

**ENHANCEMENT OF REVERSIBLE IMAGE  
STEGANOGRAPHY AND OPTIMIZATION  
OF QUANTUM IMAGE REPRESENTATION  
USING THE NEQR MODEL**

**A Thesis Submitted  
in Partial Fulfilment of the Requirements for the  
Degree of**

**MASTER OF TECHNOLOGY  
in  
INFORMATION TECHNOLOGY**

**by  
SUMITRA SINGH  
(Roll No. 2K23/ITY/26)**

**Under the Supervision of  
Prof. (Dr.) Dinesh Kumar Vishwakarma  
Head, Department of Information Technology**



**Department of Information Technology  
DELHI TECHNOLOGICAL UNIVERSITY  
(Formerly Delhi College of Engineering)  
Shahbad Daultapur, Main Bawana Road, Delhi-110042, India**

**May, 2025**

# CONTENTS

Candidate's Declaration	v
Certificate	vi
Acknowledgement	vii
Abstract	viii
List of Figures	x
List of Tables	xiii
List of Abbreviation	xiv
<b>1. REVERSIBLE IMAGE STEGANOGRAPHY .....</b>	<b>01</b>
1.1. Introduction	01
1.2. Performance Metrics for Steganography	03
1.2.1. Payload Capacity	03
1.2.2. Imperceptibility	03
1.2.3. Undetectability	06
1.2.4. Security	06
1.3. Literature Review on Reversible Image Steganography	06
1.4. Histogram Shifting Based Reversible Image Steganography	08
1.4.1. Embedding Algorithm	08
1.4.2. Extraction Algorithm	10
1.4.3. Lower Bound on PSNR	12
1.4.4. Computational Complexity	12
1.5. Image Interpolation Based Reversible Image Steganography	12
1.5.1. Neighbour Mean Interpolation	13
1.5.2. Interpolation by Neighbouring Pixels	14
1.5.3. Left Vertex Interpolation	16
1.5.4. Data Embedding and Extraction	17
1.5.5. Payload Capacity	18
1.5.6. Lower Bound on PSNR	18
1.6. Difference Expansion Based Reversible Image Steganography	19
1.6.1. Reversible Integer Transform	19

1.6.2. Expandable Difference Values	20
1.6.3. Changeable Difference Values	21
1.6.4. Embedding Algorithm	21
1.6.5. Extraction Algorithm	23
<b>2. ANALYSIS OF HISTOGRAM SHIFTING BASED REVERSIBLE IMAGE STEGANOGRAPHY .....</b>	<b>25</b>
2.1. Review of Related Work	25
2.2. Capacity Analysis	27
2.2.1. Analysis for Histogram Partitioning	28
2.2.2. Analysis for Image Partitioning	30
2.3. Histogram Analysis	31
2.4. Experimental Results	32
2.4.1. Capacity for Histogram Partitions	32
2.4.2. Capacity for Image Partitions	33
2.4.3. Results for Histogram Analysis	35
2.5. Conclusion	36
<b>3. TWO LAYER REVERSIBLE IMAGE STEGANOGRAPHY IN IMAGE HISTOGRAMS.....</b>	<b>37</b>
3.1. Introduction	37
3.2. Payload Adaptive Histogram Shifting	37
3.3. Analysis of Lower Bound on PSNR	39
3.4. Two Layer Embedding Strategy	39
3.5. Robustness to Steganalysis	43
3.6. Performance Comparison	45
3.7. Conclusion	47
<b>4. INTRODUCTION TO QUANTUM COMPUTING .....</b>	<b>48</b>
4.1. Background	48
4.2. Principles of Quantum Mechanics	51
4.2.1. Superposition	52
4.2.2. Entanglement	52
4.2.3. Decoherence	53

4.2.4. Interference	54
4.3. Quantum Logic Gates	54
4.3.1. Single-Qubit Gates	55
4.3.2. Multi-Qubit Gates	56
4.3.3. Universal Quantum Gates	58
4.4. Quantum Circuits	58
4.5. Experimental Methodology	59
4.5.1. Qiskit Installation	59
4.5.2. Quantum Circuit Creation	60
4.5.3. Ideal Simulation	60
4.5.4. Noisy Simulation	60
4.5.5. Execution on Quantum Hardware	61
<b>5. QUANTUM ARITHMETIC OPERATIONS .....</b>	<b>63</b>
5.1. Quantum Circuits for Half Adder and Full Adder	63
5.2. Quantum Circuits for Binary Adder	63
5.2.1. Toffoli-Based Adder	64
5.2.2. QFT-Based Adder	64
5.3. Quantum Comparator Circuit	65
5.4. Circuit Transpilation	65
5.5. Experimental Results	66
5.5.1. Half Adder and Full Adder Results	66
5.5.2. Toffoli-Based Adder Results	67
5.5.3. QFT-Based Adder Results	68
5.5.4. Quantum Comparator Results	69
5.6. Comparison of Toffoli and QFT Adders	70
5.7. Conclusion	71
<b>6. OPTIMIZATION AND PARALLEL IMPLEMENTATION OF NEQR.....</b>	<b>72</b>
6.1. Quantum Image Representation	72
6.2. Algorithm for Circuit Development	74
6.3. Optimization of MCX Gate Decomposition	74
6.3.1. Decomposition Algorithm	75



6.3.2. Comparative Analysis	76
6.4. NEQR Circuit for a $2 \times 2$ Image	77
6.5. Parallel Bit-Plane NEQR	78
6.5.1. Quantum Representation	78
6.5.2. Circuit for a $2 \times 2$ Image	79
6.5.3. Circuit Complexity Analysis	80
6.6. Experimental Results	81
6.6.1. MNIST Dataset	81
6.6.2. Basic Operations on NEQR Images	81
6.6.3. Comparison of Sequential and Parallel NEQR	83
6.6.4. Quantum Image Obfuscation	90
6.7. Conclusion	91
<b>REFERENCES.....</b>	<b>92-99</b>
Plagiarism Report .....	100
Publications .....	103



**DELHI TECHNOLOGICAL UNIVERSITY**  
(Formerly Delhi College of Engineering)  
Shahbad Daultpur, Main Bawana Road, Delhi-110042

### **CANDIDATE'S DECLARATION**

I, Sumitra Singh, 2K23/ITY/26 student, hereby certify that the work which is being presented in the thesis entitled “**Enhancement of Reversible Image Steganography and Optimization of Quantum Image Representation using the NEQR Model**” in partial fulfillment of the requirement for the award of the Degree of MASTER OF TECHNOLOGY in INFORMATION TECHNOLOGY, submitted to the Department of Information Technology, Delhi Technological University, Delhi is an authentic record of my own work carried out during my degree under the supervision of Professor Dinesh Kumar Vishwakarma.

The matter presented in this report/thesis has not submitted by me for the award of any other degree of this or any other Institute/University.

Place: Delhi

Sumitra Singh

Date:

(2K23/ITY/26)



**DELHI TECHNOLOGICAL UNIVERSITY**  
(Formerly Delhi College of Engineering)  
Shahbad Daultpur, Main Bawana Road, Delhi-110042

## **CERTIFICATE**

I hereby certify that the Project Dissertation “**Enhancement of Reversible Image Steganography and Optimization of Quantum Image Representation using the NEQR Model**”, submitted by Sumitra Singh, Roll No 23/ITY/26, to the Department of Information Technology, Delhi Technological University, Delhi in partial fulfilment of the requirements for the award of the Degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge, the above work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi

Date:

Prof. Dinesh K. Vishwakarma  
(SUPERVISOR)



**DELHI TECHNOLOGICAL UNIVERSITY**  
(Formerly Delhi College of Engineering)  
Shahbad Daulatpur, Main Bawana Road, Delhi-110042

## **ACKNOWLEDGEMENT**

I wish to express my sincere gratitude to **Professor Dinesh Kumar Vishwakarma**, Head of the Department of Information Technology, Delhi Technological University, for providing valuable support, guidance, consistent motivation, and supervision throughout the course of this project.

I am also thankful to the **Department of Information Technology at DTU** for providing the necessary infrastructure and laboratory facilities, which were instrumental in conducting the research required for the successful completion of this work.

In addition, I express my appreciation to the **Defence Research and Development Organization (DRDO)** for sponsoring my studies at DTU and providing me with the opportunity to pursue research in my area of interest.

Place: Delhi

Sumitra Singh

Date:

(2K23/ITY/26)

## ABSTRACT

Reversible steganography allows for exact reconstruction of the cover media after hidden data extraction, making it vital for applications such as content authentication, medical imaging, and military communications. Various reversible steganography techniques include histogram shifting, image interpolation, and difference expansion. Histogram shifting methods apply shifting to pixel-domain histograms or prediction error histograms. Prediction error histogram methods offer higher embedding capacity, but they are more complex, lack a guaranteed lower bound on PSNR, and are more susceptible to histogram-based steganalysis. Pixel-domain histogram shifting techniques, though simpler and more efficient with a theoretical PSNR bound, generally have lower embedding capacity.

Under this project, experiments are conducted on pixel-domain histogram shifting-based techniques. The capacity and histogram for varying number of non-overlapping image blocks and histogram blocks are analyzed. Experimental results show that embedding in image blocks does not significantly enhance the capacity compared to embedding in histogram blocks. Analysis of histogram blocks shows that embedding in two blocks yields the optimal results. A method is developed for making histogram shifting adaptive to payload size and a two layer embedding is developed for improved hiding capacity. Compared to previous methods, the two-layer embedding achieves higher capacity, better resistance to steganalysis, and maintains the PSNR acceptable for real-world applications.

Quantum computing is an advancing field that offers significant speed advantages for certain computational tasks over classical computing. Notable examples include Shor's algorithm, which efficiently solves integer factorization and discrete logarithm problems, and Grover's algorithm, which accelerates the search process in unstructured databases.

Quantum computing is based on quantum arithmetic operations where addition forms the core of all operations, as subtraction, multiplication, exponentiation, and division

can all be reduced to repeated or modified forms of addition. Experiments are conducted for performance analysis of quantum addition on quantum hardware. Development of quantum circuits for addition and comparison, including half adders, full adders, Toffoli-based adders, QFT-based adders (utilizing the Quantum Fourier Transform), and quantum comparators is carried out using IBM Qiskit. The circuits are first validated on ideal simulators to confirm correctness, followed by testing on noisy simulators to emulate real quantum hardware conditions. Final execution is carried out on IBM's Eagle 127-qubit Quantum Processing Unit (QPU). Results show that computation accuracy on actual hardware is limited by physical constraints such as short qubit coherence times and instability. A performance comparison shows that Toffoli-based adders outperform QFT-based adders in terms of accuracy, making them more reliable for precise arithmetic computations.

Quantum image representation provides exponential efficiency in image storage and processing. It relies on the fundamental principles of superposition and entanglement. NEQR (Novel Enhanced Quantum Representation) is a lossless encoding method used to represent digital images on a quantum computer. It is widely applicable in domains such as quantum machine learning, image steganography, and quantum image analysis.

This work introduces two enhancements to the NEQR framework: (1) Optimizing the decomposition of Multi-Controlled NOT (MCX) gates into Toffoli gates, and (2) Parallelizing the NEQR by parallel bit-plane encoding of the NEQR circuit, where the NEQR circuit is simultaneously constructed for each of the eight bit-planes of an image, thereby reducing overall circuit depth. Experimental results demonstrate that these enhancements lead to reduced circuit depth and faster execution, thereby mitigating decoherence-related errors. Additionally, quantum image processing operations that demonstrate exponential speedup over classical approaches — such as image negation, rotation, and intensity superposition — are also implemented and evaluated as part of this work.

## List of Figures

Figure No.	Figure Name	Page No.
Figure 1.1	Block diagram of a reversible image steganography system	02
Figure 1.2	Flowchart of histogram-based embedding algorithm	10
Figure 1.3	Flowchart of histogram-based extraction algorithm	11
Figure 1.4	Neighbour mean interpolation with $K = 2$	14
Figure 1.5	Interpolation by neighbouring pixels with $K = 2$	15
Figure 1.6	Left Vertex Interpolation with $K = 2$	16
Figure 1.7	Schematic of the interpolation-based reversible data hiding	17
Figure 1.8	Horizontal and vertical pixel pairing	22
Figure 2.1	Histograms of two $8 \times 8$ images	29
Figure 2.2	Embedding capacity for histograms of Figure 2.1	29
Figure 2.3	Histogram Blocks: (a) Two blocks (b) Three blocks	29
Figure 2.4	Algorithm for finding two blocks in an image histogram	30
Figure 2.5	(a) Image Blocks (b) Corresponding histograms	31
Figure 2.6	Histogram analysis for a 21-level gray-scale image	32
Figure 2.7	Standard test images for evaluation	33
Figure 2.8	Histogram variations after embedding in image blocks	35
Figure 2.9	Impact of embedding in $4 \times 4$ image blocks on histograms	36
Figure 3.1	Reduced block width in Lena image for 1000-bit embedding	38
Figure 3.2	Algorithm for finding payload adaptive peak	38
Figure 3.3	Payload adaptive PSNR for standard images	39
Figure 3.4	Stego images after multi-layer embedding	40

Figure 3.5	Stego image histograms after multi-layer embedding	40
Figure 3.6	Block diagram of two layer embedding	42
Figure 3.7	Algorithm for two layer embedding	43
Figure 3.8	Embedding rates for selected standard images	47
Figure 4.1	Quantum circuits for classical gates	59
Figure 5.1	Quantum circuits for half adder and full adder	63
Figure 5.2	Toffoli-based adder circuit for 7+7	64
Figure 5.3	QFT-based adder circuit for 3+3	64
Figure 5.4	Comparator circuits for (a) 2bit (b) 3bit (c) 4bit	65
Figure 5.5	Transpiled circuits for Toffoli and QFT adders for 1+1	65
Figure 5.6	Half adder and full adder results on simulator and QPU	66
Figure 5.7	Noisy simulator results for Toffoli-based adder	67
Figure 5.8	QPU results for Toffoli-based adder	68
Figure 5.9	Noisy simulator results for QFT-based adder	68
Figure 5.10	QPU results for QFT-based adder	69
Figure 5.11	Quantum comparator accuracy on simulator and QPU	70
Figure 5.12	Accuracy comparison of Toffoli- and QFT-based adders on QPU	71
Figure 6.1	A $2 \times 2$ image and its quantum representation	73
Figure 6.2	Algorithm for quantum image representation using NEQR	74
Figure 6.3	Algorithm for MCX to CCX decomposition	75
Figure 6.4	Optimized decomposition of 3-CX and 4-CX gates	76
Figure 6.5	(a) Original circuit (b) Optimized circuit	76



Figure 6.6	Comparison of circuit depths in MCX decomposition	77
Figure 6.7	NEQR circuit for image of Figure 6.1	77
Figure 6.8	Parallel Bit-Plane NEQR circuit for image of Figure 6.1	80
Figure 6.9	Sample MNIST dataset	81
Figure 6.10	Circuits for basic operations on QIR of MNIST images	82
Figure 6.11	(a) Original (b) Negation (c) Rotation (d) Superposition	83
Figure 6.12	Circuit depth comparison for ideal simulation	86
Figure 6.13	Circuit depth comparison for noisy simulation	88
Figure 6.14	Quantum image obfuscation and recovery	91

## List of Tables

<b>Table No.</b>	<b>Title</b>	<b>Page No.</b>
Table 2.1	Hiding capacity for varying histogram blocks	34
Table 2.2	Hiding capacity for varying image blocks	34
Table 3.1	Theoretical PSNR for varying layers of embedding	39
Table 3.2	Hiding capacity for multi-layer embedding	41
Table 3.3	PSNR for multi-layer embedding	41
Table 3.4	Statistical steganalysis on SIPI database	44
Table 3.5	Statistical steganalysis on Petitcolas database	45
Table 3.6	Performance comparison on hiding capacity and PSNR	46
Table 3.7	Comparison of adaptive histogram shifting for Lena image	46
Table 3.8	Comparison with prior methods	46
Table 4.1	Two-qubit gates and their unitary matrices	57
Table 4.2	Unitary sets of Quantum gates	58
Table 5.1	Quantum comparator results	69
Table 5.2	Noisy simulator and QPU results for Toffoli and QFT adders	70
Table 6.1	Quantum gates for NEQR-based image representation	72
Table 6.2	Circuit depths for sequential and parallel NEQR for MNIST digits	85
Table 6.3	Simulation time for sequential and parallel NEQR on ideal simulator	85
Table 6.4	Transpiled circuit depths for sequential and parallel NEQR on noisy simulator	87
Table 6.5	Observations for sequential and parallel NEQR on QPU	89

## **List of Abbreviations**

BPP	:	Bits Per Pixel
LSB	:	Least Significant Bit
MSE	:	Mean Square Error
PSNR	:	Peak Signal-to-Noise Ration
SSIM	:	Structured Similarity Measure Index
NCC	:	Normalized Cross Correlation
QPU	:	Quantum Processing Unit
NISQ	:	Noisy Intermediate-Scale Quantum
QFT	:	Quantum Fourier Transform
QIR	:	Quantum Image Representation
NEQR	:	Novel Enhanced Quantum Representation

# CHAPTER 1

## REVERSIBLE IMAGE STEGANOGRAPHY

### 1.1. Introduction

Steganography is the art and science of invisible communication. The term steganography emerged in late 15<sup>th</sup> century, but the idea has existed for thousands of years. Historically, secret messages were hidden beneath wax tablets, written on animal skins, or even tattooed onto the shaved heads of messengers. During wartime, especially for espionage, invisible ink was the popular method for covert communication. With advances in photographic processing, microdot and microfilm technology were developed and used during the Second World War.

