Eximia Journal
Vol. 3, 202-208, December, 2021
ISSN: 2784-0735
www.eximiajournal.com

Analysing Grayscale Images Using Hamming Code to Identify the Level of Bit Errors that Resulted from Alternative Noise Levels Added

Azeezah Natiq Ghanim, Riyadh Zaghlool Mahmood

aziza.20csp36@student.uomosul.edu.iq, azizanatiq1234@gmail.com, riyadh.zaghlool@uomosul.edu.iq

Abstract. Error Detection has an increased importance over the years, protecting data by encrypting or decrypting the data to preserve the desired method to the correct location without data being stolen. There are many error detection methods using various types of algorithms such as Bose-Chaudhurihocquenghem (BCH) and Hamming Code. Hamming code can further be split into three main types; Standard Hamming Code; Extended Hamming Code, and Extended Hamming Product Code. These three types can use even or odd parity to obtain the desired results, mirroring TV devices and satellite channels in this study, an image had Noise with various levels added to it, measuring the Bit Error Rate before and after correction, with Bit Error Rate resulting in 0 for Noise Levels ≥ 6. With many different types of errors ranging from single to burst errors, finding an algorithm to implement with Hamming Code to reduce Bit Error after correction without repeating the algorithm is a key target for future Error Detection.

Keywords. Hamming Code, Error Detection, Noise, Bit Error, Algorithms, Matlab, Error Correction, Noise to Error Ratio, Even Parity

Introduction

Error Detection has become increasingly more important, with the main use protecting and sending sensitive information. Errors have and can be intentional to protect sensitive transmission of information, as well as errors becoming apparent from background noise that has been added. Various types of error detection methods have been created and used such as double error correction (DEC) BCH, and Hamming codes (1).

Error correction codes (ECCs) protect data from soft errors that can cause data corruption the memories ae protected with ECC reducing the impact on memory complexity because of the impact on memory design causing the data to be coded and decoded; when the information is sent and received respectively (2). BCH codes are used to detect more complex errors than Hamming Code, because of two main reasons; 1) The greater the amount of '1' present in some parity check equations; 2) the greater the number of parity check equations that require more gates to check all values are equal to zero (2-4).

Eximia Journal
Vol. 3, 202-208, December, 2021
ISSN: 2784-0735
www.eximiajournal.com

In any environment error detection is very important, with interference occurring in forms of environmental interference, physical fault noise, electromagnetic radiation and many other types of noise and disturbances (2) (9). There are three different types of errors that can corrupt data that is being sent to a receiver which are 1) Single Bit Error, 2) Multiple Bit Errors, 3) Burst Errors. Single Bit Errors are a single bit error when bit changing or errors occur, multiple bit errors are multiple bit errors respectively. However, burst errors are in a set of bits in the transmitted codeword, then the burst error occurs. The calculation will be calculated from the first bit which is changed until the final changed bit.

Hamming code are error-detection codes designed to correct single-bit error in digital data transmission, the popularity of hamming codes is based on their ability to detect errors up to a single bit, which can be performed on the generation of several parity bits. For every data value, parity bits are added to generate the Hamming Code, represented in binary form (5).

The three types of Hamming Code are standard hamming code which was used and implemented in this study, extended hamming code, and extended hamming product code (6) (16). The standard method is dependant on a minimum hamming distance which is three among any two codewords that is needed. For example, a standard hamming code with a dimension of H (7, 4) which four data bits are encoded into seven bits by appending 3 redundancy bits (10).

The number of parity check bits is determined by the Hamming inequality rule, the number of parity check bits is known as three for four-bit data. This ratio is referred to as (7,4) hamming code with the parity check bits being appended to the data bits. In total the number of bits within a (7,4) hamming code word is seven, in which when the parity check bits are appended, they are referred to as the hamming code word (7-8), simplifying this information (7,4) code encoded four data bits into seven bits by the addition of three parity bits (4). The aim of the parity check bit and the related data bits is to achieve Even Parity. The meaning of Even Parity is the number of ‘1’ bits is identified (5)(9). Determining the parity bit can also be performed with odd parity, however in this study we focused on the use of even parity (14-15).

