NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Electrical and Electronic Engineering

DIGITAL IMAGE RESTORATION USING ERROR CONTROL CODING

Graduation Project
EE-400

Student: Fadi Hassan Shamsi (20033290)

Supervisor: Dr. Ali Serener

Nicosia-2006

ACKNOWLEDGMENT

I would like to thank Dr. Ali Serener for accepting to be my advisor. As he helped me to overcome so many difficulties in the communications field, and in the coding field, a person that could answer any question I would ask in details, and because of his guidance I felt that my learning progress was rapid and fast.

I also want to thank the management of Near East University for supplying the university library with a lot of precious books about all subjects, and for the really good care of the engineering faculty laboratories, which made it easy for my colleagues and me to learn and apply.

I would also want to thank Dr. Kadri Buruncuk and Dr. Ozgur Ozerdem for being my academic advisors during my academic years, also I would like to thank Dr. Adnan Khashman, as he taught us how to be responsible and what discipline really is, I also want to thank Prof. Dr. Fakhreddin Mamedov the dean of the electrical and electronics faculty, and Prof.Dr. Senol Bektas for his wise and kind support.

I would also like to thank my friends Mahir, Selam, Mohammed Marouf, and my fiancé for their help and support, and for sharing with me all bad and good moments.

Finally, I want to thank my family, for their support, for always giving me a reason to work, and always try harder to be better, special thanks to my parents for their emotional and financial support that always kept me on my feet.

ABSTRACT

People likes perfection; perfection in communication systems is to get use out of all the power used to transmit information, which means that when we transmit a block of data that requires a certain power to be transmitted, the data is going to be received without errors.

In real life this is impossible because of noise, so we have to increase the power of the signal that we are transmitting in order to overcome the noise power. Scientists and engineers introduced a lot of efficient methods. Like increasing the antenna power, or increasing the channel capacity by introducing different successful modulation types, but still high error rates remains. In the early 90's digital communication systems were getting more popular and started to replace most of the analog communication systems. With a lot of its special signal processing techniques, digital communication systems are much efficient than analog systems, so if we compare a terrestrial (analog) T.V. channels that is transmitted in UHF or VHF band or an old analog satellite channels (like channels on Arab-Sat which is still in duty) with a new digital T.V. satellite channels like (Nile-sat or DIGI-Turk) we will find that the analog channels get effected by weather conditions much more than digital channels. The reason behind that is the coding technique used in digital communication systems. Because as we know additive noise affects all signals either it's digital or not. However receivers in analog communication systems can not correct errors caused by noise, while in digital communication systems, because of coding the receiver has the ability to detect errors that occurred in the communication channel, and make a decision about what information has been transmitted. What makes the coding techniques very successful is increasing the signal to noise ratio, and therefore getting more efficient communication systems.

This report describes digital communication systems, types of codes used in digital communications systems, and gives details about linear block codes and cyclic codes, and their application to digital images. Comparison on communication systems will be made on models for systems that transmit images.

Table of Contents

INTRODUCTION

Because of channel coding in our present time we have very efficient digital communication systems, that is capable of transmitting and receiving data blocks with negligible bit error rates no matter what type of noise exists in the channel. A lot of good code designs have been presented, and different codes have different properties that can be associated with different types of channels.

The idea of channel coding came out with Shannon's noisy channel coding theorem that states, 'with every channel we can associate a "channel capacity" *C*. There exists error control codes such that information can be transmitted across the channel at rates less than *C* with arbitrarily low bit error rate'.

Our goal in this report is to explain the popular types of codes, with their different properties of error detection and correction, concentrating on linear block codes and cyclic codes for these codes are so efficient and simple at the same time. We show how coding can give amazing results in overcoming the noise that occurs in the communication channels.

Chapter 1 starts by describing the digital communications systems and their elements, and then we explain the encoder and the decoder part of the system, Then we mention a few types of coding techniques and some of the error correcting codes briefly.

Chapter 2 explains linear block codes, the development and use of these codes, taking in consideration the properties of these codes and why they are different. Then we go through the cyclic codes, which are a very important branch of linear block codes, and very powerful type of codes, explaining the difference between them and linear block codes.

Chapter 3 includes some brief information about images, and the processes that can be applied on them. Digital image processing is a very huge and important disciplinary of digital communication. Applying coding on digital images successfully means that we can apply these codes on any type of digital data, like digital video, digital audio, and other types of digital information.

Chapter 4 models two different digital communication systems, one using cyclic codes and another system without coding. It shows the difference in the received information, and the effect of the same noise on both.

Conclusion presents the superiority of systems with channel coding over analog systems and systems with no coding techniques, and gives examples from real daily life, and suggestions that might be useful for future life.

CHAPTER ONE

DIGITAL COMMUNICATION SYSTEMS

1.1. Digital Communication systems

A digital communication system is a means of transporting information from one party to another. The system is digital in that it uses a sequence of symbols from a finite alphabet to represent the information. The transmission of data in digital form allows for the use of a number of powerful signal processing techniques that would otherwise be unavailable [1].

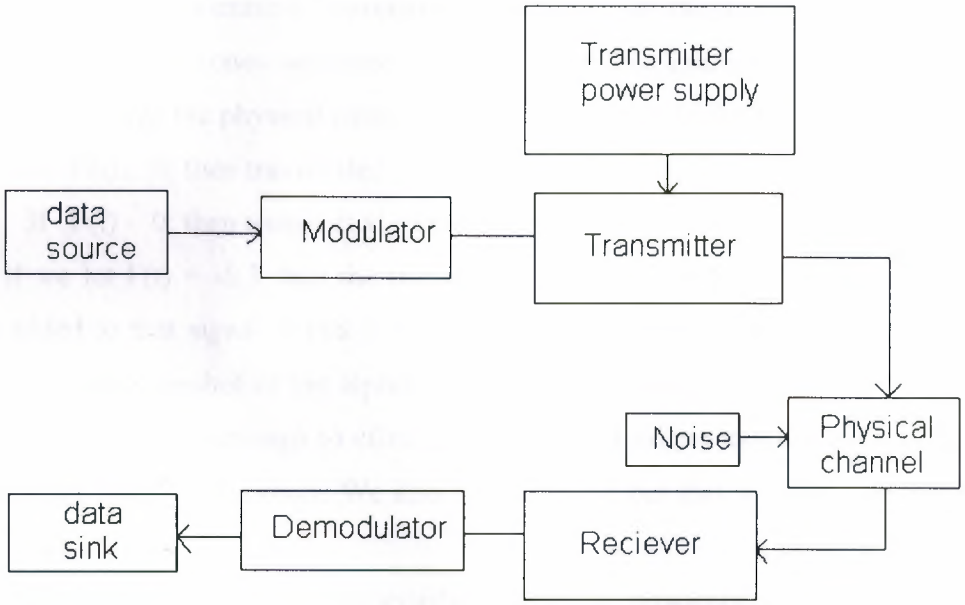

Figure 1.1 Basic elements of digital communication systems [1].

The data source block may represent any source of information like for example in digital cellular mobile telephone the data source consist of a microphone and a digital-to analog converter, in general the data source provides a stream of symbols at some average rate R_S symbols per second that is passed along to the modulator.

The modulator maps the information symbols onto signals that can be efficiently transmitted over the communication channel [1]. Therefore the selection of modulation

method depends on a lot of considerations like available power or bandwidth. Modulated information is transmitted through the physical channel that has an additive noise.

The received signal is demodulated and because of the noise in the physical channel the demodulated signal might not be the same as the original signal that we got at the information source [1].

1.2. Noise Effect on Digital Communication Systems

1.2.1. What is noise?

During transmission in communication channels if noise power is high, it might cause changes in transmitted information that ends up at the information sink as different information than the ones supposed to be received. For example if a binary sequence is being sent through the physical channel, it will be received in the following way:

If $F(t) \geq 0$; then transmitted symbol is A

If $F(t) < 0$; then transmitted symbol is B

Here if we let $F(t) = -0.3$ then the transmitted symbol is B but what happens if we have noise added to that signal is that if $n(t) = 0.4$ the resultant will be $n(t) + F(t) = 0.1$ which gives us another symbol of the alphabet which is A. In order for this to happen the noise power should be high enough to effect the transmitted information so we have a choice of transmitting with a high power. We also have filtering but that wouldn't be so efficient for some types of noise.

The term noise is used customarily to designate unwanted waves that tend to disturb the transmission and processing of signals in communication systems. As we know we have different types of noise sources like atmospheric noise, galactic noise or man-made noise those ones are external noise types but we also have another type that is the internal type of noise that arises from spontaneous fluctuations of current or voltage in electrical circuits. This type of noise represents a basic limitation on the transmission or detection of signals in communication systems involving the use of electronic devices. The two most common examples of spontaneous fluctuations in electrical circuits are shot noise and thermal noise [3].

1.2.2. Shot Noise

Shot noise arises in electronic devices such as diodes and transistors because of the discrete nature of the current flow in the devices for example in a photo detector circuit a current pulse is generated every time an electron is emitted by the cathode due to incident light from a source of a constant intensity, the electrons are naturally emitted at random times denoted by τ_K as $-\infty < \tau_K < \infty$. It is assumed that the random emissions of electrons have been going on for a long time. Thus the total current flowing through the photo detector may be modeled as an infinite sum of current pulses [3].

1.2.3. Thermal Noise

Thermal noise is the name given to the electrical noise arising from the random motions of electrons in the conductor. It is also known that the thermal noise is Gaussian distributed with zero mean [3].