With advent of digital technology and the Internet, digital steganography has emerged as a method of embedding one data type within another. It is commonly used alongside cryptography, providing an additional layer of protection to encrypted data. The digital media used for hiding, known as cover media, include text, images, audio, and video files, where redundant data in pixels or frames can be used for hiding. Various image steganography tools are available on the Internet that use images as cover media. These tools include two core components: embedding and extraction. The embedding process hides secret data in cover image, resulting in a stego, which needs to be transmitted over an insecure channel like the Internet. The receiver then applies the extraction algorithm to recover the hidden information from the stego image.

The modifications to cover during embedding are invisible to the human visual system and undetectable by statistical analysis. These modifications can either be reversible, allowing the cover to be restored after data extraction, or irreversible, preventing reconstruction of the cover. Based on this, image steganography is classified into two types: reversible and irreversible.

Irreversible techniques are primarily used in applications such as secret communication, watermarking, and fingerprinting. Widely used irreversible methods include Least Significant Bit (LSB) replacement and Pixel Value Differencing (PVD)

in images [1]. In LSB substitution, LSBs of randomly chosen pixels of cover are replaced by message. Pixel choice is typically governed by a Random Number Generator (RNG), seeded with a secret stego key. Since there is no way to determine whether a particular pixel was modified during embedding, the process is irreversible, introducing permanent, albeit imperceptible, distortion to the cover image. Video steganography methods use video files as cover media, providing significantly higher capacity than images [2].

Reversible steganography is used when the payload (data to be hidden) is associated with cover and reconstruction of cover is required for subsequent use. For instance, in medical images (CT scan, MRI, X-ray), diagnostic and patient details are embedded into the images and transmitted over the Internet among healthcare professionals, facilitating faster diagnosis and treatment while ensuring privacy. Watermarks are embedded in artwork images for copyright protection. Aerial images are embedded with relevant information. Military applications embed encrypted data within geographical maps. In these scenarios reversible steganography is useful for recovering the cover after data extraction. It is also known as lossless, invertible, or distortion-free steganography. Figure 1.1 shows a schematic of a reversible image steganographic system.

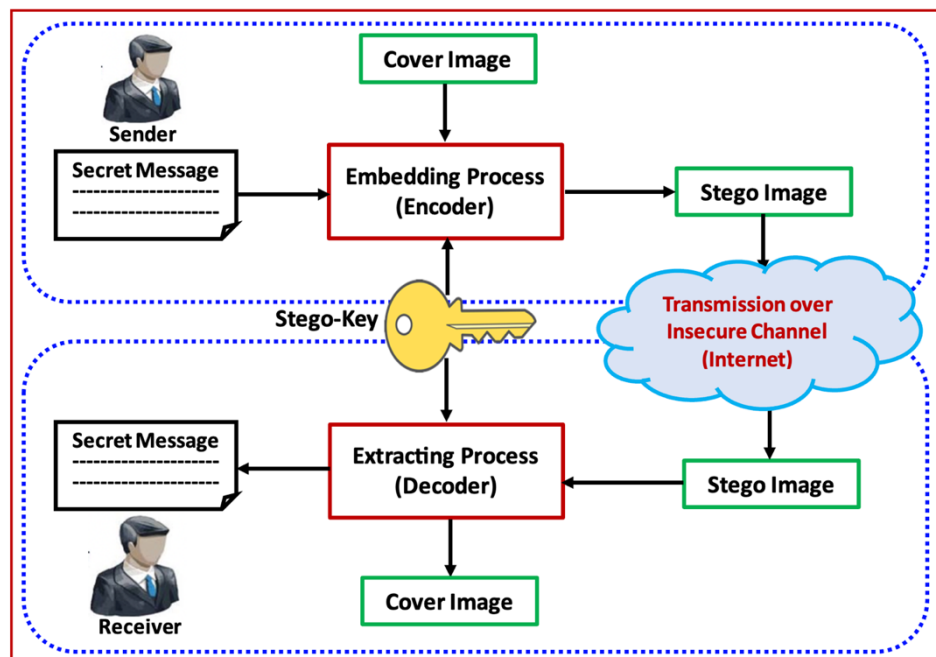


Figure 1.1: Block diagram of a reversible image steganography system

The embedding process takes a cover image and a message as inputs, generating a stego as output. During extraction, the hidden information is extracted, and then the cover is reconstructed from the stego. Reversibility is achieved by preparing overhead information, embedding it along with payload, extracting it at the receiver, and using it to restore the original cover. A comprehensive review of the current advancements in reversible data hiding methods is presented in [3].

## 1.2. Performance Metrics for Steganography

The important parameters for performance evaluation of a steganographic system are payload capacity, imperceptibility, undetectability, and security. These parameters are described as follows:

### 1.2.1. Payload Capacity

Payload capacity, measured in bits, is number of bits that can be hidden in an image. It depends on various parameters including the image size, format, and the steganographic technique applied. Bigger and coloured images have higher capacity as compared to smaller and gray images. Different techniques have different capacity. Techniques used for secret communication have higher capacity than techniques used for watermarking and fingerprinting. The theoretical capacity is expressed in bits per pixel (*BPP*). The maximum number of bits that can be concealed, depends upon the technique used for hiding, and it is represented by  $L_{max}$ . After computing  $L_{max}$ , *BPP* is computed by using the following equation, equation (1.1)

$$BPP = \frac{L_{max}}{M \times N} \quad (1.1)$$

Here  $M \times N$  is the cover image size. To achieve higher capacity, a higher value of *BPP* is required. In LSB based technique, since LSB of every pixel is replaced with a data bit if enough bits are available for hiding. Therefore, payload capacity is 1 *BPP*, which is 12.5% of the cover image size.

### 1.2.2. Imperceptibility

Imperceptibility is the absence of visual distortions into stego image created after embedding. It is the most important requirement for a practical steganographic technique. It is computed by measuring the distortion introduced into image during

data embedding. The distortion should be as low as possible. If the distortion is low imperceptibility is high. A steganographic algorithm with high imperceptibility is preferred over an algorithm with low imperceptibility. Methods that allow more data to be hidden introduce more distortion to the image, while those preserving image quality support lower data capacity.

There are various methods for measurement of distortion into stego images with reference to cover. Commonly used methods are Mean Square Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structured Similarity Index Measure (SSIM), each providing a different perspective on image quality and degradation.

#### ▪ Mean Square Error

It is a well-known metric to quantify the distortion introduced into the stego during embedding process. The difference between corresponding intensities of stego and cover image is known as error. The MSE is computed by the equation (1.2):

$$MSE = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (S(i,j) - C(i,j))^2 \quad (1.2)$$

Where  $S(i,j)$  is the intensity of a pixel at index  $(i,j)$  of the stego image  $S$ , and  $C(i,j)$  is the intensity of a pixel at index  $(i,j)$  of the cover  $C$  where  $0 \leq i \leq M$  and  $0 \leq j \leq N$ . The low value of MSE means that the distortions introduced by the steganographic algorithm during data hiding are minimal for detection by human visual system.

#### ▪ Peak Signal-to-Noise Ratio

PSNR is an important metric to measure the quality of a stego image. It represents the ratio of the maximum possible intensity (signal power) and MSE between two images. Due to the wide dynamic range of image signals, PSNR is measured in decibels ( $dB$ ) using a logarithmic scale. It is computed using the equation (1.3):

$$PSNR = 10 \log_{10} \left( \frac{C_{max}^2}{MSE} \right) \quad (1.3)$$

Where  $C_{max}^2$  is the maximum possible intensity in the cover. The largest possible value of  $C_{max}^2$  for an 8-bit image is 255 and for double-precision floating-point

representation of an image is 1. A higher PSNR is desired for a practical steganographic technique. According to different studies [37, 38, 39], the techniques introducing PSNR below 30dB are poor because the distortion is visible. The techniques with PSNR value between 30dB to 40dB are better and with PSNR value above 40dB are excellent.

#### ▪ Structured Similarity Index Measure

It is used to measure the similarity between cover and the corresponding stego image. SSIM assesses image quality based on three key features: luminance, contrast, and structure. The formula for SSIM for images is computed by equation (1.4) as below:

$$SSIM(x, y) = [L(x, y)]^\alpha \cdot [C(x, y)]^\beta \cdot [S(x, y)]^\gamma \quad (1.4)$$

Here  $x$  and  $y$  are the cover and stego images;  $L(x, y)$ ,  $C(x, y)$ , and  $S(x, y)$  are the comparison functions for luminance, contrast and structure respectively. These are computed by using the equation (1.5):

$$\begin{aligned} L(x, y) &= \frac{2\mu_x\mu_y + C1}{\mu_x^2 + \mu_y^2 + C1} \\ C(x, y) &= \frac{2\sigma_x\sigma_y + C2}{\sigma_x^2 + \sigma_y^2 + C2} \\ S(x, y) &= \frac{\sigma_{xy} + C3}{\sigma_x\sigma_y + C3} \end{aligned} \quad (1.5)$$

Here  $\mu_x$  and  $\mu_y$  are the average intensities for images  $x$  and  $y$  respectively;  $\sigma_x$ , and  $\sigma_y$  are standard deviations of intensities for these images.  $\sigma_{xy}$  is the cross-covariance for images  $x$  and  $y$ . For default values of exponents:  $\alpha = 1$ ,  $\beta = 1$  and  $\gamma = 1$ , and default value of  $C3 = 0$ , the formula is given by equation (1.6):

$$SSIM = \frac{(2\mu_x\mu_y + C1)(2\sigma_{xy} + C2)}{(\mu_x^2 + \mu_y^2 + C1)(\sigma_x^2 + \sigma_y^2 + C2)} \quad (1.6)$$

$C1$  and  $C2$  are two stabilizing parameters,  $C1 = (k_1 l)^2$ ;  $C2 = (k_2 l)^2$ ;  $k_1 = 0.01$  and  $k_2 = 0.03$ .  $l$  is the range of intensities ( $2^{BPP} - 1$ ). SSIM values close to unity means that the hiding method is secure against visual analysis.



### **1.2.3. Undetectability**

Undetectability is the ability of a stego-system to prevent the detection of hidden data by unauthorized parties through statistical analysis, machine-learning and deep-learning-based steganalysis. A steganographic method is considered undetectable if the modifications introduced to the cover image do not produce distinguishable patterns or anomalies that can be detected by any method of analysis. To ensure steganography remains undetected, it is crucial to keep the original cover medium confidential. If the cover image is publicly accessible, any direct comparison between cover and stego can easily reveal the presence of hidden information and further analysis may expose the steganographic method used.

### **1.2.4. Security**

Security in cryptography is based on Kerckhoff's principle, introduced by Auguste Kerckhoff in the 19th century. This principle asserts that the strength of a cryptographic system should depend solely on secrecy of the key, not on confidentiality of the algorithm or its implementation, as these can be discovered, analyzed, or reverse-engineered by adversaries. Security of a cryptosystem may be defined as amount of computing power needed to break the system i.e. to extract the secret key. For high security systems this computing power is high. The same law applies to steganographic techniques, where stego key is needed to extract the data.

## **1.3. Literature Review on Reversible Image Steganography**

The concept of reversibility was introduced by J. M. Barton in 1997 [4]. Since then, various reversible image steganographic techniques have been proposed, including those based on LSB compression [5, 6, 7], histogram shifting [8, 9, 10, 11, 12, 13, 14], image interpolation [15, 16, 17, 18, 19, 20], and Difference Expansion (DE) [21, 22, 23, 24, 25, 26, 27]. Each of these foundational methods has been further extended and refined through various enhancements to improve embedding capacity, imperceptibility, and reversibility. Each of these methods is based on distinct strategies to create embedding space while preserving reversibility.

In the compression of LSB based methods, before replacing LSBs of pixels with secret message bits, whole LSB plane is recorded and compressed using a lossless

compression algorithm. The compressed data serves as a location map and is appended to the message, forming the total payload. This payload is then embedded into cover by modifying LSBs of randomly chosen pixels. At receiver, the embedded payload is extracted from the stego, and then the location map is separated from the secret message. The location map is subsequently decompressed, and original cover is then perfectly restored by replacing LSBs of the stego image with decompressed original LSB plane.

Histogram shifting based method was introduced by Ni et al. in 2006 [8]. This method identifies peak and zero points in the intensity histogram of an image and shifts pixel intensities and creates a bin or space for embedding at peak point. The details of shifting of zero points are recorded for subsequent use for lossless recovery of cover file. Both data and overhead are embedded into the cover.

To improve the capacity, several variants of basic scheme are presented in [9, 10, 11, 12, 13]. These techniques divide the direct image pixels into image blocks and then apply histogram shifting on each block separately for data hiding. C. C. Chang et al. [14] further enhanced the capacity by using the first-order derivative of pixel intensities, with detection of higher peaks for data embedding.

Several reversible techniques based on image interpolation are reported in [15, 16, 17, 18, 19, 20]. The purpose of interpolation is to enlarge the cover before data embedding and divide pixels of the enlarged image into pivot and non-pivot pixels. The data is embedded into non-pivot pixels only. The reconstruction of the original image is achieved by using the pivot pixels only.

Difference expansion based technique was introduced in 2003 by Jun Tian [21]. It used the difference between pixel pairs for data embedding. The difference between adjacent pixel intensities are computed, and specific pairs with expandable differences are chosen for embedding. The average of each pixel pair is maintained during this process to ensure reversibility. The image is divided into pixel pairs, and the overhead or location map is created to record the positions of expandable pairs. This overhead is embedded along with the secret payload. During extraction, the overhead is retrieved

and used to accurately reconstruct the original image. Various DE based methods have been developed since its introduction [22, 23, 24, 25, 26, 27].

#### **1.4. Histogram Shifting Based Reversible Image Steganography**

It is a widely used approach in reversible data hiding due to its simplicity and efficiency. Prior to embedding, the image histogram is scanned to identify the peak (most frequent intensity value) and zero or minimum (least frequent intensity value) points. The pixel values falling between these two points are moved by one step toward the zero point, creating an empty slot at the peak point. This empty space is then used for data embedding.

Before moving the histogram towards left or right, the pixel coordinates having zero or minimum point are recorded as bookkeeping information, required during cover image extraction. In the following step, both the bookkeeping data and user payload are inserted into the histogram at peak point and vacant space nearby peak point. The payload capacity of this method depends on frequency of the peak and zero points in the cover histogram.

