2016

Applied Advanced Error Control Coding for General Purpose Representation and Association Machine Systems

Bowen Dai
University of Central Florida

APPLIED ADVANCED ERROR CONTROL CODING FOR GENERAL PURPOSE REPRESENTATION AND ASSOCIATION MACHINE SYSTEMS

by

BOWEN DAI
M.S. University of Central Florida, 2013
B.S. Beijing Institute of Technology, 2009

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical and Computer Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2016

Major Professor: Lei Wei

ABSTRACT

General-Purpose Representation and Association Machine (GPRAM) is proposed to be focusing on computations in terms of variation and flexibility, rather than precision and speed. GPRAM system has a vague representation and has no predefined tasks. With several important lessons learned from error control coding, neuroscience and human visual system, we investigate several types of error control codes, including Hamming code and Low-Density Parity Check (LDPC) codes, and extend them to different directions.

While in error control codes, solely XOR logic gate is used to connect different nodes. Inspired by bio-systems and Turbo codes, we suggest and study non-linear codes with expanded operations, such as codes including AND and OR gates which raises the problem of prior-probabilities mismatching. Prior discussions about critical challenges in designing codes and iterative decoding for non-equiprobable symbols may pave the way for a more comprehensive understanding of bio-signal processing. The limitation of XOR operation in iterative decoding with non-equiprobable symbols is described and can be potentially resolved by applying quasi-XOR operation and intermediate transformation layer. Constructing codes for non-equiprobable symbols with the former approach cannot satisfyingly perform with regarding to error correction capability. Probabilistic messages for sum-product algorithm using XOR, AND, and OR operations with non-equiprobable symbols are further computed. The primary motivation for the constructing codes is to establish the GPRAM system rather than to conduct error control coding per se. The GPRAM system is fundamentally developed by applying various operations with substantial over-complete basis. This system is capable of continuously achieving better and simpler approximations for complex tasks.

The approaches of decoding LDPC codes with non-equiprobable binary symbols are discussed due to the aforementioned prior-probabilities mismatching problem. The traditional Tanner graph should be modified because of the distinction of message passing to information bits and to parity check bits from check nodes. In other words, the message passing along two directions are identical in conventional Tanner graph, while the message along the forward direction and backward direction are different in our case. A method of optimizing signal constellation is described, which is able to maximize the channel mutual information.

A simple Image Processing Unit (IPU) structure is proposed for GPRAM system, to which images are inputted. The IPU consists of a randomly constructed LDPC code, an iterative decoder, a switch, and scaling and decision device. The quality of input images has been severely deteriorated for the purpose of mimicking visual information variability (VIV) experienced in human visual systems. The IPU is capable of (a) reliably recognizing digits from images of which quality is extremely inadequate; (b) achieving similar hyper-acuity performance comparing to human visual system; and (c) significantly improving the recognition rate with applying randomly constructed LDPC code, which is not specifically optimized for the tasks.

KEYWORDS: General Purpose Systems, Advanced Coding, Low-Density Parity Check Codes, Image Processing Unit

ACKNOWLEDGMENTS

I would like to express my sincere gratitude and deepest appreciation to my academic advisor, Professor Lei Wei, for his guidance, inspiration, and patience through five years of study at University of Central Florida (UCF). I feel grateful for his suggestions and discussions throughout my doctoral research. This dissertation would not have been possible without his persistent assistance and review. Financial supports from UCF are greatly appreciated from which I got a four-year fellowship.

Profound gratitude goes to my PhD committee members, Professor Mingjie Lin, Professor Nazanin Rahnavard from Department of Electrical and Computer Engineering, Professor Damla Turgut from Department of Computer Science, and Professor Qiyu Sun from Department of Mathematics, for providing invaluable advice, information, and support on different aspects of my dissertation and for their devoted cooperation.

I would like to thank the members of Signals and Communications Lab, with whom I have shared invaluable memories during my graduate studies. Particularly I would like to acknowledge Huihui Li, Alireza Sani, Mojitaba Shirazi, Toktam Amanzade for the assistance of all related discussions.

This is dedicated to my family and many friends who supported me on this memorable journey. I could not have walked alone for all these years without your love and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Motivations

Human brain is capable of executing functions such as intelligence, planning, reasoning, and abstract thought in general purposes [1] [2] [3]. And intelligence gradually emerges during evolution [4] [5]. On the other hand, artificial intelligence (AI) is typically developed aiming at specific purposes [7] [8] [9], such as modern computer which was invented in the 1940s [10]. The latter can perform with precision and speed of which human brains are not capable [11] [12]. Upon the difference observed between human brain and AI, precision and versatility needs to be considered in order to construct a machine with general purposes. To achieve precision, one or several generalized function(s) are required to represent a fixed association, and attempt to prove all other types of functions can be obtained by the generalized one(s). As for versatility, many different types of specific functions with variations needs to be included and combined to improve possible capabilities. In fact, the latter appeared in biological systems [13], which was naturally developed during evolution.

Human beings have already succeeded in handling many practical problems, prior to the establishment of theories of sciences. For instance, our ancestors gather and hunt a mammoth with a careful strategy: they ambush the animal and then drive the panic-stricken mammoth into a muck. The selection of kill site, weapon, and opportunity ensures the successfulness of hunting. In modern world, these factors are determined by computer-aided accurate calculations, with numerous mathematical or scientific models involved. And the strategy is planned and executed non-erroneously. However, the human mind is guiding and leading the decision-making by providing many widely but roughly trial thoughts. This encourages us to develop a machine to mimic this role of human mind, to make a decision based on the consideration of many parallel possibilities and different types of ideas. On the other hand, precision is ranked as lower priorities. Intuition is the main guidance during research and discovering processes, which is learned from previous experiences such as failures and random guesses.

Nowadays, scientists and engineers are still using intuition as a guide to their research and work sometimes just as our hunter-gatherer ancestor did. These intuitions utilizes stored experience [14], including failures and random guesses. And intuition plays an important role on decision making [15]. Scientists and researchers might benefit from their intuition in terms of how to approach and tackle sophisticated problems or how to develop novel theories. However, none of modern theories would be found if they completely relied on their instincts. Scientific approaches and mathematical principles are still needed to prove the correctness of possible solutions and hypotheses [16]. It is definitely amazing about how far we have achieved on Medicine [17], Neuroscience [18]. The computational tasks solved by AI lacks the intuitional part since human beings play an essential role in that process. For instance, researchers decide to implement which algorithm and utilize which feature for one specific task.

In summary, in most of the engineering task solved by AI lacks complexity factor due to the foundation of mathematical principles and engineering structure developed in [19] [20] [21]. A theory of General-Purpose Representation and Association Machine (GPRAM) is proposed in [22], which aligns with the dimension of versatility. This is inspired by the mechanism of human brain that associates the representations of external world, which enables solving problems in general purpose [23]. In our work, we want to implement a prototype of GPRAM, behind which the overall philosophy is to perform a large number of tasks with acceptable accuracy, rather than design for a specific task with high accuracy. The primary performance metrics are no long precision and speed, rather flexibility and complexity.

To implement the aforementioned prototype, we choose error control code, Low-Density Parity

Check (LDPC) codes to be specific, as the core which functions like the “brain” of GPRAM. The key reasons of this choice of error control coding are [22]:

1. The existence of good performing codes are possible since the average performance over randomly constructed long codes can be close to Shannon limit [24].
2. Plenty of randomly constructed codes are sub-optimal if generated from Tanner graphs [25] under the constraint that the graphs only contain few loops with small girth.
3. Sub-graphs can propagate and pass information among each other while complexity is at a low level [25] [26].
4. Iterative decoding is implemented in [26] and this process is noise-resilient and error-resilient.
5. McEliece *et al,* [27] have discovered connections between iterative decoding and Pearl’s belief algorithm [28], which is vital to generate information required by Bayesian networks.
6. Factor graph representation and iterative decoding can be derived as instances of a wide variety of algorithms, including algorithms in AI and signal processing [29].
7. Neurons and iterative decoding share a similar way of functioning: repetition, random permutation, and non-linear operation [30].

Based on these reasons, it is not unreasonable to state that LDPC code is indeed a good candidate for implementing GPRAM, even though some might argue that more proofs are still needed to confirm Bayesian models in cognitive science [31]. In order to understand and utilize this best known information transmission and reception mechanism to the most, we explored, studied and extended error control coding to different directions: from hamming code to LDPC codes, from equiprobable symbols to non-equiprobable ones, from error correcting codes with exclusive-OR(XOR) gate to codes with AND and OR gates, and combination of these all. There are works studied long codes [32], short codes [33] [34], and other types of codes [35].

Outline

In this dissertation, the concentration is on studying advanced error control coding, for instance, (7,4) Hamming code with AND and OR gates, and LDPC codes with non-equiprobable symbols, for the implementation of GPRAM system prototype. And an image process unit (IPU) framework is suggested which accepts images as input for GPRAM . Many tasks are selected to test against the proposed framework. There are seven chapters in this dissertation, which are organized as follows.

All the related work, techniques involved and background are presented in Chapter 2. Different logic gates are reviewed at first. Relevant error control coding theory is introduced next, including LDPC codes and (7, 4) hamming code, which is followed by human vision related visual information and also the concept of hyper-acuity.

The study and extension on linear block codes is investigated in Chapter 3. The problem of assuming equiprobable source information is stated firstly, which follows by the probability mismatch between the input and the output of encoders. The limitation of XOR operation in dealing with non-equiprobable is demonstrated. Different ways of solving the probability mismatch problem are proposed, such as quasi-XOR operation and intermediated transformation layer. How to construct codes for non-equiprobable symbols using quasi-XOR operation is explained in details.

Extension on linear block codes to a further direction is studied, which incorporates AND, OR and XOR gates in Chapter 4. Prior probabilities for each node is discussed and calculated. A study case of modifying (7,4) Hamming code using AND and OR gates is illustrated in details to investigate the properties for such codes. Traditional message passing algorithm is revised according to the changes of logic gates. Simulation results showing error handling capabilities for revised (7,4) Hamming codes are presented as well.

Chapter 5 describes the necessity of extending LDPC codes with equiprobable symbols to LDPC codes with non-equiprobable symbols. The signal and system for this application is defined at the beginning. We found the optimized constellation for non-equiprobable symbols and proved that such constellation scheme maximizes capacity and the channel mutual information. It then follows by decoding procedure for LDPC codes with non-equiprobable symbols. And we also compare the performance of LDPC codes with non-equiprobable symbols with LDPC codes with equiprobable ones. The replacement of equal space constellation with optimal constellation could gain 0.72 dB in performance. Several cases of short LDPC codes are explored and gain almost 0.4 dB.

Chapter 6 proposes a simple IPU structure for GPRAM system. A structure of IPU based on LDPC codes and image mapping is defined later. This structure includes three aspects of Visual Information Variability (VIV), LDPC parity check matrix, and power scaling process for which there are four methods discussed. The experimental procedures and algorithms are described in details. We also present the simulation results for different type of tasks and also for different type of detectors.

We summarize this dissertation by describing the time-line and future research directions in Chapter 7.

Contributions

The major contributions in this dissertation are listed as follows:

1. We presented about critical challenges in designing codes and iterative decoding for non-equiprobable symbols may pave the way for a more comprehensive understanding of bio-signal processing. (Chapter 3 and papers C1 and C2)

2. We demonstrated the limitations of XOR operation in dealing with non-equiprobable symbols. The limitation of XOR operation can be potentially resolved by applying quasi-XOR operation and intermediate transformation layer. (Chapter 3 and paper C2)

3. We showed how to construct codes for non-equiprobable symbols using quasi-XOR operation and the codes cannot satisfyingly perform with regarding to error correction capability. (Chapter 3 and paper C2)

4. We studied a new type of (7, 4) Hamming code by extending it to non-equiprobable symbols, then replaced XOR operation by other logic gates such as AND and OR. And further we computed probabilistic messages for sum-product algorithm using XOR, AND, and OR operations with non-equiprobable symbols. Large amount of operations available in the system with substantial over-complete basis will lay a foundation to develop a GPRAM system that can continuously discover better and simple approximations for complex tasks. (Chapter 4)

5. We analyzed how to decode LDPC codes with non-equiprobable binary symbols. And we proved that the message passing from check nodes to information bits and to parity check bits are diverse from conventional Tanner graph, which is with equiprobable binary symbols. (Chapter 5 and paper J3)

6. A method of optimizing signal constellation is described, which is able to maximize the channel mutual information for non-equiprobable binary symbols. We simulated LDPC codes with prior probabilities of $(0.3, 0.7)$ and was able to obtain 0.72 dB gain using our method comparing with equal space constellation. (Chapter 5 and paper J3)

7. We proposed a simple IPU structure for GPRAM system which implements a form of generalized learning which could function like an “eye” for GPRAM systems in the future. (Chapter 6)

8. We selected different tasks to test the performance ability of IPU/GPRAM with visual variability information which human beings perceive in real life. These tasks include 10 digit recognition, alphabetical letters recognition, roman numerals recognition, fine details visual discrimination and human face recognition. And we found that the IPU is capable of (a) reliably recognizing digits from images of which quality is extremely inadequate, (b) achieving similar hyper-acuity performance comparing to human visual system, and (c) significantly improving the recognition rate with applying randomly constructed LDPC code, which is not specifically optimized for the tasks. (Chapter 6 and papers J1, J2)

Paper List

Journal Papers:

J1. B. Dai, H. Li and L. Wei, “Image Processing Unit for General-Purpose Representation and Association System for Recognizing Low-Resolution Digits with Visual Information Variability,” in *IEEE Transactions on System, Man, and Cybernetics: System*, Accepted

J2. H. Li, B. Dai and L.Wei, “Image Processing Unit for GPRAM System for Recognizing Low Resolution Facial Images with Visual Imperfectness,” in *IEEE Transactions on Information Theory*, Submitted

J3. B. Dai and L. Wei, “Low-Density Parity Check Codes with Non-equiprobable Symbols,” in *IEEE Communication Letter*, vol. 17, no. 11, pp. 2124-2127, Nov. 2013.

Conference Papers:

C1. H. Li, B. Dai, S. Schultz and L. Wei, “General Purpose Representation and Association Machine, Part 3: Prototype Study Using Low-Density Parity Check codes,” in *Proceedings of IEEE Southeastcon*, Jacksonville, FL, USA, 2013, pp. 1-5.

C2. B. Dai and L. Wei, “Some Results and Challenges on Codes and Iterative Decoding with Non-equal Symbol Probabilities,” in *IEEE International Symposium on Information Theory and its Applications (ISITA)*, Hawaii, USA, Oct. 2012, pp. 116-120.

CHAPTER 2: BACKGROUND AND RELATED WORK

Logic Gates

Turing and Church investigated the characteristics of computability in the year of 1936 [19] [36], since then classical computation theory became prominent. Logic gates and logic circuits play as a major role in the theory of computation. Optimal structure of classical logic circuits have been developed [37]. A logic gate is essentially an electric circuit, and ideally implements a boolean function [38]. These logic gates take one or more logical inputs and generates one logical output. A gate is logically reversible if the input values can be uniquely determined from the output values, otherwise the gate is defined as logically irreversible. The AND and OR gates show as irreversible logical gates. The seven basic logic gates are: AND, OR, XOR, NOT, NAND (NOT AND), NOR (NOT OR), and XNOR (NOT XOR). Fig. 2.1 shows the mapping function from multiple inputs to an output. Fig. 2.2 shows the distinctive shape for several logic gates [39].

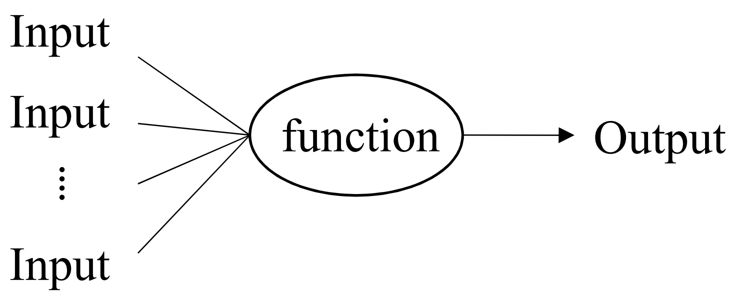
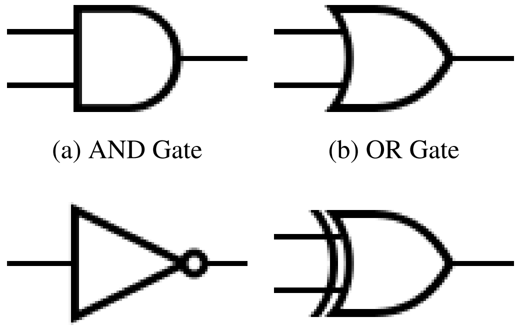


Figure 2.1: Inputs and Output for a Logic Gate with Mapping Function

There has been work reported about the similarity between logic gates and neurons [40]. McCulloch *et al.* [41] proposed that brain can be decomposed to threshold units which consist of logic gates. And the threshold units are similar to neurons. This proposed framework laid foundation for artificial neural networks [42] and machine learning theory [43] [44]. And also some suggests how brains differ from computers [45] and argues using logic-gates simplifies brain functions since they perform on pure binary elements [46].



(a) AND Gate (b) OR Gate

(c) NOT Gate (d) XOR Gate

Figure 2.2: Distinctive Shape for Different Logic Gates

Nonetheless, a logic gate provides a function mapping a set of inputs to an output. And the number of functions increases exponentially if logic gates are cascaded.

Linear Block Codes

In coding theory, the linear block codes is a subclass of all block codes. The word "linear" means any linear combination of any selection of codewords is also a codeword. There are several famous and widely used examples of block codes, such as Hamming codes, Reed-Solomon codes, Reed-Muller codes, and many more. In our study, we choose Hamming codes and LDPC codes as study cases, for the reasons stated in Chapter 1.

Each group of linear block codes is associated with two matrices, named Generator Matrix or G matrix, and Parity Check Matrix or H matrix. A codeword is essentially a tuple with length of K, which can be partitioned into two blocks. A block contains message bits with the length of $K - T$, while the other contains parity check bits with the length of T. Parity check bits are redundant information which can be used to recover original message bits after transmitting through noisy channel.

Generator matrix can be used to generate all the codewords given all the combinations of message bits. The number of possible codewords is 2^{K-T} over the field *GF(2)* with block length of K and dimension of T. Parity Check Matrix can be used for decoding and recovering original message bits through message passing algorithm. We used sum-product algorithm in our study.

Equation (2.1) shows an example of Parity Check Matrix when $K = 10$ and $T = 5$. It is worth mentioning that each row of Parity Check Matrix represents parity check constraint on codewords. An example of valid codewords for the Parity Check Matrix defined in Equation (2.1) is $[1, 1, 1, 1, 0, 0, 0, 1, 0, 1]$ since it satisfies all the parity check constraints.

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{2.1}$$

Low-Density Parity Check codes

LDPC codes were developed and introduced by Robert Gallager [47] in his Ph.D. dissertation and till Mackay revealed the good performance of LDPC codes [48]. Its associated message passing algorithm, given as belief propagation algorithm, has been applied in communities of AI [28]. A special type of belief propagation was firstly analyzed and described in [49], and was applied to hard decision decoding of LDPC codes in [50]. LDPC codes are a subset of linear block codes. In this dissertation, we only discuss binary codes and assume the block length of LDPC codes is K and dimension is T. Binary code means there are only two possible values $\{0, 1\}$ for each single element in each codeword. And that element is called bit by convention and thus in the rest of this dissertation. When the probability of message bit being 0 is equal to the probability of it being 1, this message bit is therefore referred as equiprobable, non-equiprobable otherwise. The codes are called low-density because the Parity Check Matrix contains far more 0's in comparison to the amount of 1's.

A more visualized way to represent LDPC codes is through Tanner graph [51] as shown in Fig. 2.3. A Tanner graph is a bipartite graph showing connections among variable nodes and check nodes with them aligning on two sides [52]. Each variable node corresponds to an element or a bit in every codeword and to a column in Parity Check Matrix. Each check node corresponds to a parity check constraint and to a row in Parity Check Matrix. Therefore, any codeword must satisfy all parity check constraints to be called valid. The operation for a parity check constraint is defined as modulo 2 addition or XOR. Variable nodes and check nodes are connected through edges which indicate as the ones in Parity Check Matrix in a Tanner graph.

Human Vision Hyper-Acuity

The hyper-acuity, also known as vernier-acuity [53], is described as an ability that human eye detects the misalignments of lines with extraordinary accuracy. It can be measured under various situations [54] [55] [56], for example, an object is standing, sitting, or walking. Human visual systems can surprisingly distinguish the misalignment between two lines within a misalignment of 1 arcsec [57] [58] [59], which is approximately equivalent to 1/30th of the inter-cone spacing. The term of hyper-acuity is termed by Westheimer [61] because the spatial discrimination task [60] evaluating relations between between a line or a dot and a reference are far smaller than the cone spacing. The following factors should be specified to experimentally measure the capability of human hyper-acuity capability, including distance between an observer and a screen, the screen size, the screen resolution, and stimulus conditions like luminance and edges. This is selected as one of the experimental tasks to compare the differentiation rate between IPU and human performance, in our study described in Chapter 6

Literature Review

Modern computer was invented and developed in the 1940s to satisfy the requirement of computational precision and rate that our brains are not capable of [19] [21]. Recent publications in [22] [23] highlighted the philosophic differences between the computer and a GPRAM. A GPRAM can simultaneously conduct various tasks with adequate accuracy, instead of a single task with superior accuracy. Fundamental investigations have been listed in [22] [23], from the aspects of modern error control coding, information theory, and biological/life science. The importance of self-noises in a system has been described by [62], indicating the possibility of applying LDPC codes as a representative layer in such systems. The feasibility of implementing matched filters is reported by [63] [64], this is for the purpose of achieving statistically optimal performance for a biologically inspired system. A device achieving hyper-acuity vision is proposed by Wei *et al.* in [65], under the circumstance of VIV. Iterative decoding for error control code with non-equiprobable symbols is discussed in [66] [67], because the source that generates these symbols might not be practically ideal. A GPRAM prototype system can be established by applying simple LDPC codes, and its multi-task performing can be achieved by conducting a progressive learning process, per introduction in [68]. In the presented thesis, a GPRAM is supposed to be constructed and some substantial results are exhibited to this end.

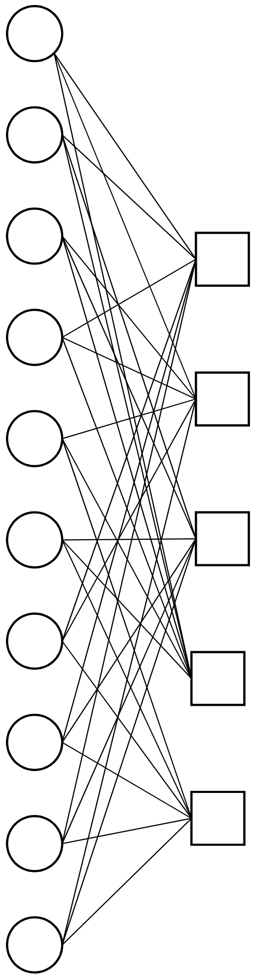


Figure 2.3: Tanner Graph with $K = 10$ and $T = 5$, Corresponding to the H matrix Defined in Equation (2.1). Circles Indicate Variable Nodes while Squares Indicate Parity Check Nodes. Connection Lines from Variable Nodes to Parity Check Nodes are Edges.

The concept of GPRAM has not been widely accepted in the existed research areas mentioned above. This is because the most competitive results with regarding to precision may not be obtained by our GPRAM approach in each highly-specified task, comparing to the existed techniques in these fields. On the contrary, a comprehensively interdisciplinary integration is required by GPRAM rather than limited combination of a few research areas, let alone the concentration on a specific field. Therefore, the higher priority would be to achieve variation, flexibility, and versatility, instead of the computational precision and rate. Furthermore, various tasks and templates can be more easily blended together in the early stage with vague boundary, where accuracy should not be considered as the critical factor in comparison. The limitation of study would be related to the current understanding of error control coding and the small-scale of computer simulation. However, it can be significantly improved from many different aspects when sufficient attention is attracted in the scientific community.

Another objective of this study is to demonstrate that VIV may be unanticipatedly beneficial for developing a multi-task visual system. This would correspond to the fact that the brain benefits from uncertainty, as reported in [69] [70]. Recognition of images (28-by-28 pixels) is selected as the testing case, including realistic VIV such as point spread, fixational movement, orientation rotation, and Gaussian noise. These visual factors and their effects are described in [65]. A GPRAM trial platform is then established by applying a randomly constructed LDPC code and iterative decoding [71]. The former secures that no particular optimization applied for any specific task. Nevertheless, the results will be described in alignment with the traditional best-known techniques for the purpose of comparison. Multiple machines will be devised under the platform in a mature GPRAM system, while each individual is equipped with its own approaches for representation and association. Therefore, GPRAM units may be dictated by different procedures and converge to each distinct aspect. The current study can then be considered as one of the numerous implemented GPRAM units. It is worth mentioning that the ultimate objective is to develop millions of tantamount rather than identical units.

The recognition of hand-written digits and optical character has been an important research topic in the field of machine learning [72], for example, to process blurred images due to camera motions [73], low resolution text, various paper quality, or other issues [74] [75]. A novel algorithm is proposed about one-shot classification using handwritten characters in [76]. Research related to computational neuroscience over the last 20 years is reviewed in [77]. An explanation from the aspect of mathematics about human vision is reported in [78]. In the neural science areas, researchers have been tried to reconstruct the visual view from animals either by directly recording neural activities [79] or through non-intruding technology such as fMRI [80]. However, the directly obtained animal view are barely recognizable and the process to form the sharp and clear image as we experience still remains a mystery. It was concluded that blurringly sampled and quantized images could actually improve the recognition in [81], which is corresponding to our approach of beneficially involving the effect of VIV in recognition process.

It is reported that human has superior capability to recognize hand-written digits [82] [83]. Various techniques have been developed with regarding to this topic over the last 20 years [84] [85]. Ciresan *et al.* [107] applied a large deep multilayer perceptron (MLP) network, achieving a computationally feasible error rate of 0.35% on the MNIST hand-written digits benchmark. The error rate could be further decreased to 0.27% in [108] with additional training time. Wang *et al.* [109] proposed a deep learning method to resolve a very-low-resolution recognition problem (16-by-16 pixels). It is worth mentioning that VIV is not considered nor applied in the studies above.

CHAPTER 3: STUDY ON ERROR CORRECTION CODES WITH NON-EQUIPROBABLE SYMBOLS

Many work and optimization problems are developed and designed under the assumption that source information is equiprobable [67] [86]. Modern telecommunication and error control coding are largely designed to handle equiprobable symbols. In this chapter, we present critical challenges in designing codes and iterative decoding for non-equiprobable symbols may pave the way for a more comprehensive understanding of bio-signal processing.

It is recommended to develop the source and channel encoders independently in a communication system according to Shannon's separation principle [87] which had an enormous impact on constructing telecommunication systems. Since the same performance can be achieved both by a jointly devised source and channel coding system and by separately developed source and channel encoders. It is almost a default assumption that data have been compressed by an ideal source encoder and hence generates an independent, identically distributed (i.i.d) sequence of equiprobable bits. Subjects such as modulation, capacity, and error correction capabilities have been well considered and researched under this assumption in [71] [88] [89] [90]. The authors presented a general method to compute the capacity of LDPC codes when the inputs are binary and the channel is memoryless in [91].

However, cases beyond this assumption have been considered in recent papers not only in telecommunication system design [92] [93] [94], but also in the understanding of biological systems [22] [23] [65]. In this Chapter, it is assumed that information source symbols have fixed, but non-equal prior probabilities. Based on this assumption, our study is focusing on designing an encoder which produces the symbols with the same prior probability settings as input symbols and hence generating sequence of non-equiprobable bits regardless of what the probability settings are. The study result can be generalized and utilized by studies on more complicated cases in GPRAM.

XOR operation is the key operation connecting parity check bit and information bit in error control coding when dealing with equiprobable symbols. However it is not an optimal operation when the source are non-equiprobable symbols. If two equiprobable symbols s_1 and s_2 are two inputs to an XOR gate, then the output of XOR operation,$s_3 = s_1 \oplus s_2$ where $\oplus$ denotes XOR, is an equiprobable symbol; however, if $p(s_1 = 0) = p(s_2 = 0) = 0.3$, where $p(.)$ denotes probability, the output symbol has the prior-probability $p(s_3 = 0) = 0.58$ which does not match with the prior-probabilities of two inputs. This issue will be demonstrated in details in next Section. Furthermore, in topics like biological systems which is related to likelihood detection theory [65], source compressing or sparse coding [95], iterative turbo-like joint estimation and decoding or factorized decoder [96], the input symbols might be non-equiprobable. Another benefit to study how to design source and channel encoding and decoding mechanisms for the systems with non-equiprobable symbols is that it might be used in bio-systems and a new type of intelligent machine GPRAM [22] [23] [68].

Therefore our work in this Chapter aims to discover potential similarities between advanced coding theory and brain functionalities, which are discussed in [27] [97] and how to use the former to represent and simulate the latter in GPRAM.

Limitation of XOR Operation

Given a simple example with $K = 3$ and $T = 2$ shown as in Fig. 3.1, in which the two message bits are (I_1, I_2) and three codeword bits are $(s_1, s_2, s_3) = (I_1, I_2, O_1 = I_1 \oplus I_2)$, where $\oplus$ denotes XOR operation. Equiprobable message bits are considered at first, i.e., $I_k \in \{0, 1\}$ with $p(I_k = 0) = p(I_k = 1) = 0.5$ for $k = 1, 2$, then prior-probabilities for the codeword bits are also equiprobable. After encoding, the transmitter can transmit codeword bits $s_k \in \{0, 1\}$ using equal-spaced constellation with $d_k \in \{\pm 1\}$ for $k = 1, 2, 3$.

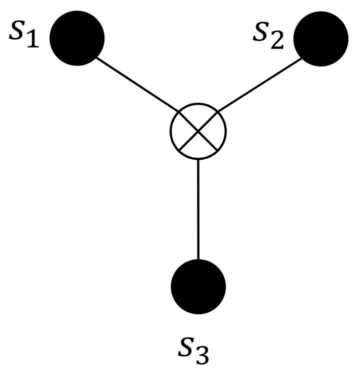


Figure 3.1: A Simple Example on Parity Check Nodes. $\otimes$ Denotes XOR.

The XOR operation has several nice properties when the inputs are equiprobable:

1. Reversible, if $s_1 \oplus s_2 = s_3$, then $s_3 \oplus s_2 = s_1$ and $s_1 \oplus s_3 = s_2$.
2. Symmetric, $s_1 \oplus s_2 = s_2 \oplus s_1$.
3. Scalable, $s_1 \oplus s_2 \oplus s_3 = (s_1 \oplus s_2) \oplus s_3$.