1.2.4. White Noise

The noise analysis of communication systems is customarily based on an idealized form of noise called white noise, the power spectral density of which is independent of the operating frequency. The adjective white is used in the sense that white light contains equal amounts of all frequencies within the visible band of electromagnetic radiation. We express the power spectral density of white noise, with a simple function denoted by ω (t), as $S_W(f) = N_0/2$. The parameter N_0 (the dimensions of N_0 are in watts per hertz), is usually referred to the input stage of the receiver of a communication system, and may be expressed as $N_0 = kT_e$, where k is Boltzmann's constant and T_e is the equivalent noise temperature of the receiver. The white Gaussian noise represents the ultimate in randomness; it has a infinite average power and not physically realizable [3].

1.3. Channel Coding

1.3.1. What is Channel Coding?

When the errors introduced by the information channel are unacceptable, then coding is necessary to reduce the error rate and improve the reliability of the transmission. Channel coding is one of the most important signal processing techniques used in digital communication systems for the reason it is maybe the most efficient way to minimize the errors without the need to use much more power. In other words coding helps us make more efficient use of power and increases the Signal-to-Noise value in a digital communication system [1][3].

The idea of coding was based on Shannon's noisy channel coding theorem that stated " With every channel we can associate a 'channel capacity' C. there exist error control codes such that information can be transmitted across the channel at rates less than C with arbitrarily low bit error rate". But Shannon didn't tell us how to achieve these codes he just stated that we can associate a code that minimize the errors under the condition that data transmission rate shouldn't exceed the channel capacity [1].

1.3.2. Fundamentals of Channel Coding

The main idea behind channel coding is adding extra digits to our digital message words at the channel encoder to have our code words that is going to be transmitted, These extra digits are called redundancy digits and will be discarded after decoding as they are only used to detect or correct errors that occurred in the physical channel during transmission of the information.

Figure1.2 shows the main parts of a digital communication system including the decoder and the encoder [1]. As we see in Figure 1.2 that the channel input and output must be in a binary format being transmitted in blocks. The source generates analog information signals that are converted by the source encoder into a digital form (message words) of length L (number of bits). The channel encoder adds redundancy to the messages to form the codewords of length N that gets modulated and transmitted through the digital channel.

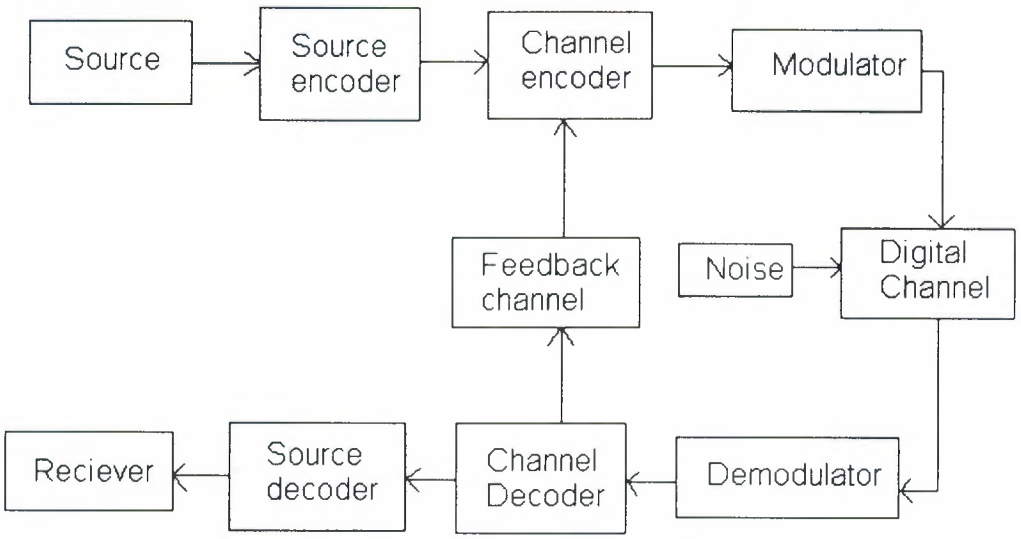

Figure 1.2 Block diagram of a digital communication systems [3].

Noise is added to the information while it goes through channel, resulting in a difference between the received and the transmitted data. The channel decoder based on the received codewords decides which codeword was transmitted. The decoder detects occurring errors, and corrects it by deciding which codeword is been transmitted.

1.3.3. Hamming Distance

Hamming distance between two binary codewords is the number of different digits between them so if we assume two codewords D1=(10110011) and the codeword D2=(10110110) then the hamming distance between both codewords is:
d(D1,D2) = 2, since the first and the third bits are different we have two different bits between them which means the Hamming distance between them is two [1].

1.3.4. Hamming Distance Decoding Rule

In Hamming distance decoding rule the decoder will check the Hamming distance between the received codeword and all the original codewords of the code. The original codeword that has the minimum hamming distance with the received codeword will be chosen by the decoder to be the transmitted codeword.

For example, if we have the following codewords D1(000) and D2(111), and received codeword has one error in the second bit R(010) then d(R,D1) = 1, d(R,D2) = 2. Therefore, the receiver will choose D1 to be the transmitted codeword [3].

1.3.5. Minimum Distance of a Code

The minimum distance of a code, is the minimum hamming distance between the codewords of a code.

Assume we have a code consisting of four codewords, which are:

D1 = (001)

D2 = (010)

D3 = (100)

D4 = (111)

The Hamming distances for the code is as follows

d(D1,D2) = 2, d(D1,D3) = 2, d(D1,D4) = 2, d(D2,D3) = 2, d(D2,D4) = 2, d(D3,D4) = 2,

Here we find the minimum distance of the code $d_{min} = 2$.

1.3.6. Code Capability of Error Detection and Correction

If we assume that a received codeword has t number of errors, then for the decoder to be able to detect it, the minimum distance of the code should be more than the number of errors t.

$$d_{min} > t \tag{1.1}$$

In order to correct a number of occurring errors, the minimum distance of the code should be more than twice of the number of errors that occurred.

$$d_{min} > 2t \tag{1.2}$$

Lets say that we have a code with a $d_{min} = 2$, and a received codeword has only one error, then the decoder will be able to detect the error (see equation 1.1), but not to correct it (see equation 1.2). We can use the code mentioned in section 1.3.3 as an example that has a $d_{min} = 2$, If we have D1(001) transmitted and we got only one error in the first bit, becoming R(011), the decoder is going to realize that an error occurred but won't be able to make the right decision about the transmitted one. This is because d(R,D1) = 1, and

d(R,D2) = 1. If we assume that we have two errors in the received codeword that makes it R(111) and we find here that the error won't be detected because the received distorted version of D1 is one of the original codewords, D4, so the decoder will assume that it is a right codeword [3].

1.3.7. Minimum Weight of a Code

The weight ω of a codeword is the number of ones within it so if we have the code word (110110) the weight of this codeword $\omega(110110) = 4$. In order to find the minimum weight of a code we find the weights of all the codewords. We choose the weight of the codeword that has the minimum weight in the code to be the minimum weight of the code. This codeword shouldn't be the zero code word. For example, if we have a code with the following codewords

D1(0000), D2(0011), D3(1110), D4(1111),

$\omega(D1) = 0$

$\omega(D2) = 2$

$\omega(D3) = 3$

$\omega(D4) = 4$

The minimum weight here is not the weight of D1 because it is the zero codeword, the minimum weight $\omega_{min} = 2$, which is the weight of D2.

1.3.8. Maximum Likelihood Decoding Rule

The decoder will receive a codeword R, then it will calculate the probability of receiving R if each of the original codewords is transmitted, and will choose the codeword that has the maximum probability to be transmitted [1][3].

1.3.9. Minimum Error Decoding Rule

The decoder has stored information about the probability of transmitting each of the codewords, when the decoder receives a codeword R; it calculates the probability of transmitting the codeword if each of the original codewords is transmitted, and then will

multiply each of the found probabilities with the probability of transmitting the corresponding codeword [1][3].

Example: Assume we have three codewords D1, D2, D3, with the following transmission probabilities p(D1) = 0.2, p(D2) = 0.1, p(D3) = 0.7, and the decoder has received the codeword R and calculated the following probabilities p(R|D1) = 0.1, p(R|D2) = 0.6, p(R|D3) = 0.3. For the maximum likelihood decoding rule D2 is going to be picked by the decoder as the transmitted codeword because it maximize the value of p(R|Dn) but for minimum error decoding rule the probabilities p(R|Dn) will be multiplied by p(Dn) and the decoder will choose the codeword that maximize p(R|Dn).p(Dn) which means in our example D3 is going to be picked by the decoder.

1.4. Error Correcting Systems

1.4.1. Automatic Repeat Request

When the codeword is received without a detectable error the channel decoder sends an ACK (acknowledgment symbol 1) to the channel encoder through the feedback channel which means codeword is received without detectable error. When the channel decoder detects an error it sends a bit 0 to the channel encoder through the feedback channel. This symbol 0 is called NAK (non acknowledgment). The channel encoder transmits the codeword again until it get an acknowledgment (ACK bit 1) from the channel decoder, which means the codeword is received without a detectable error.