In 2019, Caleb Hillier and Vipin Balyan presented three variations of Hamming codes are tested both in Matlab and VHDL. In 2015 Chukwuma Okeke and M. Eng presented a comparative study between Hamming code and Reed-Solomon (RS) code in byte error detection and correction. In 2010 Mario Blaum and Sugata Sanyal proposed that shortening hamming code can detect a maximal number of double errors constructed. Many other studies have presented various types and tests of hamming code in various situations. In this study we analysed a grey level image to understand the correlation between the amount of noise applied to the number of errors detected and corrected.

Method

A grey level image was used which was 128x128, it was in binary matrix format. The grey level image was then translated into mono matrix. The array was transformed from decimal to binary form resulting in 128 x128 x8. Four parity bits (p1-4) were then added to data word 1, 2,4 and 8 resulting in 128 x128 x12 called cord word.

The 12 bits harboured 8 original bits along with 4 parity bits, with the parity bits enabling the identification of any noise present. From the 128 arrays in intervals of 12, in which r1, r2, r3

Eximia Journal
Vol. 3, 202-208, December, 2021
ISSN: 2784-0735
www.eximiajournal.com

and r4 are extracted by using the number location, for example for r1 which was extracted from p1 one code word is taken followed by missing the adjacent code word.

Syndrome was extracted by the addition of the parity bits to the equation, creating A, B, C and D which is read from D to A. The Binary data was translated to decimal form identifying if and where the error is. If the value of 0 is present in decimal form no error is present. The presence of a value larger than 12 means there is more than one error presence, a value of less than 12 means one error is present the location of the error is identified, and that specific parity is flipped to the opposite value.

The parity bits were removed, and the binary format was translated back into decimal format and then reversed the algorithm by putting it back into Binary format resulting 128 x128.

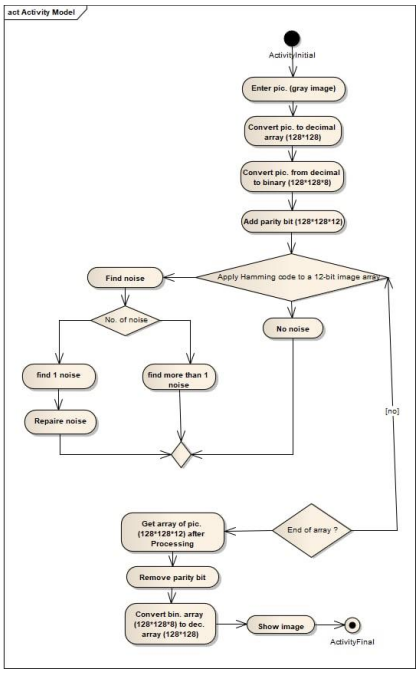

Figure 1 shows an activity model of the method used in this study to identify correlations between noise and errors detected.

Results

The results obtained from the study are shown below, with images, tables and graphs illustration the relationship between Noise and Bit Errors.

Eximia Journal
Vol. 3, 202-208, December, 2021
ISSN: 2784-0735
www.eximiajournal.com