##### **1.4.1. Embedding Algorithm**

1. Scan the cover image  $I_c$  and construct its intensity histogram  $H(X)$ , where  $X \in [0, 255]$ . Identify peak and zero points in the histogram. The peak point,  $P \in [0, 255]$ , is the intensity with highest frequency,  $H(P)$ . The zero point,  $Z \in [0, 255]$ , is the intensity with lowest frequency,  $H(Z)$ .
2. The cover image is scanned sequentially to identify and record the indices of pixels with zero intensity value  $Z$ . These indices are stored in a location map  $L$ , which is then converted into binary for embedding along with payload.
3. Read the data to be embedded from a file and convert it into binary. Then, append the location map to the binary data to form the total payload for embedding.
4. Shift the histogram between zero and peak point either to the left or to the right, based on the relation between intensity values at peak and zero points, as follows:

- If  $(P < Z)$ :  
Shift the histogram  $H(X)$ , where  $X \in [P + 1, Z - 1]$  one step to the right, creating an empty slot at index  $P + 1$ , used for embedding. To prevent overflow (which occurs when  $Z = 255$ ), the shifting range is restricted to  $Z - 1$ .
  - If  $(P > Z)$ :  
Shift the histogram  $H(X)$ , where  $X \in [Z + 1, P]$  one step to the left and create an empty slot at index  $P$  for embedding. To prevent underflow (which occurs when  $Z = 0$ ), the shifting range is restricted to  $Z + 1$ .
5. Scan the image again to embed the payload. For each pixel intensity, follow these steps:
- If  $(P < Z)$ :  
For every pixel with intensity  $P$ , if the data bit is '1', increment the pixel intensity by 1.
  - If  $(P > Z)$ :  
For every pixel with intensity  $P - 1$ , if data bit is '1', then increment the pixel intensity by 1.
6. The image generated after embedding is referred to as the stego or marked image, denoted by  $I_s$ . After completion of embedding, the peak point in the histogram typically vanishes, as its intensity values are modified to hide the data.

The flowchart for the embedding process is shown by the Figure 1.2.

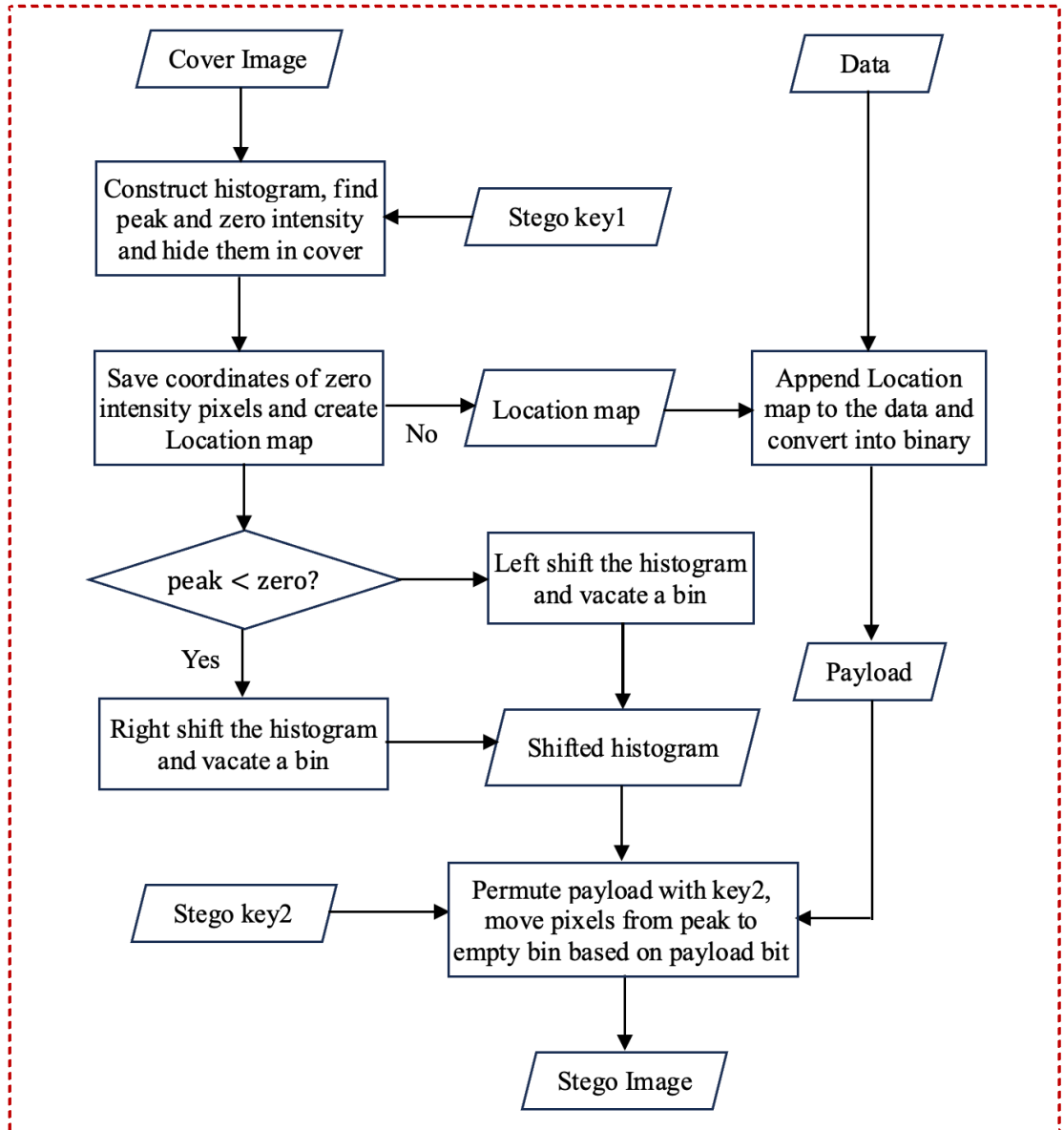


Figure 1.2: Flowchart of histogram-based embedding algorithm

#### 1.4.2. Extraction Algorithm

1. The peak and zero point are required for extraction of the data. These values may be embedded in the image using a stego key. The stego key should be transmitted to the intended recipient through a secure and trusted communication channel.
2. Scan the stego image  $I_s$  and perform following operations to extract total payload:
  - If  $(P < Z)$ :  
If the pixel intensity is  $P + 1$  then extract a bit '1' and if the pixel intensity is  $P$  then extract a bit '0'. Otherwise do nothing.

- If ( $P > Z$ ): If the pixel intensity is  $P$  then extract a bit '1' and if the pixel intensity is  $P - 1$  then extract a bit '0'. Otherwise do nothing.
3. Split the total payload into hidden data (pure payload) and the location map,  $L$ .
  4. Scan the stego image and perform the following operations to reverse the histogram shift:
    - If ( $P < Z$ ): Shift the histogram  $H(X)$  one step to the left, where  $X \in [P + 1, Z]$ .
    - If ( $P > Z$ ): Shift the histogram  $H(X)$  one step to the right, where  $X \in [Z, P - 1]$ .
  5. Replace the pixel intensities at the indices specified in the location map  $L$  with  $Z$  to recover the original image. The flowchart for extraction is shown by Figure 1.3.

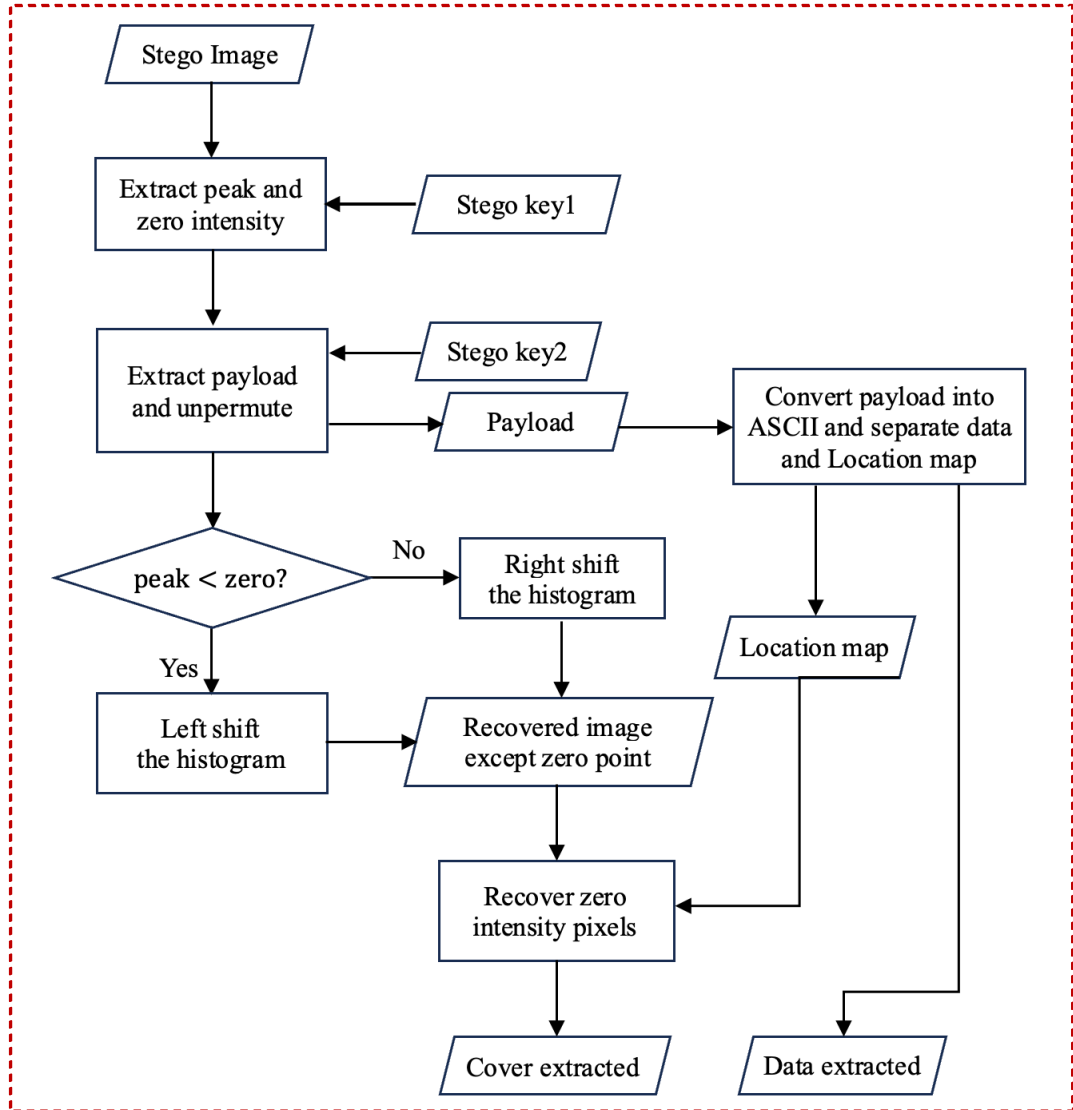


Figure 1.3: Flowchart of histogram-based extraction algorithm

### 1.4.3. Lower Bound on PSNR

The theoretical minimum PSNR value between the stego image produced by this method and the original image exceeds 48 dB. During the embedding process, pixels with grayscale values lying between the minimum and maximum points are altered by either increasing or decreasing their values by 1. Consequently, in the worst-case scenario, all pixel intensities may be adjusted by  $\pm 1$  during embedding. This implies that the error term  $(S(i, j) - C(i, j))$  in MSE formula (equation 1.2) is at most 1 for each pixel. Therefore, MSE is at most 1, indicating minimal distortion introduced into the original image. By substituting  $MSE = 1$ , and  $C_{max} = 255$ , as the largest possible intensity value into the PSNR formula (equation 1.3), the lower bound on PSNR can be computed as below:

$$\begin{aligned} PSNR &= 10 \log_{10} \left( \frac{C_{max}^2}{MSE} \right) \\ PSNR &= 10 \log_{10} (255 \times 255) \\ &= 10 * 4.8130 \\ &= 48 \text{ dB} \end{aligned}$$

This lower bound is validated through experimental results.

### 1.4.4. Computational Complexity

The algorithm maintains a relatively low computational load due to its operation entirely in the spatial domain. The main tasks are generating the histogram, identifying the zero and peak points, and adjusting intensities by  $\pm 1$  only. As a result, the algorithm demonstrates a short execution time. The image needs to be scanned three times during the embedding/extraction process, resulting in a computational complexity of  $O(3MN)$  for an image of size  $M \times N$ .

## 1.5. Image Interpolation Based Reversible Image Steganography

Interpolation is a technique used to predict missing or intermediate values at unknown points based on known data points. Image interpolation is an upsampling technique commonly used to convert low-resolution images into higher-resolution versions by estimating pixel intensities at new locations. Conceptually, the process involves overlaying a finer grid onto the original image – one with reduced pixel spacing – and

assigning intensity to the new grid points using the surrounding pixels intensities. The interpolated grid is then expanded to the desired dimensions, resulting in a resampled image with enhanced resolution.

Traditionally, three primary methods are used to assign intensities to interpolated pixels: nearest-neighbour, bilinear, and bicubic interpolation. Each of these methods represent a trade-off between computational efficiency and visual quality. Nearest-neighbour assigns the intensity of the nearest original pixel to the newly created pixel. Although it is computationally efficient, it often introduces visual artifacts such as jagged edges and distortion. Bilinear interpolation improves visual quality by considering four nearest neighbours to estimate new pixel intensity, striking a balance between computational complexity and image clarity. Bicubic interpolation further refines the result by incorporating sixteen surrounding pixels, yielding smoother transitions and improved preservation of edges and fine details.

In the context of reversible steganography, more sophisticated interpolation approaches have been explored, such as Neighbour Mean Interpolation [15], Interpolation by Neighbouring Pixels [16], and Left Vertex Interpolation [17].

### 1.5.1. Neighbour Mean Interpolation

The Neighbour Mean Interpolation (NMI) estimates the intensity of unassigned pixels by calculating the mean of a select set of neighbouring pixel values. Similar in concept to bilinear interpolation, NMI offers advantages such as reduced blurring and improved image resolution. A key feature of this method is its low computational requirement, which varies based on how many neighbouring pixels are involved in the process – more neighbours enhances accuracy but also increases processing overhead.

The scaling-up coefficient is a factor by which the image needs to be enlarged. The generalized form of the NMI method for a given scaling factor  $K$  is presented in equation (1.7). Here,  $P(i, j)$  denotes intensity at  $(i, j)$  in original image  $P$ , and  $P'(i, j)$  represents the intensity at corresponding pixel in the interpolated image  $P'$ . If the original image has dimensions  $M \times N$  then the interpolated image will have dimensions  $(K(M - 1) + 1) \times (K(N - 1) + 1)$ . The original image pixels are uniformly mapped onto the larger grid of the interpolated image at position  $(i, j)$  where



both  $i$  and  $j$  are integer multiples of  $K$ . These pixels are known as pivot pixels. Remaining pixels in the interpolated image are computed using these pivot pixels.

$$P'(i,j) = \begin{cases} P(i/K, j/K) & \text{if } i \% K == 0 \text{ and } j \% K == 0, \\ (P(i/K, j/K) + P(i/K, j/K + 1))/2, & \text{if } i \% K == 0 \text{ and } j \% K \neq 0, \\ (P(i/K, j/K) + P(i/K + 1, j/K))/2, & \text{if } i \% K \neq 0 \text{ and } j \% K == 0, \\ (P'(i-1, j-1) + P'(i-1, j) + P'(i, j-1))/3, & \text{otherwise} \end{cases} \quad (1.7)$$

Figure 1.4 illustrates the method using a scaling factor of 2. In this example, some of the interpolated image pixels such as  $P'(0,0)$ ,  $P'(0,2)$ ,  $P'(2,0)$ , and  $P'(2,2)$  are directly taken from the original image pixels  $P(0,0)$ ,  $P(0,1)$ ,  $P(1,0)$ , and  $P(1,1)$  respectively. The computations for all the interpolated pixels are presented by the equations in the figure.

