my thesis, 3-bit quantization will be implemented.

LLR	**Q-LLR**
<0.5	2
[0.5,1)	1
[1,2)	0.5
>=2	0

Figure 23 Quantization in 4 levels

LLR	Q-LLR
<0.25	3.5
[0.25,0.5)	1.875
[0.5,0.75)	1.25
[0.75,1)	1
[1,1.25)	0.685
[1.25,1.5)	0.5
[1.5,1.75)	0.4
[1.75,2)	0.3
[2,2.25)	0.25
[2.25,2.5)	0.2
[2.5,2.75)	0.15
[2.75,3)	0.1
[3,3.25)	0.0825
[3.25,3.5)	0.0625
[3.5,3.75)	0.05
>=3.75	0

Figure 24 Quantization in 16 levels

-

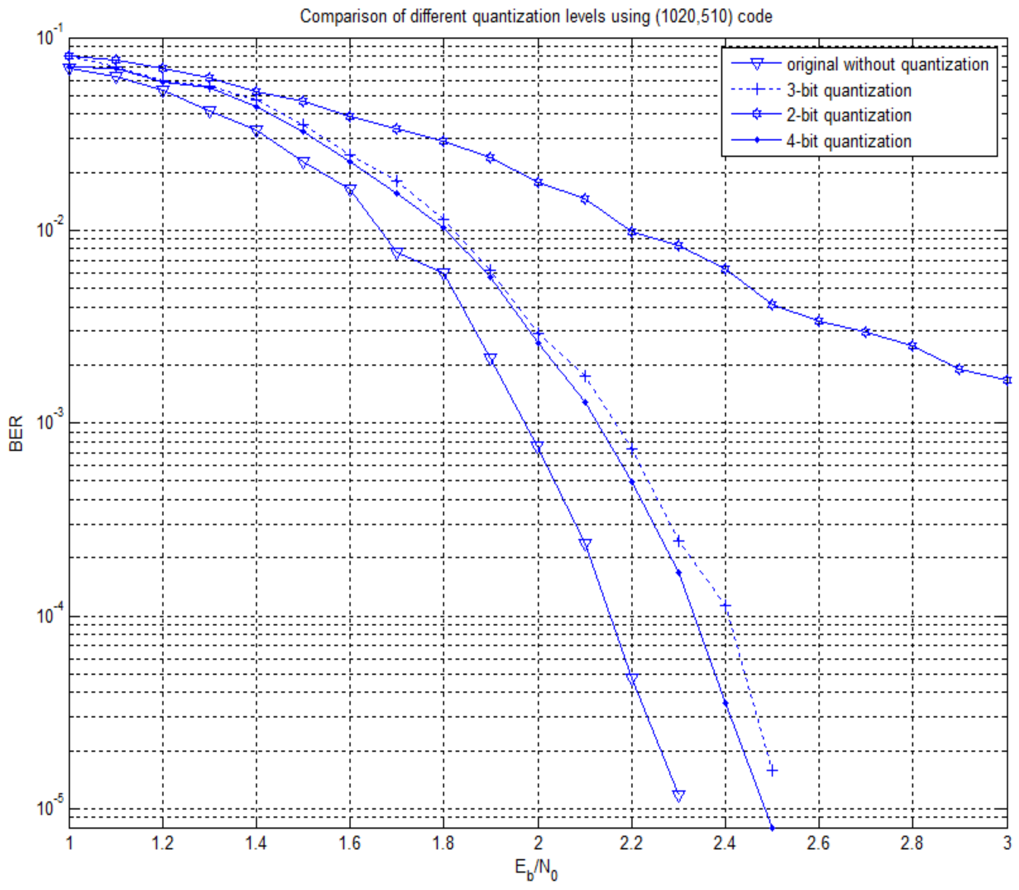


Figure 25 Comparison of different quantization levels

3.5 Noise implementation in LDPC decoding design

Computers can easily generate the Gaussian noise for us during the simulation. However, it is hard to do the same thing using FPGA. In this circumstance, a noise look-up table needs to be generated for FPGA. The noise has zero mean and unitary variance. 100 sets of quantized Gaussian noises were generated and each set contains 100 Gaussian noises. One set of quantized Gaussian noise was picked after simulation. Which is gaussian[100]={-2,0,-1,0,-1,0,-1,-1,0,1,-2,2,-1,1,1,1,-1,0,0,-1,0,0,0,0,3,0,1,0,-1,0,1,-1,-2,0,0,0,0,1,1,-1,-2,1,0,-1,1,0,0,0,1,2,0,-2,1,-1,1,-1,1,-1,-1,1,-1,1,0,0,0,2,0,0,-1,-1,-1,2,0,-1,0,1,-1,1,-1,0,-2,1,1,-1,0,0,1,0,1,0,-1,0,0,1,0,-1,0,1,0,-1}

The above set of Gaussian noise was chosen because this set fit best for the original BER performance without doing the quantization. The other sets could be found in the appendix part.

The following figure shows the pdf of the Gaussian noise.

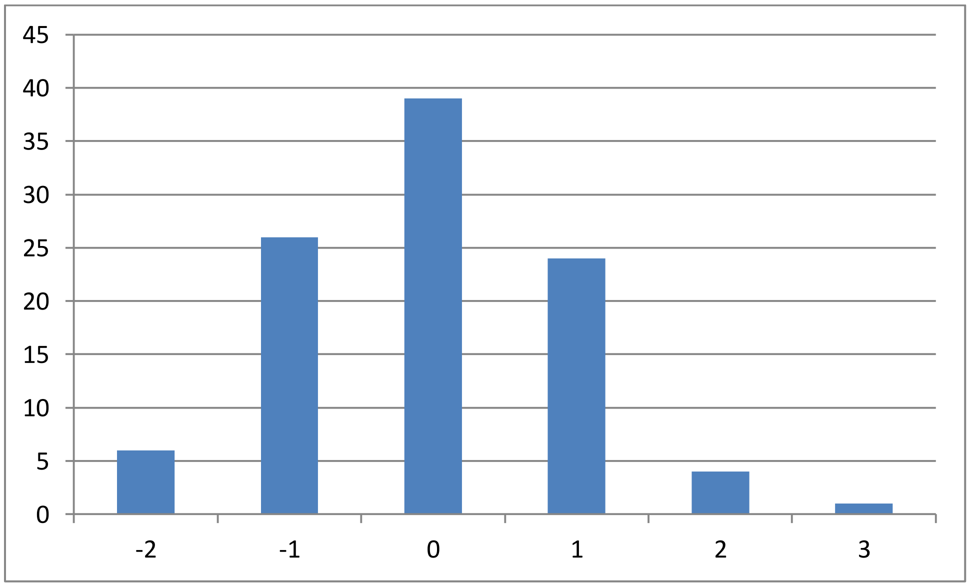


Figure 26 PDF of Gaussian Noise set which used in the simulation

3.6 Simulation Results

In the thesis, LDPC codes with length (7, 4), (40, 20) (510,255), (1020, 510) were used in the simulation. Each length code was simulated 3 times, with 10 iterations each time. The BER performance without quantized, with quantized and also the quantized noise added to the simulation.