1.4.2. Forward Error Correction

In forward error correction receiver isn't responsible only of detecting errors, because in this system the feedback channel is not present (e.g. in satellite TV broadcasting, computer storage devices, etc….), so the ability of sending an ACK or NAK is impossible. The receiver should manage to correct the errors by itself, (make an acceptable decision about the transmitted codeword, based on the received one and the coding algorithm of the decoder).

1.4.3. Hybrid ARQ

Hybrid ARQ combines both techniques of forward error correction and automatic repeat request. There are two types of hybrid ARQ systems. The first type of hybrid ARQ is the one where the receiver receives a full codeword with all parity bits for detection and correction; if it fails to decide which codeword is transmitted it requests retransmission of data. In the second type the receiver receives the codeword with the parity bits for error detection, if it detects an error within the codeword it requests the correcting parity bits and attempts to correct. If it fails to correct it request the whole code word from the transmitter to be retransmitted.

1.5. Error Correcting Code Types

There are a lot of error correcting codes types, like turbo codes, which are a combination of two convolutional codes working in parallel, Reed-Muller codes, Reed-Solomon codes. We will only study briefly some basic codes and coding techniques.

1.5.1. Repetition Codes

Repetition codes repeat the message itself certain number of times. The repeated message is the redundancy in the code word. Assuming we have two messages to be transmitted that are: 0 and 1, codewords are 000 and 111.

Message		codeword
0	----	000
1	----	111

The encoder here repeats the message word two times resulting a Hamming distance d(000,111) = 3, between the two codewords which makes the decoder capable of detecting two errors and choosing the codeword in case of only one error occurred.

1.5.2. Odd/Even Parity Check Codes

The even parity check code checks the number of ones within the message word. If it's even it adds the bit 0 to the end of the message word to keep the number of ones even in

the codeword and if it's odd it adds the bit 1 to the end of the message word so we get a codeword with an even number of ones within.

The odd parity check codes does the same job except that they add 0 bit to the message words with odd number of 1's within so the generated codewords has odd number of ones. They add 1 bit to the code words with even number of 1's within so number of 1's in the codewords will be odd.

Message possibilities	even check parity codes	odd check parity codes
00(even)	000(even)	001(odd)
01(odd)	011(even)	010(odd)
10(odd)	101(even)	100(odd)
11(even)	110(even)	111(odd)

1.5.3. Linear Block Codes

A code is said to be linear if any two codewords in the code can be added on modulo-2 arithmetic to produce a third codeword in the code. Lets assume the following code with the following codewords, D1(0000), D2(0011), D3(1100), D4(1111),

If we add on modulo-2 any of the codewords to another codeword in the code the result must be another codeword of the code.

Example:

D1(0000) + D2(0011) = D2(0011)

D1(1111) + D2(0011) = D3(1100)

D2(0011) + D3(1100) = D4(1111)

D2(0011) + D2(0011) = D1(0000)

and so on, in order to generate a linear block code we should multiply the message word by a generator matrix. The result of the multiplication is our codeword that is going to be transmitted.

1.5.4. Cyclic Codes

Cyclic codes, [1] are a subclass of linear block codes. Actually the most important linear block codes discovered until today are either cyclic codes or closely related to it [1].

The advantage of cyclic codes over most other types of codes is that they are easy to encode. A binary code is said to be a cyclic code if it exhibits two fundamental properties:

1. Linearity property: the sum of any two codewords in the code is also a codeword.
2. Cyclic property: any cyclic shift of a codeword in the code is also a codeword.

1.5.5. Convolutional Codes

Convolutional codes [1] are different from linear block codes. In convolutional coding the encoder converts the entire data stream regardless of its length, into a single codeword. This means that a convolutional encoder doesn't generate codewords block by block like in linear block codes. In convolutional codes the message bits enter the encoder serially rather than in big blocks. A convolutional encoder operates on the incoming message sequence continuously in a serial manner.

1.6. Decoding and Performance of Linear Block Codes

The main purpose of using coding in communication systems is to increase the Euclidean distance between the transmitted signals and, hence, to reduce error probability at a given transmitted power. We see that this goal is achieved by choosing the code words to be as far apart on the vertices of the cube as possible. This means that a good measure for comparing the performance of two codes is the Hamming distance between the codewords. Keeping track of all distances between any two codewords is difficult and in many cases impossible. Therefore, the comparison between various codes is usually done based on the minimum distance of the code, which for linear codes is equal to the minimum weight of the code. As we mentioned before the code with higher minimum distance is more capable of error detection and correction.

1.6.1. Soft Decision Vs. Hard Decision

Each detected element r_i of the received vector r can be described as a quantized-amplitude. [6] The decision may simply answer the question "Is the amplitude greater or less than zero?" yielding a binary decision of 1 or 0. Such a decision is a called a hard

decision because the detector firmly selects one of two levels. Sometimes the detector's decision may answer multiple questions such as "Is the amplitude or less than zero, and is greater or less than some reference level?". For binary signaling such multipart decisions are called soft decisions; they offer the decoder side information about the SNR of the corrupted analog waveform.

A soft decision might tell the decoder, "This signal has a positive amplitude, but it is not very far from the zero amplitude." or " this signal has a positive amplitude, and it's quite far from the zero amplitude." The most popular soft-decision format entails eight-level signal quantization, which can be interpreted as a hard decision plus a measure of confidence. The figure-of-merit of the error performance of a digital communication system is usually expressed as a normalized SNR, known as the ratio of bit energy to noise power spectral density, E_b / N_0. The coding gain or benefit provided by an error-correcting code to a system can be defined as the "relief" or reduction in required E_b / N_0 that can be realized due to the code [3][6].

1.6.2. Soft Decision Decoding

We know that the optimum signal-detection scheme is based on minimizing the Euclidean distance between the transmitted signal and the received signal. This means that after receiving the output of the channel and passing it through the matched filters, we choose one of the message signals that are closest to the received signal in the Euclidean distance sense [7].

1.6.3. Hard Decision Decoding

A simpler and more frequently used decoding scheme is to make hard binary decisions on the components of the received signal vector (received codeword), and then to find the codeword that is closest to it in the Hamming distance sense [7].

1.6.4. Bit Error Vs. Block Error

If we assume an 8-bit data block (codeword) being transmitted over a noisy channel, and if we assume 64 blocks transmitted over the channel (512 bits divided into blocks), the number of bits having an error over the number of all transmitted bits is the bit error rate, while the number of blocks containing errors over the number of all transmitted blocks is the block error rate.

Bit Error rate = error received bits / all transmitted bits

Block Error rate = blocks received with errors / all transmitted blocks

To understand the relation between bit error and block error rates, lets assume that a digital communication system transmitted the following codewords over an additive white Gaussian channel in the following sequence (0011,1001,1100,1111). The receiver received the following corresponding codewords respectively (0001,1001,1110,1111). In this sequence of information, we have two blocks with errors caused by two error bits, one in each of the error blocks, (3^{rd} bit in the 1^{st} block, and 3^{rd} bit in the 3^{rd} block), resulting in:

Bit Error = 2 (number of error bits) / 16 (number of all transmitted bits) = $\frac{1}{8}$ bit error rate, which means 1 bit received with error for each 8 transmitted bits, and

Block Error = 2 (number of block received with error) / 4 (number of transmitted blocks) = $\frac{1}{2}$ block error rate, which means in each two transmitted blocks one is received with errors. On the other hand let us assume for the same transmitted codes we received the following codewords: (0011,1111,1100,1111). Here we have the same bit error rate, but both bit errors occurred in one block (2^{nd} block, 2^{nd} & 3^{rd} bits). The block error rate is $\frac{1}{4}$. This would look better than if errors occurred in two different blocks. Now, lets assume that the code here has a minimum distance of 2, which means the decoder is capable of detecting 1 error in each code word. Then the errors occurring in our first case are detectable errors, while in the second case the block error is not detectable. It is obvious that the 2^{nd} received codeword, although not the one transmitted, still one of the codewords of the code.

CHAPTER TWO

LINEAR BLOCK CODES

2.1. Basics of Linear Block Codes

A binary code is a linear code if and only if the sum of two codewords is also a codeword. The minimum distance of a linear code is the minimum of the weights of the non-zero codewords.

To get the codewords of a code we multiply all the possible message words by a matrix called the generator matrix. A generator matrix for a linear block code is a binary matrix whose rows are the codewords belonging to some basis of the code. Basis for a code are the codewords that can be linearly combined to give all the codewords.

Example: {0000,0001,1000,1001} is a linear code, {0001,1000} is a basis for the code.

Possible message inputs are {00,01,10,11}

So the generator matrix becomes $G = \begin{vmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{vmatrix}$,

And the input matrix is $W = \begin{vmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{vmatrix}$,

The generated codewords are

$S = W.G$ (2.1)

$$S = \begin{vmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{vmatrix};$$

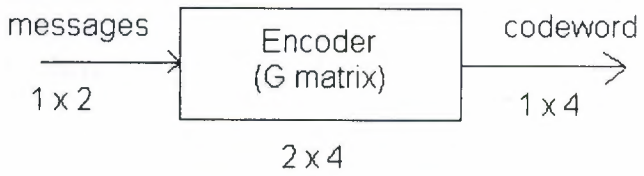

Figure 2.1 Processing of a message input of an encoder.

Figure 2.1 shows how a single message word of length 2 gets encoded, to become a codeword of length 4. The encoder for this generator matrix adds two redundancy bits to the two bits input message words.

2.2. Generator Matrix

We know that two codes with the same minimum distance have the same properties. Since two codes that can be obtained from each other and have the same minimum distance are said to be equivalent, the generator matrices of equivalent codes can be obtained of each other by applying elementary row operations and columns interchanging.