However, if the message bits becomes non-equiprobable, for instance, $p(s_k = 0) = 0.3$ for $k = 1, 2$, non-equiprobable codeword bits would be generated correspondingly, i.e., $p(s_1 = 0) = p(s_2 = 0) = 0.3$ and $p(s_3 = 0) = 0.58$ as demonstrated in Equation (3.1).

$$\begin{aligned} p(s_3 = 0) &= p(s_1 = 0) \times p(s_2 = 0) + p(s_1 = 1) \times p(s_2 = 1) \\ &= 0.3 \times 0.3 + 0.7 \times 0.7 \\ &= 0.58 \end{aligned} \tag{3.1}$$

Besides this issue, message passing from s_3 and s_2 to s_1 which is the output of $s_2 \oplus s_3$, will result in a mismatching in prior probability. since $p(s_2 \oplus s_3 = 0) = 0.468 \neq p(s_1 = 0) = 0.3$ as explained in Equation (3.2). Furthermore, it is difficult to apply optimal constellation to message bits and parity check bits because XOR operation is not capable of handling non-equiprobable symbols.

$$\begin{aligned} p(s_1 = 0) &= p(s_2 = 0) \times p(s_3 = 0) + p(s_2 = 1) \times p(s_3 = 1) \\ &= 0.3 \times 0.58 + 0.7 \times 0.42 \\ &= 0.468 \end{aligned} \tag{3.2}$$

The optimal constellations for message bits satisfy equations $d_k \in \{d_{k,1}, d_{k,2}\}$ and $0.3d_{k,1} + 0.7d_{k,2} = 0$, where $d_{k,1}$ denotes amplitude of rectangular pulse waveform transmitted when 0 occurs and $d_{k,2}$ when 1 occurs [98]. The optimal equation for constellations will be explained in Chapter 5. If equal spaced constellation is applied without considering the non-equal probabilities of message bits, it will cost 0.72 dB non-recoverable reduction in E_b/N_0 capacity.

If constellations are represented in two consecutive bits, four symbols can be obtained as shown in Table. 3.1.

Table 3.1: Four Symbols with Two Bits.

s_1	s_2	symbols	probability	constellations
0	0	(0, 0)	0.09	$\{d_{1,1}, d_{2,1}\}$
0	1	(0, 1)	0.21	$\{d_{1,1}, d_{2,2}\}$
1	0	(1, 0)	0.21	$\{d_{1,2}, d_{2,1}\}$
1	1	(1, 1)	0.49	$\{d_{1,2}, d_{2,2}\}$

In order to transmit symbols through channel, we need to assign the constellations to codeword bits. Conventionally, the same constellation $d_k \in \{d_{k,1}, d_{k,2}\}$ are used for codeword bits. However, this strategy is not optimal since the output of XOR has a different prior-probability than its inputs if they are non-equiprobable.

Therefore, XOR operation needs to be replaced so that the prior probability of each codeword bit matched to the one of each message bit. Two novel solutions are introduced to solve this probable:

1. quasi-XOR(QXOR) operation to replace XOR operation.
2. intermediate transformation to obtain symbols with prior probabilities suitable for XOR operation.

quasi-XOR Operation and Intermediate Transformation Layer

In this section, quasi-XOR Operation and Intermediate Transformation Layer are discussed. The first method is finding an alternative operation to replace XOR and yet can still be utilized in error control coding while the second is to add a layer between non-equiprobable symbols and XOR operation.

quasi-XOR Operation

Quasi-XOR operation is introduced as producing output with prior probabilities identical to the prior probability of input. In Fig. 3.2 3.3 3.4 3.5, Karnaugh maps are shown for an XOR operation, an operation with three inputs, QXOR operations with three inputs and four inputs, respectively.

The process of QXOR operation is described as follows. Firstly the XOR operation incorporates one additional input ,I_3, and then labels “1” for all slots with $I_3 = 1$ as shown in Fig. 3.3. Some of “1” are placed to other spots with the same Hamming weight. For example, “1” at the bottom left in Fig. 3.3 is moved to “1” at the top right in Fig. 3.4 since the hamming weight for the first spot $I_1I_2I_3 = 001$ is 1 and the hamming weight for second spot $I_1I_2I_3 = 010$ is 1 as well. By repeating this process, six cases can be obtained as shown in Fig. 3.4 and their representations are as follows:

$$O_1 = \bar{I}_2I_1 + I_2I_3 \tag{3.3}$$

$$O_2 = \bar{I}_1I_3 + I_2I_1 \tag{3.4}$$

$$O_3 = \bar{I}_3I_2 + I_1I_3 \tag{3.5}$$

$$O_4 = \bar{I}_2I_3 + I_1I_2 \tag{3.6}$$

$$O_5 = \bar{I}_1I_2 + I_1I_3 \tag{3.7}$$

$$O_6 = \bar{I}_3I_1 + I_2I_3 \tag{3.8}$$

Equation (3.3)(3.4)(3.5) form three feedback paths from O_1, O_2, and O_3 to I_1, I_2, and I_3 as follows.

$$I_1 = \bar{O}_2O_1 + O_2O_3 \tag{3.9}$$

$$I_2 = \bar{O}_1O_3 + O_1O_2 \tag{3.10}$$

$$I_3 = \bar{O}_3O_2 + O_3O_1 \tag{3.11}$$

by swapping I_1, I_2, I_3 with O_1, O_2, O_3, respectively. This operation is defined as $QXOR_3$.

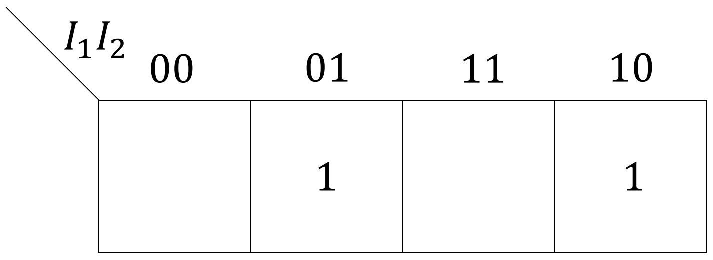


Figure 3.2: Karnaugh Map for XOR Operation.

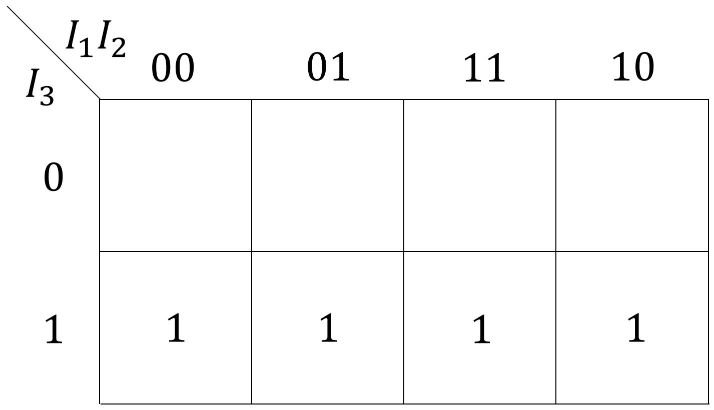


Figure 3.3: Karnaugh Map for an Operation with Three Inputs.

For four inputs, QXOR operation form as shown in Fig. 3.5.

$$O_1 = I_1I_2I_3 + I_2\bar{I}_1\bar{I}_4 + I_2I_4\bar{I}_3 + I_3I_4\bar{I}_2 \tag{3.12}$$

$$O_2 = I_1I_2I_4 + I_1\bar{I}_2\bar{I}_3 + I_3I_4\bar{I}_1 + I_1I_3\bar{I}_4 \tag{3.13}$$

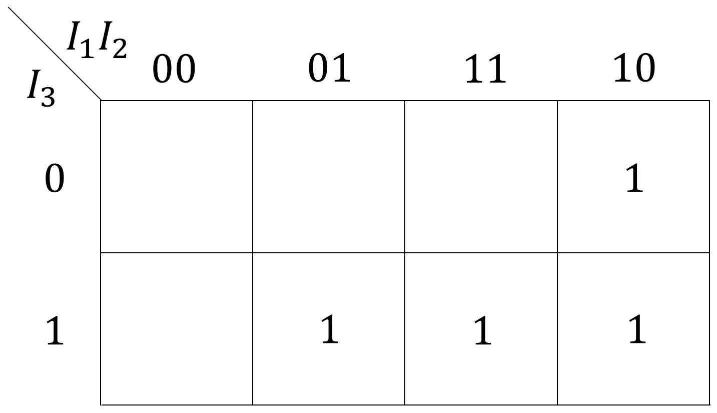


Figure 3.4: Karnaugh Map for QXOR Operation with Three Inputs.

$$O_3 = I_1 I_3 I_4 + I_3 \bar{I}_2 \bar{I}_4 + I_1 I_2 \bar{I}_3 + I_2 I_3 \bar{I}_1 \tag{3.14}$$

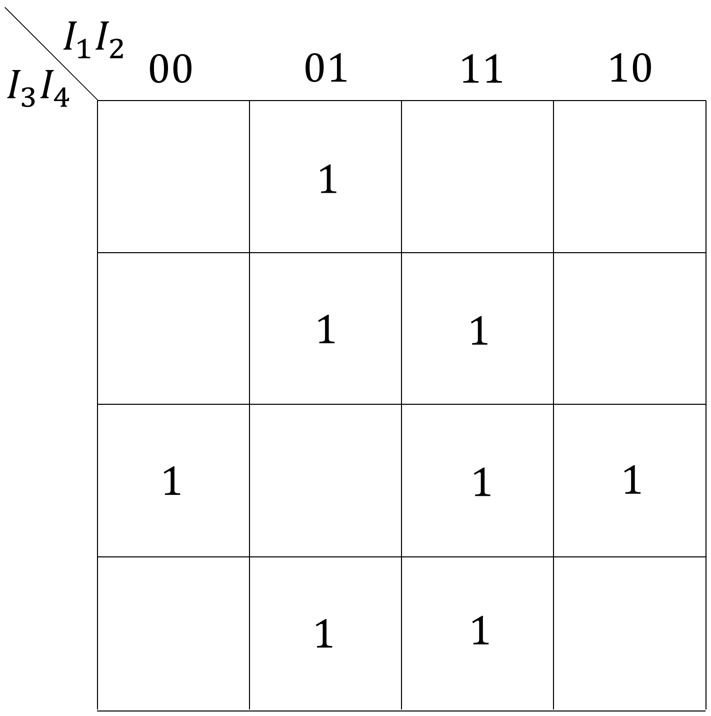
$$O_4 = I_2 I_3 I_4 + I_4 \bar{I}_1 \bar{I}_3 + I_1 I_2 \bar{I}_4 + I_1 I_4 \bar{I}_2 \tag{3.15}$$

The reversing operations can be obtained by swapping I_1, I_2, I_3, and I_4 with O_1, O_2, O_3, and O_4 respectively. And this operation is defined as $QXOR_4$.

For $QXOR_4$, there are other selections for this similar operation. Another way to operate $QXOR_4$ is listed as follows.

$$O_1 = I_4 \bar{I}_1 \bar{I}_3 + I_2 I_3 I_4 + I_1 I_2 \bar{I}_4 + I_1 I_4 \bar{I}_2 \tag{3.16}$$

$$O_2 = I_2 \bar{I}_1 \bar{I}_3 + I_1 I_2 I_4 + I_3 I_4 \bar{I}_2 + I_2 I_3 \bar{I}_4 \tag{3.17}$$



I_3I_4 \ I_1I_2	00	01	11	10
		1		
		1	1	
	1		1	1
		1	1	

Figure 3.5: Karnaugh Map for QXOR Operation with Four Inputs.

$$O_3 = I_3\bar{I}_1\bar{I}_4 + I_2I_3I_4 + I_1I_3\bar{I}_2 + I_1I_2\bar{I}_3 \tag{3.18}$$

$$O_4 = I_1\bar{I}_2\bar{I}_3 + I_1I_2I_4 + I_1I_3\bar{I}_4 + I_3I_4\bar{I}_1 \tag{3.19}$$

And reversing operations are defined as

$$I_1 = O_1O_4\bar{O}_3 + O_2O_3O_4 + O_1O_3\bar{O}_4 + O_4\bar{O}_1\bar{O}_2 \tag{3.20}$$

$$I_2 = O_2\bar{O}_3\bar{O}_4 + O_1O_2O_4 + O_2O_3\bar{O}_1 + O_1O_3\bar{O}_2 \tag{3.21}$$

$$I_3 = O_3\bar{O}_1\bar{O}_2 + O_1O_3O_4 + O_2O_3\bar{O}_4 + O_2O_4\bar{O}_3 \tag{3.22}$$

$$I_4 = O_1\bar{O}_3\bar{O}_4 + O_1O_2O_3 + O_2O_4\bar{O}_1 + O_1O_4\bar{O}_2 \tag{3.23}$$

By combining $QXOR_3$ and $QXOR_4$ operations, operations with any number (≥ 3) of inputs are able to be obtained.

Intermediate Transformation Layer

In this Section, how prior probability of inputs and output of an XOR operation is investigated. XOR operation with Φ number of inputs is defined as Φ-XOR of which form is defined in Equation (3.24).

$$I_1 \oplus I_2 \oplus I_3 \oplus, \cdots, \oplus I_\Phi \tag{3.24}$$

Define the prior probability of symbol I_ϕ as p_ϕ, then the prior probability of output symbol, denoted as $p_o(\Phi)$, can be computed recursively as follows. Since $p_o(2) = p_1p_2 + (1-p_1)(1-p_2)$, for $\phi = 3, \cdots, \Phi$,

$$p_o(\phi) = p_\phi p_o(\phi - 1) + (1 - p_\phi)(1 - p_o(\phi - 1)) \tag{3.25}$$

and finally, $p_o(\Phi)$ can be obtained by recursively computing XOR operation with adding one more input. $p_o(\Phi)$ is plotted as a function of p_ϕ for Φ-XOR in Fig. 3.6. In order to generate symbols with all possible prior probabilities between 0 and 1 for the XOR gate, p_ϕ of inputs needs to be small, i.e. $p_\phi << 0.5$. That is, the probability of satisfying parity check constraint must be small. If NOT gates are available in designing, then inputs with probabilities $p_\phi >> 0.5$ can produce desired output. In either case, the case of $p_\phi = 0.5$ should always be avoided since XOR operation will only generate equiprobable symbols when the inputs are equiprobable. Another observation is that the value of $p_o(\Phi)$ approaches the value of 0.5 over a large range when the value of p_ϕ is around of 0.5. The larger the value of Φ is, the broader this range becomes. From Equation (3.25), the following results are obtained.

$$p_o(\phi) = \begin{cases} p_\phi + (1-2p_\phi)(1-p_o(\phi-1)) \geq p_\phi & \text{if } p_\phi < 0.5 \\ p_\phi - (2p_\phi - 1)(1-p_o(\phi-1)) \leq p_\phi & \text{if } p_\phi > 0.5 \\ 0.5 & \text{if } p_\phi = 0.5 \end{cases} \tag{3.26}$$

for any $\phi \in (1, 2, \cdots, \Phi)$.

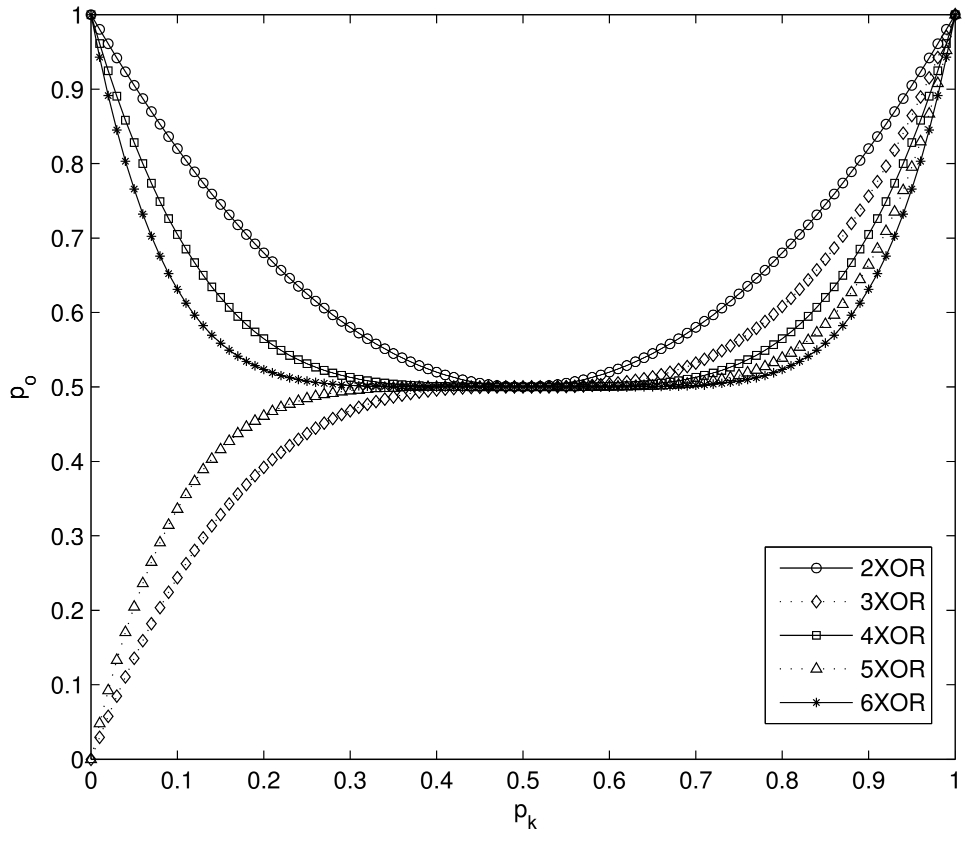


Figure 3.6: Probabilities of Output Symbols Versus Input Symbols

In order to impose code constraint, the prior probabilities of message bits and parity check bits must match with each other. For example, if $I_1 \oplus I_2 \oplus I_3 = I_4$, then $p_o(3)$ of $I_1 \oplus I_2 \oplus I_3$ must match up with p_4 of I_4.

There are three different types of intermediate transformation layer (ITL) to deal with non-equiprobable symbols. The prior probability of output from these layers are termed as p_o.

1. Front ITL (F-ITL) as shown in Fig. 3.7, applies ITL followed by Φ-XOR, which produces output symbols with desired prior-probability directly.

2. End ITL (E-ITL) as shown in Fig. 3.8, applies Φ-XOR operations to non-equiprobable symbols with probability of p_k, then uses the transformation layer to match p_o of the output symbol with p_k of the input symbols.

3. Half ITL (H-ITL) as shown in Fig. 3.9, converts non-equiprobable symbols to equiprobable symbols, then uses Φ-XOR, and finally converts equiprobable symbols back to non-equiprobable symbols.

Front-Intermediate Transformation Layer

F-ITL is demonstrated using an example of $p_k = 0.3$, or simply denoted as p since the prior probability for each bit is indistinct. If two inputs are combined together as a group, then there are three basic probabilities, $p^2 = 0.09$ for $I_1 = 0$ and $I_2 = 0$, $p(1-p) = 0.21$, and $(1-p)^2 = 0.49$. For each two inputs $I_{2\kappa+1}$ and $I_{2\kappa+2}$, where subscript κ denote the κ^{th} group of two inputs, F-ITL can be generalized to three outputs.

$$p(O_{\kappa,1} = 0) = p(I_{2\kappa+1} = 0 \bigcap I_{2\kappa+2} = 0) = 0.09 \tag{3.27}$$

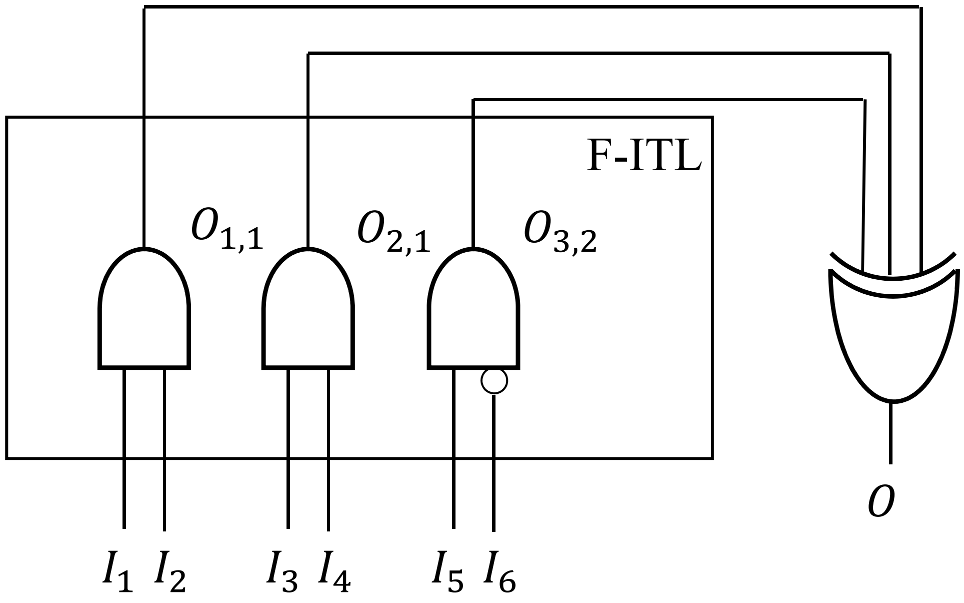


Figure 3.7: Structure for F-ITL, where $p_1 = p_2 = p_3 = p_4 = p_5 = p_6 = 0.3, p_(O_{1,1} = 0) = 0.09, p_(O_{2,1} = 0) = 0.09, p_(O_{3,2} = 0) = 0.21, p(O = 0) = 0.305$

$$p(O_{\kappa,2} = 0) = p(I_{2\kappa+1} = 0 \bigcap I_{2\kappa+2} = 1) = 0.21 \tag{3.28}$$

$$p(O_{\kappa,3} = 0) = p(I_{2\kappa+1} = 1 \bigcap I_{2\kappa+2} = 0) = 0.21 \tag{3.29}$$

where $O_{\kappa,1} = I_{2\kappa+1} + I_{2\kappa+2}$, $O_{\kappa,2} = I_{2\kappa+1} + \bar{I}_{2\kappa+2}$, and $O_{\kappa,3} = \bar{I}_{2\kappa+1} + I_{2\kappa+2}$. Applying 3-XOR with $O_{\kappa,1}$ where $p(O_{\kappa,1} = 0) = 0.09$ from the κ^{th} group, $O_{\kappa+1,1}$ where $p(O_{\kappa+1,1} = 0) = 0.09$ from the $\kappa + 1^{th}$ group and $O_{\kappa+2,2}$ where $p(O_{\kappa+2,2} = 0) = 0.21$ from the $\kappa + 2^{th}$ group will produce an output symbol with $p_o = 0.305$, which approximates $p = 0.3$ as shown in Fig. 3.7.

Combining three inputs as a group, four basic probabilities can be achieved $p^3 = 0.027, p^2(1-p) = 0.063$, $p(1-p)^2 = 0.147$, and $(1-p)^3 = 0.343$. Seven outputs can be created through these probabilities,

$$O_{\kappa,1} = I_{2\kappa+1} + I_{2\kappa+2} + I_{2\kappa+3} \tag{3.30}$$

$$O_{\kappa,2} = \bar{I}_{2\kappa+1} + I_{2\kappa+2} + I_{2\kappa+3} \tag{3.31}$$

$$O_{\kappa,3} = I_{2\kappa+1} + \bar{I}_{2\kappa+2} + I_{2\kappa+3} \tag{3.32}$$

$$O_{\kappa,4} = I_{2\kappa+1} + I_{2\kappa+2} + \bar{I}_{2\kappa+3} \tag{3.33}$$

$$O_{\kappa,5} = \bar{I}_{2\kappa+1} + \bar{I}_{2\kappa+2} + I_{2\kappa+3} \tag{3.34}$$

$$O_{\kappa,6} = \bar{I}_{2\kappa+1} + I_{2\kappa+2} + \bar{I}_{2\kappa+3} \tag{3.35}$$

$$O_{\kappa,7} = I_{2\kappa+1} + \bar{I}_{2\kappa+2} + \bar{I}_{2\kappa+3} \tag{3.36}$$

Let

$$OO_1 = O_{\kappa,1}O_{\kappa+1,1}O_{\kappa+2,1}O_{\kappa+3,1}O_{\kappa+4,1} \tag{3.37}$$

$$OO_2 = O_{\kappa,2}O_{\kappa+1,2} \tag{3.38}$$

$$OO_3 = O_{\kappa,5} \tag{3.39}$$

Applying 3-XOR with OO_1, OO_1 and OO_2, or OO_2, OO_2, and OO_3, can produce an output symbol with $p_o = 0.291$ or $p_o = 0.299$, respectively. If we define

$$OO_1 = O_{\kappa,1}O_{\kappa+1,1}O_{\kappa+2,1} \tag{3.40}$$

$$OO_2 = O_{\kappa,2} \tag{3.41}$$

$$OO_3 = O_{\kappa,1}O_{\kappa+1,5} \tag{3.42}$$

then applying 4-XOR with OO_1, OO_1, OO_2, and OO_3 will produce an output symbol with $p_o = 0.705$ which can be inverted and creates a symbol with prior probability of 0.2954.

The basic idea of F-ITL is to create symbols with small probabilities first then apply XOR operations with these symbols which will produce symbols approximately matching to the same prior probability as the inputs have.

End-Intermediate Transformation Layer

E-ITL is demonstrated using the same example with $p = 0.3$. According to Fig. 3.6, the prior-probability of output of Φ-XOR is $p_o \approx 0.5$ when $\Phi \geq 4$. Let Os denote output symbols. Define

$$OO_\kappa(\phi) = O_\kappa + O_{\kappa+1} + \cdots + O_{\kappa+\phi} \tag{3.43}$$

where κ denotes the κ-th group of combination of Os, and the prior probability of $OO_\kappa(\phi)$ is $2^{-(\phi+1)}$, denoted as $p_{OO}(\phi)$. It is not difficult to get $p_{OO}(1) + p_{OO}(4) + p_{OO}(6) = 0.296$. Combining $OO_\kappa(2)OO_{\kappa+3}(5)OO_{\kappa+9}(6)$ gives an output with prior probability of $p = 0.296$. From the described process, E-ITL is comparatively simpler than F-ITL.

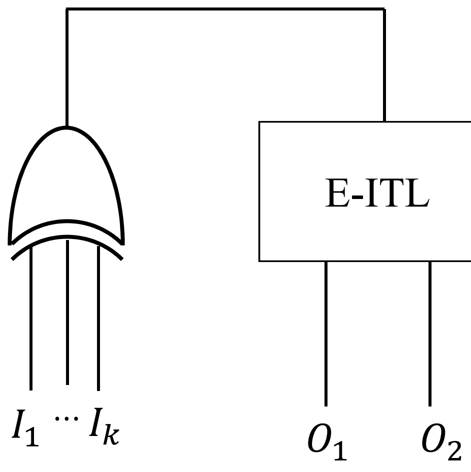


Figure 3.8: Structure for E-ITL, where $p_1 = \cdots = p_k = p_{O_1} = p_{O_2}$

Half-Intermediate Transformation Layer

The structure for H-ITL is using Φ-XOR to construct a front-layer which converts non-equiprobable symbols to equiprobable symbols; then developing an end-layer similar to E-ITL to convert equiprobable symbols back to non-equiprobable ones. The structure for H-ITL is defined in Fig. 3.9.

Code Construction and Probabilistic Messages for Non-equiprobable Symbols

In this section, the detailed process of constructing codes using QXOR is discussed. These codes are neither source codes nor error control codes. Their applications are beyond conventional source and channel coding in modern telecommunication systems and may perform functionalities similar to brain functionalities in GPRAM [22] [23].

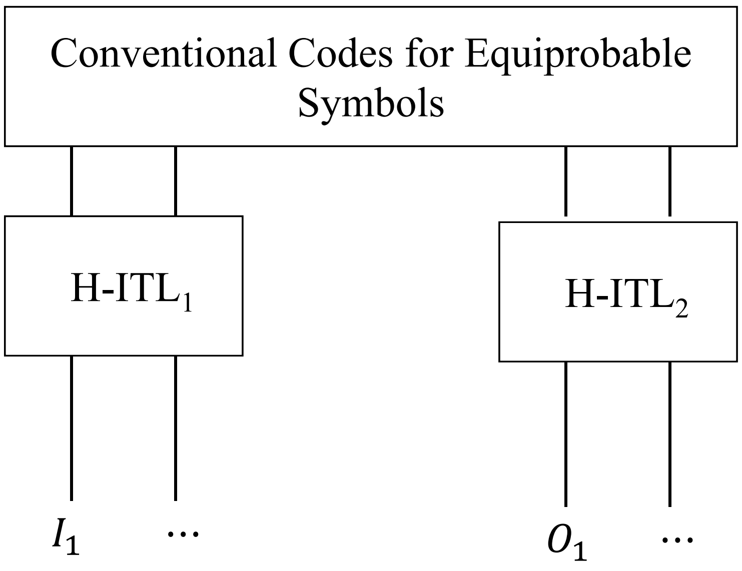


Figure 3.9: Structure for H-ITL

Tree Codes with QXOR

Two codes with $QXOR$ operations are developed as shown in Tables 3.3 and 3.5, while two other codes with XOR operations are shown in Tables 3.2 and 3.4.