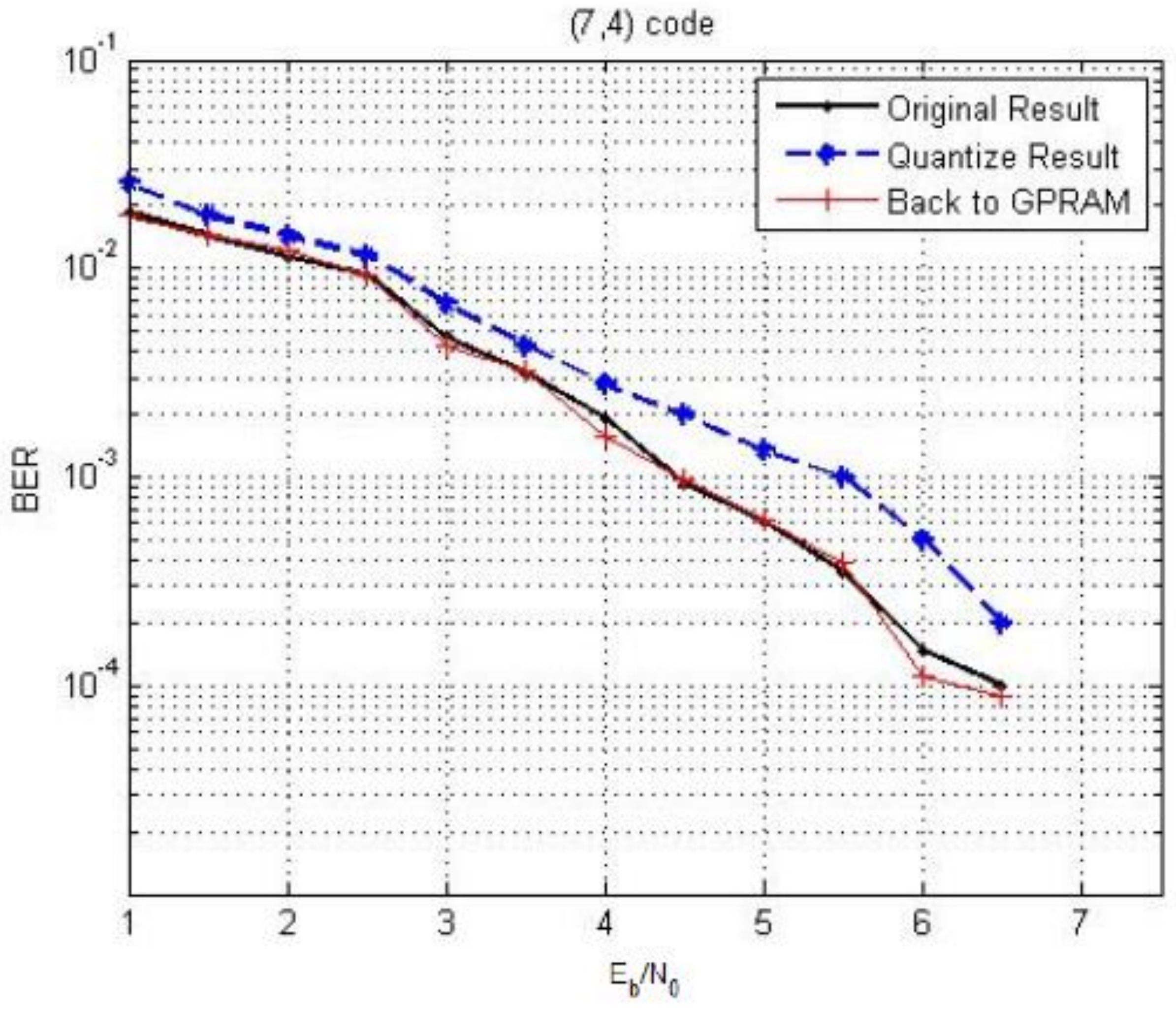


Figure 27 Simulation result of (7, 4) code

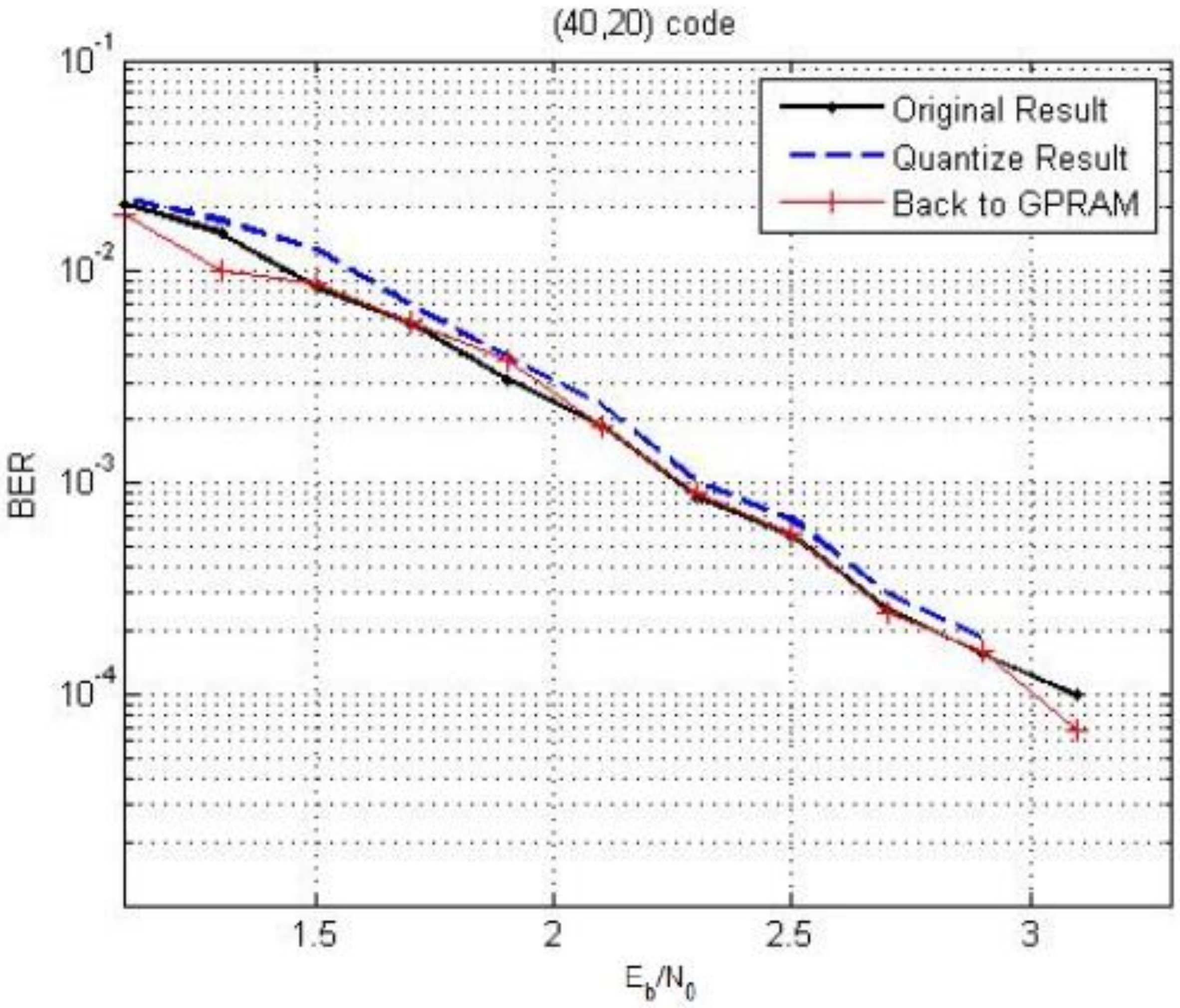


Figure 28 Simulation result of (40, 20) code

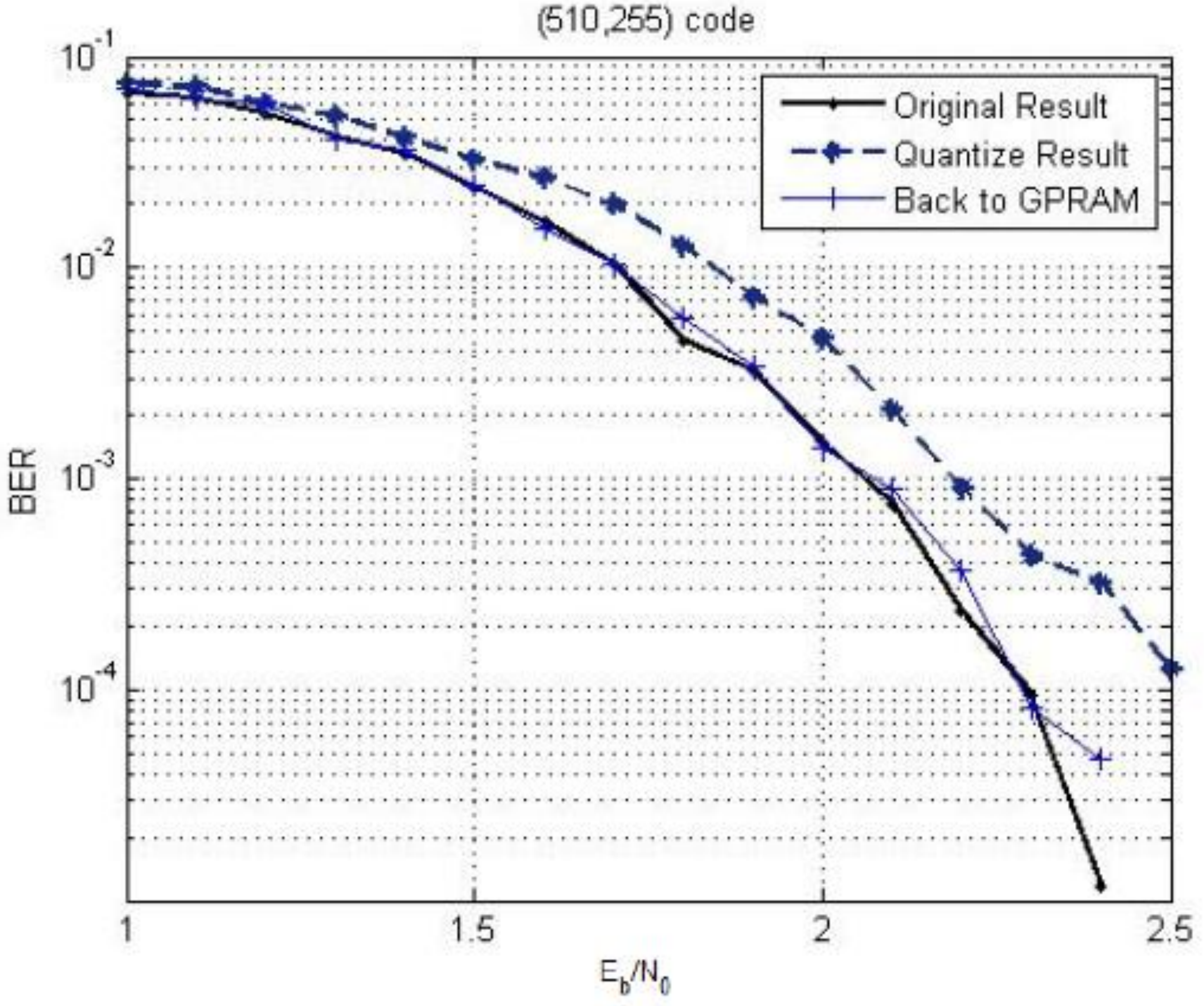


Figure 29 Simulation result of (510, 255) code

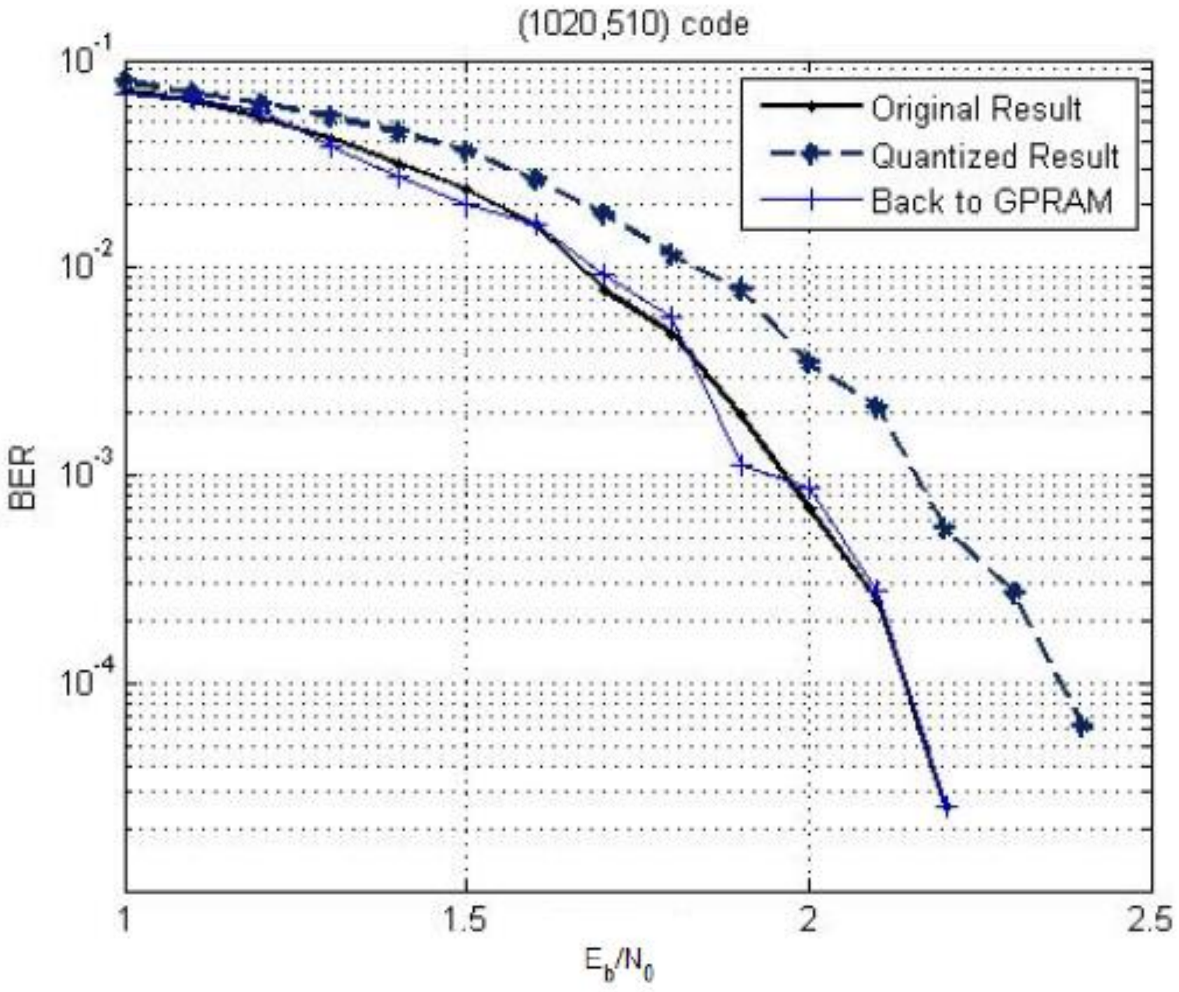


Figure 30 Simulation result of (1024, 512) code

3.7 Conclusions

From the above figures in the simulation part, the longer the code length, the BER performance is nearer to the Shannon limit. The (1020, 510) code, when the SNR is 2.5dB, it may reach to the 10^{-5} Shannon Bound. However, the (7, 4) code, even for the SNR is 7dB; it can only falls into 10^{-4}. The quantized decoding algorithm will bring 0.3dB loss at most. If we add one quantization bit into the decoding, for example, 4 bit, the result will be better. However, it will be more memory in the hardware implementation. The result is only 0.3dB, we can bare that performance. In the GPRAM design, we need to bring back this loss by adding the quantized noise. The noise set was picked which will recover the loss cost by the decoding quantization. As we have 100 noise sets, we simulated and picked one set which will have the same result on all codes no matter what length it is. This set of noise will be used as the noise look-up table in the GPRAM. As said above, the iterative decoding process in the GPRAM is different from that in telecommunication. The noise in GPRAM could change randomly in the end of each iteration. Thus, this look-up table simplifies the noise reading in the design of GPRAM. What’s more, this look-up table also brings back the performance of the iterative decoding. In conclusion, by doing the two quantized in the iterative decoding step, the problems met in FPGA design of GPRAM have been solved.