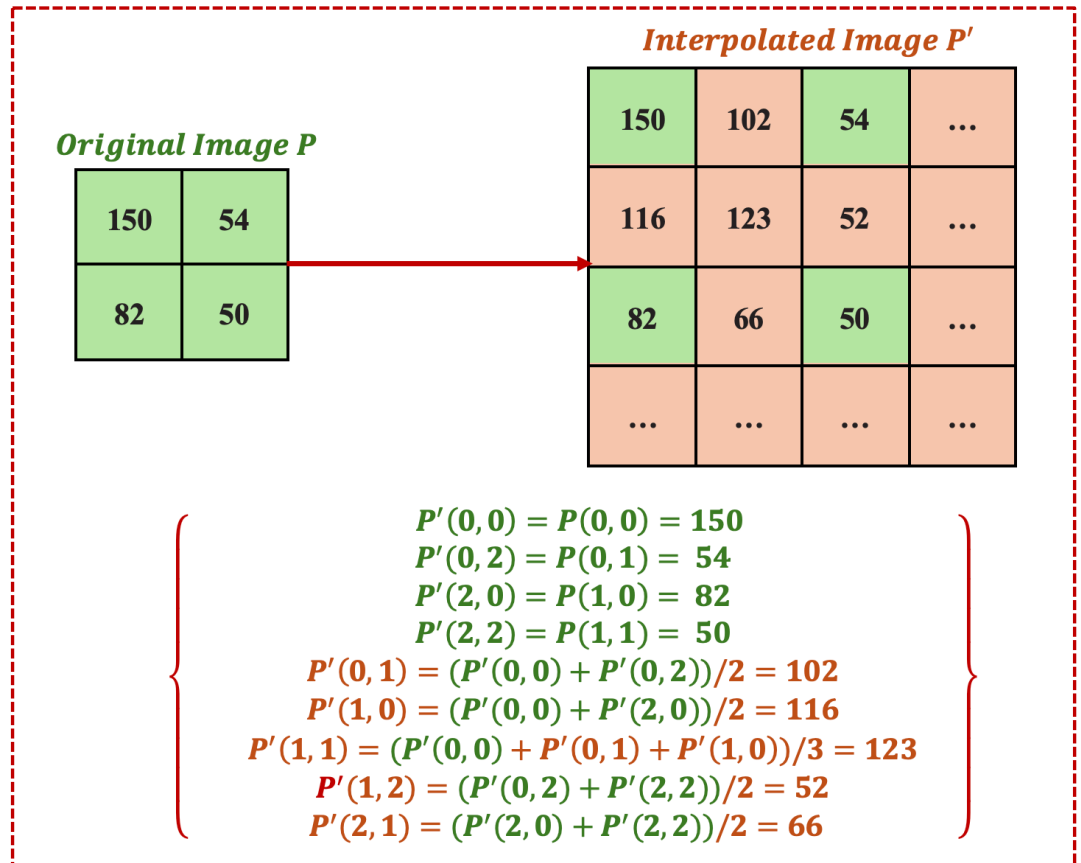


Figure 1.4: Neighbour mean interpolation with  $K = 2$

### 1.5.2. Interpolation by Neighbouring Pixels

Interpolation by Neighbouring Pixels is similar to NMI method. It uses the neighbouring pixels to compute the mean and assign it to the newly inserted pixels.

The generalized form of the INP method with a scaling factor  $K$  is given in equation (1.8). Here  $P(i, j)$  is the intensity at position  $(i, j)$  in the original image  $P$ , and  $P'(i, j)$  is the intensity at pixel  $(i, j)$  in the interpolated image  $P'$ . If the original image has dimensions  $M \times N$  then the interpolated image will have dimensions  $(K(M - 1) + 1) \times (K(N - 1) + 1)$ . The original pixels are uniformly distributed in the interpolated image at positions  $(i, j)$  where  $i$  and  $j$  are integer multiples of  $K$ .

$$P'(i, j) = \begin{cases} P(i/K, j/K) & \text{if } i \% K == 0 \text{ and } j \% K == 0, \\ (3 * P(i/K, j/K) + P(i/K, j/K + 1))/4, & \text{if } i \% K == 0 \text{ and } j \% K \neq 0, \\ (3 * P(i/K, j/K) + P(i/K + 1, j/K))/4 & \text{if } i \% K \neq 0 \text{ and } j \% K == 0, \\ (P'(i - 1, j) + P'(i, j - 1))/2, & \text{otherwise} \end{cases} \quad (1.8)$$

Figure 1.5 illustrates an example with a scaling factor of 2, where the green pixels of the interpolated image  $P'$  correspond to the original pixels  $P(0,0)$ ,  $P(0,1)$ ,  $P(1,0)$ , and  $P(1,1)$  in the input image  $P$ . These pixels serve as reference point — known as pivot pixels — for computing the intensity of newly inserted pixels.

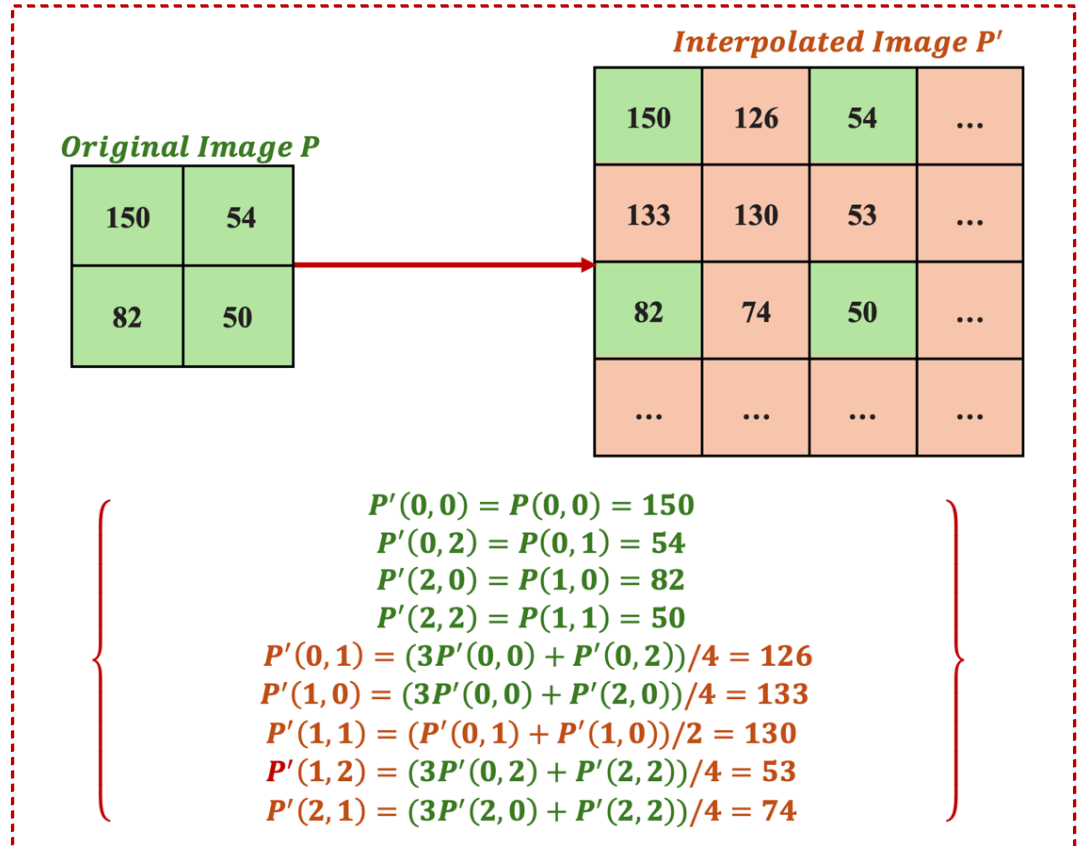


Figure 1.5: Interpolation by neighbouring pixels with  $K = 2$

### 1.5.3. Left Vertex Interpolation

Left Vertex Interpolation (LVI) replicates original image pixels into the interpolated image. If the scaling factor is  $K$ , and original image dimension is  $M \times N$ , then the size of the interpolated image becomes  $KM \times KN$ , where all the original image pixels are replicated  $K^2$  times. The generalized formula for replication is given by equation (1.9):

$$P'(i * K + l, j * K + m) = P(i, j) \quad 0 \leq l, m < K; \quad 0 \leq i < M; \quad 0 \leq j < N \quad (1.9)$$

The interpolated image can be viewed as consisting of  $K \times K$  blocks, forming a total of  $M * N$  such blocks across the image. All pixels in a block share the same intensity value. As the number of blocks in the interpolated image matches the number of pixels in the original image, each original pixel is replicated in one block in the enlarged image. For instance, the first pixel of the source image maps to the first block in the interpolated image, the second pixel maps to the second block, and so on.

A schematic representation of LVI for  $K = 2$  is shown in Figure 1.6, where four pixels from the original image  $P$  are mapped to four distinct blocks in the interpolated image  $P'$ , with the correspondence indicated by colour coding.

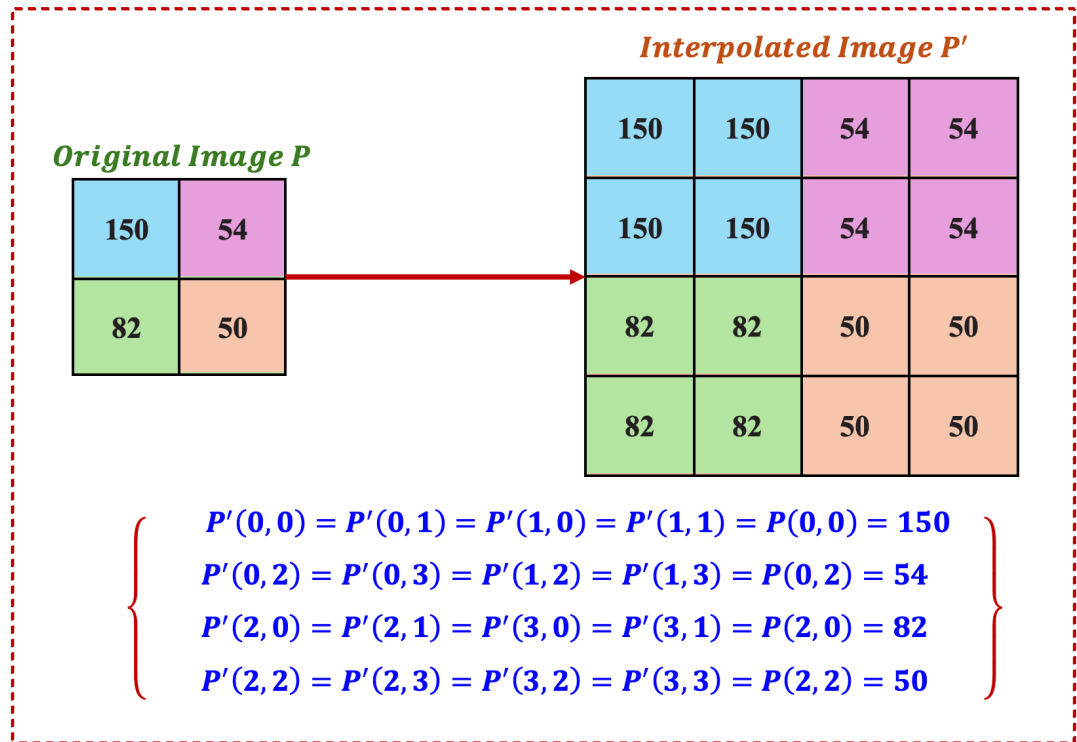


Figure 1.6: Left Vertex Interpolation with  $K = 2$

#### 1.5.4. Data Embedding and Extraction

The flowchart for embedding and extraction of data is depicted by the Figure 1.7. The process begins with enlarging the cover image through interpolation. After interpolation, secret information is embedded into the newly generated pixels using a steganographic approach, such as the basic LSB method or its enhanced versions. The original image pixels — pivot pixels — are left unmodified to preserve the integrity of the cover. During extraction, once the embedded data has been retrieved from the stego, the cover can be reconstructed using the pivot pixels of the stego version.

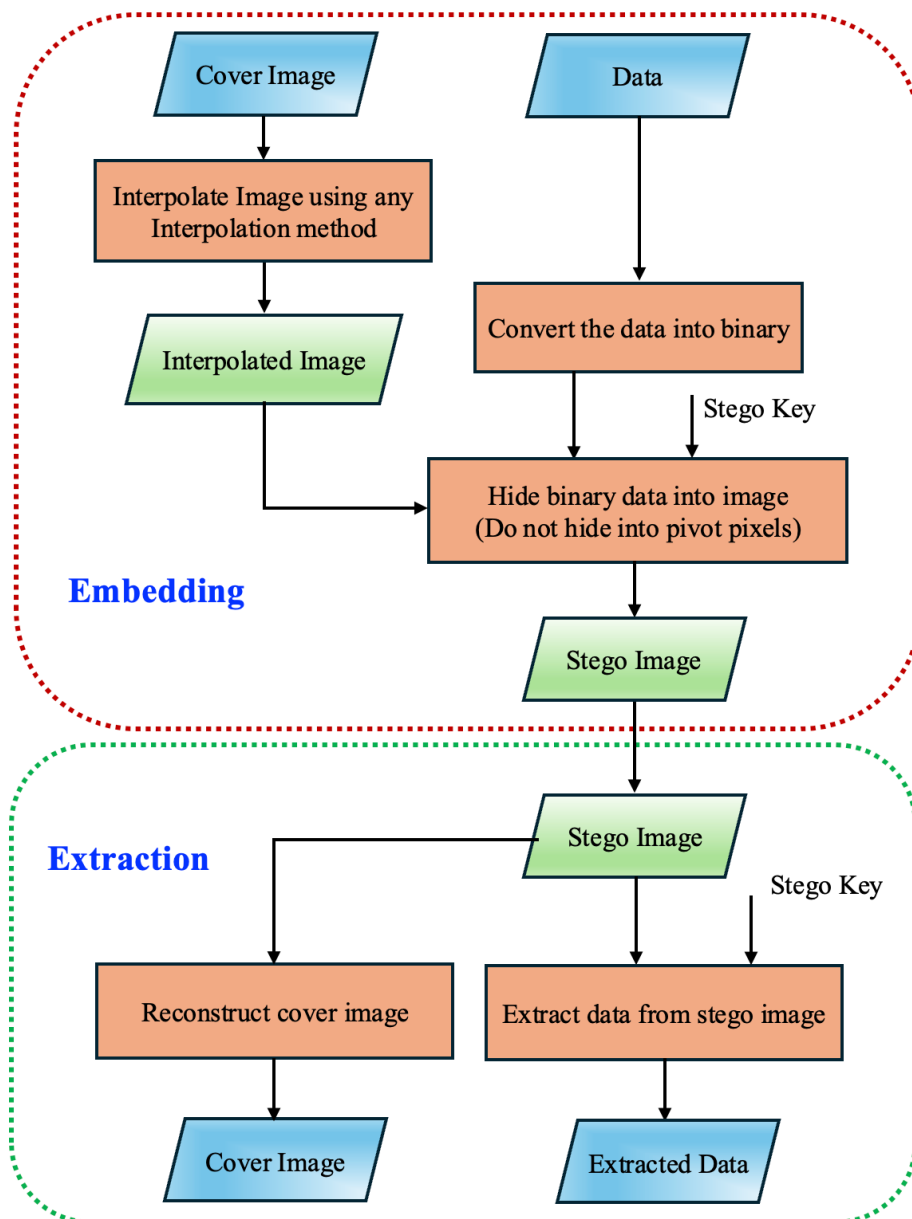


Figure 1.7: Schematic of the interpolation-based reversible data hiding

### 1.5.5. Payload Capacity

This method offers a higher capacity compared to many other methods, because it increases the dimensions of the cover image through interpolation. For example, if a scaling factor of 2 is applied on a  $256 \times 256$  cover image, the interpolated image of size  $512 \times 512$  is obtained. If LSB modification is used for hiding, one bit of data can be hidden in each non-pivot pixel of interpolated image. The payload capacity of the image, denoted by Embedding Capacity (EC), is determined by computing the number of non-pivot pixels in the interpolated image as follows:

$$\begin{aligned} \text{total pixels in interpolated image} &= 512 \times 512 = 262144 \\ \text{total pixels in cover image} &= 256 \times 256 = 65536 \\ \text{pivots in interpolated image} &= \text{number of pixels in original} \\ &\quad \text{pivots in interpolated image} = 65536 \\ \text{nonpivots in interpolated image} &= 262144 - 65536 = 196608 \\ EC = \text{nonpivots in interpolated image} &= 196608 \text{ bits} \\ &= 192Kb \end{aligned}$$

The embedding capacity, measured in *BPP*, is computed as  $196608/262144 = 0.75BPP$ . This is significantly higher than the capacity typically offered by histogram shifting-based techniques. Moreover, the embedded data consists entirely of the secret payload, as no auxiliary information is included during the embedding process.