Table 3.2: Codes based on XOR_2

I_1	I_2	I_3	O_1	O_2	O_3
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	1	0	1
0	1	1	1	1	0
1	0	0	1	1	0
1	0	1	1	0	1
1	1	0	0	1	1
1	1	1	0	0	0

Table 3.3: Codes based on $QXOR_3$

I_1	I_2	I_3	O_1	O_2	O_3
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	0	1	1
1	1	1	1	1	1

Table 3.4: Codes based on XOR_3

I_1	I_2	I_3	I_4	O_1	O_2	O_3	O_4
0	0	0	0	0	0	0	0
0	0	0	1	0	1	1	1
0	0	1	0	1	1	0	1
0	0	1	1	1	0	1	0
0	1	0	0	1	0	1	1
0	1	0	1	1	1	0	0
0	1	1	0	0	1	1	0
0	1	1	1	0	0	0	1
1	0	0	0	1	1	1	0
1	0	0	1	1	0	0	1
1	0	1	0	0	0	1	1
1	0	1	1	0	1	0	0
1	1	0	0	0	1	0	1
1	1	0	1	0	0	1	0
1	1	1	0	1	0	0	0
1	1	1	1	1	1	1	1

Codes in Table 3.2 are constructed as follows. Define

$$O_1 = I_1 \oplus I_2 = \bar{I}_2 I_1 + I_2 \bar{I}_1 \tag{3.44}$$

Table 3.5: Codes based on $QXOR_4$

I_1	I_2	I_3	I_4	O_1	O_2	O_3	O_4
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	1	1	0	0
0	1	0	0	1	0	0	0
0	1	0	1	1	0	0	1
0	1	1	0	1	0	1	0
0	1	1	1	0	1	1	1
1	0	0	0	0	1	0	0
1	0	0	1	0	1	0	1
1	0	1	0	0	1	1	0
1	0	1	1	1	0	1	1
1	1	0	0	0	0	1	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	0	1
1	1	1	1	1	1	1	1

$$O_2 = I_1 \oplus I_3 \tag{3.45}$$

$$O_3 = I_2 \oplus I_3 \tag{3.46}$$

While in Table 3.4, define

$$O_1 = I_1 \oplus I_2 \oplus I_3 \tag{3.47}$$

$$O_2 = I_1 \oplus I_3 \oplus I_4 \tag{3.48}$$

$$O_3 = I_1 \oplus I_2 \oplus I_4 \tag{3.49}$$

$$O_4 = I_2 \oplus I_3 \oplus I_4 \tag{3.50}$$

Both codes are linear with Hamming distances of 3 and 4 for Codes in Table 3.2, and 4 and 7 for Codes in Table 3.4, respectively. Despite the code constraints illustrated in prior equations, both codes introduce new constraints. For example, $O_1 \oplus O_2 \oplus O_3 = 0$ for Codes in Table 3.2 and $I_1 = O_1 \oplus O_2 \oplus O_3$ for Codes Table 3.4. These constraints form mesh networks when more inputs are incorporated. And the number of code constraints grows exponentially.

Codes in Table 3.3 and 3.5 are constructed with $QXOR_3$ and $QXOR_4$ respectively. Both codes are non-linear codes with Hamming distance (2,4,6) and (2,4,6,8) respectively. The property of Hamming distances makes both codes similar to repetition codes. However, the difference is that input symbols in the repetition codes are independent of each other, while input symbols in Codes in Table 3.3 and 3.5 are dependent on each other through constraints of Os.

Property 1. *The minimum Hamming distance of codes based on $QXOR_\Psi$ is no greater than 2 and the multiplicity is Ψ.*

Proof. From the construction process, Codes based on $QXOR_\Psi$ contain two parts Is and Os, as shown in Table 3.3 and 3.5. Since Hamming weight of a codeword is the number of 1 in that codeword. Hence Hamming weight of each codeword is the sum of Hamming weights in Is and Os. Each symbol of Ψ O bits with $O = 1$ in codes based on $QXOR_\Psi$ must not contain more than one codeword with Hamming weight of 1 in Is to enforce the the same prior probabilities of Is and Os.

For example, the only one combination has Hamming weight of 1, $I_1I_2I_3I_4 = 0100$ in Fig. 3.5 where $O_1 = 1$. There are Ψ codewords with Hamming weight of 1 in Is in total. If one of such codewords has a Hamming weight of 2 or more in Os, then there must be at least one codeword with Hamming weight of 0 in Os in these Ψ codewords. Consequently, the minimum Hamming weight is 1. If each of Ψ codewords has a Hamming weight of 1 in Os, then we have codes with minimum Hamming distance of 2 and the multiplicity is Ψ. $\square$

This proposition rules out the possibility of creating a decent error control code using $QXOR$ operation, since it produces codes with neither large minimum Hamming distance nor a favorable distribution of multiplicities.

Summary

In this chapter, we presented critical challenges in designing codes and iterative decoding for non-equiprobable symbols. We demonstrated the limitations of XOR operation in dealing with non-equiprobable symbols. The limitation of XOR operation can be potentially resolved by applying quasi-XOR operation and intermediate transformation layer. Besides We showed how to construct codes for non-equiprobable symbols using quasi-XOR operation and the codes cannot satisfyingly perform with regarding to error correction capability.

CHAPTER 4: NONLINEAR CODES WITH EXPANDED OPERATIONS

Logic and XOR operations have played essential roles in modern computer and telecommunications system design. The modern computer was developed in the 1940s to meet our need to perform computational tasks which our brains are not capable of performing with precision and speed. Visionary thoughts from Turing [19] and von Neumann [21] laid the foundation in terms of mathematical principle and engineering structure. Well-known Turning machines demonstrated in mathematics that face many problems in our life can be reduced to computational problems. During the same period, Shannon laid the foundation of the mathematical measurement of information [87]. Over the last sixty years, telecommunication engineers have learned how to build a computer and achieved the Shannon limit using coding and detection theory. During the process, we have accumulated a wealth of knowledge on how to preserve, transmit, recover and detect information.

These classic works do not include the complexity, one of key issues in our engineering activities. Even though many tasks can be completed by alternative computation in Turing sense, the complexity advantage of some non-optimal schemes should never be ignored. One of the biggest impact of Turbo invention [26] was to demonstrate that some well structured sub-optimal processing can not only achieve near Shannon-limit decoding, but also require very low computational complexity. After that, the soft-in soft-out idea embedded in SOVA [99] has swept over the entire error control coding field over the last 18 years.

Could the Turbo invention also be utilized beyond error control coding fields? Over the last sixteen years, we have been searching for its implication in understanding human brains. These lead us to our GPRAM theory [22] [23] and we aim to develop a prototype using error control coding in [68]. One of key functions in our GPRAM machine is constantly searching for simple and better approximation for any given task. For example, GPRAM system uses a group of operations to deal with a task such as approximating a function with three inputs. If in the future, it discovers a new area with a smaller group of operations which can get a better approximation, then it will abandon the old area and migrate to the new area.

To explain this concept better, a problem is given under the assumption that we want to build a machine which can approximate any given functions with an unknown number of inputs. If the desired function is known, then the design of this system can be optimized by using mathematical tools. On the contrast, if the form of this function are in oblivion and we rather have a vague idea of the function, then we can design a set of typical functions which are often called basis functions and hopefully one or combinations of them can handle the approximation and future tasks.

In the bio-systems, it has been well-known that over-complete basis functions have been used to achieve sparse coding [95]. Almost all error control coding theory known so far has been built on XOR operation [71]. However, it has been found that even the conventional XOR operation does not fit well when applying onto non-equiprobable symbols as explained in Chapter 3. In this Chapter, XOR operation will be expanded to other logic operations such as AND and OR. In future, these operations will be applied to GPRAM systems to use over-complete basis to simplify the individual computational complexity.

Modification of (7,4) Hamming code using AND and OR Gates

Nonlinear codes generated from various modifications of (7,4) Hamming codes using AND, OR and XOR gates are studied. How to encode these codes and code distance property are investigated as well.

Fig. 4.1 illustrates the Tanner graph of (7,4) Hamming code. For equiprobable symbols, the Tanner graph does not define inputs and outputs, since each XOR operation with Φ edges is equivalent to Φ XOR operation units and each edge is an output. However, for non-equiprobable symbols, inputs and outputs have to be defined in each operation due to mismatch in prior probabilities (see Chapter 3). Furthermore, by replacing XOR operation by other logic operations such as AND or OR, we can construct many codes as shown in Table 4.1 and Fig. 4.2.

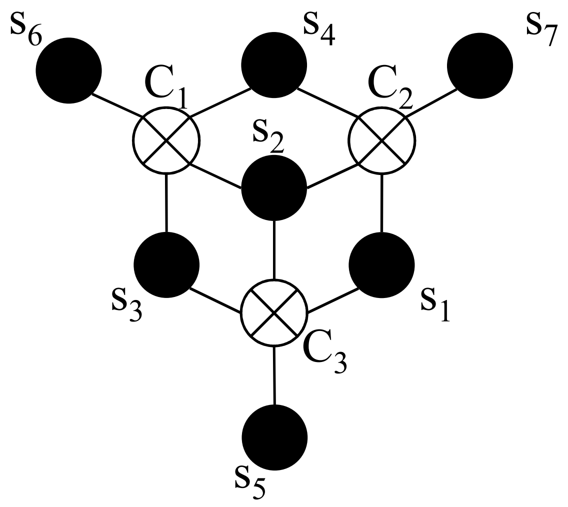


Figure 4.1: (7,4) Hamming Code

When different gates are used, directions of information must be taken into consideration. This is because if s_1 AND s_2 equals to s_3, it is not possible to get s_1 AND s_3 equals to s_2 or s_2 AND s_3 equals to s_1. In this chapter, only two directions are considered: outflow and top-down cases. In the outflow case, s_1, s_2, s_3, and s_4 are inputs (i.e., message bits) to all three C operations and s_5, s_6, and s_7 are outputs (i.e., parity check bits). While in the top-down case, s_2, s_4, s_6, and s_7 are inputs and s_1, s_3, and s_5 are outputs.

Encoder for these codes are straightforward. Taking 16 possible combinations of input bits, all code words can be created for several codes listed in Table 4.2. Readers can easily generate codewords for the remaining codes. Codes 3, 4, 5, 7, 8, and 9 are nonlinear and with minimum distance of 1.

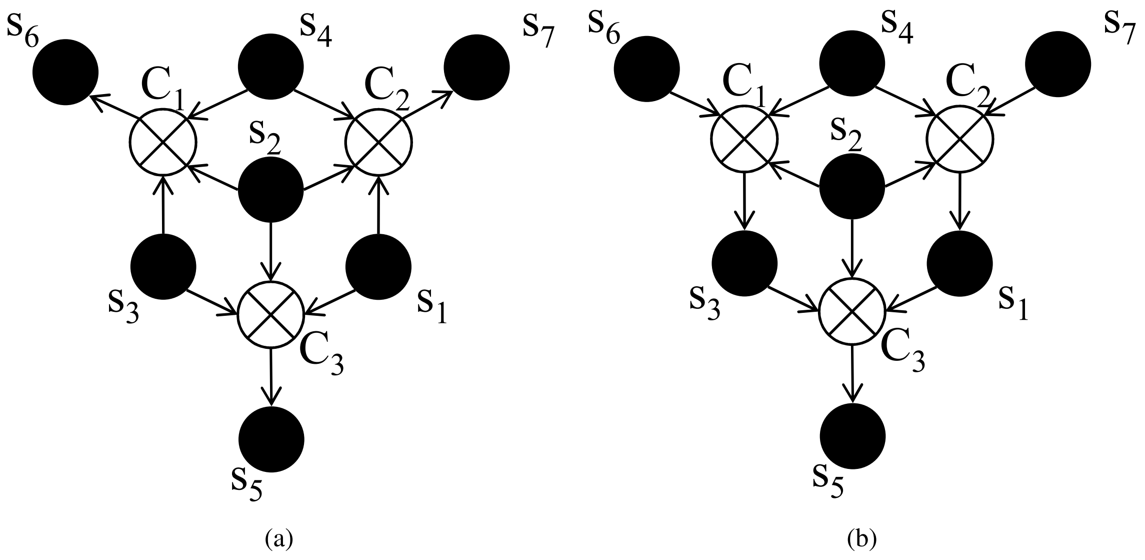


Figure 4.2: (7,4) Hamming Code with (a) Outflow (b) Top-down Directions

Table 4.1: 9 Different Codes Cases with Different Gates and Different Directions

Code	C_1	C_2	C_3	direction
1	XOR	XOR	XOR	-
2	XOR	XOR	XOR	outflow
3	XOR	AND	OR	outflow
4	AND	AND	AND	outflow
5	OR	OR	OR	outflow
6	XOR	XOR	XOR	top-down
7	XOR	AND	OR	top-down
8	AND	AND	AND	top-down
9	OR	OR	OR	top-down

Even through codes 1, 2, and 6 are identical, the decoder processing could be different, since for codes 2 and 6, the code constraints are defined by only three gates, while for code 1 the constraints are given by 12 gates. It is also worth mentioning that for non-equiprobable symbols the constraints in code 1 can not be satisfied simultaneously. The iterative decoding processing for decoder will be discussed in next Section.

Furthermore, prior probabilities for each node are different. If $p(s_k = 0) = p = 0.3$, for $k = 1, \cdots, 4$, then $p(s_k = 0) = 0.468, 0.657$, or 0.027 for XOR, AND, and OR gates, respectively, where $k = 5, 6, 7$. In Table 4.3 and 4.4 prior probabilities for codes 1 to 6 with equiprobable and non-equiprobable symbols are displayed. Prior probabilities for codes 7 to 9 can be computed based on code structure and logic gates. For non-equiprobable symbols, even through codewords for code 1, 2, and 6 are identical, the prior probabilities are very different.

Table 4.2: Codewords of Various Codes Using Tanner Graph in Fig. 4.1

1/2/6	3	4	5	7	8
0000000	0000000	0000000	0000000	0000000	0000000
0001011	0001010	0001000	0001011	0000001	0000001
0010110	0010110	0010000	0010110	0010110	0000010
0011101	0011100	0011000	0011111	0010111	0000011
0100111	0100110	0100000	0100111	0011100	0001000
0101100	0101101	0101000	0101111	0011101	0001001
0110001	0110100	0110000	0110111	0001010	0001010
0111010	0111110	0111010	0111111	0001011	0001011
1000101	1000100	1000000	1000101	0110100	0100000
1001110	1001110	1001000	1001111	0110101	0100001
1010011	1010110	1010000	1010111	0100110	0100010
1011000	1011100	1011000	1011111	0100111	0100011
1100010	1100110	1100000	1100111	0101100	0101000
1101001	1101101	1101001	1101111	1101101	1101001
1110100	1110100	1110100	1110111	0111110	0111010
1111111	1111111	1111111	1111111	1111111	1111111

Code 1 is the conventional linear (7,4) Hamming code for equiprobable symbols. Each edge does not have direction. It is equivalent to 12 XOR gates and each with 3 inputs and 1 output. For the rest of 8 codes, there are only three operations each with three inputs and one output. So, it would not be surprised that the remaining 8 codes are far weaker in term of error correction capability than code 1.

A comparison study using the (7,4,2) Hamming code of which structure can be found in Fig. 4.3 is also conducted. Because there is no loop in this structure, error propagation effect in the original (7,4) Hamming code can be estimated. Using Table 4.1 and the graph in Fig. 4.3, we can quickly establish 9 new codes based on (7,4,2) code as shown in Table 4.5 and prior probabilities in Table 4.6.

Table 4.3: Prior Probabilities for Codes with Tanner Graph in Fig. 4.1 with Equiprobable Symbols

Prior Probabilities	code 1	2	3	4	5	6	7	8	9
$p(s_1 = 0)$	0.5	0.5	0.5	0.5	0.5	0.5	0.875	0.875	0.125
$p(s_2 = 0)$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$p(s_3 = 0)$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.875	0.125
$p(s_4 = 0)$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$p(s_5 = 0)$	0.5	0.5	0.125	0.875	0.125	0.5	0.219	0.992	0.008
$p(s_6 = 0)$	0.5	0.5	0.5	0.875	0.125	0.5	0.5	0.5	0.5
$p(s_7 = 0)$	0.5	0.5	0.875	0.875	0.125	0.5	0.5	0.5	0.5

Table 4.4: Prior Probabilities for Codes with Tanner Graph in Fig. 4.1 with Non-Equiprobable Symbols

Prior Probabilities	code 1	2	3	4	5	6	7	8	9
$p(s_1 = 0)$	-	0.3	0.3	0.3	0.3	0.468	0.657	0.657	0.027
$p(s_2 = 0)$	-	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
$p(s_3 = 0)$	-	0.3	0.3	0.3	0.3	0.468	0.468	0.657	0.027
$p(s_4 = 0)$	-	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
$p(s_5 = 0)$	-	0.468	0.027	0.657	0.027	0.499	0.092	0.918	0.0002
$p(s_6 = 0)$	-	0.468	0.468	0.657	0.027	0.3	0.3	0.3	0.3
$p(s_7 = 0)$	-	0.468	0.657	0.657	0.027	0.3	0.3	0.3	0.3

Decoding of Nonlinear Codes using Iterative Decoder

To decode these nonlinear codes using iterative decoding, there are two aspects need to be considered:

1. how to map non-equiprobable symbols into signal amplitude;
2. how to update probabilistic measurement.

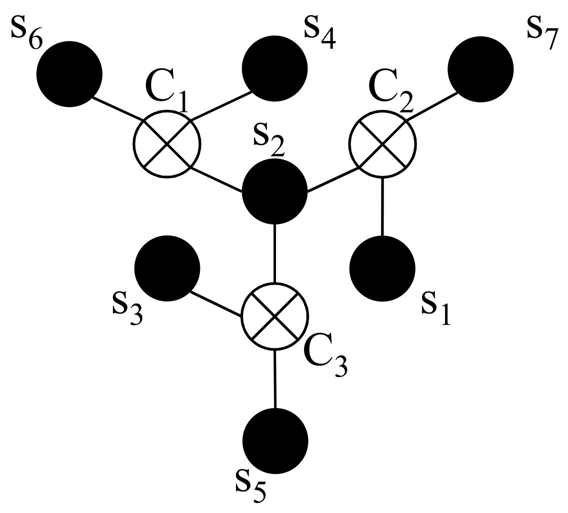


Figure 4.3: Tanner Graph for (7,4,2) Hamming Code

The first aspect will be discussed in details in Chapter 5.

For equiprobable symbols, we always map to equal amplitude with polar sign, i.e., ± 1. However, for non-equiprobable symbols, the optimal signal constellation is $r_k \in \{d_1, d_2\}$, where $pd_1 + (1-p)d_2 = 0$ as explained in Chapter 5. In this Chapter, both constellation methods are considered for the purpose of comparison. It is worth mentioning for optimal constellation, the decision threshold is not zero anymore. The threshold is defined by

$$\text{Threshold} = \frac{\sigma_1^2}{d_2 - d_1} \ln \frac{1-p}{p} + \frac{1}{2}(d_2 + d_1) \tag{4.1}$$

Assume a noise corrupted signal $b_k = r_k + n_k$, where n_k is a white Gaussian variable with mean of 0 and variance of σ_1^2. A case with three inputs and one output will be considered and explained in details and generalized later. s_1, s_2 and s_3 are inputs while s_4 is output.

Table 4.5: Codewords of Various Codes Using Tanner Graph in Fig. 4.3

1/2/6	3	4	5	7	8
0000000	0000000	0000000	0000000	0000000	0000000
0001010	0001010	0001000	0001010	0000001	0000001
0010100	0010100	0010000	0010100	0100110	0000010
0011110	0011110	0011000	0011110	1100111	0000011
0100111	0100110	0100000	0100111	0101100	0001000
0101101	0101100	0101010	0101111	1101101	0001001
0110011	0110110	0110100	0110111	0001010	0101010
0111001	0111100	0111110	0111111	0001011	1101011
1000001	1000000	1000000	1000001	0010100	0010000
1001011	1001010	1001000	1001011	0010101	0010001
1010101	1010100	1010000	1010101	0110110	0010010
1011111	1011110	1011000	1011111	1110111	0010011
1100110	1100111	1100001	1100111	0111100	0011000
1101100	1101101	1101011	1101111	1111101	0011001
1110010	1110111	1110101	1110111	0011110	0111110
1111000	1111101	1111111	1111111	0011111	1111111

Table 4.6: Prior Probabilities for Codes with Tanner Graph in Fig. 4.3 with Equiprobable Symbols

Prior Probabilities	code 1	2	3	4	5	6	7	8	9
$p(s_1 = 0)$	0.5	0.5	0.5	0.5	0.5	0.5	0.75	0.875	0.125
$p(s_2 = 0)$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.75	0.25
$p(s_3 = 0)$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$p(s_4 = 0)$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$p(s_5 = 0)$	0.5	0.5	0.25	0.75	0.25	0.5	0.25	0.875	0.125
$p(s_6 = 0)$	0.5	0.5	0.5	0.75	0.25	0.5	0.5	0.5	0.5
$p(s_7 = 0)$	0.5	0.5	0.75	0.75	0.25	0.5	0.5	0.5	0.5

Message Passing in AND Logic Gate

In this case, nodes are connected through an AND gate, Table 4.8 explains the truth table for an AND gate with three inputs and one output.

Messages are propagating from three other nodes to the left node, and the direction of message must be taken in consideration. The message passing from s_1, s_2, s_3 to s_4 can be proved.

Table 4.7: Prior Probabilities for Codes with Tanner Graph in Fig. 4.3 with Non-Equiprobable Symbols

Prior Probabilities	code 1	2	3	4	5	6	7	8	9
$p(s_1 = 0)$	-	0.3	0.3	0.3	0.3	0.5	0.649	0.657	0.027
$p(s_2 = 0)$	-	0.3	0.3	0.3	0.3	0.499	0.499	0.51	0.09
$p(s_3 = 0)$	-	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
$p(s_4 = 0)$	-	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
$p(s_5 = 0)$	-	0.499	0.51	0.51	0.09	0.5	0.15	0.657	0.027
$p(s_6 = 0)$	-	0.499	0.499	0.51	0.09	0.3	0.3	0.3	0.3
$p(s_7 = 0)$	-	0.499	0.09	0.51	0.09	0.3	0.3	0.3	0.3

Table 4.8: Truth Table with Three Inputs and One Output for AND gate

s_1	s_2	s_3	s_4
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Message From Inputs to Output

Define two sets as the possible values for r_1, r_2, r_3, r_4

$$\Gamma_1 = \left\{ \begin{array}{cccc} \{d_1, & d_1, & d_1, & d_1\} \\ \{d_1, & d_1, & d_2, & d_1\} \\ \{d_1, & d_2, & d_1, & d_1\} \\ \{d_1, & d_2, & d_2, & d_1\} \\ \{d_2, & d_1, & d_1, & d_1\} \\ \{d_2, & d_1, & d_2, & d_1\} \\ \{d_2, & d_2, & d_1, & d_1\} \end{array} \right\} \tag{4.2}$$

$$\Gamma_2 = \left\{ \begin{array}{cccc} \{d_2, & d_2, & d_2, & d_2\} \end{array} \right\} \tag{4.3}$$

$$\begin{aligned} \frac{p(r_4 = d_1|b_1b_2b_3b_4)}{p(r_4 = d_2|b_1b_2b_3b_4)} &= \frac{p(b_1b_2b_3b_4|r_4 = d_1)}{p(b_1b_2b_3b_4|r_4 = d_2)} \frac{p(r_4 = d_1)}{p(r_4 = d_2)} \\ &= \frac{p(r_4 = d_1)}{p(r_4 = d_2)} \frac{p(b_4|r_4 = d_1)}{p(b_4|r_4 = d_2)} \frac{\sum_{\{r_1,r_2,r_3\}\in\Gamma_1} p(b_1b_2b_3r_1r_2r_3|r_4 = d_1)}{\sum_{\{r_1,r_2,r_3\}\in\Gamma_2} p(b_1b_2b_3r_1r_2r_3|r_4 = d_2)} \\ &= \frac{p(b_4|r_4 = d_1)}{p(b_4|r_4 = d_2)} e^{U_4} \end{aligned} \tag{4.4}$$

Define

$$v_k^{(0)} = \ln\left(\frac{p(r_k = d_1|b_k)}{p(r_k = d_2|b_k)}\right) \tag{4.5}$$

where $k = 1, 2, 3, 4$.

$$\begin{aligned} e^{U_4} &= \frac{\text{all the cases when } r_4 = d_1}{\text{the case when } r_4 = d_2} \\ &= e^{v_1^{(0)}} e^{v_2^{(0)}} e^{v_3^{(0)}} + e^{v_2^{(0)}} e^{v_3^{(0)}} + e^{v_1^{(0)}} e^{v_2^{(0)}} + e^{v_1^{(0)}} e^{v_3^{(0)}} + e^{v_1^{(0)}} + e^{v_2^{(0)}} + e^{v_3^{(0)}} \\ &= (e^{v_1^{(0)}} + 1)(e^{v_2^{(0)}} + 1)(e^{v_3^{(0)}} + 1) - 1 \end{aligned}$$

This case is generalized for $d_c - 1$ inputs and 1 output as shown in Fig. 4.4.

Define

$$v_k^{(0)} = \ln \left(\frac{p(r_k = d_1 | b_k)}{p(r_k = d_2 | b_k)} \right) \tag{4.6}$$

$$R_k = \ln \left(\frac{p(b_k | r_k = d_1)}{p(b_k | r_k = d_2)} \right) \tag{4.7}$$

$$U_k = \ln \left(\frac{p(r_k = d_1 | \bigcap_{\substack{q=1 \\ q \neq k}}^{d_c} b_q)}{p(r_k = d_2 | \bigcap_{\substack{q=1 \\ q \neq k}}^{d_c} b_q)} \right) \tag{4.8}$$

where $k = 1, \cdots, d_c - 1$ denotes $d_c - 1$ inputs and $k = d_c$ denotes the output.

If the logic gate is AND gate, it can be shown that the message passing from input nodes to the only output node can be computed as

$$U_{d_c} = \ln \left(\prod_{k=1}^{d_c - 1} (\exp(v_k^{(0)}) + 1) - 1 \right) \tag{4.9}$$

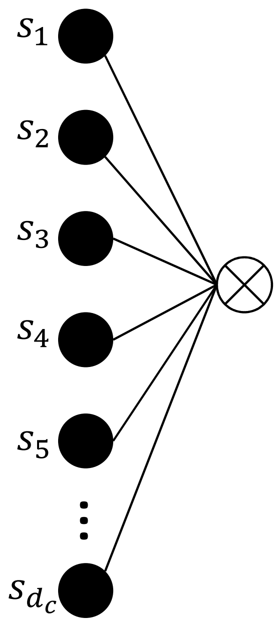


Figure 4.4: Tanner Graph for $d_c - 1$ Inputs and One Output

Message From Other Inputs and Output to an Input

While to find the message from s_2, s_3, s_4 to s_1, this formula can also be applied to from s_1, s_2, s_4 to s_2 and from s_1, s_2, s_4 to s_3. Since the labels for each node are interchangeable.

Define two sets as the possible values for r_1, r_2, r_3, r_4

$$\Gamma_3 = \left\{ \begin{array}{c} \{d_1, \quad d_1, \quad d_1, \quad d_1\} \\ \{d_1, \quad d_1, \quad d_2, \quad d_1\} \\ \{d_1, \quad d_2, \quad d_1, \quad d_1\} \\ \{d_1, \quad d_2, \quad d_2, \quad d_1\} \end{array} \right\} \tag{4.10}$$

$$\Gamma_4 = \left\{ \begin{array}{cccc} \{d_2, & d_1, & d_1, & d_1\} \\ \{d_2, & d_1, & d_2, & d_1\} \\ \{d_2, & d_2, & d_1, & d_1\} \\ \{d_2, & d_2, & d_2, & d_2\} \end{array} \right\} \tag{4.11}$$

$$\begin{aligned} \frac{p(r_1 = d_1|b_1b_2b_3b_4)}{p(r_1 = d_2|b_1b_2b_3b_4)} &= \frac{p(r_1 = d_1)}{p(r_1 = d_2)} \frac{p(b_1|r_1 = d_1)}{p(b_1|r_1 = d_2)} \frac{\sum_{\{r_2,r_3,r_4\}\in\Gamma_3} p(b_2b_3b_4r_2r_3r_4|r_1 = d_1)}{\sum_{\{r_2,r_3,r_4\}\in\Gamma_4} p(b_2b_3b_4r_2r_3r_4|r_1 = d_2)} \\ &= e^{v_1^{(0)}} \frac{e^{R_4}e^{v_2^{(0)}}e^{v_3^{(0)}} + e^{R_4}e^{v_2^{(0)}} + e^{R_4}e^{v_3^{(0)}} + e^{R_4}}{e^{R_4}e^{v_2^{(0)}}e^{v_3^{(0)}} + e^{R_4}e^{v_2^{(0)}} + e^{R_4}e^{v_3^{(0)}} + 1} \end{aligned}$$

$$e^{U_1} = \frac{e^{R_4}(e^{v_2^{(0)}} + 1)(e^{v_3^{(0)}} + 1)}{e^{R_4}(e^{v_2^{(0)}} + 1)(e^{v_3^{(0)}} + 1) + 1 - e^{R_4}} \tag{4.12}$$

Similarly, to generalize this case, the message passing from the output and other $d_c - 2$ input nodes to input node k is

$$\exp(U_k) = \frac{\exp(R_{d_c}) \prod_{\substack{q=1 \\ q\neq k}}^{d_c-1} (\exp(v_q^{(0)}) + 1)}{\exp(R_{d_c}) \prod_{\substack{q=1 \\ q\neq k}}^{d_c-1} (\exp(v_q^{(0)}) + 1) + 1 - \exp(R_{d_c})} \tag{4.13}$$

Message Passing in OR Logic Gate

In this case, nodes are connected through an OR gate, Table 4.9 explains the truth table for an OR gate with three inputs and one output using the same labels for these signals as we demonstrated for AND logic gate.