CHAPTER FOUR: IMAGING MANIPULATION FOR GPRAM

4.1 Differences between camera images and bio-visual images at retinal

In ancient time, people have started to try to explain how human visual worked. Date back to the time of Aristotle, his thoughts of human vision was that between the observer and the object, there should exits something allowing the object to be seen. He called this the media, which is known as the air today. However, during the Middle Ages time, Scholars hold different ideas of Aristotle's theory, which are the observer’s eyes sent out emission to the object and the emissions made the object being able to be observed. Although these theories may seem illogical today, the theories were based on the observation of the scholars not on today's scientific experiments. Due to the theories generated by thousands of scholars our understanding of human vision has come to a breakthrough.

Hyper acuity is important in human visual system. For example, the design of the telescope helps us to expand our scope about the universe. More and more advanced tools were developed for the image processing of human. Hyper acuity will take the resolution limitation into consideration. More and more devices with hyper acuity have been achieved. No matter how many advanced tools were designed to help human improve the resolution of the image, human eyes are the still the final step. Thus, human eyes are the limitations of human visual capability.

Take a look around the environment; you will see images and colors in huge variety. All the images can be update with no break at all, and also can judge the accurate position. These are the beauty of the human vision. Although the images we saw are seamless, they are updated continuously by the help of our brain. With the function of the brain, we can see the image more clearly and with more details. Without the help of our brain, the image we saw at retinal will be blurred. Due to the movement of the eyeball, the images are not stable. In this paper, we simulated the blurred images at the retinal and made a video which showed the exact images at the retinal without the function of the brain.

In (Wei, Levi, Li, & Klein, 2007), human visual system and hyper acuity device were discussed. Device with high visual acuity is in great need in many fields. For example, in the design of a space shuttle, a metal-polishing machine needs to examine the surface of the space shuttle, to make sure it is perfect flat, and flawless. We focus on studying that kind of machine and assume the vision part of that machine is similar to human eyes. Also, the following things were taken into consideration: point spread function, Poisson noise, eye movements with conditional of stable head.

Let’s take a short review of how human eyes work first. When light entering the eye, it passes through the first transparent layer of the eye, called cornea. After reaching cornea, it will immediately reach the eye lens. Continuous refraction of the light at the surfaces of the cornea and the eye lens together serve to focus an image on the curved rear surface of the eyeball known as the retina. Retina is the innermost layer, it is like a "screen", on which image is formed by the lens when properly focused. About 100 million photo receptors are located on the retina. We assume the receptors are arranged in hexagonal array, because, hexagonal array has the tightest character, which shows in the following figure. The diameter of receptor is the cone size which is 0.6 min of arc. The total size of receptor depends on the stimuli, range from 400 to 10000. The stimuli are two dots with distance of δ. Detector has functions of processing visual signal from receptor and also making a decision.

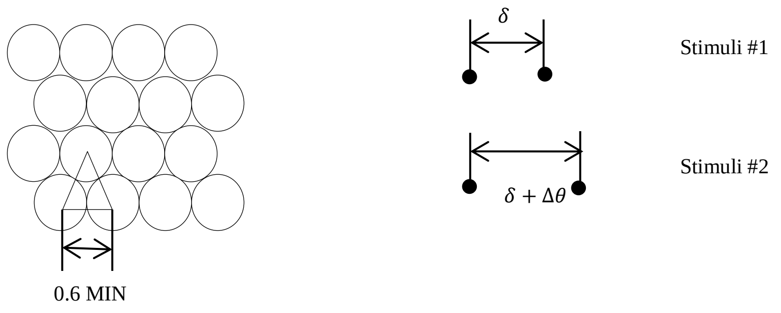


Figure 31 Receptors structure and stimuli

In (Wei, Levi, Li, & Klein, 2007), how to select stimuli is also described. We have two stimuli, shows in the above figure. We randomly select one pair of two dots and use it into the device. As stimuli presented, not all decisions are correct, about 75% decision will be made. Under the above conditional, the image will be blurred, which is the same conditional of human eyes. However, hyper acuity is in great demand. If a system with CCD (charge-coupled device) camera was used, images will be much clearer. The image is clear with no doubt, but processing the image will be slow. What if we deliver a machine, which can work well with low resolution image? This idea also brings up Dr. Wei's idea of the GPRAM machine.

Further, an ideal detector for moving stimuli is conducted. Two synchrony mechanisms have been taken into consideration. First method is by using SDE (stimulus defined exactly) detector array. Each SDE detector will record the moving distance, direction, and speed. Second method is snap shooting the all uncertain parameters for an image. After all uncertain parameters are measured and fixed, the problem will be similar as design a fixed detectors.

In my thesis, I will focus on the detector design with fixed stimuli. All parameters are the same as introduced in (Wei, Levi, Li, & Klein, 2007).

4.2 Key steps to generate coarse visual images from camera images

4.2.1 Point Spread Function

The point spread function is a calculated image of optical system to a perfect point source of light. PSF is the response of an imaging system to the point input. It's analogous to the impulse response. The flowing figure is the typical point-source situation in an imaging system.

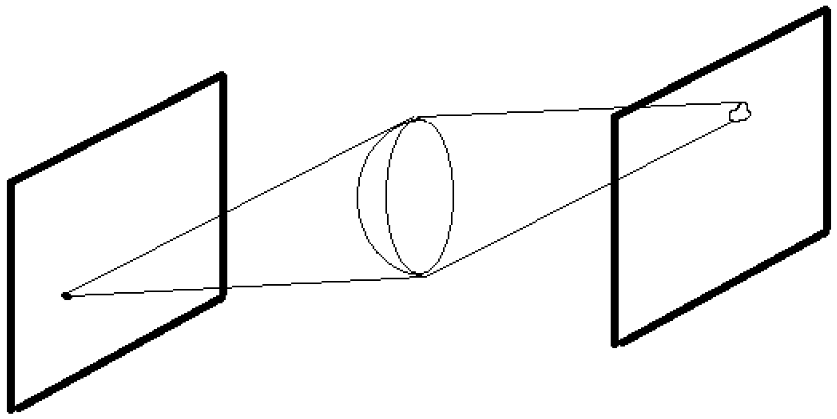


Figure 32 Typical point source situation in image system

Consider a point of light, if we have perfect visual system, the image of this light point on the retina will be exactly the same as the original point of light. It would be shown in the figure with green lines. However, the eye’s optics is not perfect, so the light point is spread on the retina as shown in the following red curve, which is called the point spread function.

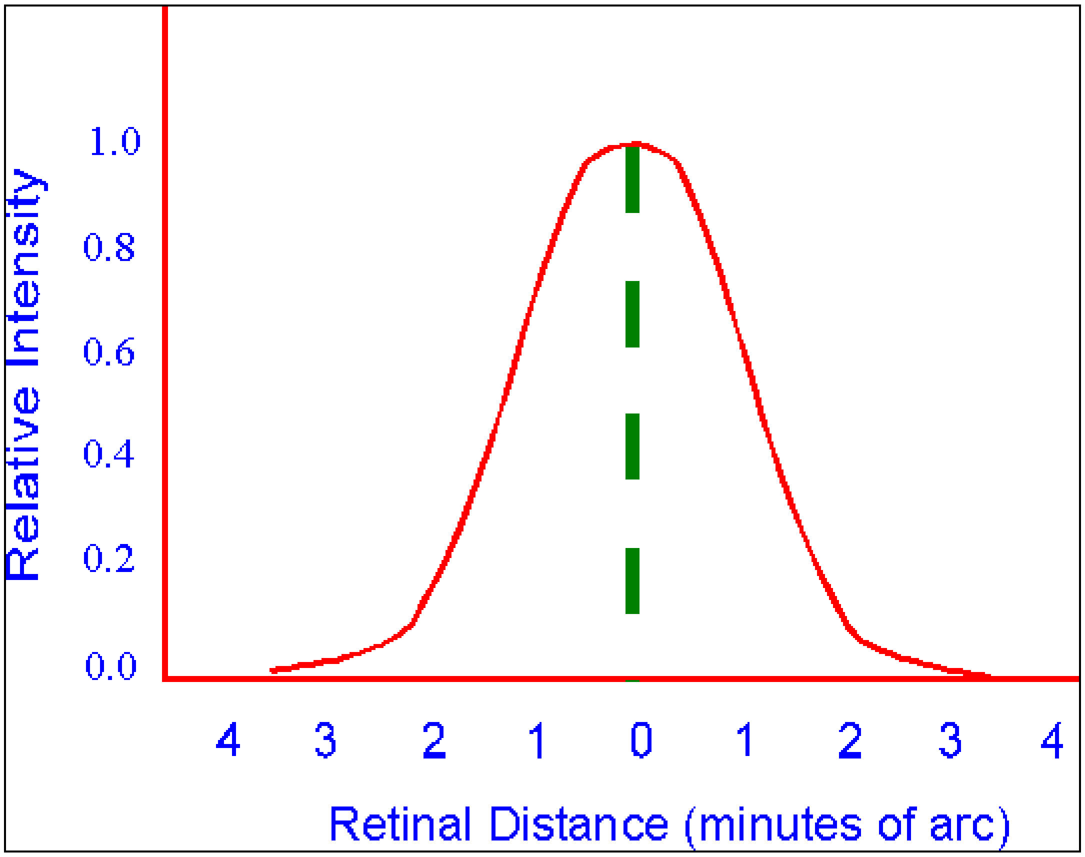


Figure 33 Point Spread Function

If we have two objects really close to each other, after the convolution with PSF, they might be smeared together and look like one object on the screen.

The output image can be achieved by the two dimensional convolution of the ideal image with point spread function. In this case, the point spread function is described as the sum of two 2-D Gaussian functions in (Geisler, 1984) (Geisler & Davilla, 1985)

$$\mathrm{h}(t) = \frac{a_1}{2\pi a_3} exp\left[\frac{-0.5t^2}{a_3^2}\right] + \frac{a_2}{2\pi a_4} exp\left[\frac{-0.5t^2}{a_4^2}\right] \quad (30)$$

In the above equation, a_1, a_2, a_3, a_4 are coefficients are a_1=0.417, a_2=0.583, a_3=0.443*scale (arc minutes)=26.58*S_{PSF} (arc seconds), and a_4=2.04* S_{PSF} (arc minutes)=122.4 *

S_{PSF} (arc seconds). S_{PSF} is scale which we can change the distribution of point spread function. S_{PSF} was scaled in range from 0 to 9. For S_{PSF} equals to 0, there is no point spread added into the image. The larger the value of S_{PSF}, the wider it spreads, the more blur the image will be. In images, we use PSF0, PSF1, PSF2 to denote no point spread case, point spread with S_{PSF}=1, 2 respectively.

In the daytime, the pupil gets narrower than in the night. The diameter of human pupil has the range from 3.0mm to 5.0 mm. In this thesis, 3.0 mm pupil was assumed. These parameters set in this thesis followed by (Wei, Levi, Li, & Klein, 2007). The diameter of the receptor is 0.6 arcmin, which is the space between cones and square shape is assumed here. The unit arcmin denotes one minute of arc, which means 1/60 of one degree. The unit arcsec will also be used in the thesis, which is 1/60 of one arcmin. In the thesis, chromatically broadband stimuli will only stimulate the middle and long wavelength cones equally.