2.3. Canonical Form of Linear Block Codes

The generator matrix G is said to be in canonical form if it is in the form

G = | I ¦ A |, where I is a K x K identity matrix and A is an arbitrary (k x (n – k))

Binary matrix. The codewords obtained from a generator matrix that is in canonical form are said to be in systematic form. If we have the generator matrix G and the message words W of length k then our codewords S are the multiplication of the message words W and the canonical generator matrix G.

$$S = W.G_c \tag{2.2}$$

If the first k-bits of the codewords are the same as the bits of W then they are called systematic.

Example: if we have the following generator matrix

$$G = \left| \begin{array}{cccc} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{array} \right|,$$

We interchange the first and the third columns to get the canonical form of this generator matrix.

$$G_c = \left| \begin{array}{cc:cc} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{array} \right|, \text{ where } A = \left| \begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array} \right| \text{ and, } I = \left| \begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right|,$$

If we assume that the input message words are

$$W = \begin{vmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{vmatrix}$$, then the codewords are

$$S = W G_c = \left|\begin{array}{cc:cc} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{array}\right|$$

It is obvious that the first two bits of the codewords are the same bits of the message words. The code generated by the canonical form of G is equivalent to the code word generated by G. This means both have the same properties of error detection and correction and have the same minimum distance.

2.4. Parity Check Matrix

The parity check matrix [1] of a linear code with (k x n) generator matrix G is the ((n – k) x n) matrix H satisfying

$$G \cdot H^T = 0 \tag{2.3}$$

where H^T is the transpose of the parity check matrix H. and 0 denotes a ((n – k) x n) zero matrix. If G is in the canonical form $G = \left| I \,\vdots\, A \right|$ then $H = \left| A^T \,\vdots\, I \right|$, where I is the (n-k) x (n-k) identity matrix. If G is not in the canonical form we can find H by reducing G to the canonical form, then find the canonical form of H then reversing the column operations used to convert G to canonical form in order to convert the canonical form of H into the parity check matrix of G.

Example: if G is a generator matrix of a linear code

$$G = \begin{vmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{vmatrix}$$, the linear code generated by this matrix is

{00000,00001,00110,00111,11000,11001,11110,11111}.

By the following operations:

1. Replace the third row by the sum of the second and the third row.
2. Replace the second row by the sum of the first and the second row.

3. Interchange between the first and the fifth columns.
4. Interchange the second and the third columns.

We find the canonical form of G to be

$$G_c = \left|\begin{array}{ccc:cc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{array}\right|$$

As we know $G_c = | I \,\vdots\, A |$ and $H_c = | A^T \,\vdots\, I |$. Here we have the matrix A = $\left|\begin{array}{cc} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{array}\right|$,

$A^T = \left|\begin{array}{ccc} 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right|$. Therefore our canonical form of H is

$$H_c = \left|\begin{array}{ccc:cc} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{array}\right|.$$

Then to get the parity check matrix of the original generator matrix we apply column operations used to reduce our generator matrix to it's canonical form in reversed order:

1. Interchange the second and the third columns.
2. Interchange the first and the fifth columns.

So we find that H = $\left|\begin{array}{ccccc} 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{array}\right|$,

If we multiply G with the transpose of H we should get a zero matrix.

$$G.H^T = 0, \left|\begin{array}{ccccc} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{array}\right| . \left|\begin{array}{cc} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 0 \end{array}\right| = \left|\begin{array}{cc} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{array}\right|.$$

If we take the equation S = WG and we multiply both sides with the transpose of the parity check matrix H^T we will have S H^T = WG H^T. We know that $G.H^T = 0$ (from equation 2.3), therefore we get S $H^T = 0$.

If this relationship is true then S is a codeword that is resulted from G, if not then S is not resulted by the generator matrix G and does not belong to the codewords resulted by G.

2.5. Syndrome Decoding

Assume the codeword S is corrupted by noise e giving X = S + e. Then the syndrome of X is given by:

$$\delta = X + e \tag{2.4}$$

To decode the received codeword X and be able to pick up a codeword of the code as the transmitted codeword we should create a syndrome table showing the codeword of the minimum weight and the syndrome. When a word that is not a codeword is received we compute its syndrome and we add it to the codeword of the minimum weight to recover the uncorrupted codeword. We know from previous section (2.4.) that $S H^T = 0$,and if this statement is not true for a word then it does not belong to the code of G and its corresponding H^T.

Example: Assume we have the following generator matrix G.

$$G = \begin{vmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{vmatrix}, \text{ and its corresponding } H^T = \begin{vmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{vmatrix}$$

This G matrix generates the following codes:

{0000,0011,1100,1111}

If we multiply the matrix X (all possibilities of received words) by the transpose of the parity check matrix we get the syndrome matrix with the syndrome corresponding to each codeword, as only words with the syndrome zero are codewords of the code generated by the generator matrix G ($S H^T = 0$).

$$S H^T = \begin{vmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{vmatrix} \cdot \begin{vmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{vmatrix} = \begin{vmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{vmatrix}.$$

We find here that the multiplication of the code words with the transpose of the parity check matrix gives a zero matrix, which means that the syndrome for all the codewords is zero. If we multiply all the possibilities of received codeword X, we will get the syndrome for each received codeword.

$$XH^T =$$

$$\begin{vmatrix} 0&0&0&0 \\ 0&0&0&1 \\ 0&0&1&0 \\ 0&0&1&1 \\ 0&1&0&0 \\ 0&1&0&1 \\ 0&1&1&0 \\ 0&1&1&1 \\ 1&0&0&0 \\ 1&0&0&1 \\ 1&0&1&0 \\ 1&0&1&1 \\ 1&1&0&0 \\ 1&1&0&1 \\ 1&1&1&0 \\ 1&1&1&1 \end{vmatrix} \cdot \begin{vmatrix} 0&1 \\ 0&1 \\ 1&0 \\ 1&0 \end{vmatrix} = \begin{vmatrix} 0&0 \\ 1&0 \\ 1&0 \\ 0&0 \\ 0&1 \\ 1&1 \\ 1&1 \\ 0&1 \\ 0&1 \\ 1&1 \\ 1&1 \\ 0&1 \\ 0&0 \\ 1&0 \\ 1&0 \\ 0&0 \end{vmatrix}$$

Now we construct the syndrome table as follows:

Syndrome	*Code Words*				$W_{\min}$ *Word*
00	0000	0011	1100	1111	0000
10	0001	0010	1101	1110	0001
01	0100	0111	1000	1011	0100
11	0110	0101	1010	1001	0101

If a syndrome value has two codewords of the minimum weight then we choose one of them randomly to be added to the decoded received words.

The process of decoding works as follows:

1. The syndrome for the received codeword is calculated.
2. The received codeword is added to the code of the minimum weight of the same syndrome.
3. The resultant codeword is the correction of the received codeword (the decision made by the decoder).

If we have the code word (0011) to be transmitted by the system and a corruption has occurred that it became (0111) we add the received codeword to the $W_{\min}$ word of the same syndrome to find the corrected codeword. The syndrome of (0111) is (01) so we add it to the codeword (0100) to get the correction for this corruption. i.e. (0111) + (0100) = (0011) so the codeword (0011) is picked by the decoder to be the transmitted codeword.

When we have two codewords of the minimum weight for the same syndrome the decoder might choose the wrong codeword to be the transmitted one. Assume that (0011) is transmitted and a corruption has occurred so that the received codeword became to be (0001) that has the syndrome (10) in the syndrome table. We add the received code word to the codeword that has minimum weight at the same syndrome level in the syndrome table as follows :(0001) + (0001) = (0000) [which is a wrong decision].

If we receive a codeword without corruptions (that means the received codeword has a syndrome of [00]), we add it the minimum weight codeword of the syndrome [00], which is (0000). For example if we receive (0011) which is a codeword and we add it to (0000) we get the same codeword.

2.6. Length of the Codeword

Assume we have a message source that generates 4-bits/sec, and we want to encode these messages and add one redundancy bit for each message. In other words we want to have a code rate of 5/4 (5 is number of bit in code word, and 4 is number of bits in message words.) which is better, to encode every 4 bits and get codewords of 5 bits, or encode every 8 bits at once and get codewords of the length 10?

Lets assume that we have a system encoding four bits messages at once, adding one redundancy bit to each, then transmitting them sequentially through a noisy channel. Also, lets assume the code used here has the capability to correct one error, and the transmitter sent two blocks of information that were received with two bit errors.

The possible scenarios at the receiver here would be:

1- One error has been received in each block; here then we have no problem because the code is capable of correcting one bit error.
2- Bit errors occurred in one block, while the other block was free of errors. Here we have a problem; we have one clean block and another uncorrected block error.

What if we combine each two message words together? In order to have eight bit message words we add two redundancy bits, so our code rate will be 10/8 which is equal to 5/4. Here we did not change the code rate we just combined every two messages together.

Because of more redundancy our code is able to fix two errors. Wherever those errors occur, we do not care in this case, because the decoder is able to fix it.

2.7. Linear Cyclic Codes

As we mentioned before cyclic codes are branch of linear block codes [1], except that they have a special property. Cyclic codes are preferred because they require a minimal amount of hardware, and their effectiveness in detecting errors is greater than 99.9% [5].