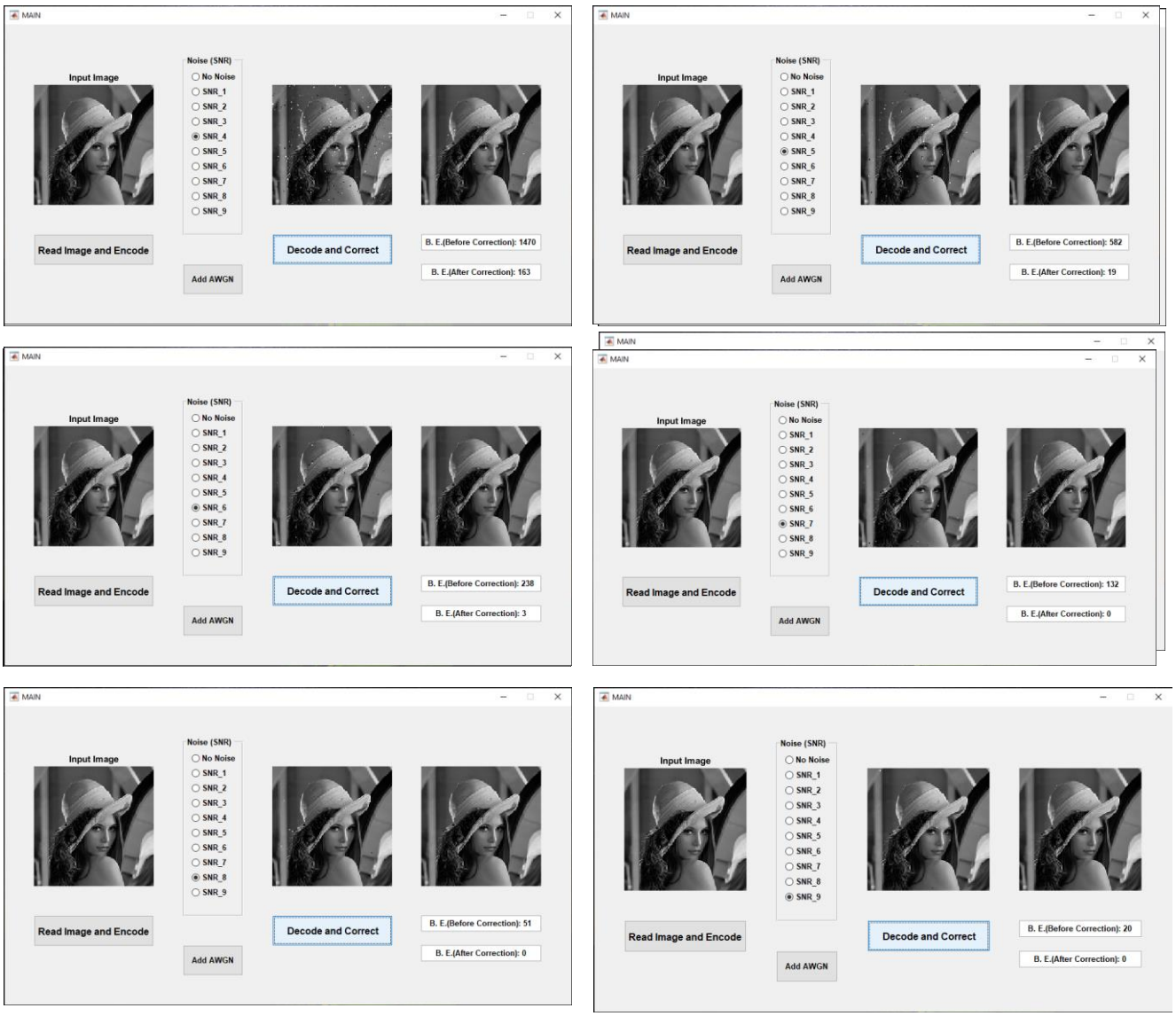

Figure 2 shows the Bit Error (BE) value before and after correction when No noise is added, and Noise (SNR) 1-9 is added. With BE Values before correction ranging from 0 to 21 801. Whilst BE values after correction ranged from 0 – 20 302.

Figure 2 shows the values of BE before and after correction depending on the SNR level, figure 1 has shown that an increase in SNR does not correlate with BE before or after correction, rather this a myth only. Rather it is negatively correlated, a Bit Error Rate between BE rate and SNR graph in figure 3 illustrates this. Figure 2 below allows visualization of BE values before and after correction on a simpler scale.

Eximia Journal
Vol. 3, 202-208, December, 2021
ISSN: 2784-0735
www.eximiajournal.com

Noise (SNR) Level Added	Bit Error (B.E) Value B. Correction	Bit Error (B.E) Value A. Correction
No Noise	0	0
SNR_1	21801	20302
SNR_2	8034	4052
SNR_3	3279	767
SNR_4	1470	163
SNR_5	582	19
SNR_6	238	3
SNR_7	132	0
SNR_8	51	0
SNR_9	20	0

Figure 3 Shows the B.E before and after correction depending on level of SNR applied. There is a clear negative correlation which is amplified in figure. 3.