4.2.2 Poisson distribution

The image we saw at the retina should be blurred, and noise should be added to the image, which is Poisson distribution. In information theory, the Poisson distribution shows the probability of a number of events happens in a fixed interval of time. In our situation, the Poisson distribution was expressed as the number of photons absorbed in the receptor during a fixed period of time.

$$\mathrm{p}(R = r) = \frac{F^r e^{-F}}{r!} \qquad (31)$$

In the above equation, r belongs to nature numbers, and F denotes the mean number of quanta absorbed in the receptor (Wei, Levi, Li, & Klein, 2007). The value of F (Geisler, 1984) dependent on optical factors and also point spread function in equation (27).

$$\mathrm{F} = \mathrm{ADST}E_{555}347.8l(t) * h(t) \qquad (32)$$

where A denotes the cross-sectional area of the receptor, the value is A=0.28 min^2. D is the duration of the stimulus, D=0.2 sec. S is the area of 2 mm pupil, which is S=3.1416 mm^2. T is the transmittance of the ocular media, T=0.68. E_{555} denotes the quantum efficiency of the photoreceptors at 555nm, E_{555}=0.5.It is assumed that half of the quanta are effectively absorbed whether they fall in a middle or long wavelength cone. l (t) is the luminance distribution of the stimulus in candelas per square meter. h (t) denotes the point spread function in equation (27). In images, we use PN0 and PN1 to denote without Poison noise and with Poison noise respectively.

4.2.3 Simulation Eye Movements

Eye movements are disordered and the eyeballs are drifting all the time. Even when we fix the head to make it not move and let the eyes gazing one object, the eyeballs are continuing moving. When walking, running, sleeping, the drift rates are different. Even two people walking with the same pace, the eye drift rate may be different. This fixation eye movements studied in this thesis gives us a better understanding of how human brain helps us to see the objects.

Quintessentially, the drift rate is 2-5 Hz, and peak to peak amplitude is less than 5 arcmin (Wei, Levi, Li, & Klein, 2007). The horizontal direction drifts are independent with the vertical direction drifts. The drifts between the two eyes are also uncorrelated. In this thesis, the drift-like eye movement could be model as the Gaussian random function, which shows in the following:

$$\mathrm{l}(t) = h(t) \otimes \phi_x(t) \quad (33)$$

$$\mathrm{r}(t) = h(t) \otimes \phi_y(t) \quad (34)$$

In the above equation, h (t) is the impulse response which can be described as follows:

$$\mathrm{h}(t) = \left(\frac{2400}{\sqrt{2\pi\sigma_i^2}} exp\left(-\frac{t^2}{2\sigma_i^2}\right) - 0.7 \right) cos\left(\frac{4\pi t}{1000}\right) \quad (35)$$

where σ_i^2=50000 ms^2.

In equation (30) & (31), $\otimes$ is the convolution operation and ϕ_x, ϕ_y denote the independent Gaussian random process with unitary variance and zero mean. In equation (33), we can change the scale of fixation movement by changing the variance of ϕ_x, ϕ_y, i.e., using two independent Gaussian process with variance S^2_{FM} and zero mean. The larger the variance, the more unstable the images. In images, we use FM0, FMx to denote no fixation movement, fixation movement with S_{FM}=x, respectively.

4.3 Flow chart of the Matlab programing

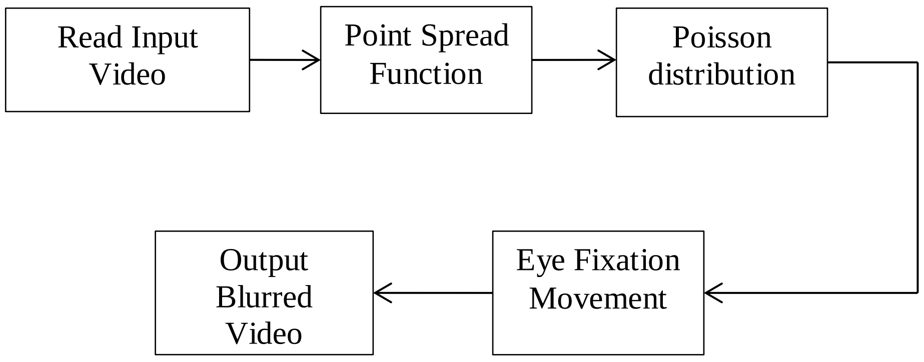


Figure 34 Flow chart of the system design

I used the Matlab to process the video which shows the result. I will capture the image of the video to show the results after doing the above three steps.