For a linear block code to be a cyclic code, any cyclic shift in a codeword of the code, must give another codeword [1][5]. Because of these codes properties, we use the generator polynomial of the code instead of the generator matrix. We can also derive the parity check polynomial from the generator polynomial. If we assume that g(x) is the generator polynomial, m(x) is the message polynomial, c(x) is the code, then from g(x) we can find the parity check polynomial h(x) such that if we have a code of length n, then $g(x).h(x) = (X^n - 1)$, And $c(x).h(x) = 0$, as $m(x).g(x) = c(x)$.

Example: Assume we want to design a cyclic code of length 7 that encodes a sequence of messages of length 3. We need to obtain a generator polynomial that is a factor of $X^7 - 1$ in GF(2)[x]. (GF(2): binary Galois field (consists of two elements) [1].). Let α be a root of the primitive polynomial $X^3 + X + 1 = 0$. The conjugacy of the powers of α with respect to GF(2) and their associated minimal polynomials are as follows:

$\{1\} \leftrightarrow X + 1$

$\{\alpha + \alpha^2 + \alpha^4\} \leftrightarrow X^3 + X + 1$

$\{\alpha^3 + \alpha^6 + \alpha^5\} \leftrightarrow X^3 + X^2 + 1$

The generator polynomials for binary cyclic code of length 7 must have one or more of the above polynomials as factors. Consider the case $g(x) = (X^3 + X + 1).(X + 1)$. The generator polynomial in this case is $g(x) = X^4 + X^3 + X^2 + 1$, the corresponding parity polynomial is $h(x) = (X^7 + 1)/g(x) = X^3 + X^2 + 1$. The message polynomials consist of all binary polynomials of degree less than or equal to 2. The corresponding code is thus a (7,3) code with the following eight code words [1].

m(x).g(x) Factorization	**Code polynomial c(x)**	**Code Word**
0.g(x)	0	(0000000)
1.g(x)	$1 + X^2 + X^3 + X^4$	(1011100)
X.g(x)	$X + X^3 + X^4 + X^5$	(0101110)

$X^2.g(x)$	$X^2 + X^4 + X^5 + X^6$	(0010111)
$(X^2+1).g(x)$	$1 + X^3 + X^5 + X^6$	(1001011)
$(X^2+X+1).g(x)$	$1 + X + X^4 + X^6$	(1100101)
$(X+1).g(x)$	$1 + X + X^2 + X^5$	(1110010)
$(X^2+X).g(x)$	$X + X^2 + X^3 + X^6$	(0111001)

In the previous example it is obvious that any cyclic shift in any of the codewords will give another codeword from the same code [1].

2.8. Generator Matrix for cyclic codes

If we have the following generator polynomial for a code of length n

$$g(x) = g_n.X^n + g_{n-1}.X^{n-1} + \cdots + g_1.X + g_0 \tag{2.5}$$

and a generator polynomial of degree k as k = (n-r):

$$h(x) = h_k.X^k + h_{k-1}.X^{k-1} + \cdots + h_1.X + h_0 \tag{2.6}$$

Then the generator matrix G and the parity check matrix H is as follows:

$$G = \begin{vmatrix} g_0 & g_1 & \cdots & g_n & & & & 0 \\ & g_0 & g_1 & \cdots & g_n & & & \\ & & \ddots & \ddots & \ddots & \ddots & & \\ & & & g_0 & g_1 & \cdots & g_n & \\ 0 & & & & g_0 & g_1 & \cdots & g_n \end{vmatrix}; \text{ and}$$

$$H = \begin{vmatrix} h_k & \cdots & h_1 & h_0 & & & & 0 \\ & h_k & \cdots & h_1 & h_0 & & & \\ & & \ddots & \ddots & \ddots & \ddots & & \\ & & & h_k & \cdots & h_1 & h_0 & \\ 0 & & & & h_k & \cdots & h_1 & h_0 \end{vmatrix}.$$

Example: The generator matrix, the parity check matrix for the generator polynomial and parity check polynomial in the previous example are as follows:

$g(x) = X^4 + X^3 + X^2 + 1$, $h(x) = X^3 + X^2 + 1$,

$$G = \begin{vmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{vmatrix}, \text{ and}$$

$$H = \begin{vmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{vmatrix}.$$

This generator matrix can be used to generate the cyclic code in the previous example, and this parity check matrix can be used to create the syndrome decoder for this code. There are also another encoding and decoding techniques for cyclic codes, like the systematic encoding algorithms and systematic decoding algorithms, and shift register encoders and decoders [1][5].

CHAPTER THREE

DIGITAL IMAGES

3.1. What is an Image?

An image is a representation of a real scene, either in black and white or in color, and either in print form or in a digital form, printed images may have been reproduced either by multiple colors and grayscale or by a single ink source [2].

3.2. What is Digital Image?

Digital images is consisted of pixels, each pixel carries the information of colors at the point it is located at, pixels all together forms the digital image that our eyes can see on the screen of a digital imaging system like a digital camera, computer, or even a digital TV receiver which converts this pixels into analogue waves to be shown on the TV screen [2].

3.3. Classification of Images

3.3.1. RGB Images

In colored images each pixel has three color vectors one for green, one for red, and one for blue, each of those color vectors has a value between 0 and 255 representing the concentration of the color within the pixel. Those three-color vectors within the pixel are mixed to give different colors that human eyes can see like (white, red, black, violet, brown, etc.). We can control the brightness of a colored image by controlling the value of the color vectors of the image pixels, if we decrease all the pixel values together the colors in the image are going to be decreased which means that the image will become darker while if we increase the color vectors of the pixels of an image the image will become brighter. We can also control the contrast, which is the degree of color mixing between the pixels of an image.

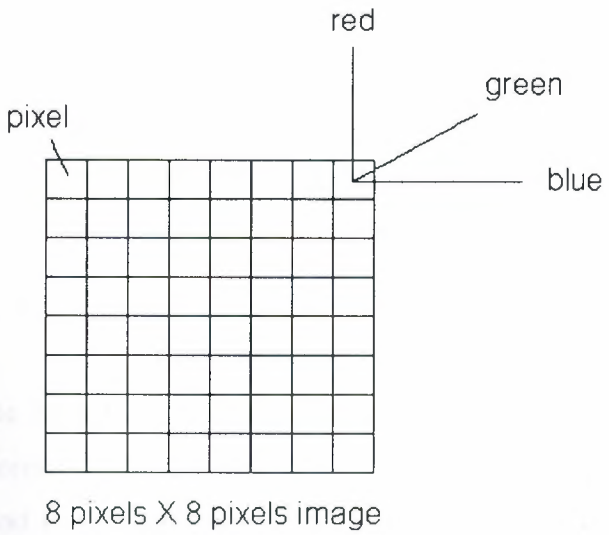

Figure 3.1 Pixels in an image

Figure 3.1 shows how color vectors in each pixel are represented. The squares are pixels, all together forming the digital image that a human eye can see. This 64 pixels image (8x8 pixels) is a simple digital image while in reality digital images consist of much more than this number of pixels. For example in mobile cellular phone cameras number of pixels might reach 3 mega pixels and even more. As mentioned before each color vector in the pixel of a colored image has a value between 0 and 255. That means we have 256 possible values and each color vector can be represented in 8-bit binary form data. In order to represent one pixel of a colored image we need 24 bits to represent the three color vectors of the image.

To understand more how colored image is displayed we can imagine three images: red scale image, blue scale image, and a green scale image, placed on each other to perform a color mixing, giving the RGB-image.

3.3.2. Grey Level Images

Gray scale images has only one color vector. The value of this color vector represents the level of the gray color in the pixel that varies between 0 for very dark gray (black) and 255 which is very bright gray (white).

As long as we have only one color vector represented with 8-bits of data in each pixel then we can represent each pixel with an 8-bits data representing the gray color value in it. We can increase the brightness of a grayscale image by increasing all the pixel values together, which will increase the color level in all the pixels uniformly. Increasing the contrast of a grayscale image will increase the color mixing between the pixels [2].

3.3.3. Black and White Images

Black and white images, also called binary images, are the first digital images found. Each pixel is formed of pure black color or pure white color. The pixel value 0 represents pure black and 1 represents pure white. Each pixel in the black and white image can have two color possibilities, black or white, 0 or 1. Therefore we only need 1-bit data to represent a pixel in a black and white image.

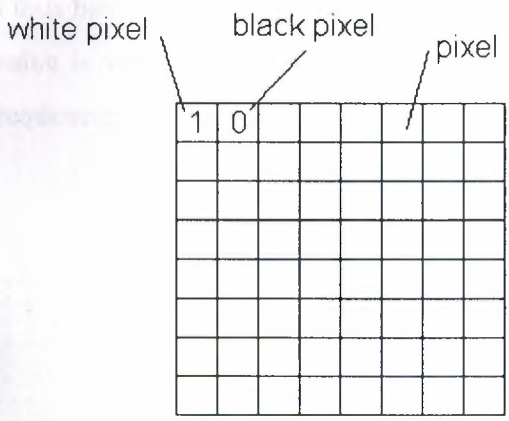

Figure 3.2 Black and white image pixels.

3.4. Edges and Noise in Digital Images

Consider sequentially plotting the gray values of the pixels of an image on the (y-axis) against pixel locations on the (x-axis) as the image is scanned. The result will be a discrete time plot of varying amplitudes showing the intensity of light at each pixel. Lets say that we are at the sixth row and looking at pixels from 81 to 96 at a 256 pixels image (16 x 16). The intensities of the pixels number 82 and 88 are so different from pixels around them and may be considered to be noise.