### 1.5.6. Lower Bound on PSNR

During LSB embedding, each non-pivot pixel may be altered by either +1 or -1. As a result, the MSE introduced in the worst-case scenario does not exceed 0.75 for a scaling factor of 2. Using this MSE along with a maximum pixel intensity  $C_{max} = 255$  in the formula for *PSNR*, the lower bound on *PSNR* is computed as below:

$$\begin{aligned} PSNR &= 10 \log_{10} \left( \frac{C_{max}^2}{MSE} \right) \\ PSNR &= 10 \log_{10} \left( \frac{255 \times 255}{0.75} \right) \\ &= 10 * \log_{10}(86700) \\ &= 10 * 4.9380 \\ &= 49dB \end{aligned}$$

## 1.6. Difference Expansion Based Reversible Image Steganography

This method computes the differences between neighbouring pixels and selects a subset of these differences for data embedding. The selected differences are expanded in such a way that the average intensity of the neighbouring pixels remains unchanged after embedding. The locations of the pixel pairs used for difference expansion are recorded as a binary overhead embedded into the cover along with payload. This map is essential for the receiver to restore the original image.

Mathematical preliminaries required for this method include reversible integer transform, expandable differences, changeable differences, and non-changeable differences. Specific conditions are imposed to prevent pixel intensity underflow and overflow, and to determine which pixel pairs are eligible for data hiding.

### 1.6.1. Reversible Integer Transform

It works by grouping the cover pixels into pairs and decomposing them into two parts: a low-pass component  $L$ , which stores the integer averages  $l$ , and a high-pass component  $H$ , which records the differences  $h$ . The integer average and the difference for an pixel pair  $(x, y)$ ,  $x, y \in \mathbb{Z}$ ,  $0 \leq x, y \leq 255$ , are computed by equation (1.10):

$$l = \left\lfloor \frac{x + y}{2} \right\rfloor, \quad h = x - y \quad (1.10)$$

This transformation is reversible, meaning the original grayscale values  $x$  and  $y$  can be accurately reconstructed from the average  $l$  and the difference  $h$  using the inverse operation outlined in equation (1.11):

$$x = l + \left\lfloor \frac{h + 1}{2} \right\rfloor, \quad y = l - \left\lfloor \frac{h}{2} \right\rfloor \quad (1.11)$$

This process is known as the Integer Haar Wavelet, also referred to as the S-transform. The equations (1.10) and (1.11) establish a one-to-one correspondence between pixel pairs  $(x, y)$  and their transformed counterparts  $(l, h)$ . To ensure that the reconstructed values of  $x, y$ , remain in valid grayscale range  $[0, 255]$ , the equation (1.12) must be satisfied:

$$0 \leq l + \left\lfloor \frac{h + 1}{2} \right\rfloor \leq 255, \quad \text{and} \quad 0 \leq l - \left\lfloor \frac{h}{2} \right\rfloor \leq 255 \quad (1.12)$$

Given that both  $l$  and  $h$  are integers, the inequalities can be simplified to the equation (1.13):

$$|h| \leq 2(255 - l), \quad \text{and} \quad h \leq 2l + 1 \quad (1.13)$$

The equation (1.13) can be simplified to the equation (1.14):

$$\begin{cases} |h| \leq 2(255 - l), & \text{if } 128 \leq l \leq 255 \\ |h| \leq 2l + 1, & \text{if } 0 \leq l \leq 127 \end{cases} \quad (1.14)$$

### 1.6.2. Expandable Difference Values

When a difference value  $h$  is expandable, a message bit  $b \in \{0, 1\}$  can be appended to LSB of  $h$ . This process increases the bit-length of the difference value by one, and is thus referred to as difference expansion. During expansion,  $h'$  is computed using the equation (1.15):

$$h' = 2 \times h + b \quad (1.15)$$

To ensure that the reconstructed pixel intensities remain in the valid grayscale range, the expanded difference  $h'$  must meet the constraint specified by the equation (1.16) as below:

$$|h'| \leq \min(2(255 - l), 2l + 1) \quad (1.16)$$

A difference  $h'$  is considered expandable if and only if the following inequality, given by equation (1.17), is satisfied for  $b \in \{0, 1\}$ . This is derived by substituting equation (1.15) into equation (1.16):

$$|2 \times h + b| \leq \min(2(255 - l), 2l + 1) \quad \text{for } b \in \{0, 1\} \quad (1.17)$$

If this condition is satisfied,  $h$  can be expanded to  $h'$ , and the intensities of new pixel pair  $(x', y')$  are computed from  $l$  and  $h'$  using the inverse integer transform as below:

$$x' = l + \left\lfloor \frac{h' + 1}{2} \right\rfloor, \quad y' = l - \left\lfloor \frac{h'}{2} \right\rfloor \quad (1.18)$$

This transformation guarantees that the new intensities remain in the valid grayscale range.

### 1.6.3. Changeable Difference Values

If a difference  $h$  is deemed changeable, a bit  $b \in \{0, 1\}$  can be embedded by substituting the LSB of  $h$  with  $b$ . As a result, a new value of  $h$ , denoted by  $h'$  is computed as follows:

$$h' = 2 \times \left\lfloor \frac{h}{2} \right\rfloor + b \quad (1.19)$$

Since LSB replacement is inherently lossy, the original LSBs of the difference values must be preserved and embedded along with the secret data to enable recovery of original image. To avoid underflow and overflow in the reconstructed pixels, the updated difference  $h'$  must satisfy the condition derived from equation (1.15),

$$|h'| \leq \min(2(255 - l), 2l + 1) \quad (1.20)$$

Substituting the expression for  $h'$  from equation (1.19) into (1.20), the changeability condition for the difference  $h$  under average  $l$ , is obtained as equation (1.21):

$$\left| 2 \times \left\lfloor \frac{h}{2} \right\rfloor + b \right| \leq \min(2(255 - l), 2l + 1) \quad \text{for } b \in \{0, 1\} \quad (1.21)$$

If this condition holds, then  $h$  is changed to  $h'$  and the new intensities of the pixel pair  $(x', y')$  are computed from  $l$  and  $h'$  using the inverse transform defined as below:

$$x' = l + \left\lfloor \frac{h' + 1}{2} \right\rfloor, \quad y' = l - \left\lfloor \frac{h'}{2} \right\rfloor \quad (1.22)$$

All expandable differences are inherently changeable, since meeting the expandability condition automatically ensures that the changeability condition is also fulfilled. A changeable difference continues to be changeable even after its LSB has been modified. On the other hand, an expandable difference might lose its expandability after embedding, but it still retains its changeability. Differences that do not meet the criteria for both the expandability and changeability are classified as non-changeable, meaning no data is embedded in these values.

### 1.6.4. Embedding Algorithm

#### ▪ Pixel Pairing

The original image is divided into pairs of pixels. The pairing can be formed in various ways such as horizontally, vertically, or based on a pattern determined by a secret key.



The pairing can be applied to the whole image or to a portion of it, depending on the embedding scheme and size of data. Figure 1.8 illustrates the horizontal and vertical pixel pairings.

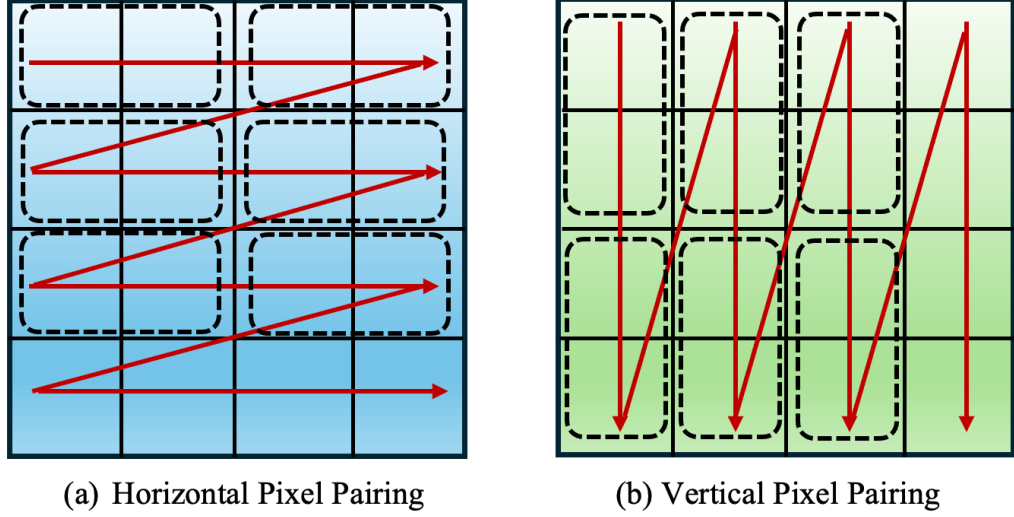


Figure 1.8: Horizontal and vertical pixel pairing

- **Finding Averages and Differences**

After pairing of the cover image pixels, reversible integer transform is applied to these pairs for computing the average  $l$  and difference  $h$  for each pair. The average values are preserved before and after data embedding.

- **Identify Expandable and Changeable Differences**

All the differences  $h$  are arranged in a one-dimension list as  $\{h_1, h_2, h_3, \dots, h_n\}$  and examined for expandability. If the expandability condition is satisfied, it is marked as expandable difference. If it is not satisfied, then the difference is examined for changeability condition. If this condition is met, the difference is marked as changeable. Differences that satisfy neither condition are considered *non-changeable* and are not used for data hiding.

- **Creating Location map**

A binary location map  $\{L = L_1, L_2, L_3, \dots, L_n\}$  is created corresponding to the list of difference  $\{h_1, h_2, h_3, \dots, h_n\}$ . If a difference is expandable, a '1' is placed, otherwise a '0' is stored at that position. This location map is embedded into the modified

differences along with actual payload. It assists the decoder during extraction in identifying expandable differences and then to recover the original values.

- **Storing LSBs of Changeable Differences**

The original LSBs of changeable differences are extracted and lossless compressed and stored in  $C$ . These compressed bits, along with location map  $L$ , and actual payload, are embedded into the difference values. The purpose is to restore the LSBs of original changeable difference values and then to restore original cover image at receiver end.

- **Embedding Data**

The location map  $L$ , compressed LSBs  $C$ , and actual payload  $P$  are combined to make a bitstream  $B$ , for embedding. The bitstream is embedded into the eligible difference values either by appending to expendable difference or by replacing LSBs of changeable difference. One bit is embedded into one changeable or expendable difference. As a result, new difference values are computed, and then new intensities for the corresponding pixel pairs  $(x', y')$  are computing using inverse transform to obtain the stego image.

#### **1.6.5. Extraction Algorithm**

The embedded bitstream is extracted from LSBs of all changeable difference of the stego image and parsed into three components: the location map, the compressed original LSBs, and the pure payload. The location map enables correct identification and restoration of all expanded differences. The compressed original LSBs allow for reconstruction of the changeable differences. Once all the expandable and changeable differences are restored, the original image is reconstructed, ensuring reversibility. The steps are as follows:

- **Pixel Pairing**

The stego image is first grouped into pairs of pixels using the same pattern that was used during embedding. This ensures synchronization between embedding and extraction processes.

- **Computing Averages and Differences**

The reversible integer transform is applied to all pixel pairs, computing the average  $l$  and difference  $h'$ . The differences are used for data extraction. Then averages are used during cover reconstruction.

- **Identifying Changeable Difference Values**

All the differences  $h'$  are arranged in a one-dimension list as  $\{h'_1, h'_2, h'_3, \dots, h'_n\}$  and each is examined for changeability by using the changeability condition. Expandability is not evaluated, since an expandable difference may no longer satisfy the expandability condition after embedding. However, it is guaranteed to remain changeable. In contrast, a changeable difference retains its changeable property after embedding.

- **Collecting LSBs of Changeable Difference**

The embedded bitstream  $B$  is obtained by extracting LSBs of all the Changeable differences. This bitstream contains three components: the location map  $L$ , the Compressed LSBs  $C$  of the original changeable differences, and the actual payload  $P$ . The desired payload is obtained after parsing these components.

- **Restoring the Original Difference**

When a value in the location map is '0', the original difference  $h$  was changeable. In such cases,  $h$  is recovered by substituting the LSB of  $h'$  with the corresponding bit from the decompressed LSBs. If the map contains a '1', this signifies that original difference was expandable, and it is restored using the following equation, equation (1.23):

$$h = \left\lfloor \frac{h'}{2} \right\rfloor \quad (1.23)$$

- **Reconstruction of Cover Image**

The inverse integer transform is applied to the transformed values – average  $l$  and difference  $h$  – to reconstruct the original pixel pairs. As a result, the original cover image is perfectly reconstructed.

## CHAPTER 2

### ANALYSIS OF HISTOGRAM SHIFTING BASED REVERSIBLE IMAGE STEGANOGRAPHY

#### 2.1. Review of Related Work

Histogram shifting based technique was introduced by Ni et al. in 2006 [8]. It identifies peak and zero points in the histogram, known as peak-zero pair. The peak point represents the most frequent intensity, while the zero point represents the least frequent intensity in the image. This pair defines a segment of the histogram that is moved before embedding. The key advantages of this approach are its simplicity, efficiency, low computational cost, and a theoretical lower bound on PSNR. It ensures that the PSNR, measured in decibels ( $dB$ ), of the stego relative to the cover remains above  $48dB$ , preserving visual quality. However, there are areas for further enhancement in the scheme:

- The PSNR remains constant regardless of the payload size. A payload-adaptive approach can be explored, where smaller payloads introduce lesser distortions.
- It is assumed that increasing the number of peak-zero pairs enhances hiding capacity; however, this assumption cannot be generalized for all types of images.
- Two peak-zero pairs in the histogram are identified by selecting zero points first and then determining their corresponding peak points, which may not always be global peaks. An alternative approach, where peak points may be identified first and their corresponding zero points being selected afterward, may improve the capacity for some images.

The techniques presented in [9, 12, 13] hide in image blocks, where the image pixels are divided into blocks, and the embedding algorithm is applied independently to each block. The limitations of these techniques are following:

- The side information required for data and cover extraction, i.e., the peak-zero pairs and overhead, increases with number of image blocks, as each block has its own histogram and corresponding peak-zero pair.

- More number of peak-zero pairs leads to higher modifications in the intensity distribution, making it detectable through histogram analysis.
- Reversibility is not verified in the given scheme [13].

The scheme described in [10] is based on multilayer embedding with varying image block sizes ( $256 \times 256$ ,  $128 \times 128$ ,  $64 \times 64$ ,  $32 \times 32$ ,  $16 \times 16$ , and  $8 \times 8$ ), achieving exceptionally high embedding capacity. However, it is considered insecure for the following reason:

- The scheme does not embed data into peak points, leaving them unaltered while using them for message extraction. This creates a vulnerability, as the peak intensity can be directly retrieved from the stego image, enabling unauthorized message extraction and compromising the security.