4.4 Simulation Results

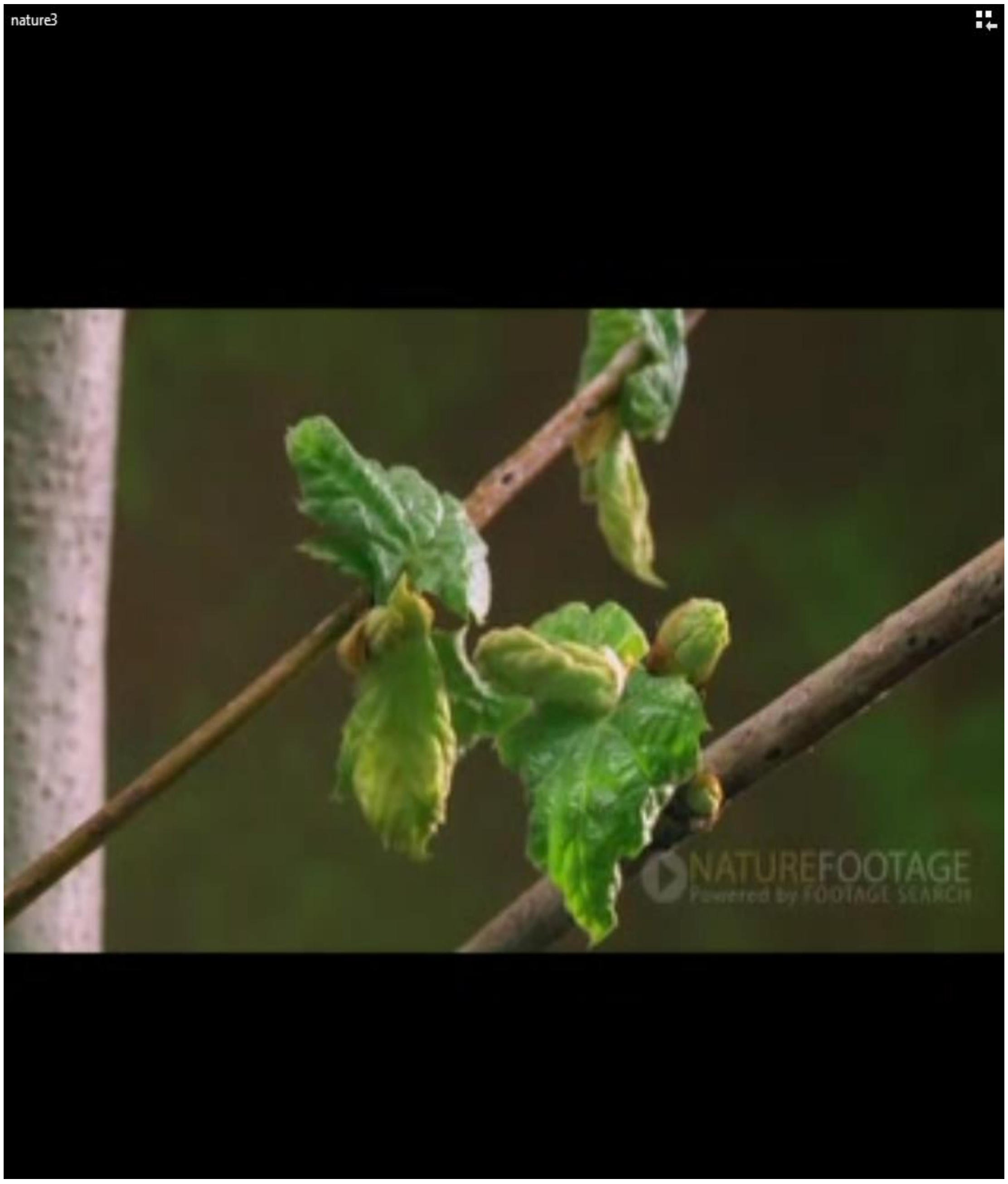


Figure 35 Original Image

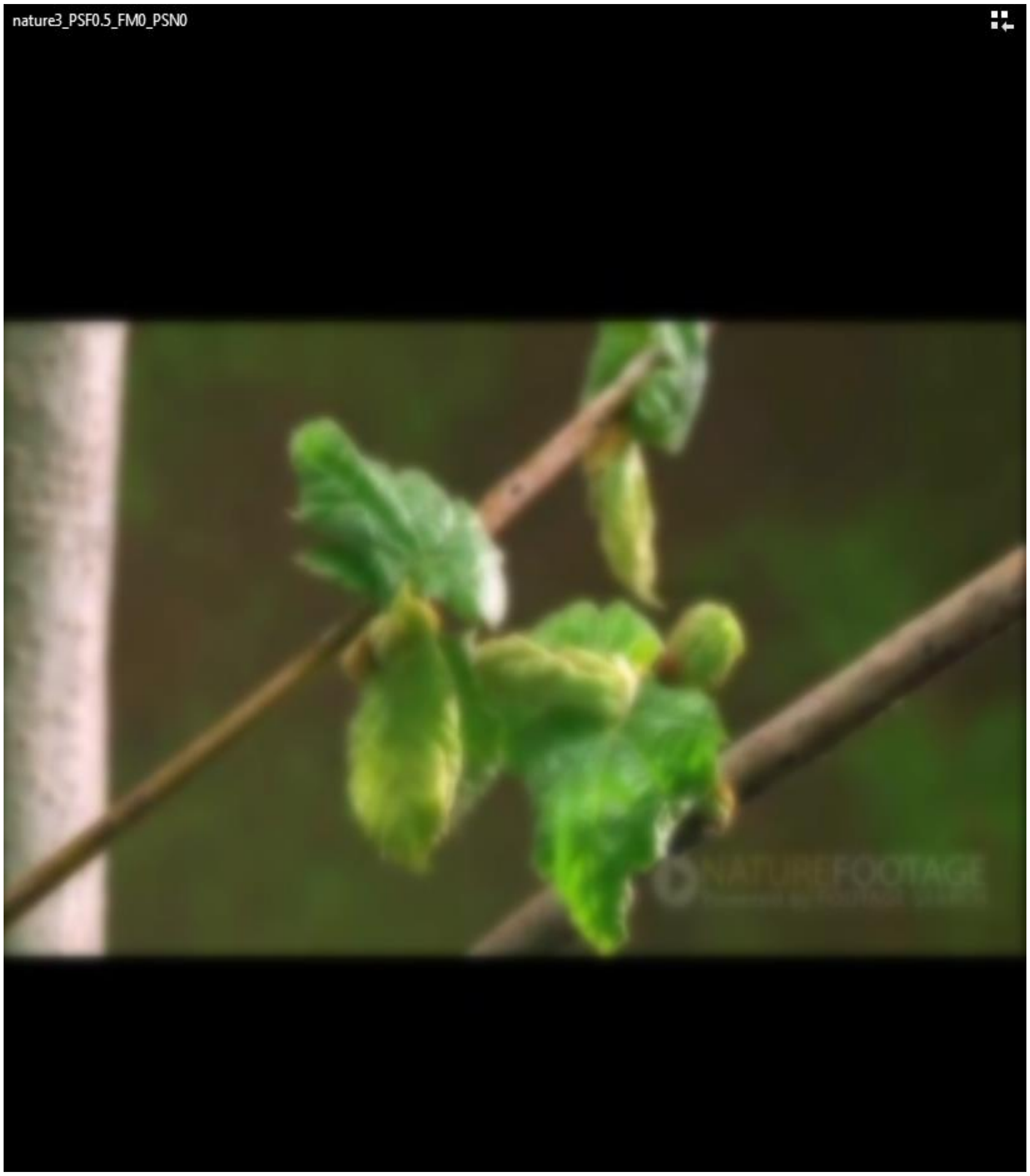


Figure 36 Image through Point Spread Function, with S_{PSF}=0.5

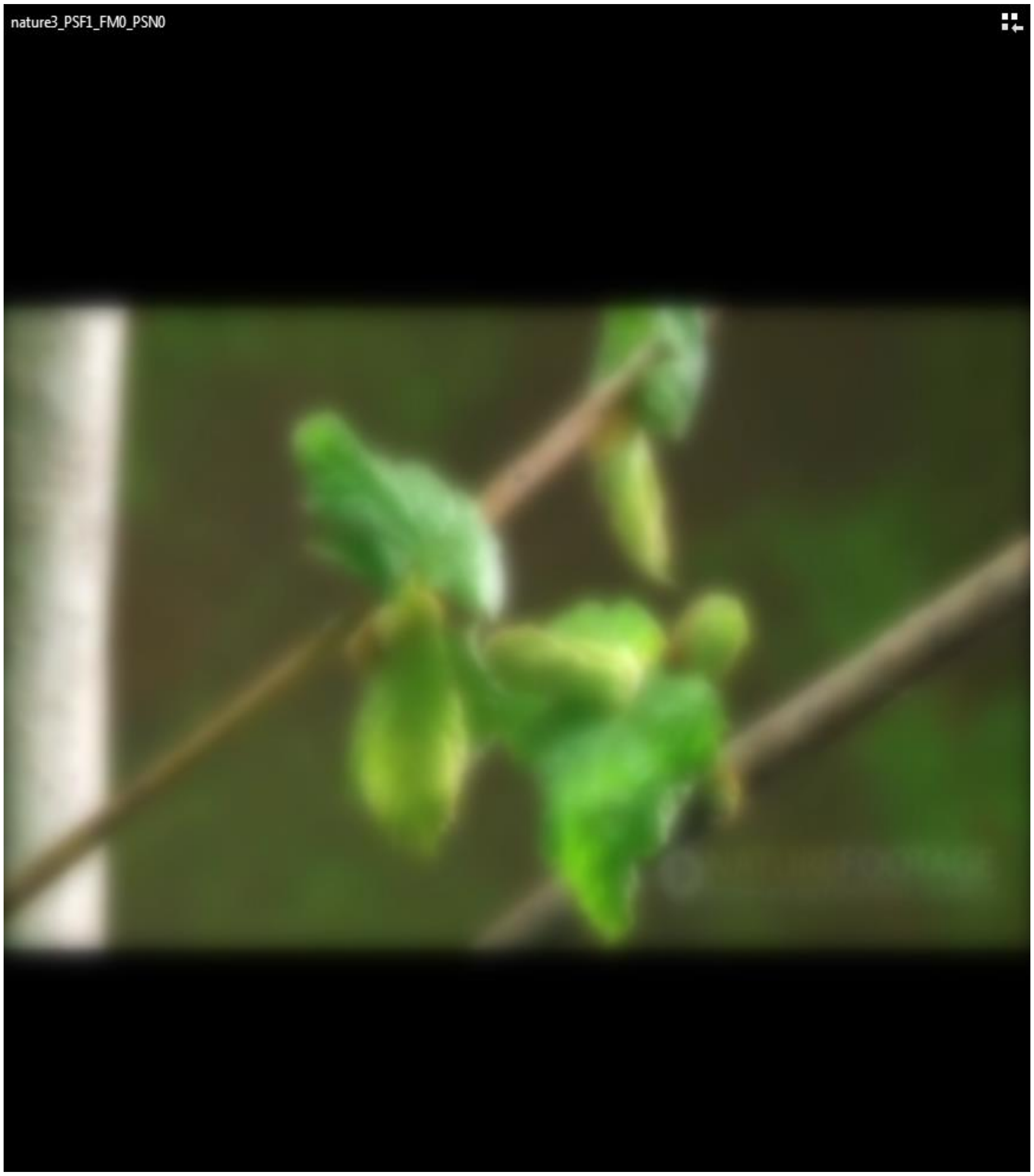


Figure 37 Image through Point Spread Function, with S_{PSF}=1

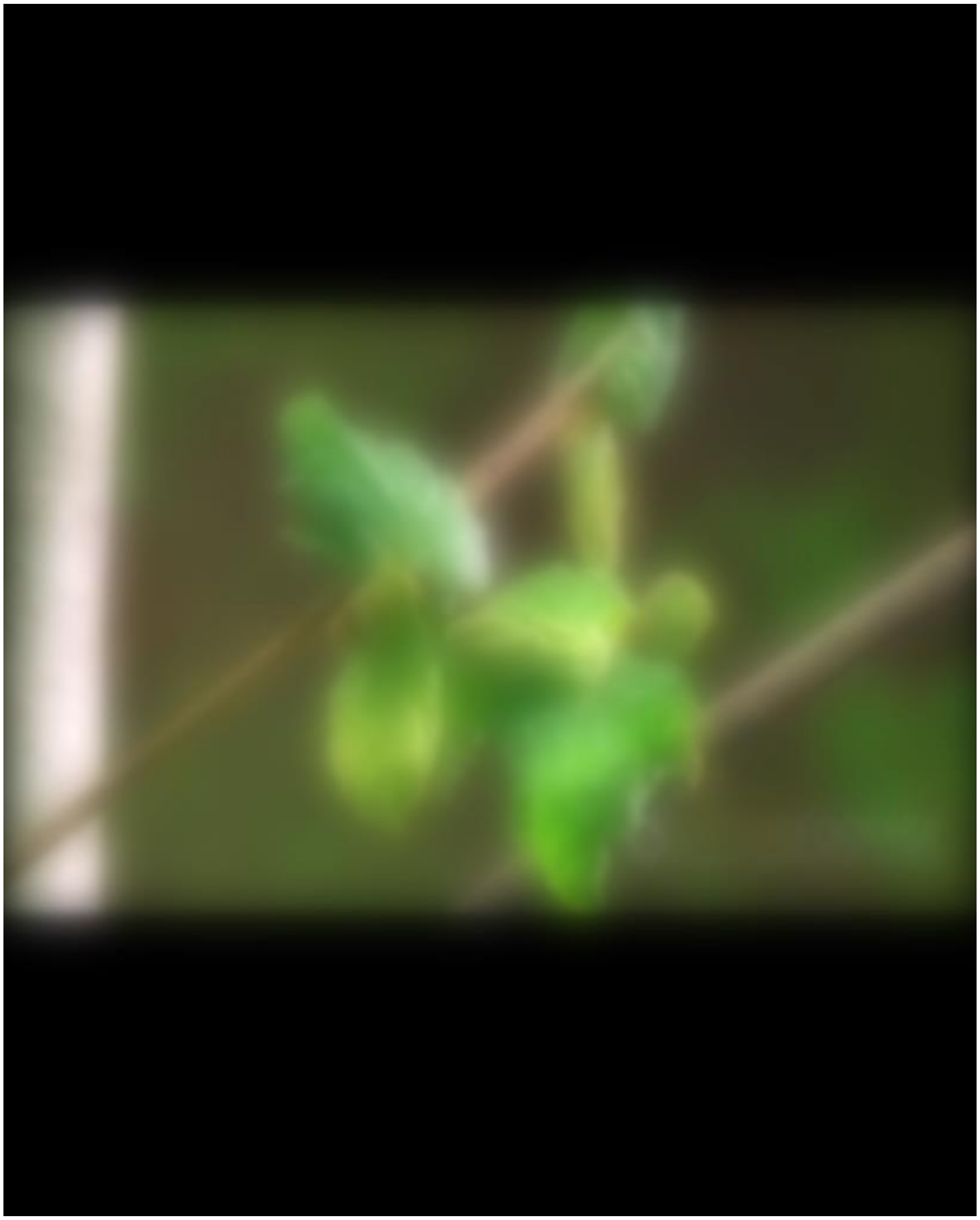


Figure 38 Image through Point Spread Function, with S_{PSF}=2

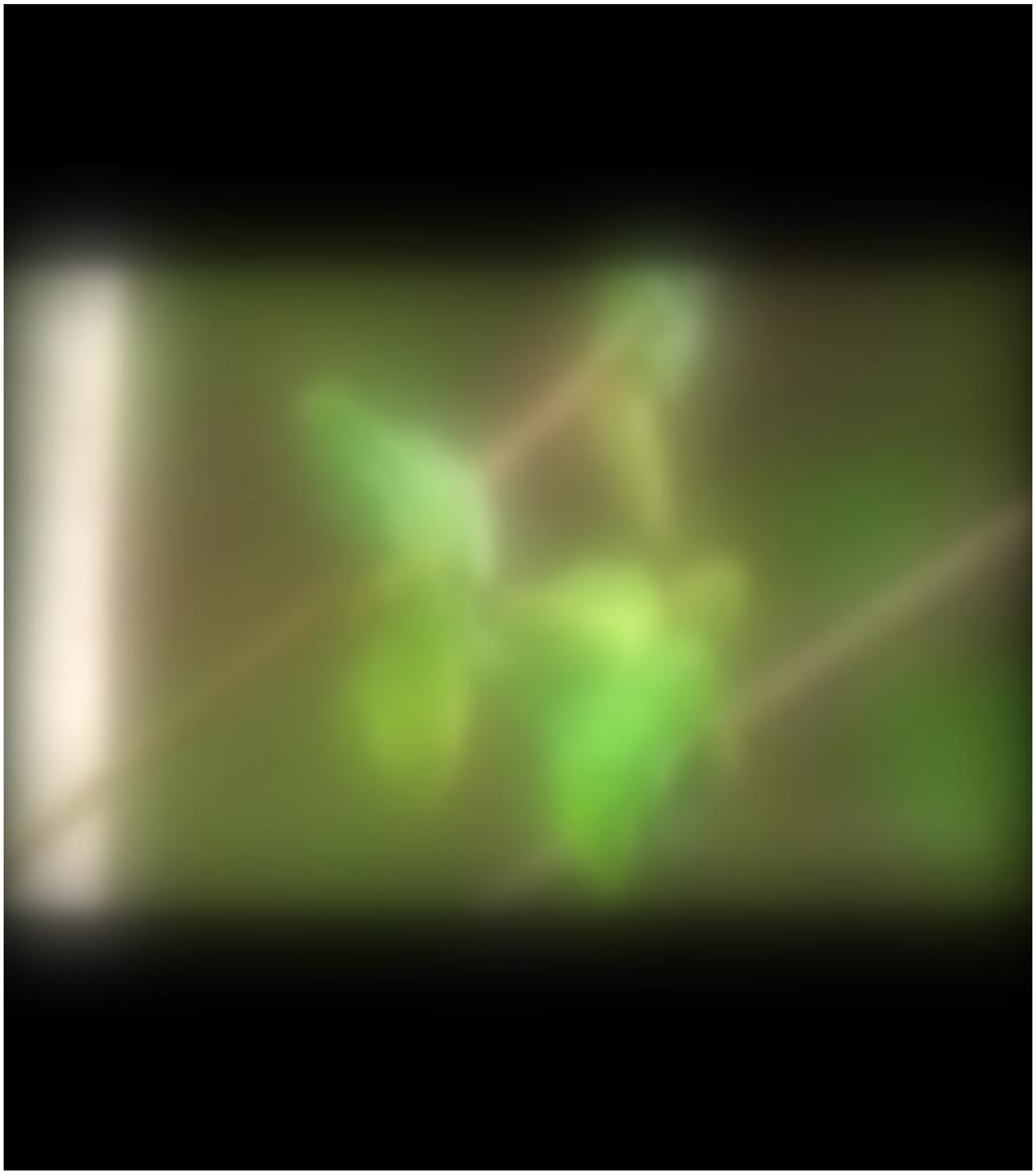