Table 4.9: Truth Table with Three Inputs and One Output for OR Gate

s_1	s_2	s_3	s_4
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Message From Inputs to Output

Define two sets as the possible values for r_1, r_2, r_3, r_4

$$\Gamma_5 = \left\{ \begin{array}{cccc} \{d_1, & d_1, & d_1, & d_1\} \end{array} \right\} \tag{4.14}$$

$$\Gamma_6 = \left\{ \begin{array}{cccc} \{d_1, & d_1, & d_2, & d_2\} \\ \{d_1, & d_2, & d_1, & d_2\} \\ \{d_1, & d_2, & d_2, & d_2\} \\ \{d_2, & d_1, & d_1, & d_2\} \\ \{d_2, & d_1, & d_2, & d_2\} \\ \{d_2, & d_2, & d_1, & d_2\} \\ \{d_2, & d_2, & d_2, & d_2\} \end{array} \right\} \tag{4.15}$$

$$\begin{aligned} \frac{p(r_4 = d_1|b_1b_2b_3b_4)}{p(r_4 = d_2|b_1b_2b_3b_4)} &= \frac{p(r_4 = d_1)}{p(r_4 = d_2)} \frac{p(b_4|r_4 = d_1)}{p(b_4|r_4 = d_2)} \frac{\sum_{\{r_1,r_2,r_3\}\in\Gamma_5} p(b_1b_2b_3r_1r_2r_3|r_4 = d_1)}{\sum_{\{r_1,r_2,r_3\}\in\Gamma_6} p(b_1b_2b_3r_1r_2r_3|r_4 = d_2)} \\ &= \frac{p(b_4|r_4 = d_1)}{p(b_4|r_4 = d_2)} e^{U_4} \end{aligned} \tag{4.16}$$

$$\begin{aligned}
e^{U_4} &= \frac{\text{the case when } r_4 = d_1}{\text{all the cases when } r_4 = d_2} \\
&= \frac{e^{v_1^{(0)}} e^{v_2^{(0)}} e^{v_3^{(0)}}}{e^{v_1^{(0)}} e^{v_2^{(0)}} + e^{v_2^{(0)}} e^{v_3^{(0)}} + e^{v_1^{(0)}} e^{v_3^{(0)}} + e^{v_1^{(0)}} + e^{v_2^{(0)}} + e^{v_3^{(0)}} + 1} \\
e^{-U_4} &= e^{-v_3^{(0)}} + e^{-v_2^{(0)}} + e^{-v_1^{(0)}} + e^{-v_1^{(0)}} e^{-v_3^{(0)}} + e^{-v_2^{(0)}} e^{-v_3^{(0)}} + e^{-v_1^{(0)}} e^{-v_2^{(0)}} + e^{-v_1^{(0)}} e^{-v_2^{(0)}} e^{-v_3^{(0)}} \\
&= (e^{-v_1^{(0)}} + 1)(e^{-v_2^{(0)}} + 1)(e^{-v_3^{(0)}} + 1) - 1
\end{aligned}$$

$$e^{U_4} = \frac{1}{(e^{-v_1^{(0)}} + 1)(e^{-v_2^{(0)}} + 1)(e^{-v_3^{(0)}} + 1) - 1} \tag{4.17}$$

This case can be generalized for $d_c - 1$ input nodes to the only output node, and the message received by the only output is

$$U_{d_c} = -\ln\left(\prod_{k=1}^{d_c-1} (\exp(-v_k^{(0)}) + 1) - 1\right) \tag{4.18}$$

Message From Other Inputs and Output to an Input

As we explained in AND logic gate, the formula for the message from s_2, s_3, s_4 to s_1 can also be applied to from s_1, s_2, s_4 to s_2 and from s_1, s_2, s_4 to s_3. Since the labels for each node are interchangeable.

Define two sets as the possible values for r_1, r_2, r_3, r_4

$$\Gamma_7 = \left\{ \begin{array}{l} \{d_1,\ \ d_1,\ \ d_1,\ \ d_1\} \\ \{d_1,\ \ d_1,\ \ d_2,\ \ d_2\} \\ \{d_1,\ \ d_2,\ \ d_1,\ \ d_2\} \\ \{d_1,\ \ d_2,\ \ d_2,\ \ d_2\} \end{array} \right\} \tag{4.19}$$

$$\Gamma_8 = \left\{ \begin{array}{l} \{d_2,\ \ d_1,\ \ d_1,\ \ d_2\} \\ \{d_2,\ \ d_1,\ \ d_2,\ \ d_2\} \\ \{d_2,\ \ d_2,\ \ d_1,\ \ d_2\} \\ \{d_2,\ \ d_2,\ \ d_2,\ \ d_2\} \end{array} \right\} \tag{4.20}$$

$$\begin{aligned} \frac{p(r_1 = d_1|b_1b_2b_3b_4)}{p(r_1 = d_2|b_1b_2b_3b_4)} &= \frac{p(r_1 = d_1)}{p(r_1 = d_2)} \frac{p(b_1|r_1 = d_1)}{p(b_1|r_1 = d_2)} \frac{\sum_{\{r_2,r_3,r_4\}\in\Gamma_7} p(b_2b_3b_4r_2r_3r_4|r_1 = d_1)}{\sum_{\{r_2,r_3,r_4\}\in\Gamma_8} p(b_2b_3b_4r_2r_3r_4|r_1 = d_2)} \\ &= e^{v_1^{(0)}} \frac{e^{R_4} e^{v_2^{(0)}} e^{v_3^{(0)}} + e^{v_2^{(0)}} + e^{v_3^{(0)}} + 1}{e^{v_2^{(0)}} e^{v_3^{(0)}} + e^{v_2^{(0)}} + e^{v_3^{(0)}} + 1} \\ &= e^{v_1^{(0)}} \frac{e^{R_4} e^{v_2^{(0)}} e^{v_3^{(0)}} + e^{v_2^{(0)}} + e^{v_3^{(0)}} + 1 + e^{v_2^{(0)}} e^{v_3^{(0)}} - e^{v_2^{(0)}} e^{v_3^{(0)}}}{e^{v_2^{(0)}} e^{v_3^{(0)}} + e^{v_2^{(0)}} + e^{v_3^{(0)}} + 1} \end{aligned}$$

$$e^{U_1} = 1 + \frac{e^{v_2^{(0)}} e^{v_3^{(0)}}}{(e^{v_2^{(0)}} + 1)(e^{v_3^{(0)}} + 1)} (e^{R_4} - 1) \tag{4.21}$$

The message passing from the output node and other $d_c - 2$ input nodes to node k can be generalized as

$$\exp(U_k) = 1 + \frac{\prod_{\substack{q=1 \\ q\neq k}}^{d_c-1} \exp(v_q^{(0)})}{\prod_{\substack{q=1 \\ q\neq k}}^{d_c-1} (\exp(v_q^{(0)}) + 1)} (e^{R_{d_c}} - 1) \tag{4.22}$$

Message Passing in XOR Logic Gate

In this case, nodes are connected through an XOR gate, Table 4.10 explains the truth table for an XOR gate with three inputs and one output using the same labels for these signals as we demonstrated for AND logic gate.

Table 4.10: Truth Table with Three Inputs and One Output for XOR Gate

s_1	s_2	s_3	s_4
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Message From Inputs to Output

Define two sets as the possible values for r_1, r_2, r_3, r_4

$$\Gamma_9 = \left\{ \begin{array}{c} \{d_1, \ d_1, \ d_1, \ d_1\} \\ \{d_1, \ d_2, \ d_2, \ d_1\} \\ \{d_2, \ d_1, \ d_2, \ d_1\} \\ \{d_2, \ d_2, \ d_1, \ d_1\} \end{array} \right\} \tag{4.23}$$

$$\Gamma_{10} = \left\{ \begin{array}{cccc} \{d_1, & d_1, & d_2, & d_2\} \\ \{d_1, & d_2, & d_1, & d_2\} \\ \{d_2, & d_1, & d_1, & d_2\} \\ \{d_2, & d_2, & d_2, & d_2\} \end{array} \right\} \tag{4.24}$$

$$\begin{aligned}
\frac{p(r_4 = d_1|b_1b_2b_3b_4)}{p(r_4 = d_2|b_1b_2b_3b_4)} &= \frac{p(r_4 = d_1)}{p(r_4 = d_2)} \frac{p(b_4|r_4 = d_1)}{p(b_4|r_4 = d_2)} \frac{\sum_{\{r_1,r_2,r_3\}\in\Gamma_9} p(b_1b_2b_3r_1r_2r_3|r_4 = d_1)}{\sum_{\{r_1,r_2,r_3\}\in\Gamma_{10}} p(b_1b_2b_3r_1r_2r_3|r_4 = d_2)} \\
&= \frac{p(b_4|r_4 = d_1)}{p(b_4|r_4 = d_2)} e^{U_4} \\
e^{U_4} &= \frac{\text{the cases when } r_4 = d_1}{\text{the other cases when } r_4 = d_2} \\
&= \frac{e^{v_1^{(0)}} e^{v_2^{(0)}} e^{v_3^{(0)}} + e^{v_1^{(0)}} + e^{v_2^{(0)}} + e^{v_3^{(0)}}}{e^{v_1^{(0)}} e^{v_2^{(0)}} + e^{v_1^{(0)}} e^{v_3^{(0)}} + e^{v_2^{(0)}} e^{v_3^{(0)}} + 1} \\
\frac{e^{U_4} - 1}{e^{U_4} + 1} &= \frac{e^{v_1^{(0)}} - 1}{e^{v_1^{(0)}} + 1} \cdot \frac{e^{v_2^{(0)}} - 1}{e^{v_2^{(0)}} + 1} \cdot \frac{e^{v_3^{(0)}} - 1}{e^{v_3^{(0)}} + 1}
\end{aligned}$$

$$\tanh \frac{U_4}{2} = \tanh \frac{e^{v_1^{(0)}}}{2} \tanh \frac{e^{v_2^{(0)}}}{2} \tanh \frac{e^{v_3^{(0)}}}{2} \tag{4.25}$$

Finally, for XOR gate, the message passing from $d_c - 1$ input nodes to the only output node is

$$\tanh \frac{U_{d_c}}{2} = \prod_{k=1}^{d_c-1} \tanh \frac{v_k^{(0)}}{2} \tag{4.26}$$

Message From Other Inputs and Output to an Input

The message from s_2, s_3, s_4 to s_1 is explained as follows. Define two sets as the possible values for r_1, r_2, r_3, r_4.

$$\Gamma_{11} = \left\{ \begin{array}{cccc} \{d_1, & d_1, & d_1, & d_1\} \\ \{d_1, & d_1, & d_2, & d_2\} \\ \{d_1, & d_2, & d_1, & d_2\} \\ \{d_1, & d_2, & d_2, & d_1\} \end{array} \right\} \tag{4.27}$$

$$\Gamma_{12} = \left\{ \begin{array}{cccc} \{d_2, & d_1, & d_1, & d_2\} \\ \{d_2, & d_1, & d_2, & d_1\} \\ \{d_2, & d_2, & d_1, & d_1\} \\ \{d_2, & d_2, & d_2, & d_2\} \end{array} \right\} \tag{4.28}$$

$$\begin{aligned} \frac{p(r_1 = d_1|b_1b_2b_3b_4)}{p(r_1 = d_2|b_1b_2b_3b_4)} &= \frac{p(r_1 = d_1)}{p(r_1 = d_2)} \frac{p(b_1|r_1 = d_1)}{p(b_1|r_1 = d_2)} \frac{\sum_{\{r_2,r_3,r_4\}\in\Gamma_{11}} p(b_2b_3b_4r_2r_3r_4|r_1 = d_1)}{\sum_{\{r_2,r_3,r_4\}\in\Gamma_{12}} p(b_2b_3b_4r_2r_3r_4|r_1 = d_2)} \\ &= e^{v_1^{(0)}} \frac{e^{R_4} e^{v_2^{(0)}} e^{v_3^{(0)}} + e^{v_2^{(0)}} + e^{v_3^{(0)}} + e^{R_4}}{e^{v_2^{(0)}} e^{v_3^{(0)}} + e^{v_2^{(0)}} e^{R_4} + e^{v_3^{(0)}} e^{R_4} + 1} \\ \frac{e^{U_1} - 1}{e^{U_1} + 1} &= \frac{e^{v_2^{(0)}} - 1}{e^{v_2^{(0)}} + 1} \cdot \frac{e^{v_3^{(0)}} - 1}{e^{v_3^{(0)}} + 1} \cdot \frac{e^{R_4} - 1}{e^{R_4} + 1} \end{aligned}$$

$$\tanh \frac{U_1}{2} = \tanh \frac{v_2^{(0)}}{2} \tanh \frac{v_3^{(0)}}{2} \tanh \frac{R_4}{2} \tag{4.29}$$

To generalize this case, the message passing from the output node and other $d_c - 2$ input nodes to node k is

$$\tanh \frac{U_k}{2} = \prod_{\substack{q=1 \\ q\neq k}}^{d_c-1} \tanh \frac{v_q^{(0)}}{2} \tanh \frac{R_{d_c}}{2} \tag{4.30}$$

The Numerical Results

The application environment of these codes in GPRAM systems is quite different from those in telecommunication systems. Supposing one information message is sent over a symbol duration, we sample the output from matched filter at the end of symbol duration and obtain a single value as decision statistics in telecommunication systems. In GPRAM, there is no way to decide how long a message will last and it can not wait until the end of message since it needs to respond in real time. The information needs to be continuously processed and to obtain an estimate based on current available signal. Thus, the iterative decoding process is revised in which n_k is independent during each iteration in GPRAM Equivalently, this process resembles a decoder in telecommunication systems which needs to make tentative decision, not at the end of symbol duration rather during a symbol duration. In real bio-systems, the noise is often correlated, which is neither independent (GPRAM case) nor totally dependent. The iteration time is set to be 10 in our simulations.

There are two ways to select constellation, (a) $d_1 = -1$, $d_2 = 1$; (b) $d_1 p = d_2(1-p)$. To decode Code 1 defined in Table. 4.1 for non-equiprobable symbols, the receiver treats all non-equiprobable symbols as equiprobable symbols. This means there is mismatches between prior symbol probabilities from transmitter to receiver. For all other codes, the receiver is assumed to know prior probabilities for symbols.

In each simulation trial, a corresponding codeword is generated and added noises. For each simulation batch, either 10000 trials will get executed or a minimum of 100 error events has been cumulated for each variable node.

In table 4.11, we show error rate for codes 1 to 9 with optimal constellation, i.e., $d_1 p = d_2(1-p)$, and in table 4.12 results for conventional constellation, $d_1 = -1$, $d_2 = 1$. In order to ensure an appropriate error rate which is no less than 10^{-3} in GPRAM systems, we select $E_b = 0.25$. As shown in Tables 4.3 and 4.5, the prior probabilities of variable nodes s_5, s_6, s_7 for codes 3, 4, 5 are affected by constellation selection. Further for the remaining codes, variable nodes s_1, s_3, s_5 for codes 6, 7, 8, 9 are affected by constellation selection. Both results in Tables 4.11 and 4.12 confirm this observation.

Error propagation will have some effects to the system. In tables 4.13 and 4.17, we present the results for codes constructed from the (7,4) Hamming code and code constructed from the (7,4,2) code. As we can see the results in 4.17 is generally better than those in 4.13.

Finally, these results show even though extended codes are weaker than the original Hamming code in term of distance profile and error correct capability, at the error rate of 10^{-2} to 10^{-3}, these extended codes are comparable. So, these codes might be good candidates because of their distance properties. There are also other tables showing the simulation results for different Hamming code structure and constellation methods.

Table 4.11: Error Probabilities of Codes using Hamming (7,4) Structure with $p = 0.5$ and Optimal Constellation

σ_1^2	Error rate (%) for Node						
	1	2	3	4	5	6	7
Code 1/2/6							
1	0.26	0.16	0.15	0.25	0.48	0.33	0.39
2	3.4	2.9	3.1	3.1	3.5	3.2	3.1
4	10	11	10	10	10	11	11
Code 3							
1	0.7	0.43	0.39	0.37	0	0.33	0
2	3.7	3.1	3.1	3.8	0.3	3.3	0.11
4	10	9.7	9.5	13	2.6	11	1.1
Code 4							
1	0.75	0.94	0.98	0.89	0	0	0.01
2	5.4	6.4	5.0	5.2	0.1	0.1	0.12
4	14	17	15	15	1.1	1.1	1.2
Code 5							
1	0.72	0.65	0.55	0.68	0.01	0	0
2	3.4	3.2	3.4	3.4	0.36	0.4	0.32
4	9.6	10	9.3	9.0	2.6	2.5	3.0
Code 7							
1	0	0.5	0.32	0.45	0.05	0.4	0.78
2	0.0 7	2.4	2.5	4.6	1.6	3.5	4.6
4	0.99	8.0	8.1	12	7.6	11	13
Code 8							
1	0	1.0	0	0.88	0	0.64	0.94
2	0.08	5.7	0.07	5.6	0	5.0	4.8
4	0.81	16	0.85	14	0	13	13
Code 9							
1	0	0.62	0	0.82	0	0.66	0.73
2	0.13	3.5	0.09	3.6	0	3.7	3.5
4	1.2	9.8	1.3	9.4	0	9.4	9.2

Table 4.12: Error Probabilities of Codes Using Hamming (7,4) Structure with $p = 0.5$ and Equal Spaced Constellation $\{\pm 1\}$

σ_1^2	Error rate (%) for Node						
	1	2	3	4	5	6	7
Code 1/2/6							
1	0.26	0.16	0.15	0.25	0.48	0.33	0.39
2	3.4	2.9	3.1	3.1	3.5	3.2	3.1
4	10	11	10	10	10	11	11
Code 3							
1	0.56	0.34	0.50	0.17	5.30	0.46	2.80
2	3.6	2.8	3.5	2.7	7.7	3.3	5.1
4	9.5	9.7	9.6	11	9.6	11	6.8
Code 4							
1	0.45	0.55	0.50	0.46	2.8	2.6	2.8
2	3.4	2.7	3.3	2.9	4.8	4.8	4.5
4	11	11	11	11	7.5	7.0	7.2
Code 5							
1	1.3	1.4	1.1	1.2	5.2	5.5	5.4
2	4.2	3.7	4.0	4.3	7.5	7.8	7.4
4	9.1	8.9	9.8	9.9	9.7	9.9	9.7
Code 7							
1	2.00	0.37	0.53	0.28	2.80	0.41	0.44
2	4.7	2.3	3.0	3.1	8.1	3.3	3.6
4	6.5	8.4	8.7	10	14	11	11
Code 8							
1	2.0	0.45	1.9	0.42	6.6	0.42	0.4
2	3.5	3.3	3.7	3.6	6.3	3.2	3.6
4	6.2	11	6.3	11	6.5	10	11
Code 9							
1	4.2	1.3	3.9	1.1	6.1	0.98	1.0
2	6.6	4.2	6.8	4.1	6.4	3.6	4.0
4	8.8	9.5	8.6	9.7	5.9	11	9.7

Table 4.13: Error Probabilities of Codes Using Hamming (7,4) Structure with $p = 0.3$ and Optimal Constellation

σ_1^2	Error rate (%) for Node						
	1	2	3	4	5	6	7
Code 1							
1	0.19	0.12	0.24	0.15	0.43	0.29	0.41
2	2.1	2.1	2.1	2.4	3.3	3.1	3.1
4	8.3	8.3	8.1	8.6	10	10	11
Code 2							
1	0.51	0.33	0.72	0.78	0.18	0.26	0.23
2	3.6	2.8	3.4	3.0	2.6	3.0	2.4
4	8.5	7.3	8.4	8.2	9.4	9.4	9.1
Code 3							
1	0.95	0.54	0.88	0.76	0	0.21	0.05
2	3.8	2.8	3.2	3.8	0	2.8	1.3
4	7.6	7.3	7.4	8.9	0	9.5	7.0
Code 4							
1	0.64	0.50	0.83	0.76	0.06	0.07	0.11
2	4.3	3.7	3.9	3.8	1.5	1.4	1.4
4	9.8	9.7	9.6	9.4	6.8	7.2	6.4
Code 5							
1	1.7	1.6	1.6	1.8	0	0	0
2	4.0	3.7	4.2	3.8	0	0	0.01
4	8.2	7.7	7.4	7.4	0.01	0	0
Code 6							
1	0.16	0.59	0.18	0.71	0.35	1.3	1.3
2	2.5	2.9	2.2	3.2	3.3	3.7	4.2
4	9.3	8.4	8.8	8.6	10	9.5	8.8
Code 7							
1	0.08	0.39	0.18	0.76	0	1.3	1.3
2	1.5	1.5	2.4	3.5	0.08	4.1	4.8
4	7.3	4.3	8.0	9.2	1.8	9.1	9.2
Code 8							
1	0.05	0.84	0	0.74	0	1.2	1.3
2	0.64	4.3	4.5	3.9	0.01	4.9	4.4
4	2.4	10	2.5	9.5	0.5.3	9.4	9.9
Code 9							
1	0	1.8	0	1.6	0	2.1	2.1
2	0	3.5	0	4.0	0	4.5	4.2
4	0.01	7.9	0	7.9	0	8.1	7.8

Table 4.14: Error Probabilities of Codes using Hamming (7,4) Structure with $p = 0.3$ and Equal Spaced Constellation $\{\pm 0.5\}$

σ_1^2	Error rate (%) for Node						
	1	2	3	4	5	6	7
Code 1							
1	0.19	0.12	0.24	0.15	0.43	0.29	0.41
2	2.1	2.1	2.1	2.4	3.3	3.1	3.1
4	8.3	8.3	8.1	8.6	10	10	11
Code 2							
1	7.0	5.9	6.9	7.1	0.74	0.72	0.78
2	11	9.6	11	11	4.1	4.0	3.9
4	14	13	15	14	11	11	11
Code 3							
1	6.4	4.0	9.4	4.7	2.6	0.84	0.1
2	10	8.9	12	8.9	2.6	4.0	1.4
4	13	13	15	14	2.4	11	6.2
Code 4							
1	2.6	0.99	2.6	2.6	0.1	0.1	0.02
2	7.5	5.1	7.9	7.7	1.2	1.7	1.5
4	13	12	13	13	6.5	6.6	6.8
Code 5							
1	14	14	14	14	2.8	2.7	2.7
2	15	14	15	15	2.5	2.5	2.8
4	15	15	16	15	2.6	2.6	2.9
Code 6							
1	0.49	5.6	0.62	7.2	0.34	10	9.6
2	3.4	9.8	3.5	11	3.1	13	13
4	11	14	9.9	15	11	15	16
Code 7							
1	0.04	4.1	0.63	4.3	12	10	6.5
2	1.3	6.6	3.5	9.0	13	13	10
4	6.7	9.6	8.9	13	14	16	15
Code 8							
1	0.05	2.6	0.06	2.7	11	6.3	6.1
2	0.73	7.2	0.78	7.1	14	11	11
4	4.6	14	4.4	13	16	14	15
Code 9							
1	2.5	14	2.5	13	0.73	14	14
2	2.5	14	2.6	15	0.71	15	15
4	2.7	15	2.8	15	0.97	16	16

Table 4.15: Error Probabilities of Codes using Hamming (7,4,2) Structure with $p = 0.5$ and Optimal Constellation

σ_1^2	Error rate (%) for Node						
	1	2	3	4	5	6	7
Code 1/2/6							
1	0.16	0	0.15	0.22	0.19	0.16	0.24
2	2.5	0.95	2.5	2.6	2.5	2.6	2.4
4	9.3	6.7	9.1	9.3	9.5	9.5	9.7
Code 3							
1	0.44	0.19	0.64	0.28	0.10	0.19	0.03
2	3.8	1.4	2.9	2.6	0.97	2.6	0.54
4	11	6.4	8.1	9.8	5.2	9.0	2.7
Code 4							
1	0.66	0.37	0.72	0.55	0.03	0.01	0.02
2	3.8	3.4	3.6	3.8	0.41	0.37	0.36
4	11	13	11	11	2.9	2.9	2.6
Code 5							
1	0.58	0.49	0.76	0.6	0.04	0.06	0.06
2	3.0	1.6	3.0	3.0	1.1	0.93	0.95
4	8.4	6.0	8.3	7.8	5.0	4.6	5.4
Code 7							
1	0.02	0.13	0.63	0.17	0.06	0.11	0.48
2	0.58	1.7	3.0	2.7	1.2	2.4	3.9
4	2.8	6.2	8.5	9.2	4.9	9.4	11
Code 8							
1	0	0	0.84	0.44	0	0.48	0.65
2	0.1	0.44	4.3	3.6	0.1	3.9	4.8
4	0.96	3.1	12	11	1.2	11	13
Code 9							
1	0.01	0.02	0.72	0.67	0	0.5	0.74
2	0.02	0.34	3.5	3.0	0.08	3.0	3.6
4	0.79	1.8	9.9	8.3	0.94	8.2	9.9

Table 4.16: Error Probabilities of Codes using Hamming (7,4,2) Structure with $p = 0.5$ and Equal Spaced Constellation $\{\pm 0.5\}$

σ_1^2	Error rate (%) for Node						
	1	2	3	4	5	6	7
code 1/2/6, p=0.5							
1	0.16	0	0.15	0.22	0.19	0.16	0.24
2	2.5	0.95	2.5	2.6	2.5	2.6	2.4
4	9.3	6.7	9.1	9.3	9.5	9.5	9.7
Code 3							
1	0.4	0.12	0.94	0.19	1.6	0.27	0.37
2	2.7	1.3	3.2	2.3	5.4	2.6	2.0
4	8.8	6.2	9.2	8.8	11	9.2	5.6
Code 4							
1	0.45	0.56	0.33	0.29	0.35	0.37	0.34
2	2.7	1.3	2.8	2.6	2.0	1.7	2.3
4	9.6	6.9	9.0	9.0	5.4	6.0	5.8
Code 5							
1	0.97	1.4	1.0	1.0	1.6	2.1	1.7
2	3.5	2.4	3.6	3.8	5.8	5.6	5.4
4	9.0	5.9	9.1	8.8	11	10	11
Code 7							
1	0.38	0.08	1.1	0.15	1.7	0.15	0.31
2	2.1	1.4	3.7	2.7	5.4	2.4	2.8
4	6.0	6.2	8.8	9.0	11	9.7	9.5
Code 8							
1	2.9	0.5	0.4	0.39	3.1	0.33	0.42
2	4.3	1.2	3.9	2.9	4.3	2.7	3.5
4	5.5	3.2	11	9.0	5.7	9.0	11
Code 9							
1	5.8	1.4	0.93	1.0	5.8	1.1	0.93
2	7.4	3.5	4.1	3.6	7.2	3.6	4.1
4	8.6	6.6	10	9.1	8.5	8.3	10

Table 4.17: Error Probabilities of Codes Using Hamming (7,4,2) Structure with $p = 0.3$ and Optimal Constellation

σ_1^2	Error rate (%) for Node						
	1	2	3	4	5	6	7
Code 1							
1	0.13	0.01	0.14	0.09	0.23	0.18	0.18
2	1.6	0.79	1.8	1.8	2.7	2.7	2.6
4	7.4	4.8	7.5	7.1	11	9.4	10
Code 2							
1	0.52	0.03	0.46	0.63	0.15	0.09	0.07
2	2.7	0.9	3.0	2.9	1.5	1.6	1.5
4	7.4	4.3	7.7	7.5	7.8	8.1	7.9
Code 3							
1	0.13	0.07	3.0	0.56	1.7	0.06	1.4
2	1.0	0.33	5.3	2.9	3.6	1.7	5.7
4	3.6	2.8	8.6	7.9	4.7	8.0	12
Code 4							
1	0.52	0.06	0.35	0.42	0.01	0.5	0.4
2	2.5	0.51	2.5	2.5	0.8	0.96	0.71
4	7.0	4.3	6.9	6.9	5.0	5.0	5.5
Code 5							
1	1.6	1.0	1.7	1.8	0	0	0
2	3.7	2.0	4.3	3.7	0.02	0.04	0
4	7.4	6.5	7.0	7.0	0.33	0.35	0.41
Code 6							
1	0.16	0.01	0.62	0.75	0.14	0.62	0.76
2	2.1	0.46	3.4	2.5	2.2	3.0	3.4
4	8.7	5.3	8.2	7.3	8.5	7.4	8.0
Code 7							
1	0.03	0.07	1.2	0.51	0	0.75	1.1
2	0.63	0.96	2.4	2.9	0.11	3.1	4.0
4	3.5	4.9	5.3	7.6	1.7	7.7	9.1
Code 8							
1	0.04	0.02	1.8	0.46	0.11	0.57	2.0
2	1.1	0.49	5.6	2.6	1.0	2.5	5.4
4	4.7	2.7	10	7.1	4.7	7.2	9.9
Code 9							
1	0	0	1.8	1.7	0	1.7	2.1
2	0	0.03	4.2	3.3	0	3.8	4.7
4	0.05	0.58	8.1	7.4	0.07	7.4	8.4

CHAPTER 5: LDPC CODE WITH NON-EQUIPROBABLE SYMBOLS

In practice, the output from source encoders after compression often contains some extent of redundancy [92] [100]. Due to these redundancies, the equiprobable symbols no longer holds that property [94]. How to optimize the system with such symbols is discussed in [93] [101]. The tight lower bounds on symbol error rate of nonuniform signaling is discussed in [102]. Extension of Turbo codes with non-equiprobable symbols can also be found in [103] [104] [105].

In this Chapter, selection and computation of optimal constellations for LDPC codes in order to maximize channel mutual information is discussed. Revised message passing algorithm for LDPC codes with non-equiprobable symbols will be reviewed and has already been presented in [104] . Also short LDPC codes with non-equiprobable signaling with different constellation schemes are simulated and the results are compared with the ones with equiprobable symbols and prior work in [103].

Signal and System

Assume each codeword be of length of K bits, s $= [s_1, s_2, \cdots, s_K]$, which contains $K - T$ message bits $\mathbf{I} = [I_1, I_2, \cdots, I_{K-T}]$ and T parity check bits $\mathbf{O} = [O_1, O_2, \cdots, O_T]$, where s $= [s_1, s_2, \cdots, s_K] = [I_1, \cdots, I_{K-T}, O_1, \cdots, O_T]$, and $\{s_k\} \in \{0, 1\}$. Let $p_{k,1} = p(s_k = 0)$, $p_{k,2} = p(s_k = 1)$, where $p(.)$ denotes probability. $\{s_k\}$ is modulated to binary constellation $\{r_k\} \in \{d_{k,1}, d_{k,2}\}$ and deteriorated by Additive White Gaussian noise (AWGN) $\{n_k\}$ with zero mean and variance of σ_1^2.

$$b_k = r_k + n_k \tag{5.1}$$

for $k = 1, 2, \cdots, K$. In an ideal case, the symbols are equiprobable, $p_{k,1} = p_{k,2} = 0.5$ and equal-spaced constellation would be used $d_{k,1} = -d_{k,2} = \sqrt{E_s}$, where E_s is the average energy per transmitted symbol. While in this Chapter, our concentration is on non-equiprobable case, i.e., $p_{k,1} = p_1 \neq p_{k,2} = p_2$, $k = 1, \cdots, K - T$ and $p_1 + p_2 = 1$.

To better explain the effect of non-equiprobable symbols, (7,4) Hamming code is used for illustration. Fig. 5.1(a) shows the conventional Tanner graph with variable nodes, s nodes and check nodes, $\mathbf{C}$. In the former case, the end nodes for each edge can both be either input or output of the parity check node. For example, edge of C_2 to s_7 can be considered as output of C_2 with inputs s_1, s_2, s_4; meanwhile, edge of s_7 to C_2 can also considered as input of C_2 with outputs to s_1, s_2, s_4 respectively. While for the latter case, because of the mismatching problem in prior probabilities, only one parity check bit associated with one check node can be selected as an output. For example, in Fig. 5.1(b), assume $p_{k,1} = 0.3$ for $k = 1, 2, 3, 4$, then $p_{6,1} = 0.468$. If s_4 is selected as output of further operation, then for the case of $p_{2,1} = p_{3,1} = 0.3$ and $p_{6,1} = 0.468$, $p_{4,1} = 0.495$ can be obtained, which is contradicted to the fact that $p_{4,1} = 0.3$. This prior probabilities mismatching problem was discussed in Chapter 3.

To study LDPC cods with non-equiprobable symbols, three problems need to be solved:

1. optimal constellations for both information bits and parity check bits;
2. message propagating from information bits to parity check bits;
3. message propagating from parity check bits to information bits.