The scheme [11] utilizes  $32 \times 32$  image blocks for embedding. For instance, a  $512 \times 512$  image is divided into 256 blocks, each having  $32 \times 32$  pixels. However, the scheme has following inherent flaws:

- It does not provide a substantial increase in hiding capacity across different images.
- It significantly increases the side information, requiring 256 peak-zero pairs.
- The algorithm stores peak intensities at fixed locations, specifically in LSBs of the initial eight pixels of each block. Additionally, it stores an indicator in LSBs of the first block to specify whether a block contains hidden data. However, an adversary can easily reverse this process without requiring additional information. As a result, the algorithm violates Kerckhoff's principle, compromising the security of the system.

In 2008, Fallahpour [28] introduced the Gradient Adjusted Predictor (GAP), a method based on embedding into prediction error histograms. Since then, various predictors have been proposed. For example, Capacity-Distortion Optimization [29] employs genetic algorithms to identify optimal peak and zero bins, while Skewed Histogram Shifting for Reversible Data Hiding [30] leverages extreme predictor pairs to decrease the distortion. Multiple Histogram-based framework [31] utilizes evolutionary algorithms for finding multiple histograms, and Pixel Residual Histogram based-

technique [32] extends Prediction Error Expansion-based schemes. However, these methods have certain drawbacks, as detailed below:

**Compute-Intensive:** Predictors generate a prediction image from the cover by performing floating-point operations on neighbouring pixels for each pixel. The prediction error matrix is then obtained by subtracting predicted image from cover. The genetic algorithms used to identify optimal peak-zero pairs involve multiple iterations of floating-point computations, leading to significant computational overhead. Consequently, these methods demand extensive computations before data hiding and extraction, unlike pixel-domain histogram-based methods, which do not require such prior computations.

**No Theoretical Lower Limit on PSNR:** Accuracy of a predictor impacts both the embedding capacity and the quality of a stego image. These methods do not guarantee a theoretical minimum PSNR value. In some cases, the PSNR falls below  $40dB$ , rendering them unsuitable for practical applications.

**Embedding Traces:** The prediction error histogram of an image typically follows Laplacian distribution, a bell-shaped distribution with peaks at zero for most predictors. After data embedding, this distribution deviates from the Laplacian model and becomes more irregular. As hiding capacity increases, the central peak flattens, making detection easier. An effective steganalysis method for prediction error histogram shifting is discussed in [33]. Although the schemes presented in [34] and [35] attempt to conceal these traces, but they result in reduced capacity and lower PSNR.

## 2.2. Capacity Analysis

The peak point in the histogram has a peak frequency, denoted as  $PF$ , and the zero point has a zero frequency denoted as  $ZF$ . Since hiding occurs at the peak points, the capacity is equal to the peak frequency, representing the maximum embedding capacity. Pixel coordinates with zero intensity are recorded in the location map as overhead. The data to be embedded is pure payload. Total payload comprises the pure payload and the overhead. The size of embedding capacity and total payload, in number of bits, are shown by equation (2.1):

$$\begin{aligned}
& \text{Embedding Capacity} = PF \\
& \text{Total Payload} = \text{Pure Payload} + \text{Overhead}
\end{aligned} \tag{2.1}$$

For successful embedding, payload must not exceed the embedding capacity, as shown by equation (2.2):

$$\begin{aligned}
& \text{Total Payload} \leq \text{Embedding Capacity} \\
& \text{Pure Payload} + \text{Overhead} \leq PF
\end{aligned} \tag{2.2}$$

Overhead size depends on frequency of zero points and on image size ( $M \times N$  pixels). It is given by equation (2.3), where *bitsperindex* is the bits required for a pixel index:

$$\begin{aligned}
& \text{Overhead} = ZF \times \text{bitsperindex} \\
& = ZF \times \log_2(M \times N)
\end{aligned} \tag{2.3}$$

Pure capacity can be computed by equation (2.4). To maximize it, the peak frequency should be as high as possible, and the zero frequency should be as low as possible.

$$\text{Pure capacity} = PF - (ZF \times \log_2(M \times N)) \tag{2.4}$$

### 2.2.1. Analysis for Histogram Partitioning

Histogram partitioning divides an image histogram into non-overlapping blocks before embedding. It is assumed that increasing the number of histogram blocks increases the hiding capacity. However, analysis shows that the improvement is not universal and it depends on the intensity distribution in the image histogram. For instance, the histograms of two  $8 \times 8$  images with 8 gray levels are presented in Figure 2.1 (a) and (b) respectively. Both images have the same capacity for a single block since they share the same peak frequency of 21 and the same zero frequency of 1. Splitting the histograms into two blocks creates two peak-zero pairs, one for each block, with the total capacity being the sum of the individual block capacities. In the figure, the green and blue regions represent the two non-overlapping histogram blocks. The blue block maintains the same peak and zero frequencies of 17 and 1, respectively, in both histograms. The green block has the same peak frequency of 21 in both the histograms, but differs in zero frequency: 3 in Figure 2.1(a) and 2 in Figure 2.1(b). Hence, Figure 2.1(a) shows higher capacity for embedding in single block, whereas Figure 2.1(b) shows higher capacity for embedding in two blocks. The computations for both

histograms for single block and two blocks are presented by Figure 2.2(a) and Figure 2.2(b), respectively.

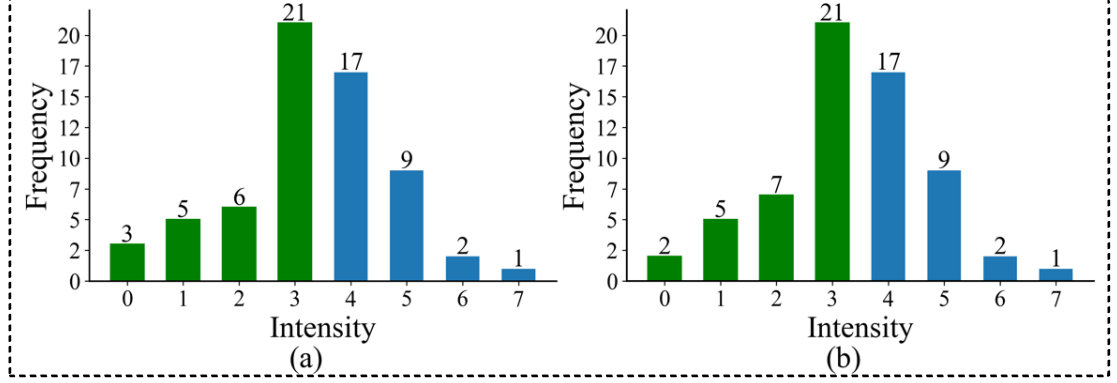


Figure 2.1: Histograms of two  $8 \times 8$  images

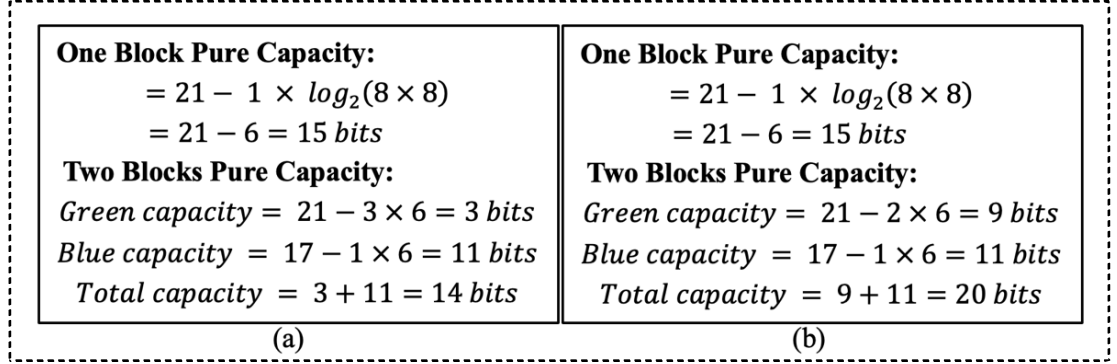


Figure 2.2: Embedding capacity for histograms of Figure 2.1

Figure 2.3(a) and Figure 2.3(b) show an histogram with two (b1, b2) and three non-overlapping blocks (b1, b2, b3), respectively.

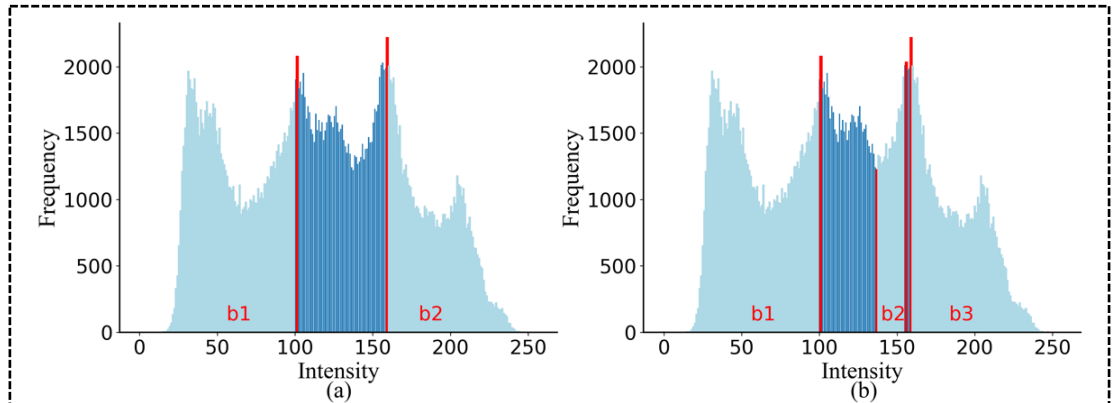


Figure 2.3: Histogram Blocks: (a) Two blocks (b) Three blocks

Experimental results show that identifying two non-overlapping blocks is generally feasible, whereas identifying three or more blocks often results in overlap. Even



without overlap, the capacity may not always increase since each added block has higher overhead due to its zero point having a higher frequency than the previous blocks.

The algorithm for two-block histogram partitioning is presented in Figure 2.4.

<b>Algorithm: Hist_Partition</b>	
<b>Input:</b>	<i>hist</i> [0,255]                      # Image Histogram with graylevels 0 to 255
<b>Output:</b>	<i>P1, Z1, P2, Z2</i> #Peak and Zero Intensities for two pairs
<ol style="list-style-type: none"> <li>1. Find two most frequent intensities in the <i>hist</i> as <i>P1, P2</i> ; where <math>P1 &lt; P2</math></li> <li>2. Divide histogram into three intervals using <i>P1, P2</i> as follows:  <math>[0, P1 - 1]</math>, <math>[P1 + 1, P2 - 1]</math> and <math>[P2 + 1, 255]</math></li> <li>3. Select least frequent intensity in the interval <math>[0, P1 - 1]</math>, denoted by <i>L1</i></li> <li>4. Select two least frequent intensities in the interval <math>[P1 + 1, P2 - 1]</math>, denoted by  <i>R1</i> and <i>L2</i> respectively</li> <li>5. Select least frequent intensity in the interval <math>[P2 + 1, 255]</math>, denoted by <i>R2</i></li> <li>6. Find frequencies for the intensities <i>L1, R1, L2, R2</i> in the histogram <i>hist</i> denoted by  <i>L1_freq, R1_freq, L2_freq, R2_freq</i> respectively, where <math>R1\_freq &lt; L2\_freq</math></li> <li>7. <b>if</b> (<math>L1\_freq &lt; R1\_freq</math>):            <math>Z1 = L1</math>            <b>if</b> (<math>R1\_freq &lt; R2\_freq</math>): <math>Z2 = R1</math>            <b>else</b>: <math>Z2 = R2</math>            <b>else</b>:            <math>Z1 = R1</math>            <b>if</b> (<math>L2\_freq &lt; R2\_freq</math>): <math>Z2 = L2</math>            <b>else</b>: <math>Z2 = R2</math></li> <li>8. <b>Return</b> <i>P1, Z1, P2, Z2</i></li> </ol>	

Figure 2.4: Algorithm for finding two blocks in an image histogram

The algorithm differs slightly from the algorithm used in original method [8], which first selects global zero points and then identifies the corresponding peak points. In contrast, this method first selects the peak points, followed by the zero points.

### 2.2.2. Analysis for Image Partitioning

Image partitioning divides image pixels into non-overlapping blocks prior to embedding. As illustrated in Figure 2.5(a), a  $2 \times 2$  partition splits the image into 4 separate regions, with Figure 2.5(b) depicting corresponding histograms. The analysis of capacity shows that while increasing the number of image blocks may improve hiding capacity for some images, the improvement is often marginal. This is because partitioning the image decreases pixel counts per block, which in turn reduces both the

histogram height and the peak frequency, thereby reducing the space available for embedding.

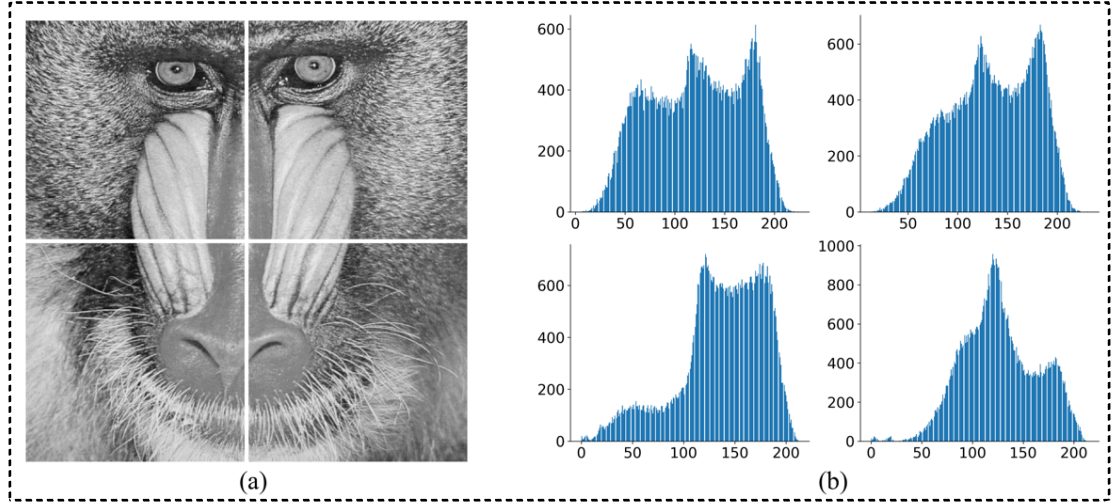


Figure 2.5: (a) Image Blocks (b) Corresponding histograms

### 2.3. Histogram Analysis

A comparison of histograms before and after embedding shows that embedding modifies the distribution of three intensities for each peak-zero pair: the peak intensity, the zero intensity, and the intensity adjacent to the peak. Partitioning an image into four image blocks, each containing two peak-zero pairs, results in a total of eight pairs, altering the distribution of 24 intensities. Further partitioning into 16 blocks yields 32 pairs, which may modify 96 intensities out of 256, a 37.5% change in the histogram. Thus, increasing the number of blocks makes the image more susceptible to detection through histogram analysis. During embedding, peak frequencies decrease, zero frequencies increase, and the frequencies of intensities adjacent to the peak become approximately half of the peak frequency, leading to histogram equalization.