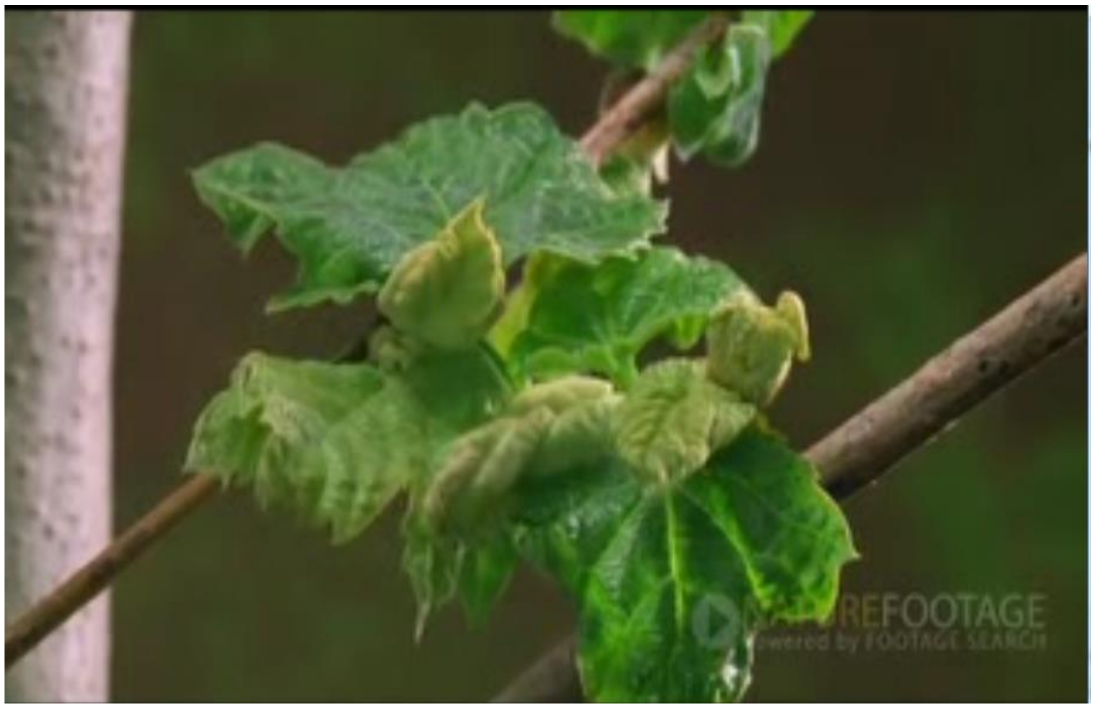
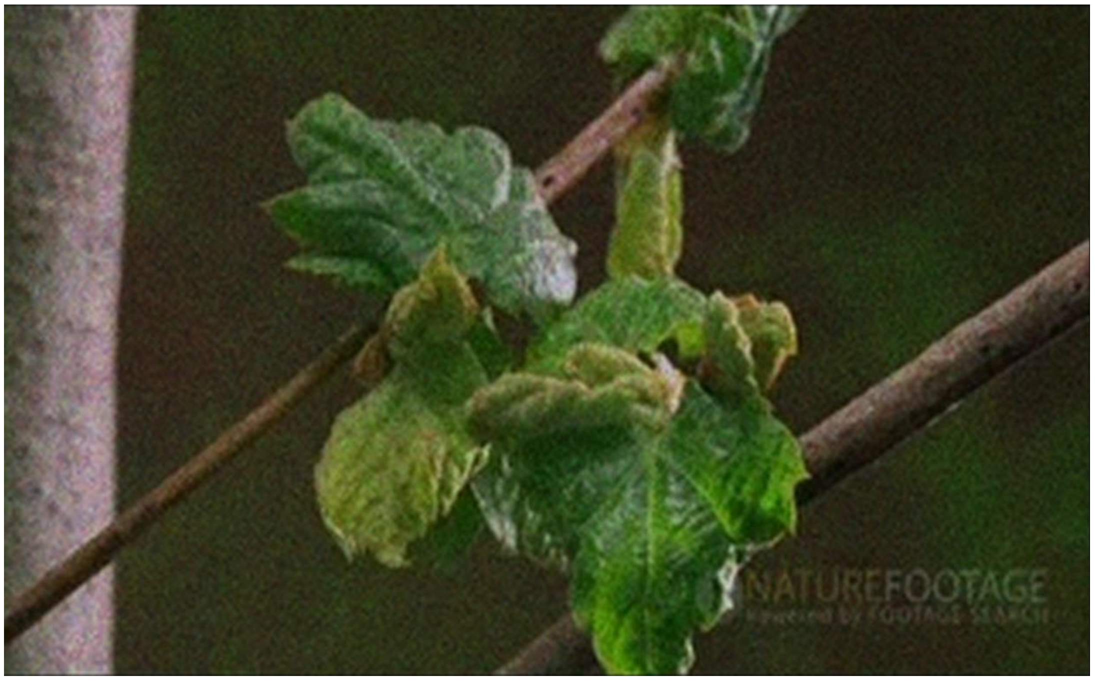


Figure 39 Image through Point Spread Function, with S_{PSF}=4



(a) without noise



(b) with noise

Figure 40 Image without Poison noise (a) and with Poison Noise (b)

well in a video.

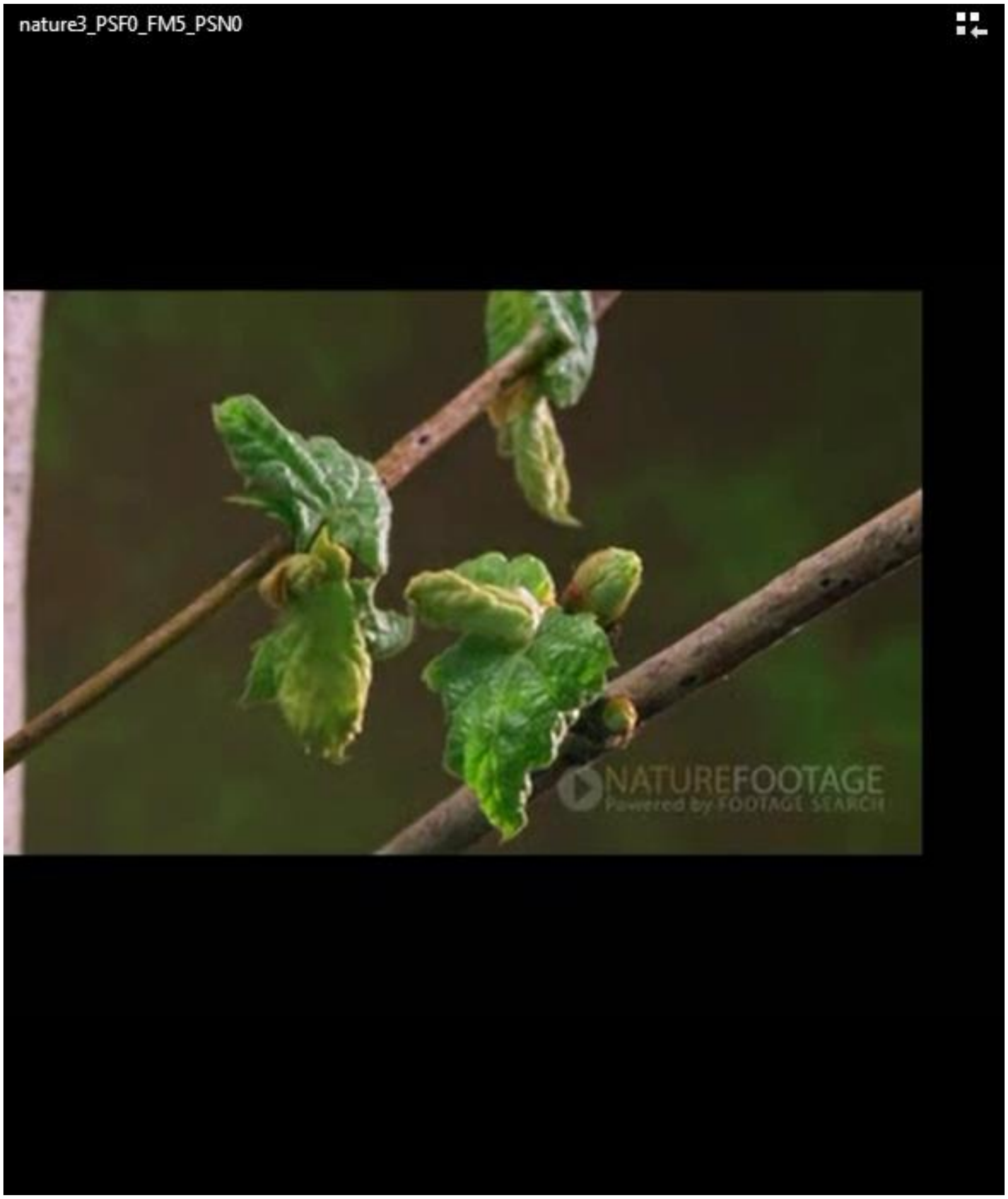


Figure 41 Image show fixation eye movement with S_{FM}=5

You may see from the image, which is not located in the center anymore. It will show well in a video.