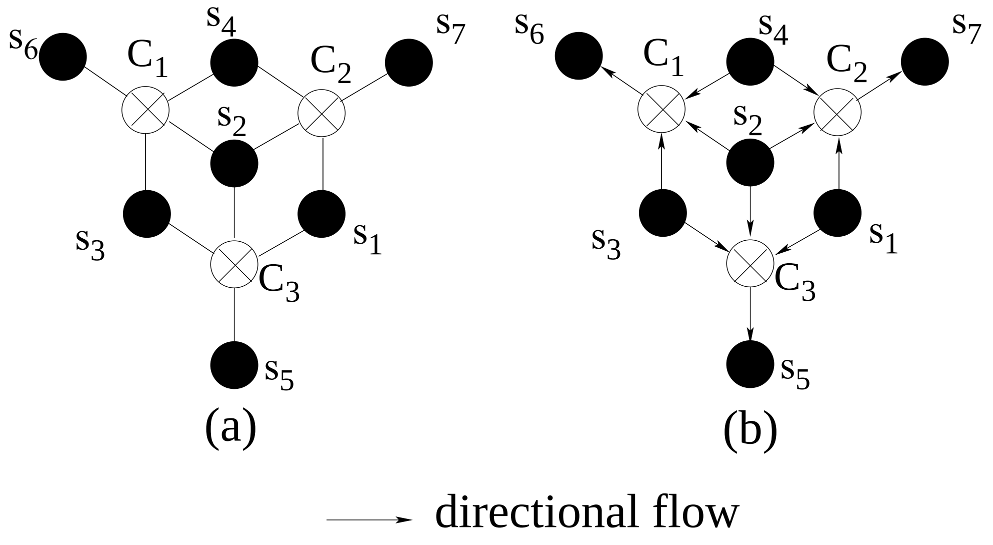


Figure 5.1: (a) Conventional Tanner Graph of (7,4) Hamming Code and Extension (b) Graph with Directions for Non-equiprobable Symbols

Signaling Optimization

In this section, how to find optimal constellations for non-equiprobable symbols in terms of maximizing channel mutual information and capacity is presented.

To make this problem more generalized, M-ary amplitude-shift keying (MASK) is considered at first, where the constellation for the LDPC codes is a special case of $M = 2$. Let $\{d_m\}$ for $m = 1, \cdots, M$ denotes the constellation points for an M-ary signal s with probabilities of $p_m = p(s_m) = p(d_m)$. The signal is passed through an AWGN channel and the output is $b = d_m + n$, where n is Gaussian distributed with zero mean and variance of $\sigma_1^2 = N_0/2$. Let $b_0 = b/\sqrt{N_0}$, and $g_m = d_m/\sqrt{N_0}$, and the capacity can be represented as

$$I(D;B) = \sum_{m=1}^{M} \int_{-\infty}^{\infty} \frac{p_m}{\sqrt{\pi}} e^{-(b_0-g_m)^2} \log_2 \frac{e^{-(b_0-g_m)^2}}{\sum_{\alpha=1}^{M} p_\alpha e^{-(b_0-g_\alpha)^2}} db_0 \tag{5.2}$$

In order to optimize $I(D;B)$ with given $\{p_m\}$ and

$$\sum_{m=1}^{M} p_m g_m^2 = \frac{E_s}{N_0} \tag{5.3}$$

, the object function is defined as

$$\Omega = I(D;B) + \lambda \left(\sum_{m=1}^{M} p_m g_m^2 - \frac{E_s}{N_0} \right) \tag{5.4}$$

where λ is a Lagrange multiplier. The optimal g_m and λ could be found by solving $\partial\Omega/\partial g_m = 0$ and $\partial\Omega/\partial\lambda = 0$ for $m = 1, 2, \cdots, M$, where

$$\begin{aligned} \frac{\partial I(D;B)}{\partial g_a} = & \sum_{\substack{m=1 \\ m\neq a}}^{M} \int_{-\infty}^{\infty} 2p_m p_a \\ & \left[\frac{(g_a - g_m - b_0)\exp\left(-b_0^2 + (g_a - g_m)(2b_0 - g_a + g_m)\right)}{\sqrt{\pi}\ln 2 \sum_{\alpha=1}^{M} p_\alpha \exp\left((g_\alpha - g_m)(2b_0 - g_\alpha + g_m)\right)} + \right. \\ & \left. \frac{(g_a - g_m + b_0)\exp\left(-b_0^2 + (g_m - g_a)(2b_0 - g_m + g_a)\right)}{\sqrt{\pi}\ln 2 \sum_{\alpha=1}^{M} p_\alpha \exp\left((g_\alpha - g_a)(2b_0 - g_\alpha + g_a)\right)} \right] db_0 \end{aligned} \tag{5.5}$$

Proposition 1: For $M = 2$, optimal constellation satisfies $p_1 g_1 + p_2 g_2 = 0$.

Proof : Taking $\partial\Omega/\partial g_m = 0$ for m =1,2,

$$\frac{\partial I(D;B)}{\partial g_m} + 2\lambda p_m g_m = 0 \tag{5.6}$$

To complete the proof, only $\frac{\partial I(D;B)}{\partial g_1} = -\frac{\partial I(D;B)}{\partial g_2}$ need to be proved. From (5.5),

$$\begin{aligned}\frac{\partial I(D;B)}{\partial g_1} =& \int_{-\infty}^{\infty} \frac{p_1 p_2 e^{-(d+g_1-g_2)^2} 2(g_1 - g_2 + d)}{\sqrt{\pi} \ln 2(p_1 + p_2 e^{(g_1-g_2)(g_1-g_2+2d)})} dd \\ &- \int_{-\infty}^{\infty} \frac{2 p_1 p_2 e^{-(d+g_2-g_1)^2} 2(g_2 - g_1 + d)}{\sqrt{\pi} \ln 2(p_2 + p_1 e^{(g_2-g_1)(g_2-g_1+2d)})} dd \\ =& -\frac{\partial I(D;B)}{\partial g_2}\end{aligned}$$

This proposition shows that the constellation not only minimizes bit error rates [93] [106] but also maximizes the capacity. The optimal values of constellation points are:

$$g_{1_o} = \sqrt{\frac{E_s}{N_0} \frac{p_2}{p_1}} \tag{5.7}$$

$$g_{2_o} = -\sqrt{\frac{E_s}{N_0} \frac{p_1}{p_2}} \tag{5.8}$$

$$G_o = g_{1_o} - g_{2_o} = \sqrt{\frac{E_s}{N_0}} \left(\sqrt{\frac{p_1}{p_2}} + \sqrt{\frac{p_2}{p_1}} \right) \tag{5.9}$$

where E_s is the average energy per transmitted symbol. The optimal constellation to every single bit is solely dependent on its prior probability distribution when E_s is fixed. The scheme used computing prior probabilities of parity check bits is identical to [103].

$Proposition$ 2: When $E_s/N_0 \to \infty$, the capacity, $I(D, B)$, converges to $H(D)$ regardless signal constellations where $H(D) = \sum_{m=1}^{M} -p_m \log_2(p_m)$.

$Proof$: Putting $\{g_{m_0}\}$ (with distinct M_0 values and assume $M_0 \geq 2$) in order from least to greatest, a new sequence of a total of $M_0 \leq M$ elements, $\{g'_1, g'_2, \cdots, g'_{M_0}\}$, where $g'_{m_0} < g'_{m_1}$ if $m_0 < m_1$ can be arranged. Ordering $\{p_{m_0}\}$ accordingly and combining probabilities together for those points with same d_{m_0}. It is easy to show that $H(D) = H(D')$, $H(D;B) = H(D';B)$, where simplifying Equation (5.2).

$$
\begin{aligned}
I(D';B) = & \sum_{m_0=1}^{M_0} \int_{-\infty}^{\infty} -\frac{p'_{m_0}\exp(-b_0^2)}{\sqrt{\pi}} \\
& \log_2\left(p'_{m_0} + \sum_{\substack{m_1=1 \\ m_1 \neq m_0}}^{M} p_{m_1} \exp\left((g'_{m_1} - g'_{m_0})(2b_0 - g'_{m_1} + g'_{m_0}) \right) \right) db_0 \\
\leq & \sum_{m_0=1}^{M_0} \int_{-\infty}^{\infty} -\frac{p'_{m_0}\exp(-b_0^2)}{\sqrt{\pi}} \log_2(p'_{m_0}) db_0 = H(D') \qquad (5.10)
\end{aligned}
$$

$$
\lim_{\frac{E_s}{N_0}\to\infty} I(D';B) = \lim_{|g'_{m_0} - g'_{m_1}|\to\infty} I(D';B) = H(D')
$$

Message Passing with Non-equiprobable Symbols and Nonlinear Codes

Starting from the simplest example of message passing for non-equiprobable symbols, we have two information bits and one parity check bit, $\mathbf{s} = [I_1, I_2, O] = [s_1, s_2, s_3]$ and $O = I_1 \bigoplus I_2$. Let v_k represent the message being propagated from variable node k to the check node, while u_{1k} represent the message being propagated from the check node to variable node k, where $k = 1, 2, 3$ in this case. Hereafter, the superscript denotes iteration number, and $(.)^{(0)}$ is the initial value of $(.)$.

Define

$$
v_k^{(0)} = \ln \frac{p(I_k = 0|b_k)}{p(I_k = 1|b_k)} = \ln \frac{p(d_{k,1}|b_k)}{p(d_{k,2}|b_k)} \qquad (5.11)
$$

for $k = 1, 2$, both are initial messages propagated from s_1, s_2 to the check node. Now, the message propagated from check node to the parity check bit s_3 is

$$u_{13}^{(1)} = \ln \frac{p(O=0|b_1b_2)}{p(O=1|b_1b_2)} = \ln \frac{p(d_{3,1}|b_1b_2)}{p(d_{3,2}|b_1b_2)} \tag{5.12}$$

which can be calculated as

$$\tanh \frac{u_{13}^{(1)}}{2} = \tanh \frac{v_1^{(0)}}{2} \tanh \frac{v_2^{(0)}}{2} \tag{5.13}$$

The posterior probability ratio is

$$\ln \frac{p(d_{3,1}|b_1b_2b_3)}{p(d_{3,2}|b_1b_2b_3)} = \ln \frac{p(b_3|d_{3,1})}{p(b_3|d_{3,2})} + u_{13}^{(1)} \tag{5.14}$$

$$v_3^{(0)} = \ln \frac{p(b_3|d_{3,1})}{p(b_3|d_{3,2})} \tag{5.15}$$

which is not identical to $\ln \frac{p(d_{3,1}|b_3)}{p(d_{3,2}|b_3)}$ for non-equiprobable symbols.

$$\ln \frac{p(d_{3,1}|b_1b_2b_3)}{p(d_{3,2}|b_1b_2b_3)} = v_3^{(0)} + u_{13}^{(1)} \tag{5.16}$$

Similarly the message propagated from check node to the k-th information bit is defined as

$$\tanh \frac{u_{1k}^{(1)}}{2} = \tanh \frac{v_3^{(0)}}{2} \tanh \frac{v_{3-k}^{(0)}}{2} \tag{5.17}$$

for $k = 1, 2$. These steps demonstrate that information bits and parity check bits for non-equiprobable symbols should be treated separated in message passing algorithm.

In general for all information bits, we have

$$v_k^{(0)} = \ln \frac{p(d_{k,1}|b_k)}{p(d_{k,2}|b_k)} \tag{5.18}$$

which is the initial message of information bits, and

$$v_k^{(0)} = \frac{(2d_{k,1} - 2d_{k,2})b_k + d_{k,2}^2 - d_{k,1}^2}{N_0} + \ln \frac{p_{k,2}}{p_{k,1}} \tag{5.19}$$

where $1 \leq k \leq K - T$. And for all parity check bits,

$$v_k^{(0)} = \ln \frac{p(b_k|d_{k,1})}{p(b_k|d_{k,2})} \tag{5.20}$$

which is the initial message for parity check bits, and

$$v_k^{(0)} = \frac{(2d_{k,1} - 2d_{k,2})b_k + d_{k,2}^2 - d_{k,1}^2}{N_0} \tag{5.21}$$

where $K - T + 1 \leq k \leq K$. $\ln \frac{p_{k,2}}{p_{k,1}}$ is termed as biased term.

The iterative decoding process is identical to traditional message passing algorithm,

$$\tanh \frac{u_{tk}^{(l)}}{2} = \prod_{\substack{q=1 \\ q \neq k}}^{d_c} \tanh \frac{v_{tq}^{(l-1)}}{2} \tag{5.22}$$

and

$$v_k^{(l)} = \sum_{q=1}^{d_v} u_{qk}^{(l-1)} + u_k^{(l)} \tag{5.23}$$

where $1 \leq k \leq K$, d_v is the degree of each variable node, s nodes, and d_c is the degree of each check node, **C** nodes.

In next section, improvement due to this modification will be displayed.

Numerical Procedure for Simulation

Optimal Constellation

LDPC codes with binary bits where $M = 2$ and $\{p_k\} = [0.3, 0.7]$ is selected. In Fig. 5.2, the maximum achievable rate is plotted as a function of E_s/N_0 for five cases: (a) AWGN channel, (b) binary input AWGN (BIAWGN) channel with equiprobable symbols and optimal constellations, (c) BIAWGN channel with equiprobable symbols and constellations $\{d_k\} = \pm\sqrt{E_s}$, (d) BIAWGN channel with non-equiprobable symbols ($\{p_1\} = 0.3$) and optimal constellation, $d_1 = \sqrt{E_s \frac{p_2}{p_1}}$, $d_2 = -\sqrt{E_s \frac{p_1}{p_2}}$, (e) BIAWGN channel with non-equiprobable symbols ($\{p_1\} = 0.3$) and constellations $\{d_k\} = \pm\sqrt{E_s}$. When the value of E_s/N_0 is comparatively high, the rate for cases (b) and (c) converge to 1, while the rate of cases (d) and (e) converge to $H(p_1)$ as proved in $Proposition$ 2. When the value of E_s/N_0 is comparatively small, the rate with optimal constellation of case(d) converges to that of case (b), but the rate of non-optimal constellation of case(e) does not converge with other cases. The loss of E_s/N_0 due to non-optimal constellation is approximately 0.72 dB.

Simulation of short LDPC Codes

5 code cases given in Table 5.1 are chosen in simulation. A regular-(3, 6) LDPC code with the length of 1024 is used. Bit error rate is plotted as a function of E_s/N_0 for cases 1 to 4 in Fig. 5.3. By comparing case 1 and case 2, the application of optimal constellation over equal-spaced constellation could gain 0.4 dB. Moreover, the gap between case 1 and case 3 is due to the existence of biased term $\ln(p_1/p_2)$ or $\ln(p_2/p_1)$ in case 3. Besides revising the constellation, sum-product algorithm also needs to be revised when the prior probabilities of input symbols are non-equiprobable, the process is described in previous Section.

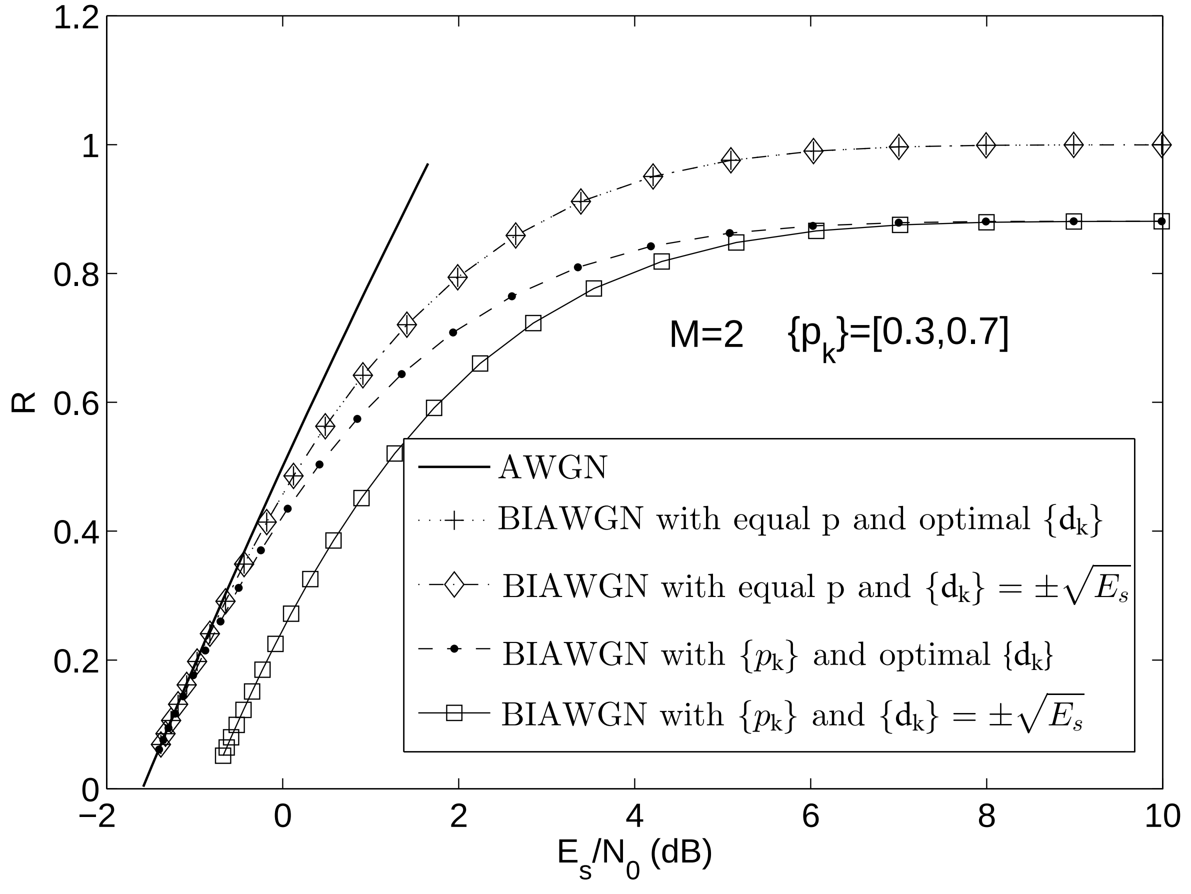


Figure 5.2: R of Systems with M=2, Equiprobable and Non-equiprobable ($\{p_k\}$=[0.3, 0.7]) Symbols as a Function of E_s/N_0

A comparison is made among the five cases in terms of energy per information bit. For codes in cases 1-3, each transmitted bit delivers $H(0.3) = 0.88$ information bits. Fig. 5.4 shows that case 5 has approximately 0.2dB gain compared with case 1. However, such a gain can only be achieved with optimal source coding.

Codes with $K = 1000$ for $p_1 = 0.1$ and 0.2 are also simulated. Our results match to those in [103].

Table 5.1: Simulation Cases

cases	p_1	constellation	code rate	(K - T, K)
1	0.3	optimal	1/2	(512, 1024)
2	0.3	$(\sqrt{E_s}, -\sqrt{E_s})$	1/2	(512, 1024)
3	0.3	optimal without biased term	1/2	(512, 1024)
4	0.5	-	1/2	(512, 1024)
5	0.5	-	0.44	(511, 1161)

Summary

In this Chapter, selection and computation of optimal constellations for LDPC codes in order to maximize channel mutual information was reviewed. The message passing from information bits to parity check bits and from parity check bits to information bits were computed as well. A gain of 0.72 dB in performance can be achieved if optimal constellation $(\sqrt{E_s p_1/p_2}, -\sqrt{E_s p_2/p_1})$ is used instead of $(-\sqrt{E_s}, \sqrt{E_s})$.

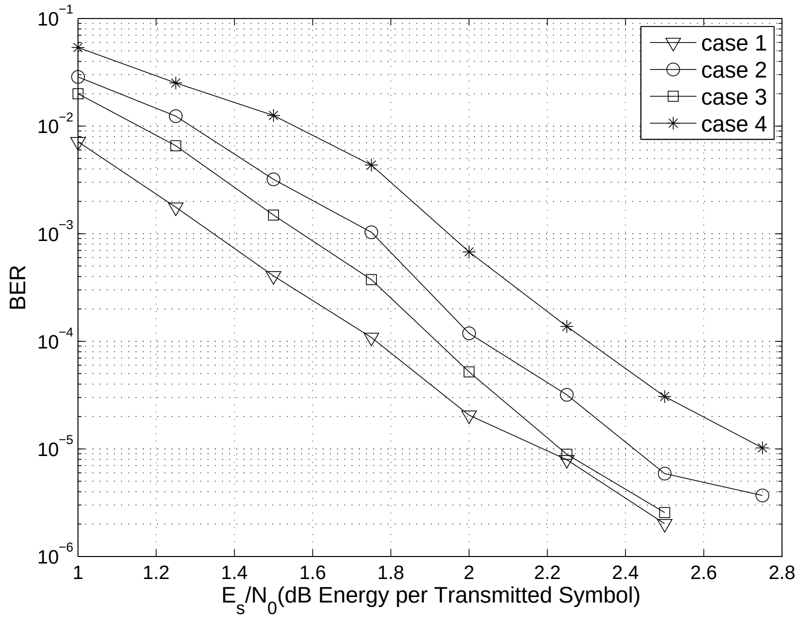


Figure 5.3: Bit Error Rate of (3, 6) LDPC Codes for Case 1 to Case 4 as a Function of E_s/N_0

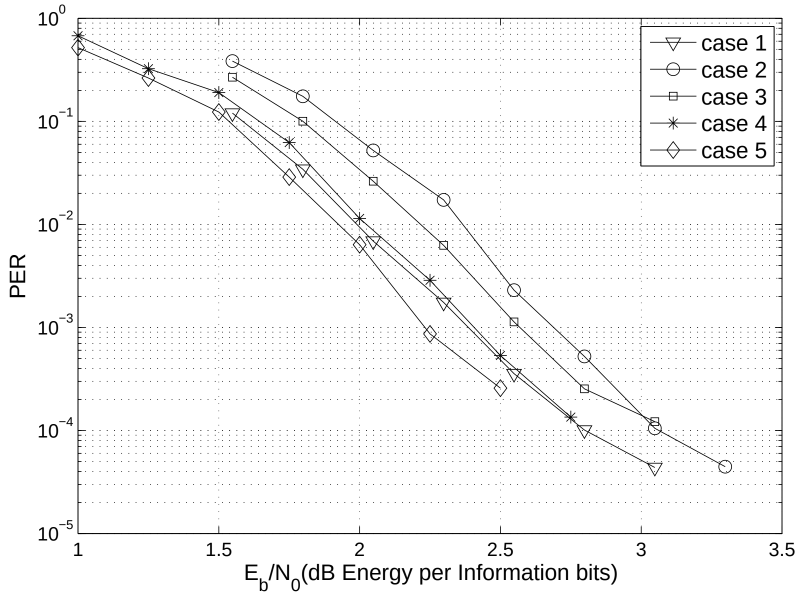


Figure 5.4: Package Error Rate of (3, 6) LDPC Codes for Case 1 to Case 5 as a Function of E_b/N_0

CHAPTER 6: IMAGE PROCESSING UNIT FOR GENERAL PURPOSE REPRESENTATION AND ASSOCIATION MACHINE

In this chapter, a novel framework which supports multiple recognition tasks and includes VIV is presented [110]. This framework is an IPU which consists of a randomly constructed LDPC coding structure, an iterative decoder, a switch, scaling and decision device. VIV is introduced to degrade the quality of input images which mimics human visual systems. The degraded images are then fed to the IPU.

Each task contains a pool of candidate images. At training step, these images are inputted into IPU to generate reusable information which will be utilized in testing step. At testing step, a randomly selected candidate is recognized through the IPU with the information generated in previous step. These tasks include digits recognition, alphabetical letters recognition, roman numerals recognition, hyper-acuity task and low-resolution human face recognition.

Three aspects are the focus of our study in this Chapter and listed as follow.

1. Whether the IPU unit can recognize different candidates in a VIV environment.
2. Whether the IPU unit is capable of hyper-acuity task.
3. Whether randomly constructed code can improve the differentiation capability.

Based on GPRAM principles, the non-optimized LDPC coding structure is totally randomly constructed which might lay a foundation for individuality of different units. The results presented latter in this Chapter provide promising answers to these aspects.

Structure of IPU for GPRAM

An IPU for GPRAM structure is introduced in this Section and a testable platform is founded due to the superior capability of LDPC code dealing with internal noise in the system [62]. Fig. 6.1 illustrates different modules for IPU.

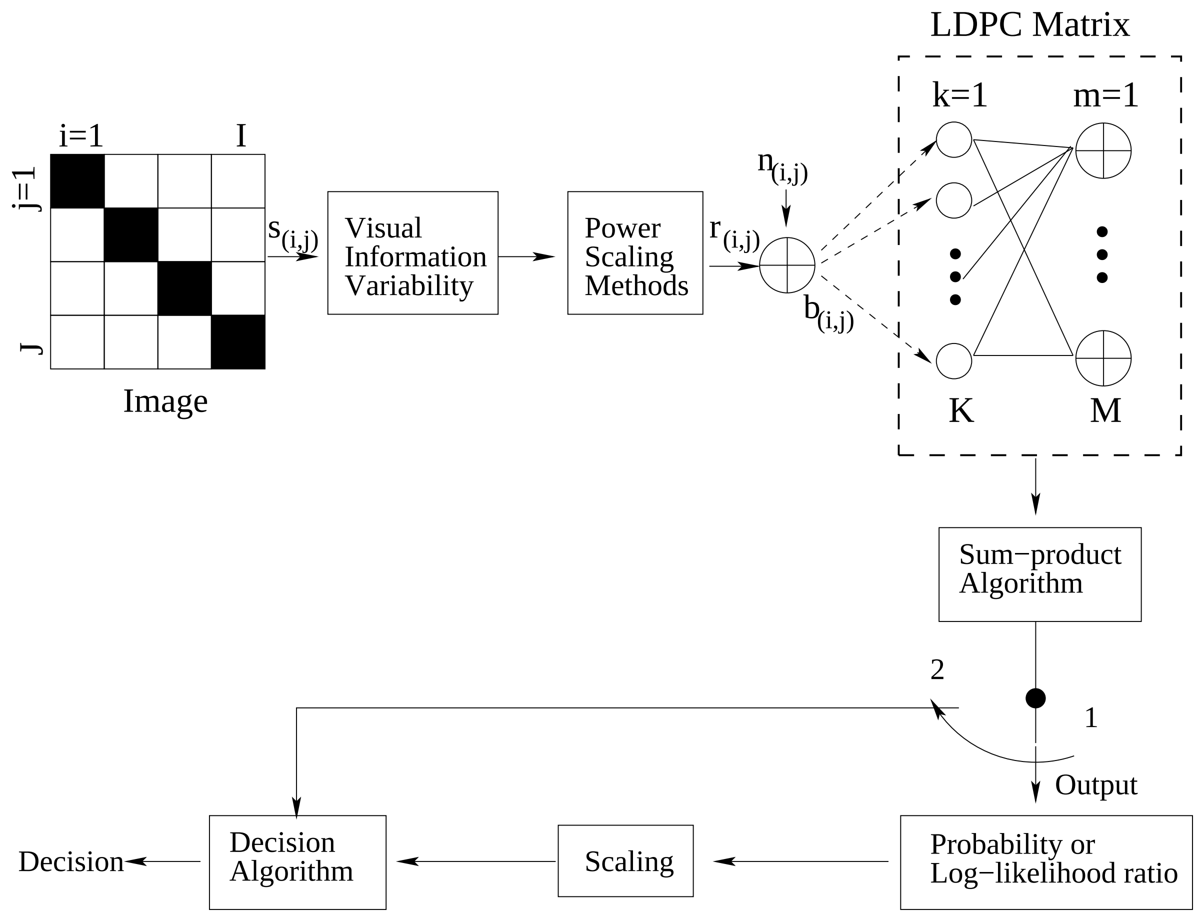


Figure 6.1: Structure of IPU for GPRAM for One Frame

Each pixel in an input image is connected to a photo-sensor unit in IPU (e.g., s_1, s_2, $\cdots$). Hence for a picture with K-pixels, there are K sensor units in the system. For a gray scaled picture, the value for each pixel is ranging from 0 to 255. These pixels can be normalized into a range of [0,1] when divided by 255. To better explain the structure of IPU, simpler images -binary pictures- are assumed as inputs to the unit in this and the next section. Furthermore, the value for a pixel in an image with dimensions of I by J can be represented by $s_{(i,j)}$ or s_k, which locates at the i-th row and j-th column, and $k = (i-1) \times I + j, 0 \leq i \leq I, 0 \leq j \leq J$.

The IPU is shown in Fig. 6.1. For every task, the switch is first at position 1 for the system to compute a template for every input image in an image set, which could be a tuple with length K of either log-likelihood ratios (LLR) or measured probabilities of variable nodes. The system performs the real differentiation task while the switch is at position 2. Uncoded cases can be obtained by removing the LDPC coding structure for comparison, which is shown as the dashed box in Fig. 6.1.

The detailed transformation from s_k to r_k under VIV will be discussed in next Section. Without VIV, $r_k = s_k$. Assuming the input to unit is without influence of VIV, then black pixel is mapped to 0 and white pixel is mapped to 1 which are latter modulated to $\{-1, 1\}$, respectively; and AWGN noise is added to the pixels to generate inputs to the LDPC coding structure where we have $b_k^{(l)} = n_k^{(l)}$ if the variable node k is not connected to any sensor input and $b_k^{(l)} = 1 + n_k^{(l)}$ or $b_k^{(l)} = -1 + n_k^{(l)}$ if node k is connected to a sensor input and the input value is 1 or 0, respectively. $n_k^{(l)}$ is Gaussian distributed with zero mean and variance of σ_1^2, $n_k^{(l)} \sim \mathcal{N}(0, \sigma_1^2)$, and (l) indicates the iteration number. Each iteration uses one frame of image information.

Define $u_k^{(l)} = 2b_k^{(l)}/\sigma_2^2$ as the initial LLR of variable node k. The value of σ_2 is chosen to be different than the value of σ_1 and $\sigma_2 > \sigma_1$ is enforced for robustness purposes [111]. A sum-product algorithm is used to propagate messages as in conventional iterative decoding [112] [113] [114],

$$v_{kt}^{(l)} = \begin{cases} u_k^{(l)} & l = 0 \\ u_k^{(l)} + \sum_{\substack{q=1 \\ q \neq t}}^{d_v} u_{qk}^{(l)} & l > 0 \end{cases} \tag{6.1}$$

$$\tanh \frac{u_{tk}^{(l)}}{2} = \prod_{\substack{q=1 \\ q\neq k}}^{d_c} \tanh \frac{v_{tq}^{(l-1)}}{2} \tag{6.2}$$

where $v_{kt}^{(l)}$ is the LLR from the variable node k to the check node t, $u_{tk}^{(l)}$ is the LLR from the check node m to the variable node k, and d_v and d_c are the degrees of the variable and check nodes, respectively. At the end of each iteration, the status of each variable node is decided by

$$c_k^{(l)} = \begin{cases} 1 & v_k^{(l)} > 0 \\ 0 & v_k^{(l)} \leq 0 \end{cases} \tag{6.3}$$

where $v_k^{(l)} = u_k^{(l)} + \sum_{q=1}^{d_v} u_{qk}^{(l-1)}$.