In general, noise is information that does not belong to the surrounding environment. The intensities of pixels 86 and 90 are also different from the neighborhood pixels and may indicate a transition between the object and the background; thus these pixels can be constructed as representing the edges of the object (see figure 3.3) [2].

3.5. Image Filtering and Noise Reduction

Although we are discussing a discrete signal, this signal may be transformed into a large number of sines and cosines with different amplitudes and frequencies that if added together, will construct a signal. Slowly changing signals (such as small changes between succeeding pixel gray values) will require few sines and cosines in order to be constructed, and thus have low frequency content. On the other hand quickly varying signals (such as large differences between pixel gray levels) will require many more frequencies to be reconstructed, and thus have high frequency content. Both noises and edges are instances in which one pixel value is very different from the neighboring ones, thus noises and edges create the larger frequencies of a typical frequency spectrum [2].

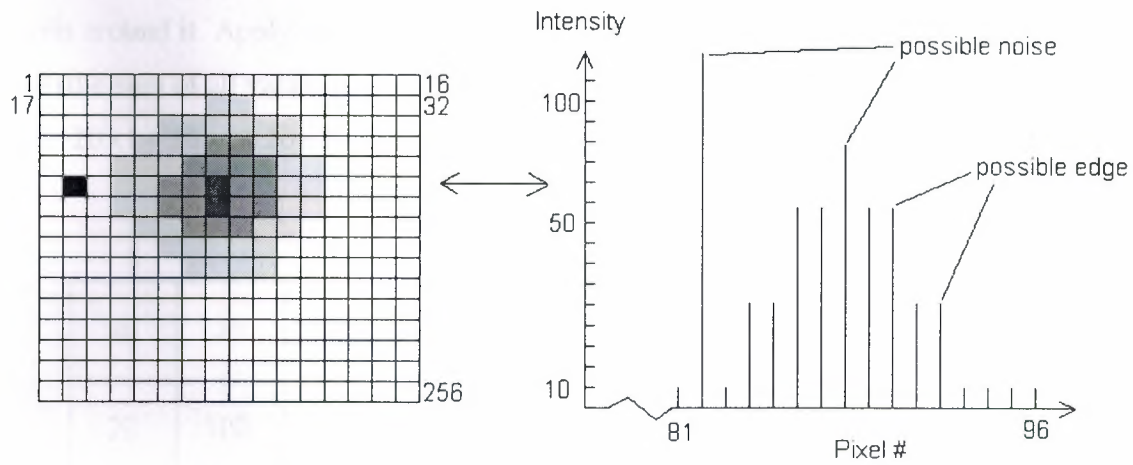

Figure 3.3 Noise and edge information in an intensity diagram of an image

3.5.1. Image through Low-Pass Filter

If a large frequency is passed through a low-pass filter (a filter that allows lower frequencies to go through without much attenuation in amplitude, but that severely

attenuates the amplitudes of the higher frequencies in the signal) the filter will reduce the influence of all high frequencies, including the noises and edges. This means that, although a low pass filter will reduce noise, it will also reduce the clarity of an image by attenuating the edges, thus softening the image throughout.

3.5.2. Image Through High-Pass Filter

A high-pass filter will increase the apparent effect of higher frequencies by severely attenuating the low-frequency amplitudes. In such cases, noises and edges will be left alone, but slowly changing areas will disappear from the image.

3.5.3. Neighborhood Averaging

Neighborhood averaging can be used to reduce noise in an image, but it also reduces the sharpness of the image. Consider the (3 x 3) mask shown in the Figure 3.4 together with its corresponding values, as well as a portion of an imaginary image with its gray levels indicated. As it can be seen, all the pixels but one are at gray value of 20. The pixel with a gray level of 100 may be considered to be noise, since it is different from the pixels around it. Applying the mask over the corner of the image with a normalizing value of 9 (the sum of all values in the mask), yields:

$$X = (20\times1 + 20\times1 + 20\times1 + 20\times1 + 100\times1 + 20\times1 + 20\times1 + 20\times1 + 20\times1)/9 = 29$$

20	20	20	20		
20	100	20	20		
20	20	20	20		

1	1	1
1	1	1
1	1	1

Figure 3.4 Neighborhood averaging mask.

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

1	2	1
2	4	2
1	2	1

Figure 3.5 (5×5) and (3×3) Gaussian averaging filters.

As a result of applying the mask on the indicated corner, the pixel with the value 100 changes to 29 and the large difference between the noisy pixels and the surrounding pixels (100 vs. 20) becomes much smaller (29 vs. 20), thus reducing the noise. With this characteristic, the mask acts as a low pass filter. This averaging low-pass filter will also reduce the sharpness of edges, making the resulting image softer and less focused. We can also use (5 x 5) filters, which give better results, but require a bit more processing.

There are other averaging filters, such as the Gaussian averaging filter (also called the mild isotropic low-pass filter), which is shown in Figure 3.5. This filter will similarly improve an image, but with slightly different results.

3.5.4. Frequency Domain

When the Fourier transform of an image is calculated, the frequency spectrum might show a clear frequency for the noise, which in many cases can be selective eliminated by proper filtering.

3.5.5. Median Filters

One of the main problems in using neighborhood averaging is that, along with removing noises, the filter will blur edges [2]. A variation of this technique is to use a

median filter, in which the value of the pixel is replaced by the median of the values of the pixels in a mask around the given pixel, stored in ascending order. A median is the value such that half of the values in the set are below and half are above the median. Since, unlike an average, the median is independent of the value of any single pixel in the set, the median filter will be much stronger in eliminating spike-like noises without burring the object or decreasing the overall sharpness of the image. Consider the image in Figure 3.6 (a). The gray values, in ascending order, are 1, 2, 3, 4, 5, 6, 7, 8, and 9. The middle value is 5, resulting in the image (b) in Figure 3.6. Observe that the image has become grainy because the pixel sets with similar values appear longer (as in 5 and 5) [2].

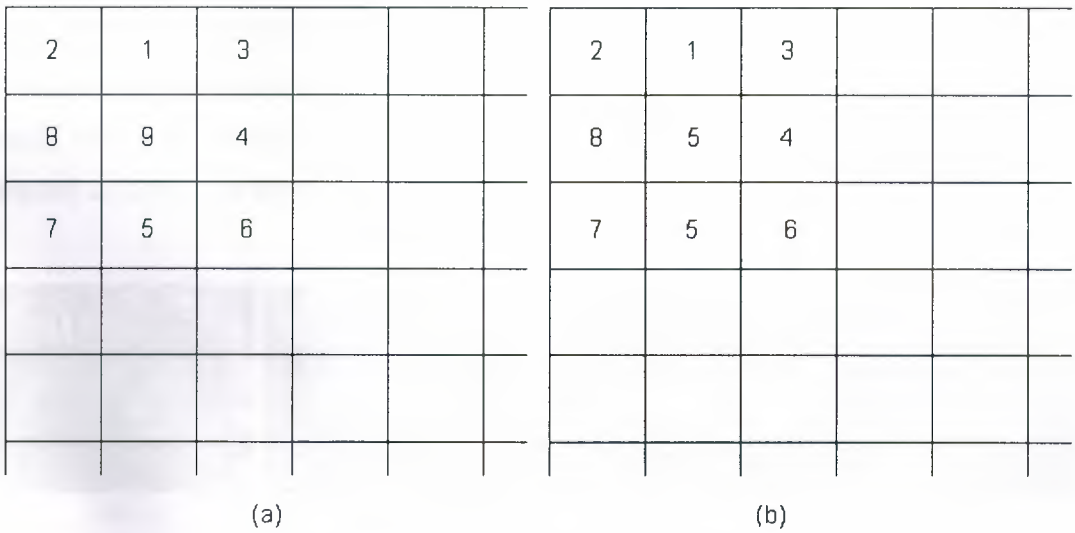

Figure 3.6 Application of a median filter.

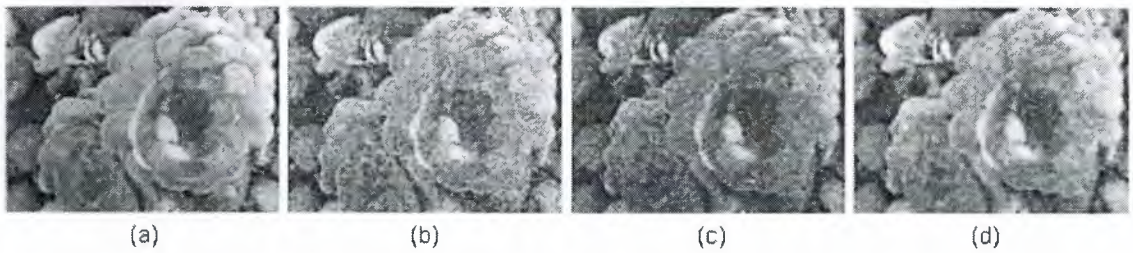

Figure 3.7 (a) original image. (b) the image corrupted with a random Gaussian noise. (c) the image improved b a (3×3) median filter. (d) the same image improved by a (7×7) median filter.

In Figure 3.7 we can see how that the effect of a (7 x 7) filter on a corrupted image is better than the effect of a (3 x 3) filter mask, but as we mentioned before the bigger the mask we use the more processing we need.