For example, a 21-level grayscale image, shown in Figure 2.6 as cover image, transforms into a 38-level stego image after embedding into  $2 \times 2$  (4 image blocks), and a 42-level stego image after embedding into  $4 \times 4$  (16 image blocks). The original histogram, along with the histograms after embedding into  $2 \times 2$  and  $4 \times 4$  blocks, are shown in the figure

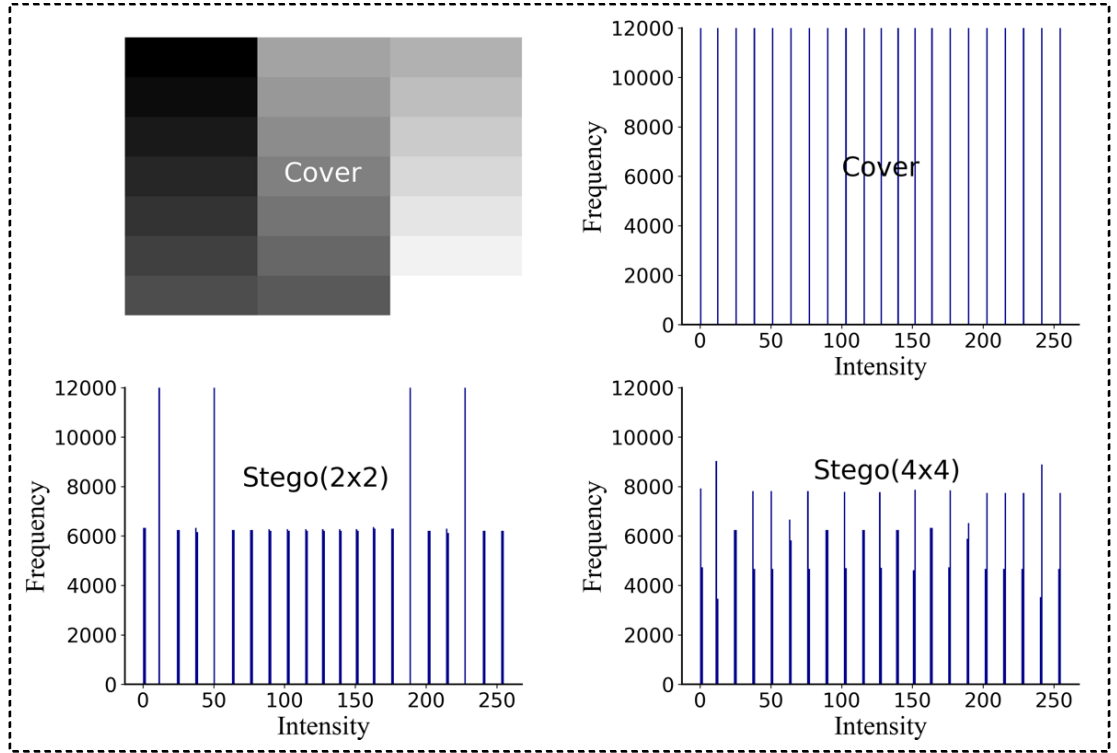


Figure 2.6: Histogram analysis for a 21-level gray-scale image

## 2.4. Experimental Results

Observations are presented for the standard  $512 \times 512$  grayscale images shown in Figure 2.7, sourced from the USC-SIPI image database [36]. This dataset offers images with diverse intensity distributions, textures, and patterns, making it suitable for evaluating data hiding techniques under realistic conditions. The experiments are conducted on the images in lossless formats such as Portable Network Graphics (PNG) and Tagged Image File Format (TIFF).

### 2.4.1. Capacity for Histogram Partitions

Hiding capacity observations on histogram partitioning are presented in Table 2.1. The findings from the table are as follows:

- Splitting histogram into two blocks increases the hiding capacity over single-block use for all images except for the Moon.
- Splitting into three blocks often results in violations of the non-overlapping constraints. In cases where three non-overlapping blocks are found, the overhead of third block exceeds the peak frequency, thereby negating the capacity gains (e.g., Barbara). Only Cat image shows a slight capacity improvement.

- The capacity does not increase consistently with number of blocks, and it is found that the optimal histogram block size is 2.



Figure 2.7: Standard test images for evaluation

#### 2.4.2. Capacity for Image Partitions

Table 2.2 presents the observations for varying image block sizes. The block size  $n \times n$  denotes the partitioning of images into  $n$  blocks along both the horizontal and vertical axes. It is found that hiding capacity does not consistently increase with number of blocks. Furthermore, the optimal block size for maximum hiding capacity is image-dependent. For the analyzed images, the optimal block sizes were found to be  $4 \times 4$ ,  $8 \times 8$ , or  $16 \times 16$ , with the corresponding maximum hiding capacities highlighted in bold in the table.

Table 2.1: Hiding capacity for varying histogram blocks

Image Name	Histogram Partitions		
	1 block	2 blocks	3 blocks
Barbara	2277	4456	Overhead
Pepper	2686	5314	Overlapping
Baboon	2699	5245	Overlapping
Lena	3058	6129	Overlapping
Sailboat	3766	7489	Overlapping
Tiffany	3808	7343	Overlapping
Fishing boat	5733	11322	Overlapping
Jet	8249	15943	Overlapping
Airplane	30393	59882	Overlapping
Sunset	68105	68105	Overlapping
Cat	96219	100963	105622
Moon	122908	122908	Overlapping

Table 2.2: Hiding capacity for varying image blocks

Image Name	$2 \times 2$	$4 \times 4$	$8 \times 8$	$16 \times 16$	$32 \times 32$	$64 \times 64$
Barbara	3577	5961	7946	<b>8069</b>	5532	1988
Pepper	3879	5671	<b>7517</b>	5716	430	0
Baboon	2826	<b>3432</b>	3192	1107	5	0
Lena	5725	8568	12087	<b>12292</b>	7718	2459
Sailboat	4466	7436	<b>8331</b>	5970	1227	0
Tiffany	6681	9273	<b>12052</b>	10890	3160	218
Fishing boat	5931	7407	<b>8410</b>	5860	621	0
Jet	12501	14991	<b>18188</b>	17848	9402	409
Airplane	50740	74731	88315	<b>90637</b>	75278	6566

### 2.4.3. Results for Histogram Analysis

Image partitioning divides the image pixels into multiple blocks, creating more peak-zero pairs than histogram partitioning and results in higher modifications to the histogram during embedding. Figure 2.8 presents a cover image and its histogram, partitioning of the image into  $2 \times 2$  (four blocks) and corresponding stego image histogram, and partitioning of the image into  $4 \times 4$  (16 blocks) and corresponding stego image histogram. The modifications in the stego histograms are visible in the figure, highlighting the impact of embedding on intensity distributions.

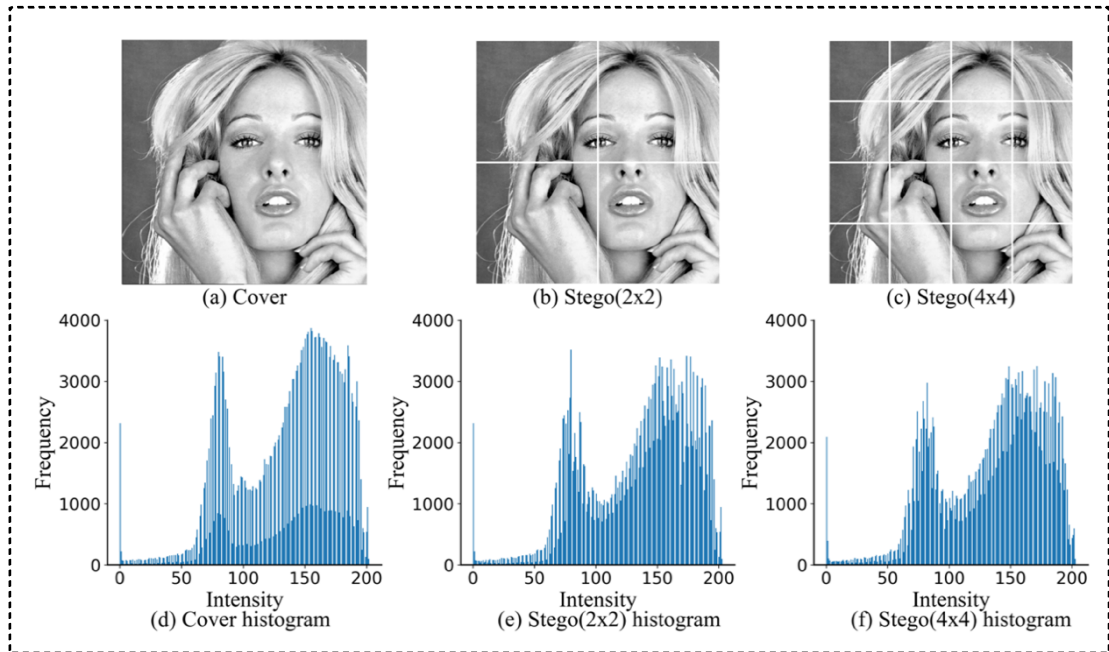


Figure 2.8: Histogram variations after embedding in image blocks

Figure 2.9 illustrates results for three images from the SIPI dataset. The first column shows the original images, the second column displays their corresponding histograms, and the third column depicts histograms after embedding into  $4 \times 4$  image blocks. The results show that embedding into image blocks leads to localized histogram flattening, thereby increasing vulnerability to histogram-based steganalysis.

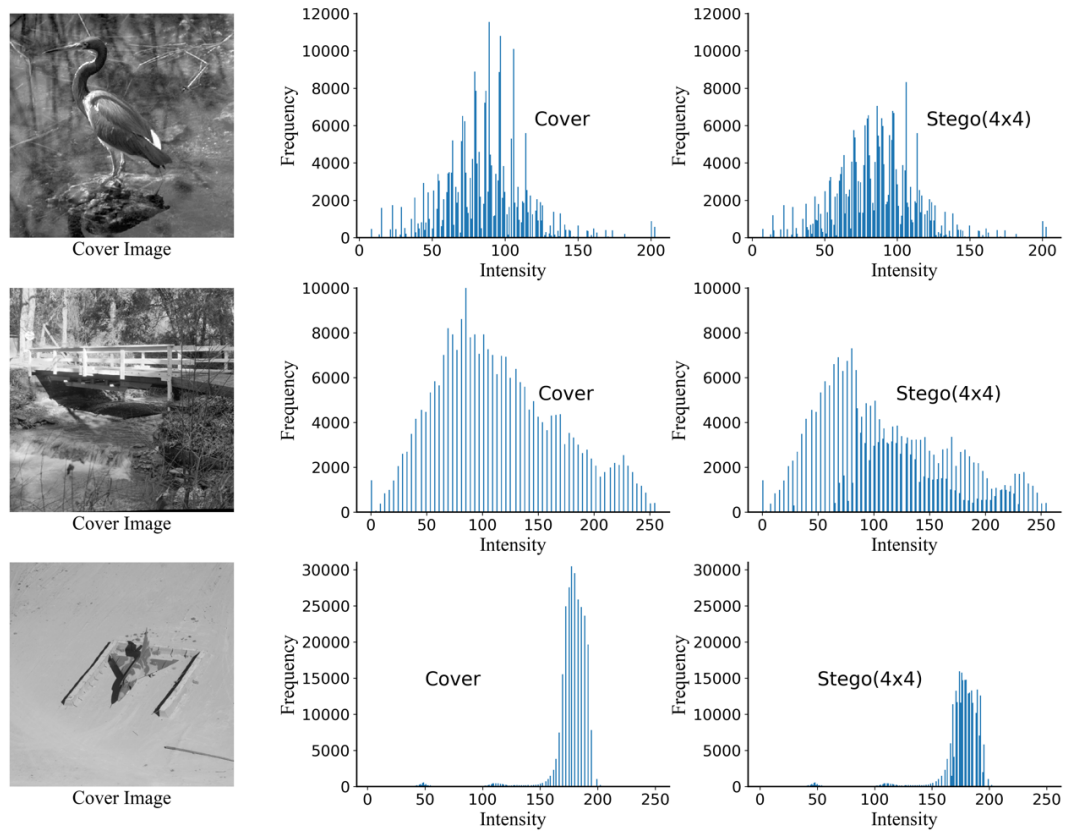


Figure 2.9: Impact of embedding in  $4 \times 4$  image blocks on histograms

## 2.5. Conclusion

This study analyzed reversible data hiding using histogram shifting for both image partitioning and histogram partitioning. The findings are that the optimal block size for image partitioning is image-dependent and cannot be generalized. Additionally, increasing the number of blocks results in more side information required for data extraction and cover restoration, further the more number of peak-zero pairs leads to increased histogram modifications, reducing resistance to steganalysis. For histogram partitioning, it was observed that dividing the histogram into two blocks provides the optimal hiding capacity.

## **CHAPTER 3**

### **TWO LAYER REVERSIBLE IMAGE STEGANOGRAPHY IN IMAGE HISTOGRAMS**

#### **3.1. Introduction**

Multi-layer embedding in reversible image steganography aims to enhance data hiding capacity while preserving the ability to fully recover the original cover image. Although this layered approach can significantly improve payload capacity, it also introduces challenges such as managing side information, maintaining high PSNR, and ensuring that reversibility is not compromised. Therefore, multi-layer embedding must be carefully designed to balance capacity, distortion, and reversibility.

#### **3.2. Payload Adaptive Histogram Shifting**

In this work, a novel payload-adaptive histogram shifting technique is developed, which dynamically adjusts the histogram shifting range based on the actual size of the data which needs to be embedded. Unlike traditional methods that use a fixed shifting range regardless of payload size, the proposed approach enhances both flexibility and embedding efficiency by tailoring the shifting range to the payload requirements. The shifting range is defined between peak and zero points. In this method, the global peak is updated by identifying a local peak with the minimum frequency that still satisfies the required payload size.

Figure 3.1 illustrates this change, showing the original and updated blocks, with the latter having narrower widths for the Lena image at a payload size of 1000 bits. Since the updated peak point is closer to zero point than original peak, the width of histogram block defined by peak and zero points is reduced, resulting in decreased shifting of the histogram.



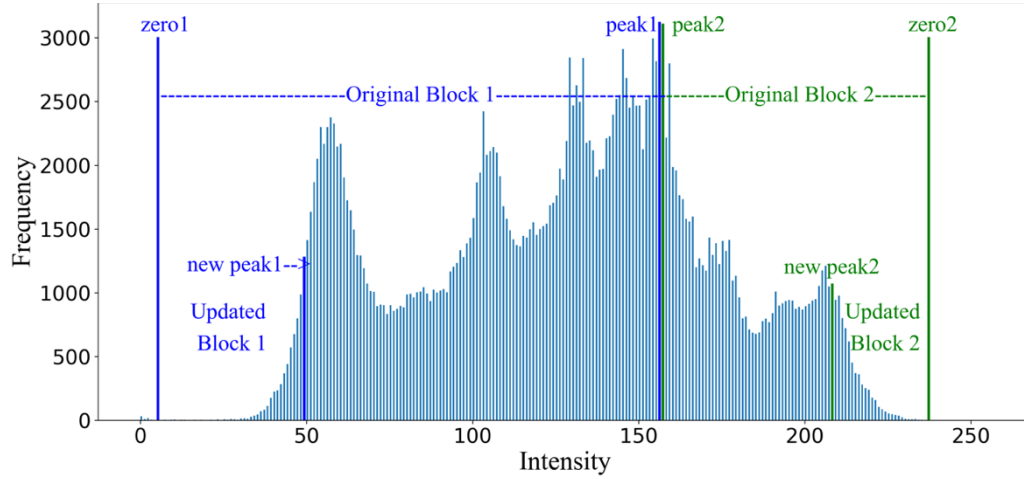


Figure 3.1: Reduced block width in Lena image for 1000-bit embedding

The algorithm, presented by Figure 3.2, begins by selecting the initial peak and zero intensity. It then scans from the zero intensity toward the global peak to locate a suitable local peak that can accommodate the data, thereby enabling an adaptive embedding process.