Visual Information Variability

Line-spread functions [115] is applied to evaluate the distribution of luminance on the retina, which can be obtained by summing two Gaussian functions [116] as an approximation. The same parameters used in [65] for a 3-mm pupil are strongly related to the data from bio-visual systems [115], being applied in the system.

$$\begin{aligned} h(x,y) = {} & \frac{a_1}{2\pi a_3} \exp\left[-\frac{0.5(x^2+y^2)}{a_3^2}\right] \\ & + \frac{a_2}{2\pi a_4} \exp\left[-\frac{0.5(x^2+y^2)}{a_4^2}\right] \end{aligned} \tag{6.4}$$

where the coefficients $a_1 = 0.417$ and $a_2 = 0.583$ and the variances $a_3 = 0.443\beta_1$ and $a_4 = 2.04\beta_1$; the intensity of this movement is denoted as β_1.

The second factor is drift-type fixational motion. The drifting pattern and rate are strongly dependent on the observer and their status and might vary from person to person [117]. The drifting, on the other hand, is uncorrelated between (a) horizontal and vertical directions, and (b) two eyes [118]. The drift-type motion is therefore modeled, as follows:

$$\begin{aligned} x_0(\tau) &= g(\tau) \bigotimes n_x(\tau) \\ y_0(\tau) &= g(\tau) \bigotimes n_y(\tau) \end{aligned} \tag{6.5}$$

where $\bigotimes$ denotes the linear convolution operation, $g(\tau)$ is the impulse response, and n_x and n_y are independent Gaussian noise vectors with zero mean and unitary variance. This model fits well with the observation of the human visual system in [119] [120]. The effect of fixation eye movements is explained in [121]. An examination of this model is conducted in [65].

The form of $g(\tau)$ is

$$g(\tau) = \left(\frac{40}{\sqrt{2\pi} a_5} \exp\left[-\frac{\tau^2}{2a_5^2} \right] - 0.0117 \right) \cos(4\pi\tau/1000) \tag{6.6}$$

where $a_5 = 223.61\beta_2$ ms and β_2 is the scale factor.

The third one is orientational movement, which causes rotation along clockwise and counterclockwise directions. This motion imitates side-wise head tilting. A simple filter is used:

$$\theta(\tau) = 0.95\theta(\tau - 1) + 0.05 n_0(\tau) \frac{\pi}{9} \beta_3 \tag{6.7}$$

where $n_0(\tau) \sim \mathcal{N}(0, 1)$ and the initial value of $\theta(\tau)$, $\theta(0)$, is a uniformly distributed random variable in the interval of $[-20°, 20°]$.

Three scale factors, β_1, β_2, and β_3, denote the intensities for all three movement. The higher a value for a factor, the more intense that movement is.

All the previous processes are described in continuous domains. Discrete models of these three have to be studied in order to be implemented in our unit which will be introduced in following Sections.

Point spread

Define

$$H(x,y) = s^{(l)}(x,y) \bigotimes h(x,y) \tag{6.8}$$

where $\bigotimes$ denotes convolution. After that, the discrete values for $H(x,y)$ can be computed as

$$H^{(l)}_{(i,j)} = \int_{(i-1)h_s}^{ih_s} \int_{(j-1)h_s}^{jh_s} H(x,y)dxdy. \tag{6.9}$$

Alternatively, discrete convolution operation is defined as in

$$H^{(l)}_{(i,j)} = s^{(l)}_{(i,j)} \bigotimes h(i,j) \tag{6.10}$$

where $h(i,j) = h(ih_s - 1/2h_s, jh_s - 1/2h_s)$.

Drift-like motion

The drift-like motion is modeled as described in Equation (6.5). The discrete model for drift-like motion is

$$
\begin{aligned}
x_0^{(l)} &= \left\lfloor \frac{x_0(lT_s)}{\beta_2 h_s} \right\rfloor \\
y_0^{(l)} &= \left\lfloor \frac{y_0(lT_s)}{\beta_2 h_s} \right\rfloor
\end{aligned} \tag{6.11}
$$

where T_s is the time duration between two frames and $\lfloor x \rfloor$ takes the largest integer smaller than or equal to x.

Orientation movement

Orientation is represented as

$$
f_{(i,j)}^{(l)} = H_{(i_1+x_0^{(l)},j_1+y_0^{(l)})}^{(l)} \tag{6.12}
$$

where

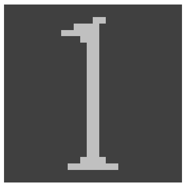
$$
\begin{aligned}
i &= \lfloor i_1 \cos(\theta(t)) - j_1 \sin(\theta(t)) \rfloor \\
j &= \lfloor i_1 \sin(\theta(t)) + j_1 \cos(\theta(t)) \rfloor
\end{aligned} \tag{6.13}
$$

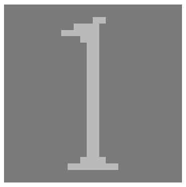
This movement can change the orientation of input figures.

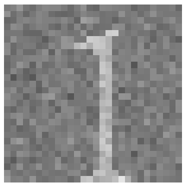
Fig. 6.2 illustrates the effects of these VIV movements adding upon an image of standard digit “1”. σ_1 is typically set to 1 in this study, which is far noisier than what can be observed in Fig.6.2(t).

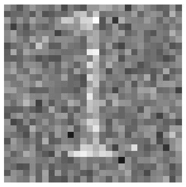
Power Scaling Methods

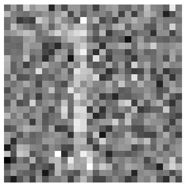
The pixel values significantly vary from frame to frame due to the effect of VIV. Therefore, each frame is supposed to be scaled into a range before inputted into the system. There are four scaling methods.

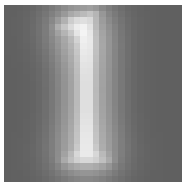

(a) $\beta_1 = 0$, $\beta_2 = 0$, $\beta_3 = 0$, $\sigma_1 = 0$

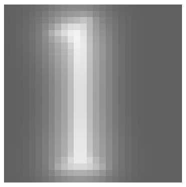

(b) $\beta_1 = 0$, $\beta_2 = 1$, $\beta_3 = 0$, $\sigma_1 = 0$

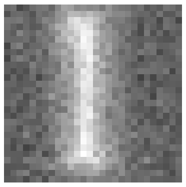

(c) $\beta_1 = 0$, $\beta_2 = 1$, $\beta_3 = 0$, $\sigma_1 = 0.2$

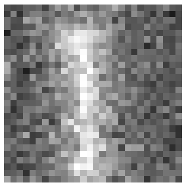

(d) $\beta_1 = 0$, $\beta_2 = 1$, $\beta_3 = 0$, $\sigma_1 = 0.4$

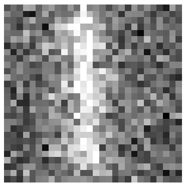

(e) $\beta_1 = 0$, $\beta_2 = 1$, $\beta_3 = 0$, $\sigma_1 = 0.6$

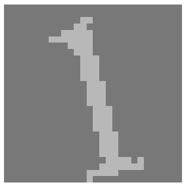

(f) $\beta_1 = 1$, $\beta_2 = 0$, $\beta_3 = 0$, $\sigma_1 = 0$

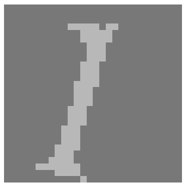

(g) $\beta_1 = 1$, $\beta_2 = 1$, $\beta_3 = 0$, $\sigma_1 = 0$

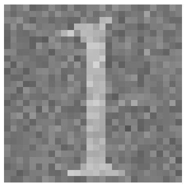

(h) $\beta_1 = 1$, $\beta_2 = 1$, $\beta_3 = 0$, $\sigma_1 = 0.2$

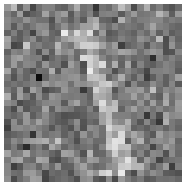

(i) $\beta_1 = 1$, $\beta_2 = 1$, $\beta_3 = 0$, $\sigma_1 = 0.4$

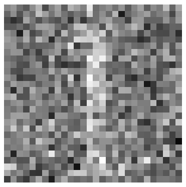

(j) $\beta_1 = 1$, $\beta_2 = 1$, $\beta_3 = 0$, $\sigma_1 = 0.6$

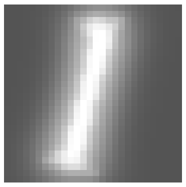

(k) $\beta_1 = 0$, $\beta_2 = 0$, $\beta_3 = 1$, $\sigma_1 = 0$

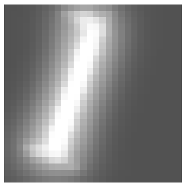

(l) $\beta_1 = 0$, $\beta_2 = 1$, $\beta_3 = 1$, $\sigma_1 = 0$

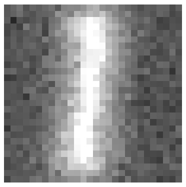

(m) $\beta_1 = 0$, $\beta_2 = 1$, $\beta_3 = 1$, $\sigma_1 = 0.2$

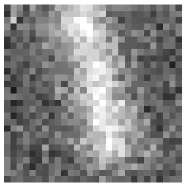

(n) $\beta_1 = 0$, $\beta_2 = 1$, $\beta_3 = 1$, $\sigma_1 = 0.4$


(o) $\beta_1 = 0$, $\beta_2 = 1$, $\beta_3 = 1$, $\sigma_1 = 0.6$


(p) $\beta_1 = 1$, $\beta_2 = 0$, $\beta_3 = 1$, $\sigma_1 = 0$


(q) $\beta_1 = 1$, $\beta_2 = 1$, $\beta_3 = 1$, $\sigma_1 = 0$


(r) $\beta_1 = 1$, $\beta_2 = 1$, $\beta_3 = 1$, $\sigma_1 = 0.2$


(s) $\beta_1 = 1$, $\beta_2 = 1$, $\beta_3 = 1$, $\sigma_1 = 0.4$

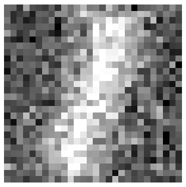

(t) $\beta_1 = 1$, $\beta_2 = 1$, $\beta_3 = 1$, $\sigma_1 = 0.6$

Figure 6.2: Effects of Point Spread (β_1), Drift-like Motion (β_2), Head Orientation Rotation (β_3) and Gaussian Noise (σ_1)

Method 1: No mean nor energy normalization; output from VIV is the input to the LDPC code, where $r^{(l)}_{(i,j)} = f^{(l)}_{(i,j)}$.

Method 2: This is the mean normalization, as defined by

$$m^{(l)} = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} f^{(l)}_{(i,j)}}{I \times J} \tag{6.14}$$

$$q^{(l)}_{(i,j)} = f^{(l)}_{(i,j)} - m^{(l)} \tag{6.15}$$

Zero can then be considered as a watershed for dividing all pixels into two sets after subtracting their mean value. The mean value for each group can be found as:

$$\begin{aligned} m^{(l)}_p &= \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} q^{(l)}_{(i,j)}}{\sum_{i=1}^{I}\sum_{j=1}^{J} Z\left(q^{(l)}_{(i,j)} > 0\right)} \text{ if } q^{(l)}_{(i,j)} > 0 \\ m^{(l)}_n &= \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} q^{(l)}_{(i,j)}}{\sum_{i=1}^{I}\sum_{j=1}^{J} Z\left(q^{(l)}_{(i,j)} \leq 0\right)} \text{ if } q^{(l)}_{(i,j)} \leq 0 \end{aligned} \tag{6.16}$$

where

$$Z(x) = \begin{cases} 1 & x \text{ is true} \\ 0 & x \text{ is false} \end{cases} \tag{6.17}$$

Finally,

$$r^{(l)}_{(i,j)} = \frac{q^{(l)}_{(i,j)}}{\min(m^{(l)}_p, |m^{(l)}_n|)} \tag{6.18}$$

$$b^{(l)}_{(i,j)} = r^{(l)}_{(i,j)} + n^{(l)}_{(i,j)} \tag{6.19}$$

where $n^{(l)}_{(i,j)} \sim \mathcal{N}(0, \sigma_1^2)$. This step is to ensure the group with a smaller mean, i.e., $\min(m^{(l)}_p, |m^{(l)}_n|)$ should be higher than a lower bound lest the decoder generate irregular results with all zeros or all ones.

Method 3: The value of $r^{(l)}_{(i,j)}$ is further scaled per energies of both positive and negative groups following the mean-normalization described in $Method\ 2$.

$$E^{(l)}_p = \sum_{i=1}^{I}\sum_{j=1}^{J}\left[q^{(l)}_{(i,j)}\right]^2 \text{ if } q^{(l)}_{(i,j)} > 0$$
$$E^{(l)}_n = \sum_{i=1}^{I}\sum_{j=1}^{J}\left[q^{(l)}_{(i,j)}\right]^2 \text{ if } q^{(l)}_{(i,j)} \leq 0 \tag{6.20}$$

Define $S^{(l)}_p$ and $S^{(l)}_n$ as scaling factors to the positive and negative groups

$$\begin{aligned} S^{(l)}_p &= \min\left(1, \sqrt{\frac{E^{(l)}_n}{E^{(l)}_p}}\right) \\ S^{(l)}_n &= \min\left(1, \sqrt{\frac{E^{(l)}_p}{E^{(l)}_n}}\right) \end{aligned} \tag{6.21}$$

In addition, each $r^{(l)}_{(i,j)}$ is updated as

$$r^{(l)}_{(i,j)} = \begin{cases} q^{(l)}_{(i,j)} \times S_p(l) & q^{(l)}_{(i,j)} > 0 \\ q^{(l)}_{(i,j)} \times S_n(l) & q^{(l)}_{(i,j)} \leq 0 \end{cases} \tag{6.22}$$

This step is to ensure the energies for both groups are the same. Therefore, the iterative decoder will not fail to converge for given σ_1^2.

Method 4: It is observed that the total energy drastically varies for each digit after applying $Method\ 2$. In order to figure out the impact on this variation, we set $S^{(l)}_p$ and $S^{(l)}_n$ based on

$$S^{(l)}_p = \sqrt{\frac{784}{E^{(l)}_p}}$$
$$S^{(l)}_n = \sqrt{\frac{784}{E^{(l)}_n}} \tag{6.23}$$

Therefore, the new $E_p^{(l)}$ and $E_n^{(l)}$ for each candidate under any condition will be identical after scaling. $r_{(i,j)}^{(l)}$ can be obtained by replacing Equation (6.23) by Equation (6.22) to obtain for $Method\,4$. Conclusively $Method\,1$ excludes the scaling process, which is for the purpose of comparison, and $Method\,2$ is the prerequisite for both $Method\,3$ and $Method\,4$.

Algorithms

In this section, learning and testing procedures will be introduced, for the task of standard digits recognition. The procedures can be transplanted to other tasks with adjusting the number of candidates only. z is denoted as a digit, referring to "0", "1", "2", $\cdots$ "9". Each digit is considered as a 28-by-28 pixels image, where each pixel is granted to binary values. It is worth mentioning that the values would no longer be binary but with gray level after the involvement of VIV.

Training

The image stream will be input frame by frame as an LDPC code and decoded by a sum-product algorithm. The image matrix is then converted into an image vector, by relocating pixels (i, j) in the matrix at $(i-1) \times J + j$ in the vector. Each digit will be consecutively learned. Two different outputs can be obtained: the averaged LLR (i.e., the soft decision, given in Equation (6.24)) and the multiplicity (i.e., the hard decision, given in Equation (6.25)) of each pixel.

As for the first type, the averaged likelihood of a bit (784 in total in our case) is the mean of $v_k^{(l)}$ over $I_a - 20$ iterations (I_a is 320 in program). Only the last 300 iterations will be counted in lest of error propagation.

$$v_k(z) = \frac{\sum_{l=21}^{I_a} v_k^{(l)}}{I_a - 20} \tag{6.24}$$

where z refers to the digits “0”, “1”, $\cdots$, “9” and $v_k(z)$ denotes the averaged likelihood of the k variable in the case of digit z.

For the second type, the status of a bit will be decided at each iteration, i.e., making a hard decision, which is $c_k^{(l)}(z)$. A probability of 0 and 1 can be calculated for each bit by dividing the occurrences of zeros and ones by the number of iterations, after I_a iterations.

$$
\begin{aligned}
p(c_k = 0|z) &= \frac{\sum_{l=21}^{I_a} Z\left(c_k^{(l)} = 0\right)}{I_a - 20} \\
p(c_k = 1|z) &= \frac{\sum_{l=21}^{I_a} Z\left(c_k^{(l)} = 1\right)}{I_a - 20}
\end{aligned} \tag{6.25}
$$

where $p(c_k = 0|z)$ denotes the probability of 0 for the variable k in the case of digit z.

Algorithm 1 summarizes the training step of our proposed IPU.

Algorithm 1 Training Step

```
1:  for $z \leftarrow 1, 10$ do
2:      Add VIV to image $z$
3:      Scale image $z$ with a method discussed in Section 6
4:      Initialize $u_k^{(0)} = 2b_k^{(0)}/\sigma_2^2$, $u_{tk}^{(0)} = 0$, $v_{kt}^{(0)} = u_k^{(0)}$
5:      for $l \leftarrow 1, I_a$ do
6:          $v_{kt}^{(l)} = u_k^{(l)} + \sum_{\substack{q=1 \\ q \neq t}}^{d_v} u_{qk}^{(l)}$, $u_{tk}^{(l)} = 2 \times \arctan \prod_{\substack{q=1 \\ q \neq k}}^{d_c} \tanh \frac{v_{tq}^{(l-1)}}{2}$
7:          Store the value of $c_k^{(l)}$ based on Equation (6.3)
8:      end for
9:      Output $p(c_k = 0)$ and $p(c_k = 1)$ or $v_k$ for image $z$
10: end for
```

Testing

Scaling for LLR

The 4 scaling methods described in previous steps are applied to images. And to determine whether taking away the mean value of $\mathbf{L}(z)$ and scaling it based on the energy of positive group and negative group will effect the performance of system, we studied three scaling methods of LLR: *Method* 1 does not have mean or energy normalization for the purpose of comparison; *Method* 2 has mean normalization, but no-energy normalization; *Method* 3 has both mean and energy normalization.

Method 1: This step is exactly the same as in Equation (6.35)(6.36).

Method 2: Mean value of $\mathbf{L}(z)$ for each z from each pixel is deducted.

Define

$$m_L(z) = \frac{\sum_1^K L_k(z)}{K} \tag{6.26}$$

as mean of likelihood. And the adjusted LLR is

$$L'_k(z) = L_k(z) - m_L(z) \tag{6.27}$$

for every $1 \leq k \leq K$.

Substitute **L(z)** in Equation (6.36) with $\mathbf{L}'_\mathbf{k}(\mathbf{z})$

$$\begin{aligned}
Decision &= \arg\min_{0\le z\le 9}\sum_{l=1}^{I_l}\sum_{k=1}^{K}\left[L'_k(z)-v_k^{(l)}\right]^2\\
&= \arg\min_{0\le z\le 9}\sum_{l=1}^{I_l}\sum_{k=1}^{K}\left[L_k(z)-m_L(z)-v_k^{(l)}\right]^2\\
&= \arg\min_{0\le z\le 9}\sum_{l=1}^{I_l}\left\{\sum_{k=1}^{K}\left[(L_k(z)-v_k^{(l)})^2-2m_L(z)(L_k(z)-v_k^{(l)})\right]+Km_L(z)^2m_L(z)\right\}\\
&\Rightarrow \arg\min_{0\le z\le 9}\sum_{l=1}^{I_l}\left\{\sum_{k=1}^{K}\left[(L_k(z)-v_k^{(l)})^2\right]+\sum_{k=1}^{K}\left[-2m_L(z)(L_k(z)-v_k^{(l)})\right]\right\} \qquad (6.28)
\end{aligned}$$

If $\sum_{k=1}^{K}\left[(L_k(z)-v_k^{(l)})^2\right] \gg \sum_{k=1}^{K}\left[-2m_L(z)(L_k(z)-v_k^{(l)})\right]$, then the detector will not be affected by $m_L(z)$.

Method 3: After mean-normalization in *Method* 2, we further adjust the value of likelihood based on the total energy of positive group and negative group. For $1 \le k \le K$, $L_k(z)$ is either grater than, or less than or equal to 0.

For $L_k(z)$ which is larger than 0, its subscript belongs to set $I(z)$ of which length is $K_1(z)$. For $L_k(z)$ which is less than or equal to 0, its subscript belongs to set $J(z)$ of which length is $K_2(z)$. $I_\lambda(z)$ is the λ-th element in set $I(z)$. To simplify the notation, let $L_{I_\lambda}(z)$ be the $I_\lambda(z)$-th element of **L(z)**. So we have

$$\begin{aligned}
&L_{I_\lambda}(z) > 0, \text{where } 1 \le \lambda \le K_1(z)\\
&L_{J_\lambda}(z) \le 0, \text{where } 1 \le \lambda \le K_2(z)\\
&K_1(z)+K_2(z)=K\\
&I(z)\cup J(z)=\{1,2,\cdots,K\} \qquad (6.29)
\end{aligned}$$

We compute energy for each group as

$$\begin{aligned} E^{+}(z) &= \sum_{\lambda=1}^{K_1(z)} \left[L_{I_\lambda}(z)\right]^2 \\ E^{-}(z) &= \sum_{\lambda=1}^{K_2(z)} \left[L_{J_\lambda}(z)\right]^2 \end{aligned} \tag{6.30}$$

Take scaling the value of $L_k(z)$ into consideration based on $E^{+}(z)$ and $E^{-}(z)$. By definition

$$\begin{aligned} S^{+}(z) &= \min\left(1, \sqrt{\frac{E^{-}(z)}{E^{+}(z)}}\right) \\ S^{-}(z) &= \min\left(1, \sqrt{\frac{E^{+}(z)}{E^{-}(z)}}\right) \end{aligned} \tag{6.31}$$

Then we have

$$\begin{aligned} L''_{I_\lambda}(z) &= L_{I_\lambda}(z) S^{+}(z) \\ L''_{J_\lambda}(z) &= L_{I_\lambda}(z) S^{-}(z) \end{aligned} \tag{6.32}$$

So after scaling, the energy of $\mathbf{L''_I(z)}$ is equal to the one of $\mathbf{L''_J(z)}$. The reason of this step is because that for each digit, the one with more one (white) in its original digit will have a larger ratio of $E^{+}(z)$ to $E^{-}(z)$. Rewrite Equation (6.36) with the definition in Equation (6.30)

$$\begin{aligned}
Decision &= \arg\min_{0\leq z\leq 9}\sum_{l=1}^{I_l}\left\{\sum_{\lambda=1}^{K_1(z)}\left[L_{I_\lambda}(z)-v_{I_\lambda}^{(l)}\right]^2+\sum_{\lambda=1}^{K_2(z)}\left[L_{J_\lambda}(z)-v_{J_\lambda}^{(l)}\right]^2\right\}\\
&= \arg\min_{0\leq z\leq 9}\sum_{l=1}^{I_l}\left\{\sum_{\lambda=1}^{K_1(z)}\left[(L_{I_\lambda}(z))^2-2L_{I_\lambda}(z)v_{I_\lambda}^{(l)}+(v_{I_\lambda}^{(l)})^2\right]\right.\\
&\quad\left.+\sum_{\lambda=1}^{K_2(z)}\left[(L_{J_\lambda}(z))^2-2L_{J_\lambda}(z)v_{J_\lambda}^{(l)}+(v_{J_\lambda}^{(l)})^2\right]\right\}\\
&= \arg\min_{0\leq z\leq 9}\sum_{l=1}^{I_l}\left\{E^{+}(z)+E^{-}(z)-2\sum_{\lambda=1}^{K_1(z)}L_{I_\lambda}(z)v_{I_\lambda}^{(l)}-2\sum_{\lambda=1}^{K_2(z)}L_{J_\lambda}(z)v_{J_\lambda}^{(l)}\right\}
\end{aligned} \tag{6.33}$$

Substitute $\mathbf{L_{I_\lambda}(z)}$ and $\mathbf{L_{J_\lambda}(z)}$ in Equation (6.33) with $\mathbf{L''_{I_\lambda}(z)}$ and $\mathbf{L''_{J_\lambda}(z)}$.

$$\begin{aligned}
&Decision\\
&= \arg\min_{0\leq z\leq 9}\sum_{l=1}^{I_l}\left\{\sum_{\lambda=1}^{K_1(z)}\left[L''_{I_\lambda}(z)-v_{I_\lambda}^{(l)}\right]^2+\sum_{\lambda=1}^{K_2(z)}\left[L''_{J_\lambda}(z)-v_{J_\lambda}^{(l)}\right]^2\right\}\\
&= \arg\min_{0\leq z\leq 9}\sum_{l=1}^{I_l}\left\{\sum_{\lambda=1}^{K_1(z)}\left[L_{I_\lambda}(z)S^{+}(z)-v_{I_\lambda}^{(l)}\right]^2.+\sum_{\lambda=1}^{K_2(z)}\left[L_{J_\lambda}(z)S^{-}(z)-v_{J_\lambda}^{(l)}\right]^2\right\}\\
&= \arg\min_{0\leq z\leq 9}\sum_{l=1}^{I_l}\left\{\sum_{\lambda=1}^{K_1(z)}\left[\left(L_{I_\lambda}(z)S^{+}(z)\right)^2-2L_{I_\lambda}(z)S^{+}(z)v_{I_\lambda}^{(l)}+\left(v_{I_\lambda}^{(l)}\right)^2\right]\right.\\
&\quad\left.+\sum_{\lambda=1}^{K_2(z)}\left[\left(L_{J_\lambda}(z)S^{-}(z)\right)^2-2L_{J_\lambda}(z)S^{-}(z)v_{J_\lambda}^{(l)}+\left(v_{J_\lambda}^{(l)}\right)^2\right]\right\}\\
&\Rightarrow \arg\min_{0\leq z\leq 9}\sum_{l=1}^{I_l}\left\{\left(S^{+}(z)\right)^2E^{+}(z)+\left(S^{-}(z)\right)^2E^{-}(z)-2\sum_{\lambda=1}^{K_1(z)}S^{+}(z)v_{I_\lambda}^{(l)}-2\sum_{\lambda=1}^{K_2(z)}S^{-}(z)v_{J_\lambda}^{(l)}\right\}
\end{aligned} \tag{6.34}$$

Testing Procedure

A candidate will be selected from digit “0” to digit “9” and iteratively decoded for I_l iterations. The distances between the LLR of $\mathbf{v}^{(l)}$ (vector of $v_k^{(l)}$) and $\ln \frac{\mathbf{p}(c_k=1|z)}{\mathbf{p}(c_k=0|z)}$, or $\mathbf{v}(z)$, represented as $\mathbf{L}(z)$ in either case, can be computed during each iteration. Therefore, $L_k(z) = v_k(z)$ in Detector 1, whereas $L_k(z) = \ln \frac{p(c_k=1|z)}{p(c_k=0|z)}$ in Detector 2.

Four aforementioned scaling methods in learning step are applied for image inputs, whereas the scaling in this testing step is applied for the LLR computation.

The output from scaling in the testing step remains to be denoted as $\mathbf{L}(z)$. Define $\xi^{(l)}(z)$ as the distance between $\mathbf{v}^{(l)}$ and $\mathbf{L}(z)$ of the z-th digital during the l-th iteration:

$$\xi^{(l)}(z) = \sum_{k=1}^{I*J} \left(v_k^{(l)} - L_k(z)\right)^2 \tag{6.35}$$

The test decision is based on the minimum sum of distances over I_l (30 in our program) iterations.

$$Decision = \arg\min_{0\leq z\leq 9} \sum_{l=1}^{I_l} \xi^{(l)}(z) \tag{6.36}$$

The probability of a correct decision, which matches the input, can be obtained by repeating this testing procedure 10000 times.

Algorithm 2 summarizes the testing step of our proposed IPU.

Algorithm 2 Testing Step

```
1:  for $i \leftarrow 1, 10000$ do
2:      Randomly select an image $z$
3:      Add VIV to image $z$
4:      Scale image $z$ with a method discussed in Section 6
5:      Initialize $u_k^{(0)} = 2b_k^{(0)}/\sigma_2^2$, $u_{tk}^{(0)} = 0$, $v_{kt}^{(0)} = u_k^{(0)}$
6:      for $l \leftarrow 1, I_l$ do
7:          $v_{kt}^{(l)} = u_k^{(l)} + \sum_{\substack{q=1 \\ q\neq t}}^{d_v} u_{qk}^{(l)}$, $u_{tk}^{(l)} = 2 \times \operatorname{arctan} \prod_{\substack{q=1 \\ q\neq k}}^{d_c} \tanh \frac{v_{tq}^{(l-1)}}{2}$
8:          Calculate the distance between $\mathbf{v}^{(l)}$ and $\mathbf{L}(z)$ based on Equation (6.35)
9:      end for
10:     Decide the input image based on Equation (6.36)
11: end for
12: Output error rate
```

Simulation Results and Discussion

The tests are conducted using a computer with an Intel Core2 Duo 2.00 GHz processor, 4 GB of RAM, and 64-bit Windows 7 operating system. Two steps are required to develop an IPU, and performed only once: (a) generating an H matrix and (b) training to obtain reference templates. It takes approximately 15 milliseconds for step (a) and 110 seconds for step (b). The training process requires 320 frames for each digit and 10 digits in total. The time anticipated to perform one recognition test is approximately 1.1 seconds, generating 30 frames of each candidate. It is worth mentioning that all information about VIV is practically intrinsic rather than generated as in this work.

Several randomly constructed (3, 6) LDPC codes with lengths of 784 variable nodes are compared, exhibiting very similar error performance results. Therefore, our IPU is code independent for these tasks. One LDPC code was selected to perform the following tasks.

Performance on 2 digits

First, several pairs of digits are selected, of which one has a similar shape with the other, such as 0 and 6, 0 and 8, 0 and 9, 1 and 4, 1 and 7, 3 and 8, 6 and 9, 7 and 9, 8 and 9. 10000 trials are conducted for each pair with $\sigma_1 = 1$, $\sigma_2 = 1$, and $\beta_1 = \beta_2 = \beta_3 = 1$. The results in Table 6.1 show that the recognition errors of most pairs are approximately zero.