3.6. Image Sampling and Resolution

Sampling an image [2] is representing an image in a digital form (pixels), as each pixel of the image represents a part of it. As we mentioned before the pixel is the smallest piece of a digital image.

The resolution of an image is the number of samples forming it. If we sample an image of an object at a rate of 64-kilo pixels (having 65536 pixels forming the image), and we sample an image of the same object at a rate of 256 pixels (256 pixels forming the image), the image sampled at 64-kilo pixel resolution is going to be more clear than the one sampled at 256 pixels resolution.

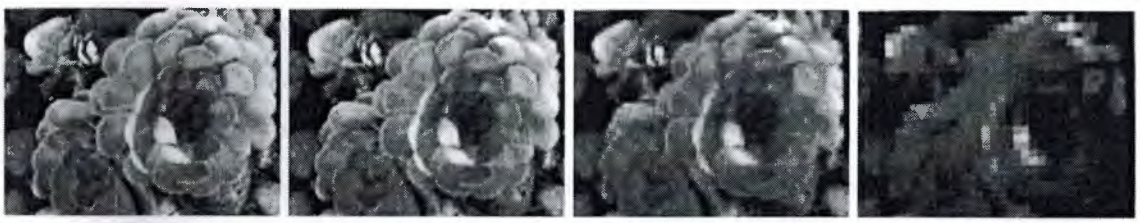

Figure 3.8 Effect of different sampling rates on an image. At (a) 432×576 pixels, at (b) 108×144 pixels, at (c) 54×72 pixels, at (d) 27×36 pixels. As the resolution decreases, the clarity of the image diminishes accordingly.

3.7. Storage of Digital Images

After the image is converted to its digital form by any imaging device (scanner, digital camera, etc.) [4] The image can be stored in a hard disk or any other storage device in different formats each has different properties and advantages, the reason behind that are the different compression methods used for each. Some compression methods discard too much information in order to have a small file size. This type of compression causes a lot of information loss, so when we attend to view the image again, we are not going to have

an image with the same quality of the original. Since loss less compression does not discard much information, we are not going to have much reduction in the image size, but at the same time we will keep the quality of the image. Some image file formats do not do any kind of compression, like .raw files (uncooked images), They just discard some information related to the printer and some other applications.

Storage of computer files on the hard disk also includes some coding techniques. This includes image files and also the communication of data between CD drives, random access memory, and other devices within the computer, etc. The reason behind using coding within the computer is to overcome the magnetic interference, which cannot be eliminated by isolating the main board layers.

3.8. Image Restoration

Image restoration methods are used to improve the appearance of an image by application of restoration process that uses a mathematical model for image degradation. The modeling of the degradation process differentiates restoration from enhancement where no such model is required [4].

Examples of the types of degradation considered include blurring caused by motion or atmospheric disturbance, geometric distortion caused by imperfect lenses, superimposed interference patterns caused by mechanical systems, and noise from electronic sources. It is assumed that the degradation model is known, or can be estimated, the degradation process is modeled and its inverse is applied to restore the original image.

3.9. Image Enhancement

Image enhancement techniques are employed to emphasize, sharpen, and/or smooth image features for display and analysis. Image enhancement is the process of applying these techniques to facilitate the development of a solution to a computer imaging problem, Consequently, the enhancement methods are application specific and are often developed empirically. Enhancement methods operate in the spatial domain manipulating the pixel data, or in the frequency domain by modifying the spectral components, some enhancement algorithms use both [4].

The type of technique include point operations, where each pixel is modified according to a particular equation that is not dependent on other pixel values, mask operations where each pixel is modified according to the values in a small neighborhood (sub image), or global operations where all the pixel values in the image are taken into consideration. Spatial domain processes include all three types, but frequency domain operations, by nature of the frequency (and sequence) transforms are global operations, and can become mask operations by performing the transform on small image blocks instead of the entire image.

Examples of image enhancement techniques include edge detection, modifying the brightness and/or the sharpness of the image, increasing or decreasing the image contrast and many other possible processes [4].

CHAPTER FOUR

RESULTS

4.1. A System Without Coding

We took a 256 by 256 pixels image (figure 4.1) and converted it into binary blocks. Each block is of 16 bits length. We transmitted it through a noisy channel that adds a random binary noise pattern to the 16 bits binary data blocks. The receiver will receive the message and reform the image back. In this model we are not using any type of coding technique. We will just try to filter the received noisy image using a median filter.

Figure 4.1 Original image

Figure 4.2 Noisy form of the image

In Figure 4.2 we can see the effect of adding random binary noise pattern to our original image. Subtracting the noisy image from the original image will give us the noise pattern image. Figure 4.3 shows the noise pattern added to the image. If we filter the received noisy image with a median filter we are going to have very good results. Figure 4.4, shows the same received noisy image after filtering it with a median filter. The only problem is that the median filter we are using is going to blur the edges, which might be obvious in the image.

Figure 4.3 Noise pattern

Figure 4.4 Image filtered with Median filter

Figure 4.5 shows the remaining noise pattern in the filtered image. It is obvious that the median filter has filtered most of the noise, but some losses in the edges have occurred. Because of the way the median masks operates, pixel values are going to be averaged and therefore, our image is not going to be exactly the same of our original image. However, differences are not going to be recognized by the human eye. The median filter as we discussed before causes the edges to be blurred (see section 3.5.5.).

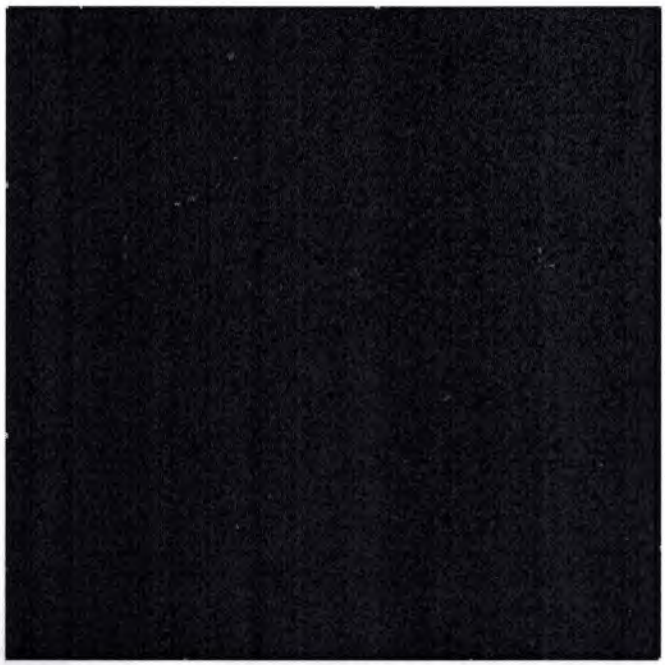

Figure 4.5 Noise Pattern after filtering

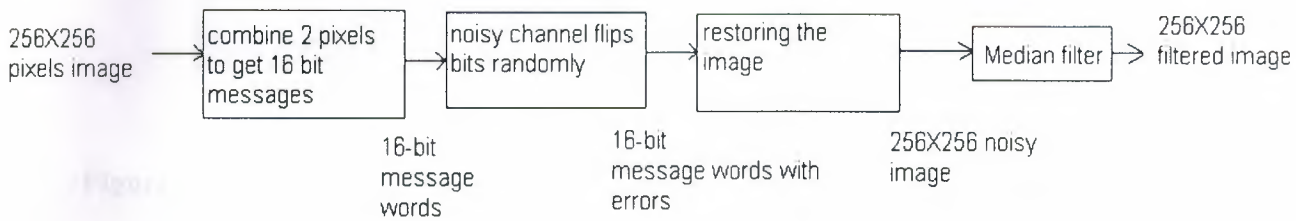

Figure 4.6 Block diagram of the system in section 4.1

4.2. A System With Channel Coding

If we encode the 16 bit messages using cyclic coding technique by adding 2 redundancy bits, we will get 18 bit codewords. After adding noise to the transmitted information in the channel, the receiver will receive the information, decode it and try to correct the occurring errors in the channel. The result is shown in Figure 4.6.

Figure 4.7 Image received by a decoder of a system using 2-redundency bits

As we can see, not many errors are corrected when only 2 redundancy bits are added to the message words. We can get better results by adding more redundancy. Lets try adding 5 bits of redundancy to get 21 bit codewords. The received image when using 5 bits of redundancy is totally free of errors (figure 4.7).

Figure 4.8 Image Received by the decoder of a 5-redundency bits system

Figure 4.9 Black image (empty noise pattern)

In Figure 4.8 we can see the empty noise pattern, when we used a 5 bit redundancy. As we mentioned before, the noise pattern we are taking here is the received image subtracted from the original image. Here we can see that we do not have noise at all in our received image.

Of course it differs from person to another when it comes to design a system. Some engineers for some applications prefer low redundancy accepting the errors that is going to occur. Some of them prefer not to have errors at all so they use more redundancy. There are also a lot of code types that are so powerful like turbo codes that are able to correct more errors with the same amount of redundancy. Still a lot of research and experiments are being done in order to correct as many errors as possible of errors with the least amount of redundancy bits.