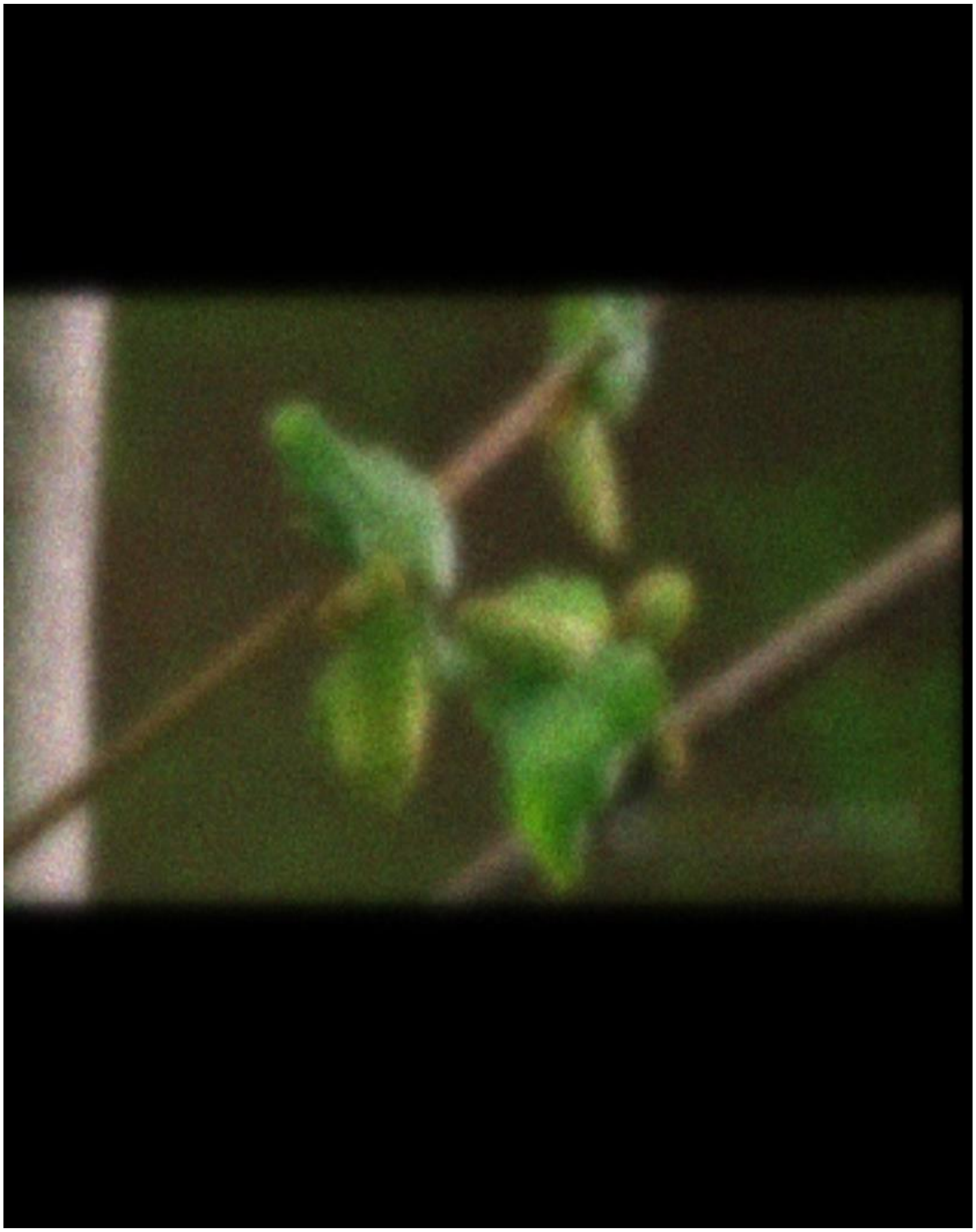


Figure 42 Combine three functions together, with S_{PSF}=1, PN=1, S_{FM}=1

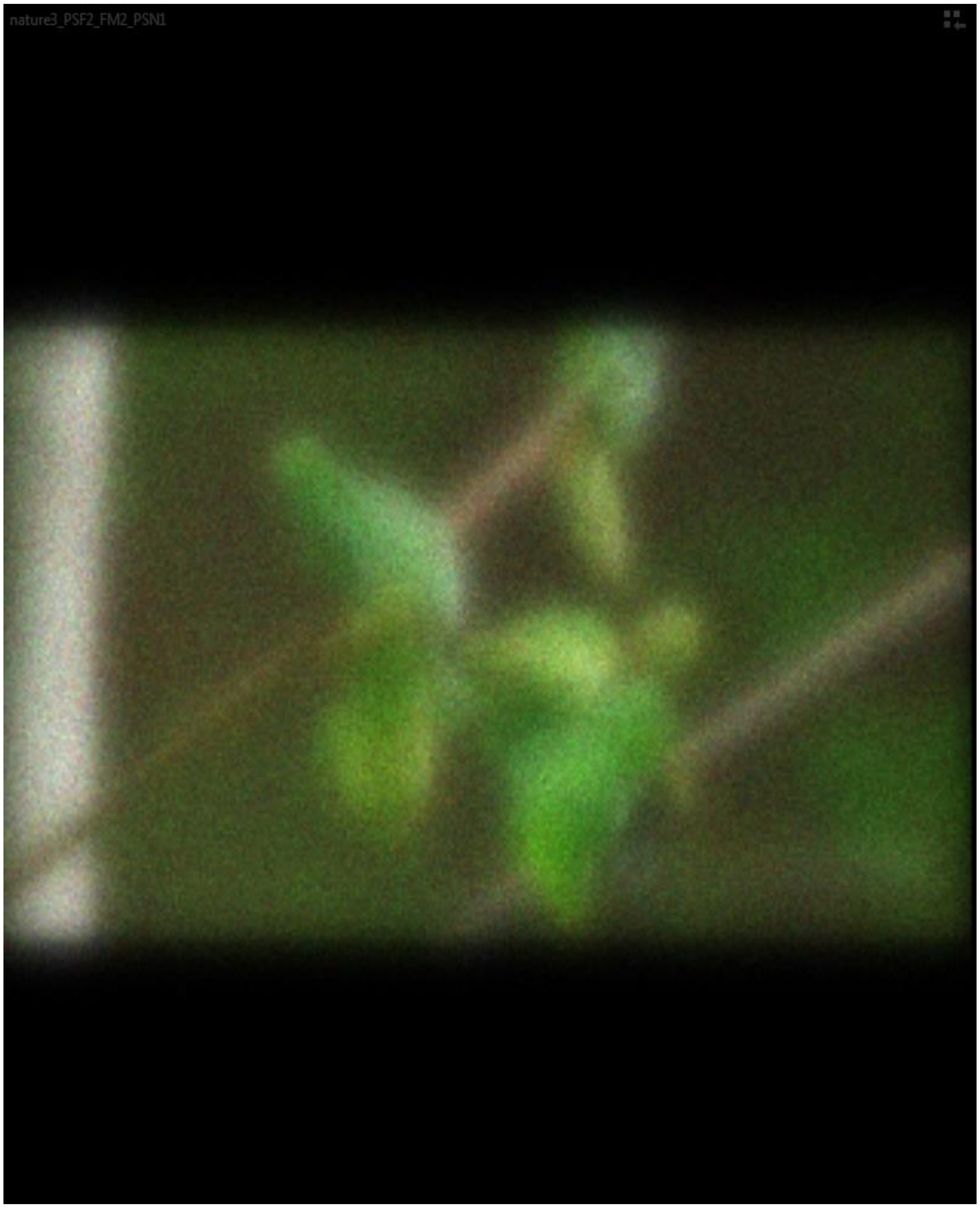


Figure 43 Combine three functions together, with S_{PSF}=2, PN=1, S_{FM}=2

CHAPTER FIVE: CONCLUSION

In the thesis, I introduce the basic design idea of the GPRAM, which is different from the conventional machine. GPRAM systems use the idea of coding instead of the logic design. Error control coding plays an important role in the design of GPRAM. The GPRAM system uses LDPC coding and decoding theory. Thus, the quantization in the iterative decoding part is needed in the FPGA design of the GPRAM. The information updates between check nodes and variable nodes have been changed to the integer number due to the quantization of the decoding. After the quantization of the iterative decoding, the BER performance is 0.3dB loss compare to the un-quantized performance. What's more, noise quantization is also needed in the FPGA design of GPRAM. A Gaussian noise look-up table was created, and could be read one by one in the end of every iteration. This quantized noise brings back the BER performance compare to the quantized performance. These noises may be not perfect match the Gaussian distribution due to the small quantities. The goal is to pick a set of noise which will match the un-quantized BER performance. We may tolerance the noises are not exactly match the Gaussian distribution, because we need the BER performance match the un-quantized one in the GPRAM design.

In the human vision part, the images we see from our eyes are blurred. The role of our brain is as an image processing center, which may recover the blurred images to the fine images. GPRAM is a general purpose machine which is like the human brain, so the images input into the GPRAM should be blurred ones. Another reason of needed for the blurred images is the following, the higher resolution of the image, the more data are required. Thus, we need to mimic the images merely see from our eyes, which are blurred images. Three steps are introduced in this thesis, which are point spread function, Poisson distribution, fixed eye movements. These three steps are independent. One image pass through these three steps, coarse images are generated.

What we did in the thesis is not enough; many other practical ideas should be added in the GPRAM. We will build a robot-like system which can display behaviors similar to some small animals. But to achieve this goal, we will need to extend error control coding beyond our current understanding. What’s more, music signal is an interesting part and needed in the GPRAM. The GPRAM tit uses singing songs to enhance its capability. For example, in the future, we may use the mind-mouth-ear-mind loop to make its music-like signal base more coherent, longer strings, more variations, so it may better predict the future. For the Vision part, after we get the coarse images, we may take only a few pixels and do the coding, after iterative decoding; the GPRAM may take a good guess and determine what is in an image.

APPENDIX: MATLAB CODE OF VISION PART

PSF function

```
        function h=pointspread(hs,scale)

        hs=0.6; %space between cones,in arc min, assume square shape here

        scale=6;

        a_1=0.417;

        a_2=0.583;

        a_3=0.443*scale;

        a_4=2.04*scale;

        x=[-10*hs*scale:hs:10*hs*scale];

        y=[-10*hs*scale:hs:10*hs*scale];

        h=zeros(length(x),length(y));

        for i=1:length(x)

           for j=1:length(y)

              h(i,j)=a_1/(2*pi*a_3)*exp(-0.5*(x(i)^2+y(j)^2)/a_3^2)+a_2/(2*pi*a_4)*exp(-
0.5*(x(i)^2+y(j)^2)/(a_4^2));

           end

        end

        h=h/h(floor(length(x)/2)+1,floor(length(y)/2)+1);

          surf(x,y,h);

         axis([-10*scale 10*scale 0 1.3]);

          xlabel('retinal distance, x (min)');

          ylabel('relative intensity');

          text(-5,1.2,'Point spread function');
```

x

Fixation Eye movement

```
function [xo,yo]=fixation_eye_mov(frame_rate,nframes,hs,scale)

%frame_rate=30;nframes=1000;scale=16;hs=0.6;

prnt=0;

xo=zeros(1,nframes,'int16');

yo=zeros(1,nframes,'int16');

frame_msec=floor(1000/frame_rate/scale+0.5);

tot_msec=frame_msec*nframes;


x_scale=1000;

x=[1:1:x_scale];

x=x-x_scale/2;

sigma2=50000;

ht=(2400/sqrt(2*pi*sigma2)*exp(-x.*x./2/sigma2)-0.7).*cos(2*pi*x/500);

%temp=sum(ht);

if prnt==1

    figure

    subplot(2,1,1),plot(x,ht,'k-')

    axis([-x_scale/2 x_scale/2 -3 5]);

    grid;

    legend('fixational eye movement')
    xlabel('msecond');
    ylabel('Amplitude (SEC OF ARC)');
    text(200,4,'impluse response');

   figure
    subplot(2,1,2), plot(20*log10(abs(fft(ht))));
    axis([0 200 -20 60]);
    legend('Power Spectral Density of h(t)')
    xlabel('Hz');
    ylabel('PSD (dB)');
end
ii=1;
for t=1: tot_msec+10
    if t==1
        ax=randn(x_scale);
    end
    yt(t)=0.;
    for i=1:x_scale
        yt(t)=yt(t)+ht(i)*ax(x_scale-i+1);
    end
    if mod(t,frame_msec)==1
        xo(ii)=int16(yt(t)/60/hs*scale);
        ii=ii+1;
    end
    for i=1:x_scale-1
        ax(i)=ax(i+1);
    end
    ax(x_scale)=randn(1);
end
ii=1;
for t=1: tot_msec+10
    if t==1
        ax=randn(x_scale);
    end
    yt(t)=0.;
    for i=1:x_scale
        yt(t)=yt(t)+ht(i)*ax(x_scale-i+1);
    end
    if mod(t,frame_msec)==1
        yo(ii)=int16(yt(t)/60/hs*scale);
        ii=ii+1;
    end
    for i=1:x_scale-1
        ax(i)=ax(i+1);
    end
    ax(x_scale)=randn(1);
end


if prnt==1
    xx=[1:1:tot_msec+10]
    figure
    plot(xx,yt,'k-')
    axis([1 tot_msec+10 -200 200]);
    legend('fixational eyemovement')
    ylabel('Amplitude (SEC OF ARC)');
    xlabel('msec');
    xx=[1:1:length(xo)]
    figure
    plot(xx,xo,'k-')
    axis([1 length(xo) min(min(xo)) max(max(xo))]);
    legend('fixational eyemovement')
    ylabel('Amplitude (SEC OF ARC)');
    xlabel('msec');
end
filr_gram
function flnm_para()
```

```
global Mv Mvname Shp_nm lenH lenW scl_res scl_PSF scl_FM scl_PSN;
Mv='';
while(length(Mv)==0)%(Mv~='0')&&(Mv~='1'))
    Mv = input('For Movie, input 1, pattern, input 0: \n','s');
    if Mv~='0' && Mv~='1'
        fprintf('wrong input, try again; \n');
        Mv='';
    end
end
if Mv=='1'
    Mvname='';
    while(length(Mvname)==0)
        Mvname = input('input movie name, for example nature3.mpg: \n','s');
    end
end
if Mv=='0'
    Shp_nm='';ind=0;
    while(length(Shp_nm)~=3||ind==0)
        Shp_nm = input('input shape name: dot,sqr for squar,ovl for oval\n','s');
        ind=0;
        if (length(Shp_nm)==3)
            if(Shp_nm=='dot')
                Mvname='dot';
                ind=1;
            end
            if(Shp_nm=='sqr')
                ind=1;
                Mvname='square';
            end
            if(Shp_nm=='ovl')
                ind=1;
                Mvname='oval';
            end
            if ind==1
                reply=input('length:\n','s');
                lenH=str2num(reply);
                reply=input('width:\n','s');
                lenW=str2num(reply);
            end
        end
        if(ind==0)
            fprintf('wrong input, try again;\n');
            Shp_nm='';
        end
```

```
    end
end
reply='';
while (length(reply)==0)
    reply = input('input PSF, 0.0 to 9.9: \n','s');
    scl_PSF=str2num(reply);
    if (scl_PSF<0)||(scl_PSF>9.9)
        fprintf('out of range, try again;\n');
        reply='';
    end
end

reply='';
while (length(reply)==0)
    reply = input('input Fixational Movement, 0.0 to 9.9: \n','s');
    scl_FM=str2num(reply);
    if (scl_FM<0)||(scl_FM>9.9)
        fprintf('out of range, try again;\n');
        reply='';
    end
end
reply='';
```

```
while (length(reply)==0)
    reply = input('input Poisson, 0.0 to 9.9: \n','s');
    scl_PSN=str2num(reply);
    if (scl_PSN<0)||(scl_PSN>9.9)
        fprintf('out of range, try again;\n');
        reply='';
    end
end
```