Table 6.1: Performance of Pairwise Detector using Method 3

Pair	occurrence	error rate
0	5009	4.771%
6	4991	4.167%
0	5009	0.180%
8	4991	0.020%
0	4998	4.732%
9	5002	1.423%
1	4955	0.000%
4	5055	0.000%
1	5000	0.000%
7	5000	0.000%
3	5004	0.000%
8	4996	0.006%
6	4978	0.000%
9	5022	0.000%
7	5002	0.000%
9	4998	0.000%
8	5000	0.020%
9	5000	0.060%

Performance on 10 digits

Fig. 6.3 shows the digit images used in this task. Two types of detectors are defined in Table 6.2.

Detector 1 takes soft decisions from the training phase, while Detector 2 uses hard decisions. Both detectors, with 28×28 nodes, are called full-scale Detector 1 and 2, whereas the other detectors are named reduced-scale Detector 1 and 2. The latter are for comparison purposes because it may lessen the number of pixels accessed by the detectors to reduce the complexity and to indicate the importance of certain nodes.

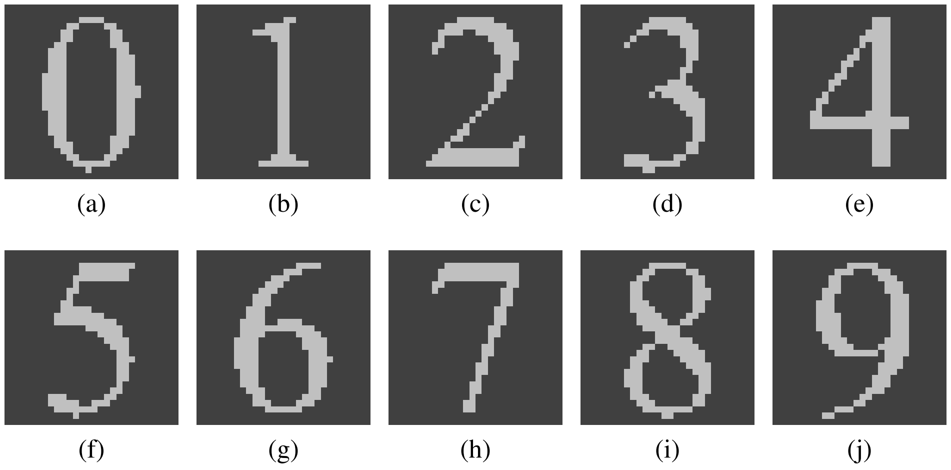


Figure 6.3: 10 Standard Digits

Table 6.2: Detector Types

Detector	Detector Input	σ_2	Number of Nodes
1	$v_k(z)$	$1.5\sigma_1$	5, 10, 20, 40, 28×28
2	$p(c_k = 0\|z)$	$1.5\sigma_1$	5, 10, 20, 40, 28×28

Simulation Results

In this section, the performance of detectors introduced in Fig. 6.2 is indicated by 0.00% resulting error rates in Fig. 6.4 for full-scale Detector 1 and 0.52% for full-scale Detector 2. The number of nodes is reduced from 784 to specific numbers of nodes, i.e., 40, 20, 10, and 5, in the reduced-scale Detectors 1 and 2, resulting a degradation of the performance.

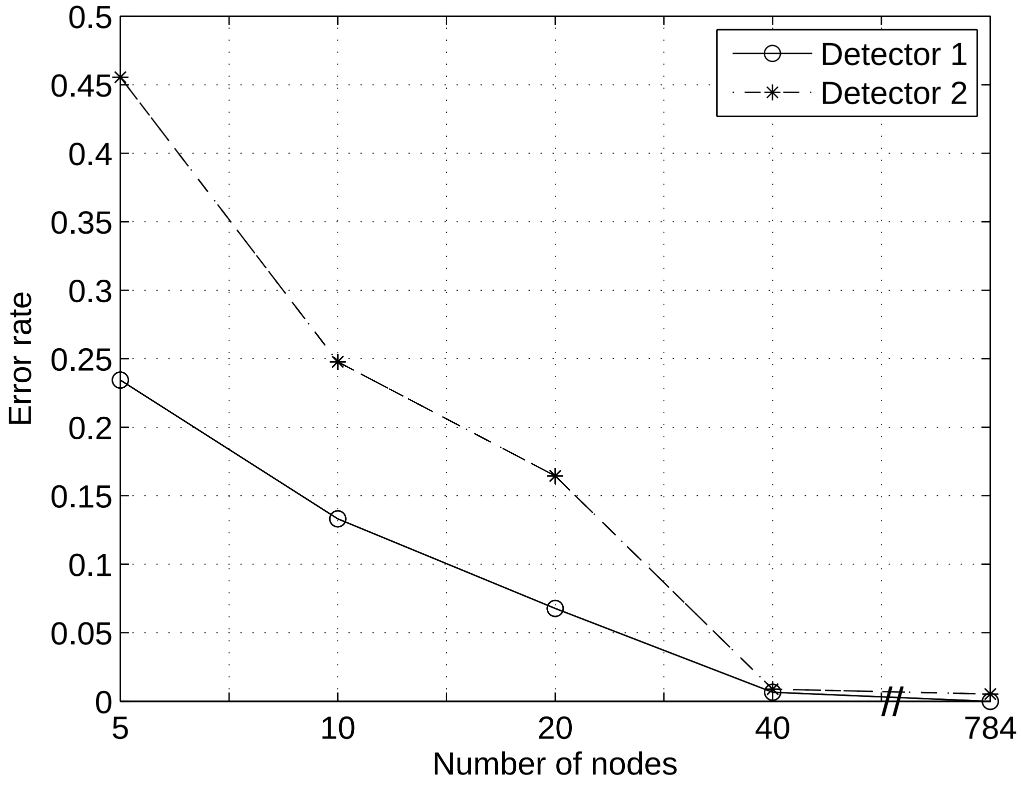


Figure 6.4: Error Rates of 2 Types of Detectors

From Fig. 6.4, the error rate is below 1% if 40 nodes are retained. Only full-scale detectors will be considered because of their possibly better performance compared to reduced scale detectors. In Table 6.4, the energy equalization method 4 is excluded (i.e., to preserve constant energy for different numbers of nodes) because it is not practical in bio-systems, and the coding systems reduce the total amount of energy in 784 bits into a number of bits as small as 40, 20, 10 or 5 bits, respectively.

100 batches are processed under different system configurations to obtain the histograms in Fig. 6.5.

In each batch, 10,000 randomly selected digits (from 0 to 9) are sent to the IPU and the number of error decisions is counted and recorded. Errors ranging from 0 to 200 or to 2000 are evenly divided into 10 intervals, i.e., $[0, 20)$, $[20, 40), \cdots$ in Fig. 6.5(a). The median number of each range is marked at each interval center along the horizontal axis. The number of errors is counted and adds up under each interval. The same procedure is conducted to depict an error range from 0 to 2000 and a step of 200 in Fig. 6.5(b). It can be observed that the performance of IPU detection is improved by the LDPC code constraint, per comparison between Fig. 6.5(a) and Fig. 6.5(b). It is reiterated that the LDPC code is randomly constructed and is not specifically optimized for this task. Scaling Method 3 is applied to generate the results in Fig. 6.5.

The same experiments are conducted under different parameter settings with the LDPC H matrix involved in order to verify the importance of VIV, such as (a) $\beta_1 = 0$, $\beta_2 = 1$, $\beta_3 = 1$, $\sigma_1 = 1$, and $\sigma_2 = 1.5$; (b) $\beta_1 = 1$, $\beta_2 = 0$, $\beta_3 = 1$, $\sigma_1 = 1$, and $\sigma_2 = 1.5$; and (c) $\beta_1 = 1$, $\beta_2 = 1$, $\beta_3 = 0$, $\sigma_1 = 1$, and $\sigma_2 = 1.5$; (d) $\beta_1 = 0$, $\beta_2 = 1$, $\beta_3 = 0$, $\sigma_1 = 1$, and $\sigma_2 = 1.5$; (e) $\beta_1 = 0$, $\beta_2 = 0$, $\beta_3 = 1$, $\sigma_1 = 1$, and $\sigma_2 = 1.5$ as shown from Fig.6.6-6.10. Comparing to Fig. 6.5, it can be concluded that the uncertainty in these movements can be beneficial for the IPU detection capability, due to a sub-optimal property of the detector. It is noticed that the system is not ideal comparing to telecommunication systems, where it is assumed that (a) the receiver knows the exact SNR, (b) the system controls the signal transmission power to maintain the required minimum SNR, and (c) the receiver maximizes the SNR with the matched filter [90]. The IPU is unable to control the received signal power or noise power in bio-visual systems. Therefore, it will be unable to build an optimal receiver for minimizing the error rates. This can be verified by applying Method 4 to normalize the energy for all frames. The errors are remarkably reduced comparing to Fig .6.5 when Method 4 is used. This indicates that the IPU is substantially closer to the optimal detector under Method 4 than Method 3. The distributions are further investigated by removing one or two VIV factors. It can be observed that additional VIV factors result in greater error distributions, which is consistent with our conventional wisdom. If the IPU is accepted as a sub-optimal detector, then it would not be surprising that the VIV may be beneficial. Contradictory results are often observed in bio-systems, implying that bio-coding systems can be sub-optimal.

Therefore, this output in Fig. 6.5 exhibits high similarity to phenomena observed in bio-neural systems, in which neurons are typically firing stochastically. Fig.6.11 and 6.12 is of comparison while scaling *Method* 4 is used, where the error rate is smaller comparing to Fig. 6.5 for both with and without LDPC H matrix involved.

Performance on Alphabetic Letters and Roman Numerals

Recognition tasks on both alphabetic letter ("a" to "z" as shown in Fig. 6.13) and Roman numerals ("I" to "X" as shown in Fig. 6.14) are also performed. The full-scale Detector 2 was applied with following environment settings: $\beta_1 = 1$, $\beta_2 = 1$, $\beta_3 = 1$, $\sigma_1 = 1$ and $\sigma_2 = 1.5$. The error rate for alphabetic letter recognition is 0.74%, and 0.00% for Roman numeral recognition.

Performance on Hyper-Acuity

The IPU unit is evaluated under a simplified configuration, assuming that the photo-receptor in the IPU is identical to the image pixel in terms of both size and shape. The ability of IPU is evaluated, about differentiating a misalignment up to 1/30th of a pixel size.

The IPU unit monitors the difference between two candidates at a time to perform hyper-acuity detection, while the input experiences independent point-spreads, fixational movements, orientations and noises.

If the differences are inputted into the IPU simultaneously, then both of them will experience identical visual uncertainty, i.e., $s_k^{(l)}(z_{12}) = s_k^{(l)}(z_1) - s_k^{(l)}(z_2)$ and $s_k^{(l)}(z_{21}) = -s_k^{(l)}(z_{12}) = s_k^{(l)}(z_2) - s_k^{(l)}(z_1)$.

When $s_k^{(l)}(z_{12})$ is the input into the IPU, $b_k^{(l)}(z_{12})$ can be obtained and then $u_k^{(l)}(z_{12})$. $v_k(z_{12})$ and $p(c_k|z_{12})$ is yielded from (6.24) and (6.25) after iterative decoding, as defined in

$$v_k(z_{12}) = \frac{\sum_{l=21}^{I_a} v_k^{(l)}}{I_a - 20} \tag{6.37}$$

$$\begin{aligned} p(c_k = 0|z_{12}) &= \frac{\sum_{l=21}^{I_a} Z\left(c_k^{(l)} = 0\right)}{I_a - 20} \\ p(c_k = 1|z_{12}) &= \frac{\sum_{l=21}^{I_a} Z\left(c_k^{(l)} = 1\right)}{I_a - 20}. \end{aligned} \tag{6.38}$$

Similarly, $v_k(z_{21})$ and $p(c_k|z_{21})$ can be obtained when $s_k^{(l)}(z_{21})$ is the input.

Now, "z_{12}" or "z_{21}" is inputted into the IPU during the testing stage. However, the IPU has no knowledge as to whether z is z_{12} or z_{21}. The following testing procedure is identical as in Section 6; Fig.6.15 presents three different cases.

The performance of hyper-acuity detector is tested under various settings. Each result data point in Table 6.3 is obtained from an average of 1000 trials. Thresholds of 3% and 1% of a photo-receptor are chosen as the right bar offset from the left one, as illustrated in Fig.6.15.

A 79.1$\%$ correct detection rate can be achieved by our detector with a 3% threshold. However, it is only approximately 50% with a 1% threshold. It has been reported by [57] that humans are capable of detecting a 1/30th threshold; however, no report is found claiming that humans can do better than 3%. The 3% threshold case can be improved under a reduced level of σ_1 of 0.5; however, the

1% case remains unable to be detected. The latter can be considered as one of the benchmarks to determine whether human vision systems are compliant with our IPU principle. It is found that the IPU can not differentiate the 1% threshold because the energy in input $S_k^{(l)}(z_{21})$ for this case is significantly smaller than that for the 3% threshold. The capability of the IPU to detect the 1% threshold can be acquired after normalizing the energy of $S_k^{(l)}(z_{21})$ under the 1% threshold identical to that under the 3% threshold, as mentioned in Method 4. However, noises are inseparable from signals in practice, as mentioned earlier, which implies that Method 4 is to be used for comparison study only, and it is inapplicable to bio-systems to boost signal energy without affecting the noise power. In Table 6.3, scaling Method 3 is compared to Method 4 in the hyper-acuity detector. The fixed energy in Method 4 is chosen to be approaching to the value of $E_p^{(l)}$ and $E_n^{(l)}$ under the 3 % threshold (0.09) when $\beta_1 = 1, \beta_2 = 1, \beta_3 = 1.$

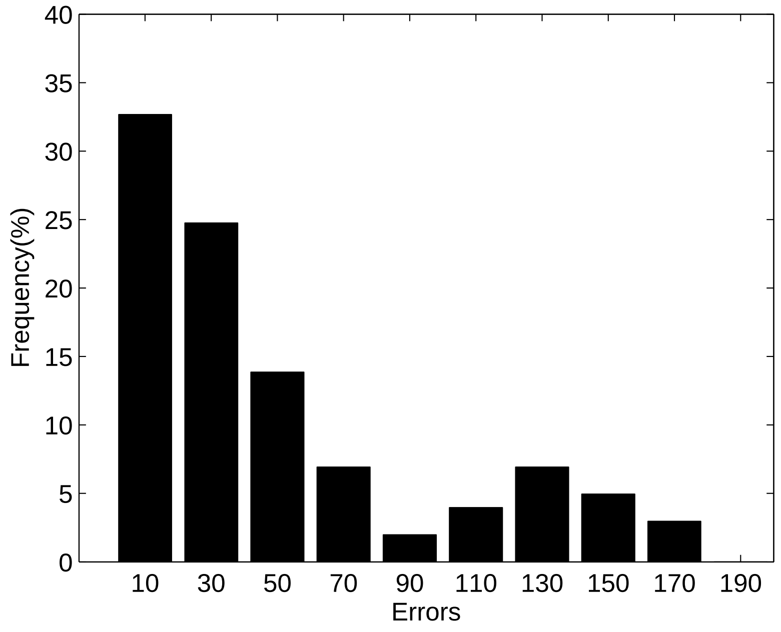
Performance on Low-Resolution Human Face Recognition

The last recognition task that IPU performs on is low-resolution face recognition under VIV conditions. The experimental procedures in this task is identical as the one presented in [122] with AT&T ORL database [123]. The SSSL approach proposed by Cai *et al.* [122] could obtain 97.4±1.2% recognition accuracy for 5 training and 5 testing task. Cai *et al.* [122] have also reported the performance of other methods such as Eigenface (87.5±2.5%), TensorPCA (88.1±2.5%), Fisherface (94.3±1.4%), 2DLDA (95.8±1.2%), Laplacianface (93.0±1.9%), and NPE (93.4±1.8%). All these methods do not incorporate VIV.

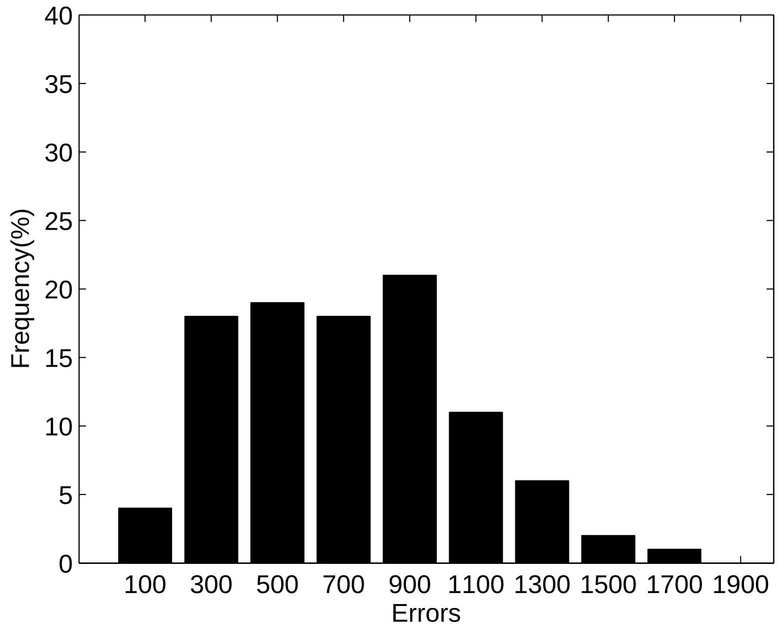
Using our IPU (Detector 2 with 32-by-32 pixels) with the VIV ($\beta_1 = 0.5, \beta_2 = 0.5, \beta_3 = 0.5, \sigma_1 = 0.8, \sigma_2 = 1$), recognition accuracy of 93.67±1.53% can be achieved.

Table 6.3: Performance of Hyper-Acuity Detector under Various Settings using Scaling Method 3 and 4 ($\sigma_2 = 1$)

	Offset	β_1, β_2, β_3	error rate when σ_1	
			1	0.5
Method 3	0.03	1, 1, 1	35.1%	4.5%
	0.03	1, 1, 0	32.5%	6.1%
	0.03	1, 0, 1	31.4%	3.3%
	0.03	0, 1, 1	29.6%	4.9%
	0.03	1, 0, 0	26.9%	1.5%
	0.03	0, 1, 0	34.2%	3.6%
	0.03	0, 0, 1	31.0%	2.6%
	0.01	1, 1, 1	49.8%	50.3%
	0.01	1, 1, 0	51.4%	49.7%
	0.01	1, 0, 1	48.9%	50.5%
	0.01	0, 1, 1	49.9%	50.1%
	0.01	1, 0, 0	49.6%	48.5%
	0.01	0, 1, 0	49.9%	51.2%
	0.01	0, 0, 1	49.6%	47.9%
Method 4	0.03	1, 1, 1	37.1%	4.2%
	0.03	1, 1, 0	34.1%	3.2%
	0.03	1, 0, 1	28.0%	3.9%
	0.03	0, 1, 1	29.9%	5.0%
	0.03	1, 0, 0	32.4%	0.8%
	0.03	0, 1, 0	27.7%	5.0%
	0.03	0, 0, 1	26.5%	1.9%
	0.01	1, 1, 1	34.3%	9.5%
	0.01	1, 1, 0	32.4%	10.7%
	0.01	1, 0, 1	33.6%	12.0%
	0.01	0, 1, 1	36.0%	10.6%
	0.01	1, 0, 0	26.5%	2.5%
	0.01	0, 1, 0	35.1%	11.8%
	0.01	0, 0, 1	32.9%	5.4%



(a)



(b)

Figure 6.5: Histogram of Errors when $\beta_1 = 1$, $\beta_2 = 1$, $\beta_3 = 1$, $\sigma_1 = 1$, and $\sigma_2 = 1.5$ (a) with the LDPC H Matrix Involved and (b) without the LDPC H Matrix Involved

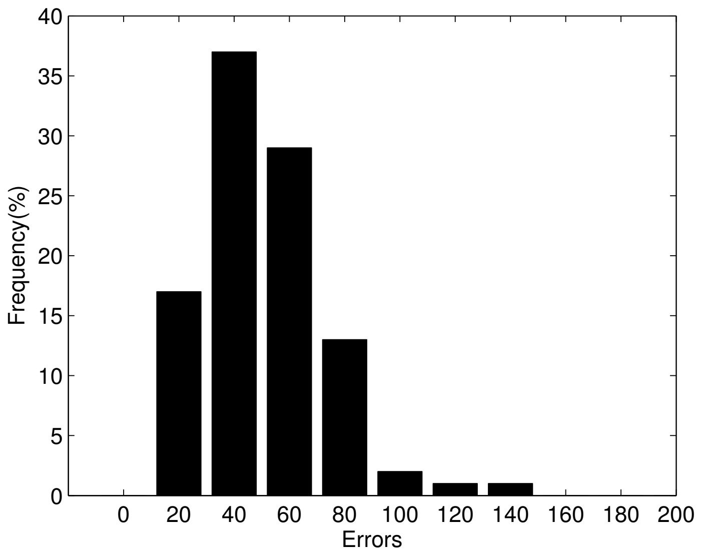


Figure 6.6: Histogram of Errors when $\beta_1 = 0$, $\beta_2 = 1$, $\beta_3 = 0$, $\sigma_1 = 1$, and $\sigma_2 = 1.5$ with the LDPC H Matrix Involved.

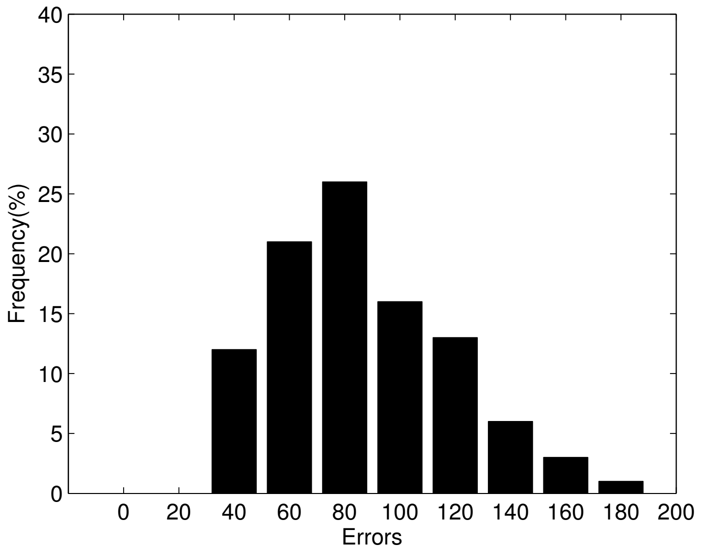


Figure 6.7: Histogram of Errors when $\beta_1 = 0$, $\beta_2 = 0$, $\beta_3 = 1$, $\sigma_1 = 1$, and $\sigma_2 = 1.5$ with the LDPC H Matrix Involved.

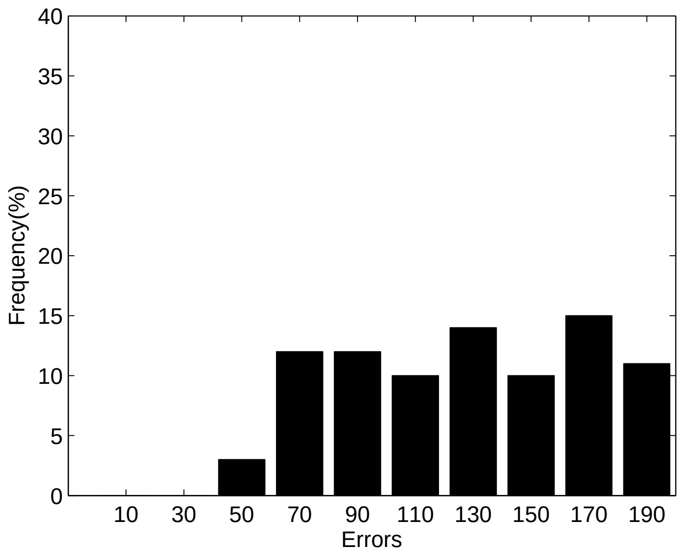


Figure 6.8: Histogram of Errors when $\beta_1 = 0$, $\beta_2 = 1$, $\beta_3 = 1$, $\sigma_1 = 1$, and $\sigma_2 = 1.5$ with the LDPC H Matrix Involved.

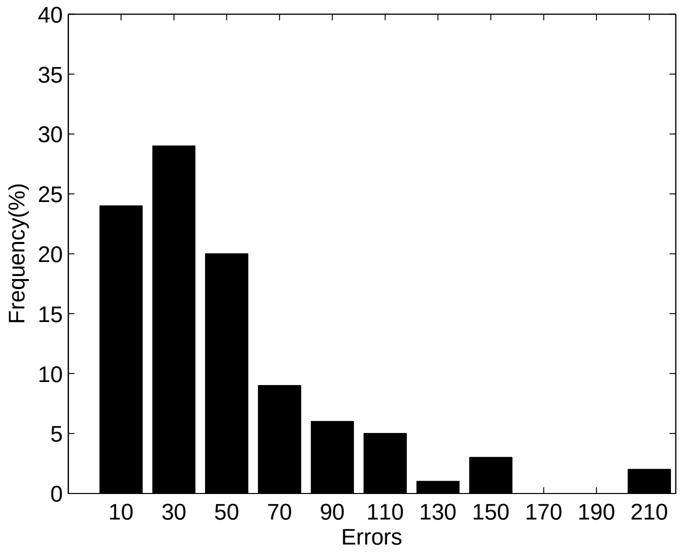


Figure 6.9: Histogram of Errors when $\beta_1 = 1$, $\beta_2 = 0$, $\beta_3 = 1$, $\sigma_1 = 1$, and $\sigma_2 = 1.5$ with the LDPC H Matrix Involved.

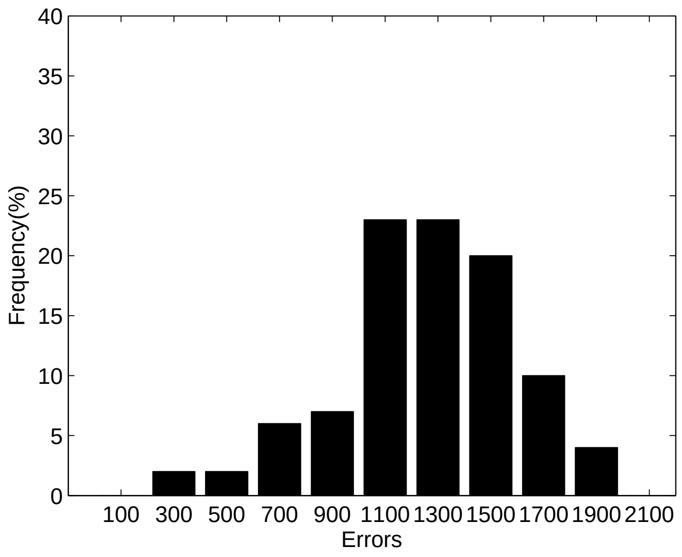


Figure 6.10: Histogram of Errors when $\beta_1 = 1$, $\beta_2 = 1$, $\beta_3 = 0$, $\sigma_1 = 1$, and $\sigma_2 = 1.5$ with the LDPC H Matrix Involved.

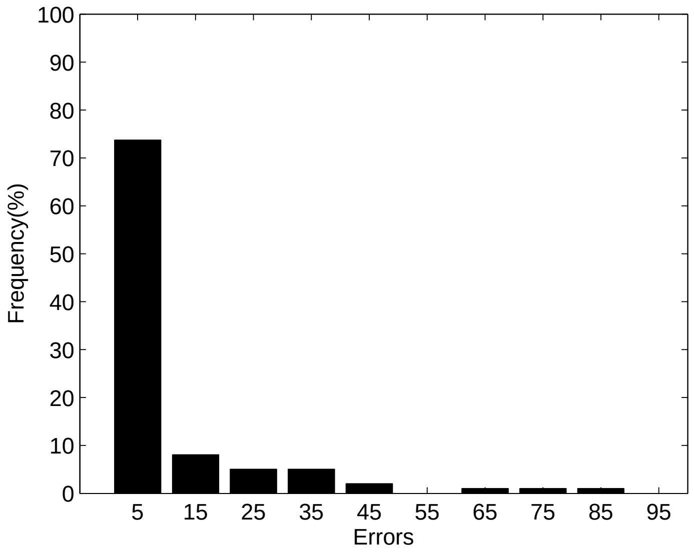


Figure 6.11: Histogram of Errors when $\beta_1 = 1$, $\beta_2 = 1$, $\beta_3 = 1$, $\sigma_1 = 1$, and $\sigma_2 = 1.5$ using Scaling Method 4 with LDPC H Matrix Involved.

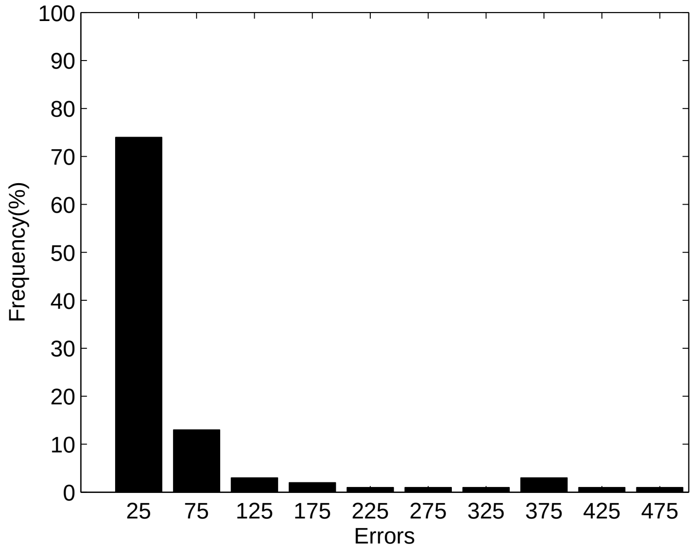


Figure 6.12: Histogram of Errors when $\beta_1 = 1$, $\beta_2 = 1$, $\beta_3 = 1$, $\sigma_1 = 1$, and $\sigma_2 = 1.5$ using Scaling Method 4 without LDPC H Matrix Involved.