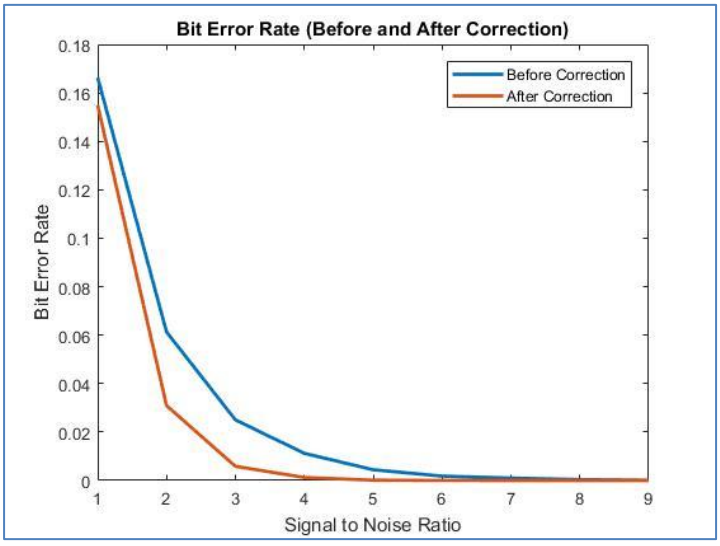

Figure 4 Shows the relationship between B.E Rate and SNR.

Conclusions

This study enabled the identifications that most error detection and correction algorithms have the ability to detect error caused by noise during the processes of transmission of data along electronic channels of only one binary order, however, to correct these errors more than one binary order is required to improve the complex algorithms which are still identified as classic models. Most of the algorithms applied in this study returned a BER value of 0, when the value of SNR was ≥ 6. This not only was shown in this study but has been seen across other algorithms such as LDPC and MINSUM (8).

The higher the SNR value the lower the image distortion from background noise, meaning the blocks used in the algorithm which had a size of eight binary orders were exposed to noise from only one binary order, after the detection of these and corrections the values returned as 0. To obtain lower BER values a repetition of this algorithm process or other algorithms can be done, however this comes at the expense of time and resources therefore finding a algorithm that is more adaptable and reduces noise on a greater level will increase productivity and use of Hamming code. More so it can be noted when only one binary rank is distorted to each block of data and after applying the error correction algorithms, BER will always equal 0.

References

[1] C. Hiller and V. Balyan, “Error Detection and Correction On-Board Nanosatellites Using Hamming Codes,” 2019.
[2] P. Reviriego et al, “Efficient error detection in Double Error Correction BCH codes for memory applications,” 2011.
[3] R. Naseer and J. Draper, “DEC ECC Design to Improve Memory Reliability in Sub-100nm Technologies,” 2008.
[4] P. Ankolekar, “Error Correction Methods for Latency-Constrained Flash Memory Systems,” 2008.
[5] A. H. Saleh, “Design of Hamming Code For 64 Bit Single Error Detection And Correction Using VHDL,” 2014.
[6] Dr. A. K. Singh, “Error Detection and Correction by Hamming Code,” 2016.
[7] C. Chan and C. Chang, “An efficient image authentication method based on Hamming code,” 2005.
[8] U. K. Kumar and B. S. Umashankar, “Improved Hamming Code for Error Detection and Correction,” 2007.
[9] S. Hekmat, “Communication Networks,” [Online]. Available: http://www.pragsoft.com/books/CommNetwork.pdf [Accessed: 09-Feb-22].
[10] L. Williams, “Hamming code: Error Detection and Correction with Examples” 2017. [Online]. Available : https://www.guru99.com/hamming-code-error-correction-example.html [Accessed: 10-Feb-22]
[11] C. Hiller and V. Balyan, “Error Detection and Correction On-Board Nanosatellites Using Hamming Codes,” 2019.
[12] C. Okeke and M. Eng, “A Comparative Study between Hamming Code and Reed-Solomon Code in Byte Error Detection and Correction,” 2015.

[13] M. Blaum and S. Sanyal, “Shortening Hamming Codes maximizing Double Error Detection,” 2010. 14
[14] Fauzi et al, “Bit Error Detection and Correction with Hamming Code Algorithm,” 2021.
[15] A. Fauzi et al, “Bit Error Detection and Correction with Hamming Code Algorithm,” 2022.
[16] Reviriego, “Hamming SEC-DAED and Extended Hamming SEC-DED-TAED Codes Through Selective Shortening and Bit Placement,” 2014.