Main program

```
% this is program to convert any shape and movie file into
% retinal images, including point spread blur, fixation movement, and
% poisson noise

close all
clear all
global Mv Mvname Shp_nm lenH lenW scl_res scl_PSF scl_FM scl_PSN;
Mv='0';Mvname='nature3';Shp_nm='dot';
lenH=1;lenW=1;scl_PSF=0.0;scl_FM=0.0;scl_PSN=0.0;

flnm_para();
```

```
Mvname1=[Mvname '.mpg'];
hs=0.6; % cone size
if Mv=='1'
   xyloObj=mmreader(Mvname1);
   nFrames = xyloObj.NumberOfFrames;
   vidHeight = xyloObj.Height;
   vidWidth = xyloObj.Width;
   frame_rate=xyloObj.FrameRate;
%
% Preallocate movie structure.
   nFrames=300;
   mov(1:nFrames) = ...
      struct('cdata', zeros(vidHeight, vidWidth, 3, 'uint8'),...
   'colormap', []);

% Read one frame at a time.
   aa=zeros(vidHeight, vidWidth, 3,'uint8');
else
   nFrames=100;
   vidHeight= 100;
   vidWidth = 200;
   frame_rate=29.97;
```

```
    aashape=zeros(vidHeight, vidWidth, 3, 'uint8');
    LH=int16(vidHeight/2-lenH/2);
    LW=int16(vidWidth/2-lenW/2);
    for i=LH:LH+lenH-1
        for j=LW:LW+lenW-1
            aashape(i,j,:)=128;
        end
    end
   % mov(1:nFrames) = ...
      %  struct('cdata', zeros(vidHeight, vidWidth, 3, 'uint8'),...
       % 'colormap', []);
end
if scl_PSF~=0.0
    h=pointspread(hs,scl_PSF);
    bb=zeros(length(h)-1+vidHeight,length(h)-1+vidWidth, 3,'uint8')
    cc=zeros(vidHeight, vidWidth,'single')
    dd=zeros(length(h)-1+vidHeight, length(h)-1+vidWidth,'single')
end
if scl_FM~=0.0
    [xo,yo]=fixation_eye_mov(frame_rate,nFrames,hs,scl_FM);
    xo=xo-min(xo);
    yo=yo-min(yo);
    end

    for k = 1 : nFrames
      if Mv=='1'
        aa=read(xyloObj, k);
      else
        aa=aashape;
      end
      if scl_PSF~=0.0
        for j=1:3
          cc=single(aa(:,:,j))*2^(-8);
          ccmax=max(max(cc));
          dd=conv2(cc,h);
          ddmax=max(max(dd));
          bb(:,:,j)=uint8(dd*ccmax/ddmax*2^8);
        end

aa=bb(floor(length(h)/2)+1:vidHeight+floor(length(h)/2),floor(length(h)/2)+1:vidWidth+floor(le
ngth(h)/2),:);
      end
      if scl_PSN~=0.0
        ee=poissrnd(single(aa)*2^(-8)*200);
```

```
            aa=uint8(ee);
            %aa=uint8(ee/max(max(max(ee)))*2^8);
         end
         if scl_FM~=0.0
            mov(k).cdata(1:vidHeight-xo(k),1:vidWidth-yo(k),:) =
aa(xo(k)+1:vidHeight,yo(k)+1:vidWidth,:);
         else
            mov(k).cdata = aa;
         end
      end

      Cpsf=num2str(scl_PSF);
      Cfm=num2str(scl_FM);
      Cpsn=num2str(scl_PSN);
      if Mv=='1'
         Filename=['rawdata/' Mvname '_PSF' Cpsf '_FM' Cfm '_PSN' Cpsn '.avi'];
         FilenameM=['rawdata/' Mvname '_PSF' Cpsf '_FM' Cfm '_PSN' Cpsn '.mat'];
      else
         CH=num2str(lenH);
         CW=num2str(lenW);
         Filename=['rawdata/' Mvname '_H' CH '_W' CW '_PSF' Cpsf '_FM' Cfm '_PSN' Cpsn
'.avi'];
```

```
        FilenameM=['rawdata/' Mvname '_H' CH '_W' CW '_PSF' Cpsf '_FM' Cfm '_PSN'
Cpsn '.mat'];
    end
    movie2avi(mov, Filename, 'compression', 'None');
    save(FilenameM);
```

LIST OF REFERENCES

Bahl, L., Cocke, J., Jelinek, F., & Raviv, J. (1974, Mar.). Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Trans. Inform. Theory, vol. IT-20(2)*, pp. 284-287.

Berrou, A., Glavieux, A., & Thitimajshima, P. (1993). Near Shannon Limit Error-Correcting coding and decoding: Turbo-codes.1. *IEEE International Conference on Communication,* (pp. 1064-1070). Geneva, Switzerland.

Brink, S. T. (1999, May). Convergence of Iterative Decoding. *Electronics Letters*, pp. 10-35.

Chung, S. Y., & Forney, G. D. (2001, Feb). On the design of Low-Density Parity-Check Codes within 0.0045 dB of the Shannon limit. *IEEE Communications Letters,Vol.5, No.2*.

Dolinar, S. (1988). *A New Code for Galileo.* Pasadena, California: Jet Propulsion Laboratory.

Elias, P. (1955). Coding for noisy channels. *Proc. IRE Conv.Rec. part 4*, pp. 37-46.

Fano, R. M. (1963, April). A heuristic discussion of probabilistic decoding. *IEEE Trans Inform. Theory 9*, pp. 64-74.

Forney, G. D. (1973a, Mar.). The viterbi algorithm. *Proceedings of the IEEE., vol. 61*, pp. 268-278.

Forney, G. D. (2001b, Feb.). Codes on graphs: normal realization. *IEEE Trans. Inform. Theory, vol. 47*, pp. 520-548.

Fox, M. D., & Raichle, M. E. (2007, Sep). Spontaneous fluctuations in brain activity observed with functional magnetic resonance imaging. *Nature Reviews Neuroscience 9*, pp. 700-711.

Gallager, R. G. (1963a). *Low-Density Parity-Check Codes.* Cambridge, MA: M.I.T. Press.

Gallager, R. G. (1968). Information Theory and Reliable Communication. New York: Wiley.

Gardner, H. (1983). *Frames of mind: The theory of multiple intelligences.* New York: Basic Books.

Hagenauer, J., & Hoeher, P. (1989). A Viterbi algorithm with soft-decision outputs and its applications. *Globecom,* (pp. 47.1.1-7). Dallas, TX.

Heller, J. A., & Jacobs, I. M. (1971, Oct). Viterbi decoding for satellite and space communication. *IEEE Trans. Commun. Technol., vol. COM-19,* pp. 835-848.

Klein, S. A., & Levi, D. M. (1985). Hyperacuity thresholds of 1 sec: theoretical predictions and empirical validation. *J. Opt. Soc. Am 75,* pp. 1170-1190.

Kschischang, F. R., Frey, B. J., & Loeliger, H. A. (2001, Feb). Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory, vol. 47,* pp. 498-519.

Levy, W. B., & Steward, O. (1983, Apr.). Temporal contiguity requirements for longterm associative potentiation/depression in the hippocampus. *Neuroscience 8 (4),* pp. 791-797.

Li, H. H., & Wei, L. (2013c). General Purpose Representation and Association Machine Part 4: Improve Learning for Three States and Multi-tasks. *IEEE Southeastcon*.

Li, H. H., Dai, B., Schultz, S., & Wei, L. (2013). General Purpose Representaion and Association Machine Part 3: Prototype Study using LDPC codes. *IEEE Southeastcon*.

Lin, S., & Costello, D. J. (2004). *Error Control Coding, Edition 2.* Prentice Hall.

MacKay, D. J., & Neal, R. M. (1996, Aug). Near Shannon limit performance of low-density parity-check codes. *Elect. Lett.,vol.32,* pp. 1645-1646.

Mackay, D. J., McEliece, R. J., & Cheng, J. F. (1997). Turbo decoding as an instance of Pearl's belief propagation algorithm. *IEEE J. Select. Areas Commun*.

Mackworth, A., & Poole, D. (1988). Artificial Intelligence: Foundations of computational agents. Cambridge University.

Neumann, J. v. (1958). *The computer and the brain.* USA: Yale University Press.

Olshausen, B. A., & Field, D. J. (1996a). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature, 381*, 607-609.

Olshausen, B. A., & Field, D. J. (1996a). Emergence of Simple-Cell Receptive Field Properties by Learning a Sparse Code for Natural Images. *Nature, 381*, 607-609.

Olshausen, B. A., & Field, D. J. (1997b). Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Res.*, 3311-3325.

Pearl, J. (1988). Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. San Mateo, CA: Morgan Kaufmann.

Rehn, M., & Sommer, F. T. (2007). A network that uses few active neurones to code visual input predicts the diverse shapes of cortical receptive fields. *Journal of Computational Neuroscience 22*, 135-146.

Richardson, T. J. (2001, Feb). Design of Capacity-Approching Irregular Low-Density Parity-Check Codes. *IEEE Trans. Inform Theory, Vol 47, No.2*.

Shannon, C. E. (1948, July). A mathematical theory of commnication. *Bell System Technical Journal*, pp. 379-423.

Tanner, R. M. (1981, Sep). A recursive approach to low complexity codes. *IEEE Trans. Inform. Theory, vol. 27,* pp. 533-547.

Viterbi, A. J. (1967, Apr.). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inform. Theory, vol. IT-13,* pp. 260-269.

Wei, L. (2003c). Connectivity Reliability of Large Scale Random Ad Hoc Networks. *Procs of MILCOM*. Boston, USA.

Wei, L. (2012a, Mar.). General Purpose Representation and Association Machine Part1: Introduction and Illustrations and Part 2: Biological Implications. *IEEE Southeastcon*.

Wei, L. (2012b). General Purpose Representation and Association Machine Part 2: Biological Implications. *Sout*.

Wei, L., & Qi, H. (2000b, July). Near Optimal Limited Search Decoding on ISI/CDMA channels and decoding of long convolutional codes. *IEEE Trans. Inform. Theory, vol. 46-4*, pp. 1459-1482.

Wei, L., Levi, D. M., Li, R., & Klein, S. (2007, April). Feasibility Study on a hyperacuity Device with Motion Unvertainty: Two-point Stimuli. *IEEE Trans. on Systems. Man and Cybernetics*, pp. 385-397.

Westheimer, G. (1975). Visual acuity and hyperacuity. *Invest Ophthalmol 14*, pp. 570-572.

Yang, D., & Froemke, R. C. (2002b, Jan. 15). Spike timing-dependent synaptic modification induced by natural spike trains. *Nature*, pp. 433-438.

Yang, D., & Poo, M. M. (2006a). Spike timing-dependent plasticity from synapse to perception. *Physiol Rev 86*, pp. 1033-1048.

Yuen, J. H., & Vo, Q. D. (1985). In search of a 2 dB Coding Gain. *TDA Progress Report*, (pp. 42-83). Pasadena, California.