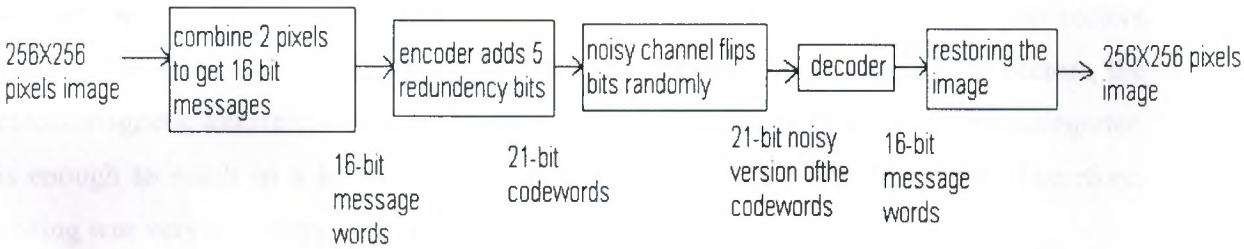

Figure 4.10 Block diagram for the system in section 4.2

CONCLUSION

Digital revolution has introduced a lot of efficient signal processing techniques that can not be performed in analog systems. These techniques, like applying masks on images, require that the image is in spatial domain and can not be applied on continuous waveforms. A very important of these techniques is coding.

Coding technique is used in communication systems, computer storage, and also a lot of other applications like the communication between a digital camera and a computer, and a lot of similar devices.

In computer storage whenever we want to store information on the hard disk, the computer divides the information into blocks of data, adds the redundancy bits and stores it. When we open the file that contains the information stored, a decoder will decode this data so we can use it. The coding technique has helped a lot in keeping the information resistant against losses, especially given that hard disks are very sensitive devices and their sectors can get damaged easily. Even if the hard disk does not have any damages in its sectors, the electromagnetic interferences between the main board layers, and the parts of the computer, is enough to result in a lot of errors while storing the data to the hard disk. Therefore, coding was very necessary for computer storage.

To apply coding in a digital communication system, we need to set an encoder at the transmitting part, and a decoder at the receiving part. An encoder generates codewords depending on the message words entering the input side. The codewords generated by the encoder usually have some extra bits in order for the decoder to be capable of detecting and correcting errors. Even though decoders make wrong decisions some times about what codeword has been transmitted, still the coding technique is so efficient and these situations do not occur very frequently.

Linear block codes, and cyclic codes encode each block of data (message word) separately by multiplying the data block by the generator matrix. The result of this multiplication is the codeword corresponding to the message word.

The longer the codewords generated by a certain code, the better error detection and correction capabilities it has, and the more channel capacity it requires, since there is more information to be transmitted.

If an image is transmitted through a noisy channel, the receiver can apply filtering techniques that would reduce the noise and blur the edges within the image. The can be encoded and transmitted through the noisy channel, which would decrease the number of occurring errors within the received image.

The more redundancy bits used to encode the message words, the better results we get, but the more channel capacity required. Therefore, to design a code the channel capacity available must be concerned.

REFERENCES

[1] Wicker S. B., Error Control Systems for Digital communication and storage, Prentice Hall, Upper Saddle River, New Jersey, 1995.

[2] Niku S. B., Introduction to Robotics, Analysis, Systems, Applications, Prentice hall Inc., Upper Saddle River, New Jersey, 2001.

[3] Haykin S., Communication Systems, John Wiley & Sons Inc., Toronto, 1994.

[4] Umbaugh S. E., Computer imaging: digital image analysis and processing, CRC Press, Boca Raton, Florida, 2005.

[5] Mamedov F., Telecommunications, Near East University press, Nicosia, 2000.

[6] Sklar B. and Harris F. J., "The ABCs of Linear Block Codes", *IEEE Signal Processing Magazine,* Vol. 21, Issue 4, July 2004, pp. 14-35.

[7] Proakis J. G., Salehi M., Communication Systems Engineering, Prentice Hall, Upper Saddle River, New Jersey, 2002.

APPENDIX A

A matlab program that model a communication system transmitting an image. It adds noise in the channel, receives it, and applies the median filtering technique to clean the noise within. It displays the original image, received noisy image, noise pattern of received image, filtered image, noise pattern of filtered image:

```
clc
clear
M = Imread('elaine.256.tif');
W = zeros(256,256);
for i = 1:256
  for j = 1:256
     c = M(i,j);
     W(i,j) = c;% changing the image array to a double array
  end
end
A = de2bi(W);%convert the double array into pixel sequence of 8 bits
msg = zeros(32768,16);%create msg matrix
x = 1;
for i = 1:32768
   for j = 1:8
      c = A(x,j);
      msg(i,j)=c;
   end
   for j = 9:16
      c = A(x+1,j-8);
      msg(i,j)=c;
   end
    x = x + 2;%fill msg matrix to become info sequene of 16 bits
end
%encoder side
%decoder side
errval = randerr(32768,16,[0 1]);%noise values
noisycode=rem((msg+errval),2);%adding noise
rseq=noisycode;
rmsg=zeros(65536,8);
l = 1;
t = 2;
for i = 1:32768
   for j = 1:8
      y=rseq(i,j);
      rmsg(l,j)=y;
   end
   l=l+2;
   for j = 9:16
```

```
      y=rseq(i,j);
      rmsg(t,j-8)=y;
    end
    t=t+2;
end
Q=bi2de(rmsg);
recim=zeros(256,256);
x=1;
for i = 1:256
    for j = 1:256
        y = Q(x,1);
        recim(j,i)=y;
        x=x+1;
    end
end
recimag = uint8(recim);%we got our image back as a uint8 matrix
figure(1),
imshow(M);
figure(2),
imshow(recimag);
NP=imsubtract(M,recimag);
figure(3),
imshow(NP);
MFimage=medfilt2(recimag,[3,3]);%filtering the image with a median filter
figure(4),
imshow(MFimage);
MP=imsubtract(M,MFimage);
figure(5),
imshow(MP);
```

APPENDIX B

Matlab program modeling a communication system that encodes an image using cyclic codes with 2 redundancy bits, transmits it, adds noise to the codewords being transmitted, receives and decodes the image, Displays the original image, relieved image, noise pattern:

```
clear
clc
M = Imread('elaine.256.tif');
W = zeros(256,256);
for i = 1:256
  for j = 1:256
     c = M(i,j);
     W(i,j) = c;% changing the image array to a double array
  end
end
A = de2bi(W);%convert the double array into pixel sequence of 8 bits
msg = zeros(32768,16);%create msg matrix
x = 1;
for i = 1:32768
   for j = 1:8
      c = A(x,j);
      msg(i,j)=c;
   end
   for j = 9:16
      c = A(x+1,j-8);
      msg(i,j)=c;
   end
    x = x + 2;%fill msg matrix to become info sequene of 16 bits
end
%encoder side
code = encode(msg,18,16,'cyclic/binary');%adding 2 bit redundency,
errval = randerr(32768,18,[0 1]);
noisycode=rem((code+errval),2);
%decoder side
newmsg = decode(noisycode,18,16,'cyclic');
rseq=newmsg;
%to image form
rmsg=zeros(65536,8);
l = 1;
t = 2;
for i = 1:32768
   for j = 1:8
      y=rseq(i,j);
      rmsg(l,j)=y;
   end
```

```
    l=l+2;
    for j = 9:16
        y=rseq(i,j);
        rmsg(t,j-8)=y;
    end
    t=t+2;
end
Q=bi2de(rmsg);
recim=zeros(256,256);
x=1;
for i = 1:256
    for j = 1:256
        y = Q(x,1);
        recim(j,i)=y;
        x=x+1;
    end
end
recimag = uint8(recim);%we get our image back as a uint8 matrix
figure(1),
imshow(M);
figure(2),
imshow(recimag);
NP=imsubtract(recimag,M);
figure(3),
imshow(NP);
```

APPENDIX C

Matlab program modeling a communication system that encodes an image using cyclic codes with 2 redundancy bits, transmits it, adds noise to the codewords being transmitted, receives and decodes the image, displays the original image, received image, noise pattern:

```
clear
clc
M = Imread('elaine.256.tif');
W = zeros(256,256);
for i = 1:256
  for j = 1:256
     c = M(i,j);
     W(i,j) = c;% changing the image array to a double array
  end
end
A = de2bi(W);%convert the double array into pixel sequence of 8 bits
msg = zeros(32768,16);%create msg matrix
x = 1;
for i = 1:32768
   for j = 1:8
      c = A(x,j);
      msg(i,j)=c;
   end
   for j = 9:16
      c = A(x+1,j-8);
      msg(i,j)=c;
   end
    x = x + 2;%fill msg matrix to become info sequene of 16 bits
end
%encoder side
code = encode(msg,21,16,'cyclic/binary');%adding 5 bit redundency,
errval = randerr(32768,21,[0 1]);
noisycode=rem((code+errval),2);
%decoder side
newmsg = decode(noisycode,21,16,'cyclic');
rseq=newmsg;
%to image form
rmsg=zeros(65536,8);
l = 1;
t = 2;
for i = 1:32768
   for j = 1:8
      y=rseq(i,j);
      rmsg(l,j)=y;
   end
```

```
  l=l+2;
  for j = 9:16
     y=rseq(i,j);
     rmsg(t,j-8)=y;
  end
  t=t+2;
end
Q=bi2de(rmsg);
recim=zeros(256,256);
x=1;
for i = 1:256
   for j = 1:256
      y = Q(x,1);
      recim(j,i)=y;
      x=x+1;
   end
end
recimag = uint8(recim);%we get our image back as a uint8 matrix
figure(1),
imshow(M);
figure(2),
imshow(recimag);
NP=imsubtract(recimag,M);
figure(3),
imshow(NP);
```