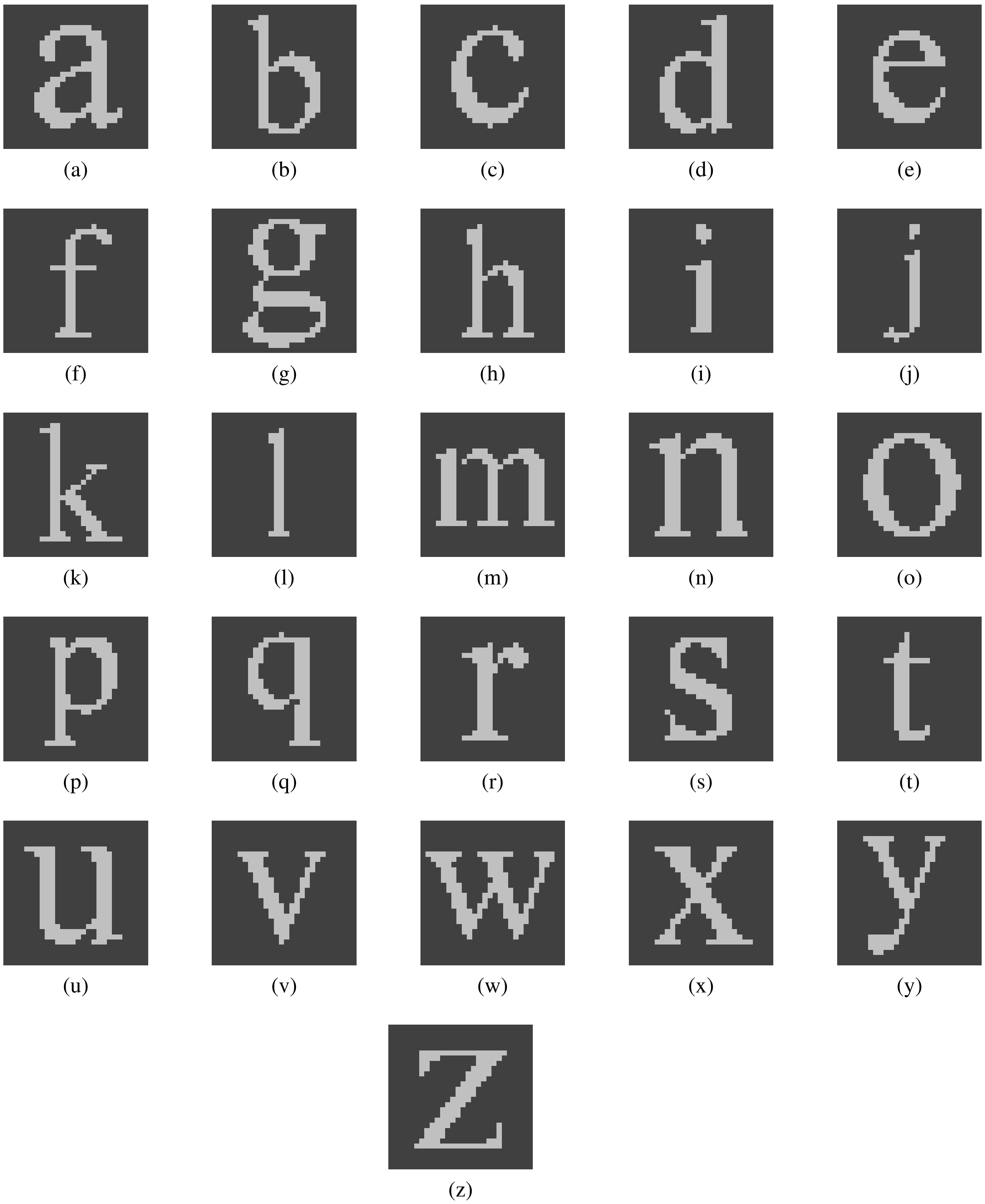


Figure 6.13: Alphabetic Letters “a” through “z”

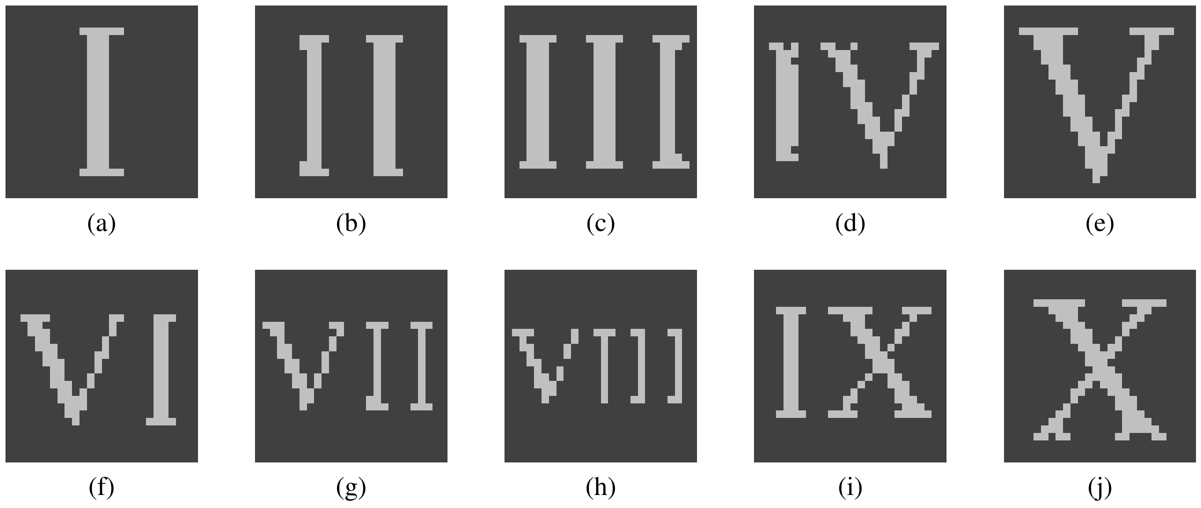


Figure 6.14: Roman Numerals "I" to "X"

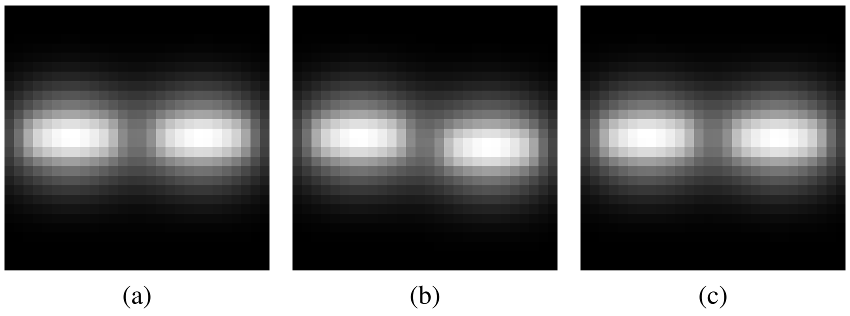


Figure 6.15: (a) Two Bars Lined up Perfectly, (b) Right Bar Shifted downward with a Hyper-acuity Threshold of 1 Photo-receptor, (c) Right Bar Shifted downward with a Hyper-acuity Threshold of 1/10 of a Photo-receptor

CHAPTER 7: CONCLUSION

Summary

This research addressed the implemented coding for GPRAM system. The challenges in designing codes and iterative decoding for non-equiprobable symbols were discussed. Two possible approaches were proposed to tackle the limitation of XOR operation relation to non-equiprobable symbols: quasi-XOR operation and intermediate transformation layer. The former can be used to construct nodes for non-equiprobable symbols, and it was proved the codes are less capable of error correction.

Non-linear codes with expanded operations were then developed, aiming to establish the GPRAM instead of conducting error control coding. Conventional Hamming code was extended for non-equiprobable symbols and XOR operation was replaced by other logic gates such as AND and OR. An error rate of 10^{-2} to 10^{-3} could be achieved by applying the extended codes despite their worse distance property, indicating the effects of constellations and small loops. The ones with bio-implications need further exploring.

The LDPC codes decoding with non-equiprobable symbols was reviewed. The message passing to check nodes is varied between from information bits and parity check bits, and an approach was proposed to optimize signaling constellation maximizing channel capacities. A 0.72 dB gain in performance could be reached if using optimal instead of equal-spaced constellation for symbols with prior probabilities of $p(0) = 0.3$. Several cases of short LDPC codes were simulated and 0.4 dB could be approximately gained. An additional 0.2 dB gain could be achieved if optimal source coding was applied.

A simple IPU for GPRAM was proposed and evaluated, consisted of a LDPC coding structure, an iterative decoder, a switch, and decision unit. The IPU mimics bio-visual systems as a gateway to information processing in biological brain, on the basis of fundamental principles of neural networks addressing information and communication. The IPU devision was guided by the GPRAM concept and philosophy, which is to compete against other computational machines on factors of variation, flexibility, and versatility rather than on precision and rate. The IPU was realistically configured to demonstrate its huge potential, while an immense amount of resources is required to completely invigorate GPRAM in the future. The IPU was tested by assigning tasks of digit and hyper-acuity recognition, where significantly debased digit images were inputted in order to mimic the VIV experienced by human visual system. The IPU exhibits strong capability in reliably recognizing such inputs, and achieving comparable hyper-acuity to human visual system. The recognition capability of IPU can be greatly improved by applying randomly (not specifically optimized for given tasks) constructed LDPC code, comparing to the IPU without applying any coding.

Future Works

Further challenges and works are outlined as follows.

1. **Beyond XOR:** The necessity of extending the basic operation beyond XOR, for example, including AND and OR gates, or the combination of these gates. It will complicate the design but help to deeply understand bio-systems.
2. **Code construction using intermediate transformation layer:** It is current unavailable to precisely match the targeted prior probabilities with those in intermediate transformation layer. Only an approximation can be achieved, of which effect on performance of codes remains unknown.

3. **Encoder:** Well-known issue in constructing encoder from parity check matrix of general LDPC codes. It can be worse for codes with XOR, AND, and OR gates with non-equiprobable symbols.

4. **Application of advanced error correction codes:** Variety of approaches to apply these codes in GPRAM design [68]. A simple IPU is developed and studied for GPRAM. The IPU is equipped with regular LDPC codes, which can be altered to advanced codes.

5. **Comparison and optimization:** Comparison to best know codes like non-uniform Turbo codes, by applying optimized codes using density evolution method. A good way of designing LDPC codes can be found in [124] [125].

6. **Extension of the logic operations into vague operations:** Insufficiency of simple logic operations in GPRAM systems. A complex tasks can be decomposed into a combination of simpler tasks being resolved by applying an amount of simple logic gates, upon which conventional systems are established. On the contrary, the GPRAM system pursues to achieve better and simpler groups of approximations towards the complex tasks. As an approximation, it can be gradually transformed from one variation to another, resulting two different logical functions. Therefore, an over-completed function basis is required in GPRAM systems, each of which has different degrees of variations. The accomplishment of generating this type of functions would be assisted by investigating those two different logical functions.

7. **Improvement on the IPU/GPRAM system:** Compatibility between IPU and LDPC codes and iterative decoders. They are commonly more appropriated for 10^{-5} bit error rates cases than IPU, which typically requires an accuracy of 70% - 99%. The IPU can be considered as a sub-optimal case for the tasks per se, because computations specified in it do not completely befit the concept of GPRAM in this stage. Currently, this work is being extended to recognize hand-written digits. This effort and results help use to understand how brains work and how

humans recognition abilities function, which is described in [126].

LIST OF REFERENCES

[1] J. Hawkins and S. Blakeslee. *On Intelligence*. Times Books, 2004.

[2] H. B. Barlow. Intelligence, guesswork, language. *Nature*, 304:207-209, 1983.

[3] M. D. Fox and M. E. Raichle. Spontaneous fluctuations in brain activity observed with functional magnetic resonance imaging. *Nature Reviews Neuroscience*, 9:700-711, 2007.

[4] G. Roth and U. Dicke. Evolution of the brain and intelligence. *Trends in Cognitive Sciences*, 9(5):250-257, 2005.

[5] Y. Li, Y. Liu, J. L, W. Qin, K. Li, C. Yu and T. Jiang. Brain anatomical network and intelligence. *PLoS Computational Biology* 5(5):e1000395, 2009.

[6] W. H. Calvin. *How brains think: evolving intelligence, then and now*. Basic Books, 1996.

[7] N. J. Nilsson. *Principles of artificial intelligence*. Springer, 2004.

[8] G. F. Luger. *Artificial intelligence: structures and strategies for complex problem solving*. Pearson, 6th edition, 2008.

[9] A. Mackworth and D. Poole. Artificial intelligence: foundations of computational agents. Cambridge University Press, 2010.

[10] K. G. Zuse, Z3 computer on May 12th, 1941.

[11] W. K. Estes. Is human memory obsolete? Comparisons of computer memory with the picture of human memory emerging from psychological research suggest basic differences in modes of operation, with little likelihood that one can replace the other. *American Scientist*, 68(1):62-69, 1980

[12] World's most powerful computer as of 2016 can perform around 93,000 trillion calculations per second at it peak.

[13] H. Gardner. Frames of mind: The theory of multiple intelligences. Basic Books, 1983.

[14] A. L. Blackler, V. Popovic and D. P. Mahar. Empirical investigations into intuitive interaction: a summary. *Mensch-Maschine-Interaktion Interaktiv*, 13:4-24, 2007.

[15] N. Khatri and H. A. Ng. The role of intuition in strategic decision making. *Human Relations*, 53(1):57-86, 2000.

[16] J. R. Platt. Strong inference: certain systematic methods of scientific thinking may produce much more rapid progress than others. *Science*, 146(3642):347-353, 1964.

[17] Y. Jia, R. Argueta-Morales, M. Liu, Y. Bai, E. Divo, A. J. Kassab and W. M. Decampli. Experimental study of anisotropic stress/strain relationships of the piglet great vessels and relevance to pediatric congenital heart disease. *The Annals of Thoracic Surgery*, 99(4):1399-1407, 2015.

[18] B. D. Smedt, M. P. Noël, C. Gilmore and D. Ansari. *Trends in Neuroscience and Education*, 2:48?55, 2013.

[19] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. In *Proceedings of the London Mathematical Society*, 2(42):230-265, 1937.

[20] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem: a correction. In *Proceedings of the London Mathematical Society*, 2(43):544-546, 1937.

[21] von J. Neumann. *The computer and the brain.* Yale University Press, 2nd edition, 2000.

[22] L. Wei. General purpose representation and association machine-part 1: introduction and illustrations. In *Proceedings of the 2012 IEEE Southeastcon*, pages 1-5, 2012.

[23] L. Wei. General purpose representation and association machine-part 2: biological implications. In *Proceedings of the 2012 IEEE Southeastcon*, pages 1-5, 2012.

[24] R. G. Gallager. *Information theory and reliable communication*. Wiley, 1968.

[25] R. M. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*,27:533-547, 1981.

[26] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: turbo codes. In *IEEE International Conference on Communications (ICC)*, pages 1064-1070, 1993.

[27] R. J. McEliece, D. J. C. MacKay and Jung-Fu Cheng. Turbo decoding as an instance of Pearl's "belief propagation" algorithm. *IEEE Journal on Selected Areas in Communications*, 16(2):140-152, 1998.

[28] J. Pearl. *Probabilistic reasoning in intelligent systems*. Morgan Kaufmann, 1988.

[29] F. R. Kschischang, B. J. Frey and H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47:498-519, 2001.

[30] G. D. Forney, Jr.. Codes on graphs: normal realization. *IEEE Transactions on Information Theory*, 47:520-548, 2001.

[31] F. Eberhardt and D. Danks. Confirmation in the cognitive sciences: the problematic case of Bayesian models. *Minds and Machines*, 21:389-410, 2011.

[32] L. Wei and H. Qi. Near optimal limited search decoding on ISI/CDMA channels and decoding of long convolutional codes. *IEEE Transactions on Information Theory*, 46(4):1459-1482, 2000.

[33] L. Wei. Several properties of short LDPC codes. *IEEE Transactions on Communications*, 51(5):721-727.

[34] C. A. Cole, S. G. Wilson, E. K. Hall and T. R. Giallorenzi. Analysis and design of moderate length regular LDPC codes with low error floors. In *40th Annual Conference on Information Sciences and Systems*, 2006.

[35] L. Wei. High-performance iterative viterbi algorithm for conventional serial concatenated codes. *IEEE Transactions on Information Theory*, 48(7):1759-1771, 2002.

[36] A. Church. An unsolvable problem of elementary number theory.*American Journal of Mathematics*, 58:345-363, 1936.

[37] F. E. Hohn. *Applied boolean algebra: an elementary introduction*. The Macmillan Company, 1966.

[38] R. C. Jaeger and T. N. Blalock. *Microelectronic circuit design*, McGraw-Hill, 4th edition, 2011.

[39] IEEE Graphic Symbols for Logic Functions. *IEEE Std 91/91a-1991*, 1994.

[40] A. V. M. Herz, T. Goollisch, C. K. Machen and D. Jaeger. Modeling single-neuron dynamics and computations: a balance of detail and abstraction. *Science*, 314: 80?85, 2006.

[41] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5:115-133, 1943.

[42] B. Krose and P. van der Smagt. *An introduction to neural networks*. University of Amsterdam Press, 1993.

[43] Y. Bengio. Learning deep architectures for AI.*Foundations and Trends in Machine Learning*,2(1):1-127, 2009.

[44] T. Hastie, R. Tibshirani and J. Friedman. *The elements of statistical learning*, Springer, 2009.

[45] J. R. Searle. Is the brain's mind a computer program? *Scientific American*, pages 26-31, 1990.

[46] A. Goldental, S. Guberman, R. Vardi and I. Kanter. A computational paradigm for dynamic logic-gates in neuronal activity. *Frontiers in Computational Neuroscience*, 29:8-52, 2014.

[47] R. G. Gallager. *Low-density parity-check codes*. MIT Press, 1963.

[48] D. J. C. Mackay. Good error-correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory*, 45(2):399-431, 1999.

[49] M. Luby, M. Mitzenmacher and A. Shokrollahi. Analysis of random processes via and-or tree evaluation. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 364-373, 1998.

[50] M. Luby, M. Mitzenmacher, A. Shokrollahi and D. Spielman. Analysis of low density codes and improved designs using irregular graphs. *IEEE Transaction on Information Theory*, 47:585-598, 2001.

[51] R. M. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27:533-547, 1981.

[52] L. Yang. Advanced coding and modulation for ultra-wideband and impulsive noises. *Electronic Theses and Dissertations*, Paper 3420, 2007.

[53] E. A. Wülfing. Über den kleinsten Gesichtswinkel. *Zeitschrift für Biologie*, 29:199-202, 1892.

[54] G. Westheimer and S. P. McKee. Spatial configurations for visual hyperacuity. *Vision Research*, 17(8): 941-947, 1977.

[55] G. Westheimer and S. P. McKee. Visual acuity in the presence of retinal-image motion. *Journal of the Optical Society of America*, 65:847-850, 1975.

[56] D. Marr, T. Poggio and E. Hildreth. Smallest channel in early human vision. *Journal of the Optical Society of America* 70:868-870, 1980.

[57] S. A. Klein and D. M. Levi. Hyper-acuity thresholds of 1 sec: Theoretical predictions and empirical validation. *Journal of the Optical Society of America A: Optics and Image Science*, 7(2): 1170-1190, 1985.

[58] H. Wang and D. M. Levi. Spatial integration in position acuity. *Vision Research*, 34(21):2859-2877, 1994.

[59] D. R. Williams and N. J. Coletta. Cone spacing and the visual resolution limit. *Journal of the Optical Society of America A: Optics and Image Science*,4(8):1514-1523, 1987.

[60] W. S. Geisler and K. D. Davila. Ideal discriminators in spatial vision: two-point stimuli. Journal of the Optical Society of America, 2(9):1483-1497, 1985.

[61] G. Westheimer. Visual acuity and hyperacuity. *Investigative Ophthalmology and Visual Science*, 14(8):570-572, 1975.

[62] L. Wei. Robustness of LDPC codes and internal noisy systems. In *Proceedings of the 41st Annual Allerton Conference on Communication, Control, and Computing*, pages 1665-1674, 2003.

[63] L. Wei. Biologically inspired statistical matched filter. In *IEEE International Conference on Communications(ICC'05)*, 2:810-814, 2005.

[64] L. Wei. Biologically inspired amorphous communications. In *IEEE International Symposium on Information Theory (ISIT'05)*, page 1005, 2005.

[65] L. Wei, D. M. Levi, R. Li and S. Klein. Feasibility study on a hyper-acuity device with motion uncertainty: two-point stimuli. *IEEE Transactions on Systems, Man, and Cybernetics:Cybernetics*, 37(2):385-397, 2007.

[66] B. Dai and L. Wei. Low-density parity check codes with non-equiprobable symbols. *IEEE Communication Letter*, 17(11):2124-2127, 2013.

[67] B. Dai and L. Wei. Some results and challenges on codes and iterative decoding with non-equal symbol probabilities. in *IEEE International Symposium on Information Theory and its Applications(ISITA)*, pages 116-120, 2012.

[68] H. Li, B. Dai, S. Schultz, and L. Wei. General purpose representation and association machine, part 3: prototype study using LDPC codes. In *Proceedings of the 2013 IEEE Southeastcon*, pages 1-5, 2013.

[69] E. Yaakobi and J. Bruck. On the uncertainty of information retrieval in associative memories. In *Proceedings of IEEE International Symposium on Information Theory*, pages 106-111, 2012.

[70] D. C. Knill and A. Pouget. The Bayesian brain: the role of uncertainty in neural coding and computation. *Trends in Neurosciences*, 27(12):712-719, 2004.

[71] L. Shu and D. Costello. *Error Control Coding*. Prentice Hall, 2nd edition, 2004.

[72] A. Kae, G. B. Huang, C. Doersch, and E. Learned-Miller. Improving state-of-the-art OCR through high-precision document-specific modeling. In *Computer Vision and Pattern Recognition(CVPR'10)*, pages 1935-1942, 2010.

[73] N. Joshi, S. B. Kang, C. L. Zitnick, and R. Szeliski. Image deblurring using inertial measurement sensors. In *ACM Transactions on Graphicss (Proc. SIGGRAPH)*, 29(3), 2010.

[74] C. Jacobs, P.Y. Simard, P. Viola, and J. Rinker. Text recognition of low-resolution document images. In *International Conference on Document Analysis and Recognition(ICDAR'05)*, pages 695-699, 2005.

[75] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D. J. Wu and A. Y. Ng. Text detection and character recognition in scene images with unsupervised feature learning. In *International Conference on Document Analysis and Recognition(ICDAR'11)*, 2011.

[76] B. M. Lake, R. Salakhutdinov and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332-1338, 2015.

[77] B. A. Olshausen. 20 years of learning about vision: Questions answered, questions unanswered, and questions not yet asked. *Twenty Years of Computational Neuroscience*. Springer, pages 243-270.

[78] E. Simoncelli. Vision's grand theorist. *Science*, 314(5796):78-79, 2006.

[79] G. Felsen, and Y. Dan. A natural approach to studying vision. *Nature Neuroscience*, 8:1643-1646, 2015.

[80] S. Nishimoto, A. Vu, T. Naselaris, Y. Benjamini, B. Yu, and J. Gallant. Reconstructing visual experiences from brain activity evoked by natural movies. *Current Biology*, 21(19):1641-1646, 2011.

[81] L. D. Harmon and B. Julesz. Masking in visual recognition: effects of two-dimensional filtered noise. *Science*, 180(4091): 1194-1197, 1973.

[82] I. Chaaban and M. R. Scheessele. Human performance on the USPS database. *Technical Report*, Indiana University, 2007.

[83] D. Cireşan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, 2012.

[84] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Müller, E. Säckinger, P. Simard and V. Vapnik. Comparison of learning algorithms for handwritten digit recognition. In *International Conference on Artificial Neural Networks(ICANN)*, pages 53-60, 1995.

[85] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 86(11):2278-2324, 1998.

[86] A. D. Liveris, Z. Xiong and C.N. Georghiades. Joint source-channel coding of binary sources with side information at the decoder using IRA codes. In *2002 IEEE Workshop on Multimedia Signal Processing*, pages 9-11, 2002.

[87] C. E. Shannon. A Mathematical theory of communication. *The Bell System Technical Journal*, 27:370-423, 1948.

[88] J. M. Wozencraft and I. M. Jacobs. *Principles of Communication Engineering*. John Wiley and Sons, 1965.

[89] M. K. Simon, M. K. Hinedi and W. C. Lindsey. *Digital Communication*. Prentice-Hall, 1995.

[90] J. B. Proakis and M. Salehi. *Digital Communications*. McGraw-Hill, 5th edition, 2008.

[91] T. J. Richardson and R. L. Urbanke. The capacity of Low-Density Parity-Check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 47(2):599-618, 2001.

[92] W. Xu, J. Hagenauer and J. Hollmann. Joint source-channel decoding using the residual redundancy in compressed images. In *IEEE International Conference on Communications(ICC'1996)*, 1996.

[93] I. Korn, J. P. Fonseka and S. Xing. Optimal binary communication with nonequal probabilities. *IEEE Transactions on Communication*, 51(9):1435-1438, 2003.

[94] H. Kuai, F. Alajaji and G. Takahara. Tight error bounds for nonuniform signaling over AWGN channels. *IEEE Transactions on Information Theory*, 46:2712-2718,2000.

[95] B. A. Olshausen and D. J. Field. Emergence of simple cell receptive field properties by learning a sparse code for nature images. *Nature*, 381:607-609, 1997.

[96] Y. Burak, U. Rokni, M. Meister and H. Sompolinsky. Bayesian model of dynamic image stabilization in the visual system. *Proceedings of the National Academy of Sciences(PNAS)*, 107:19525-19530, 2010.

[97] C. Mead. *Analog VLSI and Neural Systems*. Addison-Wesley, 1989.

[98] L. Wei. Channel capacity and constellation optimization of M-PAM input AWGN with non-equal symbol probabilities. Available at: http://people.cecs.ucf.edu/lei/letter_ieee_capacityMPSK_v2.pdf

[99] J. Hagenauer and P. Hoeher. A viterbi algorithm with soft-decision outputs and its applications. In *IEEE Global Telecommunications Conference(GLOBECOM'89)*, pages 1680-1686, 1989.

[100] F. Alajaji, N. Phamdo and T. Fuja. Channel codes that exploit the residual redundancy in CELP-encoded speech. *IEEE Transactions on Speech Audio Processing*, 4:325-336, 1996.

[101] L. Wei. Optimized M-ary orthogonal and bi-orthogonal signaling using coherent receiver with non-equal symbol probabilities. *IEEE Communication Letters*,16(9):794-796, 2010.

[102] S. Yousefi and B. Holmes. Tight lower bounds on the symbol error rate of uncoded nonuniform signaling over AWGN channel. In *International Conference on Wireless Networks, Communications and Mobile Computing*, pages 606-611, 2005.

[103] I. Ochoa, P. M. Crespo and M.Hernaez. LDPC codes for non-uniform memoryless sources and unequal energy allocation. *IEEE Communication Letters*, 14(9):794-796, 2010.

[104] F. Cabarcas, R. D. Souza, and J. Garcia-Frias. Turbo coding of strongly nonuniform memoryless sources With unequal energy allocation and PAM signaling. *IEEE Transactions on Signal Processing*, 54(5):1942-1946, 2006.

[105] G. I. Shamir and J. J. Boutros. Non-Systematic Low-Density Parity-Check Codes for Nonuniform Sources. In *Proceedings of IEEE International Symposium on Information Theory*, 2005.

[106] L. Wei and I. Korn. Optimal M-ASK/QASK with non-equal symbol probabilities. *IET Communications*, 5(6):745-752, 2011.

[107] D. C. Ciresan, U. Meier, L. M. Gambardella and J. Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12):3207-3220, 2010.

[108] D. Ciresan, U. Meier, L. Gambardella and J. Schmidhuber. Convolutional neural network committees for handwritten character classification. *2011 International Conference on Document Analysis and Recognition(ICDAR)*, pages 1135-1139, 2011.

[109] Z. Wang, S. Chang, Y. Yang, D. Liu and T. S. Huang. Studying very low resolution recognizing using deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 1-9, 2016.

[110] B. Dai, H. Li and L. Wei. Image processing unit for General-purpose representation and association system for recognizing low-resolution digits with visual information variability. *IEEE Transactions on System, Man, and Cybernetics: System*, 2016

[111] H. Saeedi and A. H. Banihashemi. Performance of belief propagation for decoding LDPC codes in the presence of channel estimation error. *IEEE Transactions on Communications*, 55(1):83-89, 2007.

[112] S.-Y. Chung. On the construction of some capacity-approaching coding schemes. Ph.D. dissertation, Massachusetts Institute of Technology, 2000.

[113] G. D. Forney, Jr.. On iterative decoding and the two-way algorithm. In *Proceedings of International Symposium on Turbo Codes and Related Topics*, 1997.

[114] J. Hagenauer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *IEEE Transactions on Information Theory*, 42:429-445, 1996.

[115] F. W. Campbell and R. W. Gubisch. Optical quality of the human eye. *The Journal of Physiology*,186(3):558-578, 1966.

[116] W. S. Geisler. Physical limits of acuity and hyperacuity. *Journal of the Optical Society of America. A, Optics, image science, and vision*,1(7):775-782, 1984.

[117] E. Kowler. The stability of gaze and it implications of vision. *Eye Movements*, R. H. S. Carpenter, pages 71-92, 1991.

[118] J. Krauskopf, T. N. Cornsweet and L. A. Riggis. Analysis of eye movements during monocular and binocular fixation. *Journal of the Optical Society of America*, 50(6):572-578, 1960.

[119] L. A. Riggis and J. C. Armington. Motions of the retinal image during fixation. *Journal of the Optical Society of America*, 44(4):315-321, 1954.

[120] M. Eizenman, P. E. Hallett, and R. C. Frecker. Power spectra for ocular drift and tremor. *Vision Research*, 25(11):1635-1640, 1985.

[121] S. Martinez-Conde, S. L. Macknik and D. H. Hubel. The role of fixational eye movements in visual perception. *Nature Reviews Neuroscience*, 5:229-240, 2004.

[122] D. Cai, X. He, Y. Hu, J. Han and T. S. Huang. Learning a spatially smooth subspace for face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 138-142, 2007.

[123] F. Samaria, A. Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of 2nd IEEE Workshop on Applications of Computer Vision*, 1994.

[124] T. J. Richardson, M. A. Shokrollahi and R. L. Urbanke. Design of capacity-approaching irregular Low-Density Parity-Check codes. *IEEE Transactions on Information Theory*, 47(2):619-637, 2001.

[125] S.-Y. Chung, G. D. Forney, Jr., T. J. Richardson and R. Urbanke. On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit. *IEEE Communications Letters*, 5(2):58-60, 2001.

[126] P. Sinha, B. Balas, Y. Ostrovsky, and R. Russell. Face recognition by humans: nineteen results all computer vision researchers should know about. In *Proceedings of the IEEE*, 94(11):1948-1962, 2006.