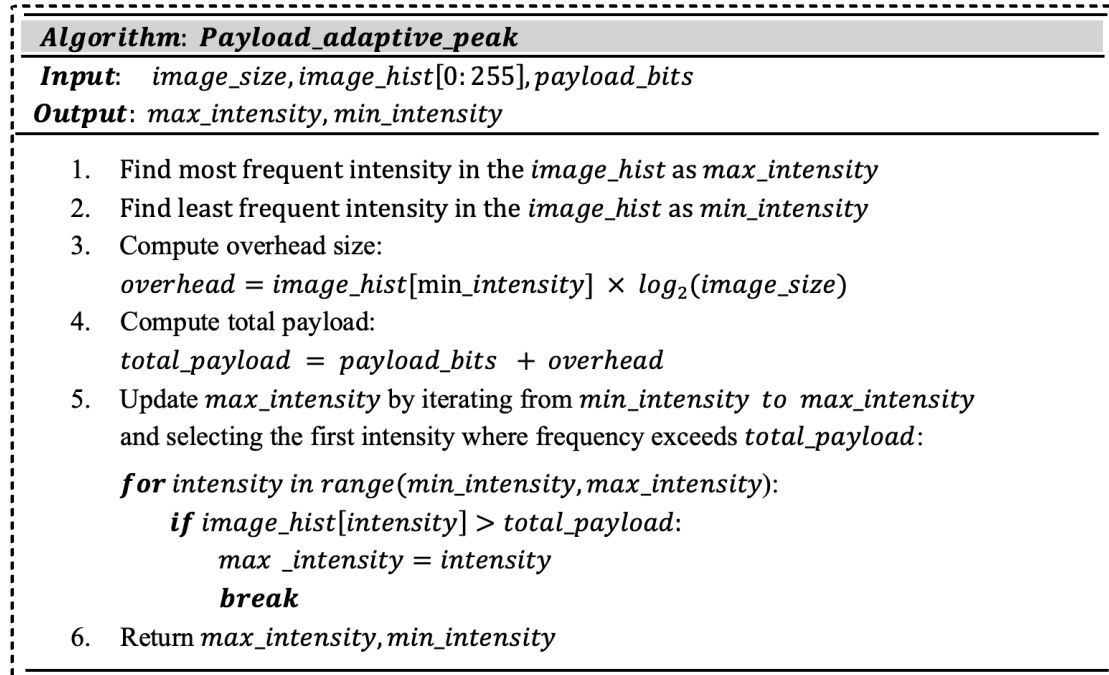


Figure 3.2: Algorithm for finding payload adaptive peak

Figure 3.3 presents the PSNR values for varying payload sizes for Baboon, Pepper, Lena and Sailboat images. The PSNR decreases with increase in payload size up to embedding capacity.

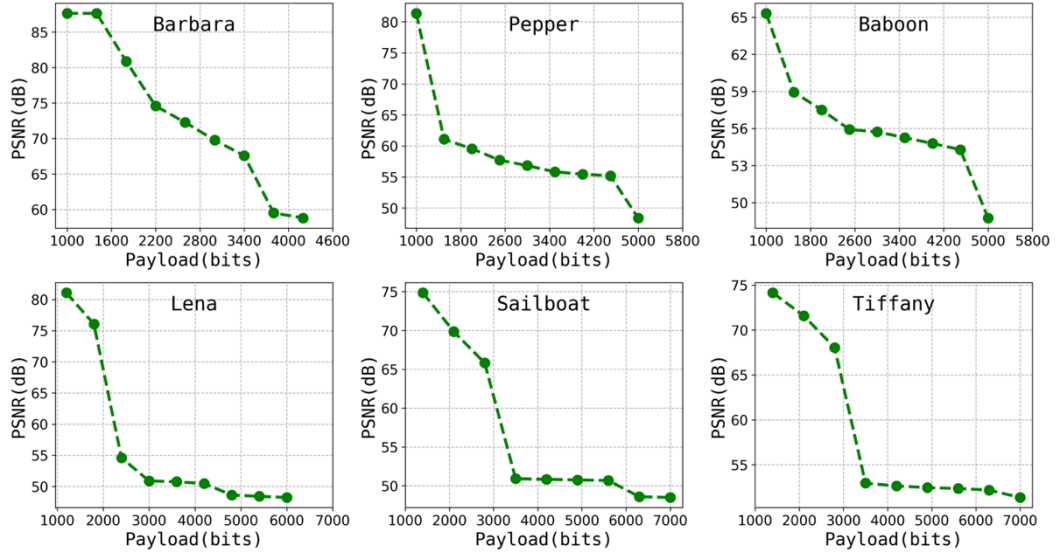


Figure 3.3: Payload adaptive PSNR for standard images

### 3.3. Analysis of Lower Bound on PSNR

Theoretical lower bound on PSNR for embedding layers up to 7 is computed. Each embedding layer shifts the histogram by one, resulting in a maximum pixel difference of  $n$ , and the MSE of  $n^2$ , for  $n$ -layer embedding. Table 3.1 presents the MSE and PSNR, computed using equations (3.1) and (3.2), respectively. In these equations  $C_{max} = 255$  is the maximum possible intensity,  $C$  and  $S$  are cover and stego images, and  $M \times N$  is the image size.

$$MSE = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (S(i,j) - C(i,j))^2 \quad (3.1)$$

$$PSNR = 10 * \log_{10} \left( \frac{C_{max}^2}{MSE} \right) \quad (3.2)$$

Table 3.1: Theoretical PSNR for varying layers of embedding

Layer	1	2	3	4	5	6	7
MSE	1	4	9	16	25	36	49
PSNR	48.13	42.11	38.58	36.08	34.15	32.56	31.22

### 3.4. Two Layer Embedding Strategy

Experiments are conducted for embedding up to five layers. Figure 3.4 shows the original image and corresponding stego images after each layer of embedding. Figure 3.5 shows the original histogram and corresponding stego image histograms.

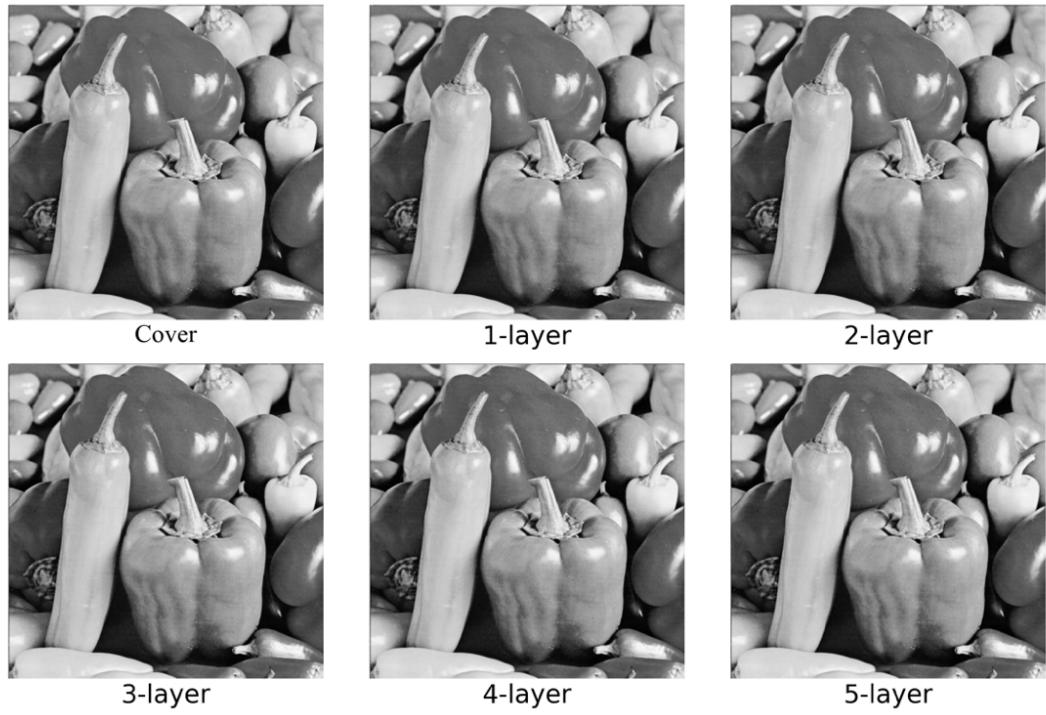


Figure 3.4: Stego images after multi-layer embedding

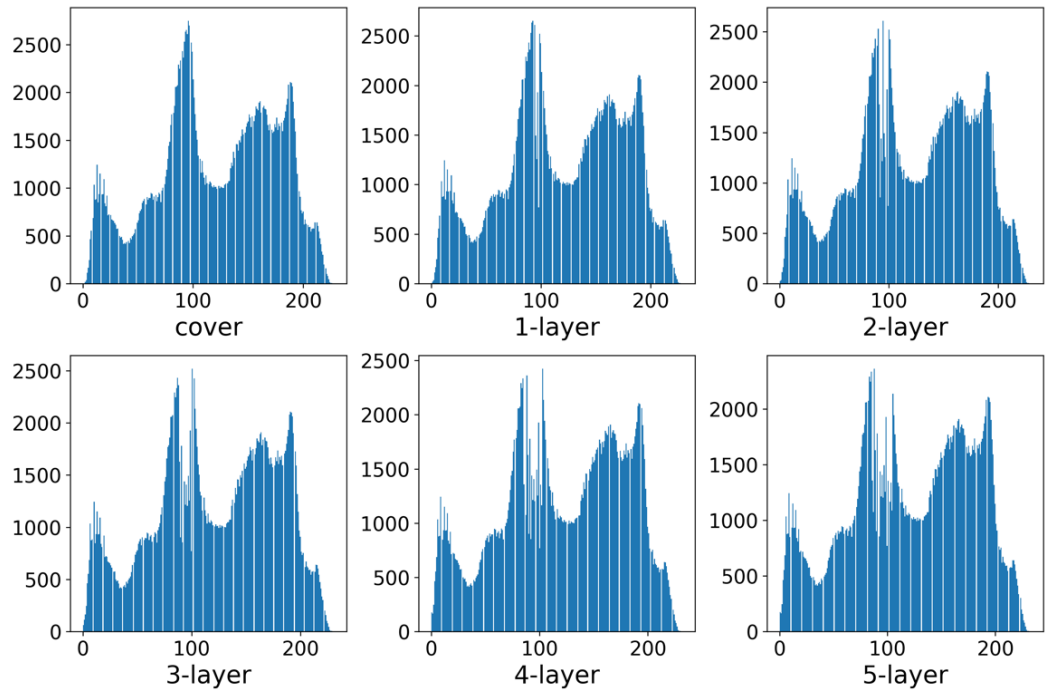


Figure 3.5: Stego image histograms after multi-layer embedding

The hiding capacity and PSNR observations are presented by Table 3.2 and Table 3.3 respectively. The capacity increases approximately linearly with each layer of embedding and the PSNR decreases gradually.

Table 3.2: Hiding capacity for multi-layer embedding

Image Name	1-Layer	2-Layer	3-Layer	4-Layer	5-Layer
Barbara	4456	8750	12888	16981	21041
Pepper	5314	10377	14933	18653	21015
Baboon	5245	10308	15229	20082	24679
Lena	6129	11938	17526	23043	28239
Sailboat	7489	14355	20851	26795	32597
Tiffany	7343	14499	21558	28547	35478
Fishing Boat	11322	21891	31691	40640	49277
Jet	15943	30427	43785	56493	68954
Airplane	59882	113241	162913	206093	238743
Sunset	68105	107716	136146	152023	164514
Cat	100963	197149	261144	309161	341770
Moon	122908	190257	245757	282994	313459

Table 3.3: PSNR for multi-layer embedding

Image Name	1-Layer	2-Layer	3-Layer	4-Layer	5-Layer
Barbara	53.45	47.24	43.16	39.90	38.64
Pepper	48.17	42.26	38.76	36.34	34.99
Baboon	48.22	42.27	38.77	36.33	34.47
Lena	48.17	42.55	39.71	37.14	35.27
Sailboat	48.19	42.26	38.82	37.62	37.20
Tiffany	48.23	42.46	39.00	36.54	34.73
Fishing Boat	48.21	42.60	39.23	36.84	35.02
Jet	51.10	42.45	38.98	36.79	34.92
Airplane	48.84	45.06	41.31	38.42	36.92
Sunset	50.53	42.76	39.44	36.98	35.14
Cat	49.81	47.89	45.77	44.09	41.64
Moon	49.41	43.23	40.19	37.48	35.67

To ensure high quality stego images, a PSNR of  $40dB$  or higher is preferred [37, 38, 39]. Three layer embedding reduces the PSNR below  $40dB$ , while two layer embedding maintains it above  $42dB$ . Therefore, two layer embedding is developed and analyzed for its robustness to steganalysis. It is shown by block diagram of Figure 3.6. In the first layer, Data1 is embedded into cover, creating an intermediate stego image. In the second layer, Data2 is embedded into the intermediate stego, producing the final stego image.

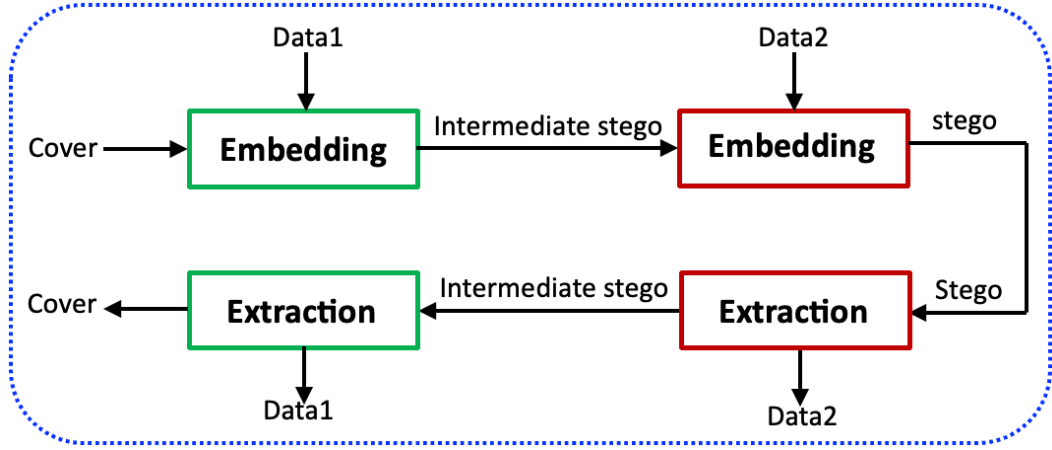


Figure 3.6: Block diagram of two layer embedding

During extraction, the process is reversed: Data2 is extracted first, followed by restoration of the intermediate stego. Subsequently, Data1 is extracted, and the cover is restored.

The pseudocode for two layer embedding is provided by Figure 3.7. The time complexity for single layer embedding is  $O(mn)$ , where  $m \times n$  is the image size. For multi-layer embedding, the time complexity scales linearly with number of layers, resulting in  $O(lmn)$ , where  $l$  denotes the number of layers. The space complexity remains  $O(mn)$  and is independent of layers.

**Algorithm: Two\_layer\_embedding****Input:**  $cover[1:M, 1:N], data[0:n], beg = 0$       *#beg is the pointer in data***Output:**  $stego[1:M, 1:N]$       *#  $M \times N$  is the image size*

1. **For** each embedding layer, perform the following:      *#two layers*
  - a. Construct histogram of  $cover$ , denoted by  $cover\_hist[0,255]$
  - b. Find two peak-zero pairs using Algorithm1:       $\#0 \leq P1, Z1, P2, Z2 \leq 255$   
 $P1, Z1, P2, Z2 = hist\_partition(cover\_hist[0,255])$
  - c. **For** each peak-zero pair  $(P, Z)$ , perform the following:
    - i. Compute payload capacity in bits by using equation (4):  
 $PF = cover\_hist[P]$       *#PF is frequency of peak point*  
 $ZF = cover\_hist[Z]$       *#ZF is frequency of zero point*  
 $hc = PF - (ZF \times \log_2(M \times N))$       *#hc is pure payload capacity*
    - ii. Create an array,  $payload[0:PF]$  and copy bits from  $data[0:n]$  to it  
 $payload[0:hc] = data[beg:(beg + hc)]$  and  $beg = beg + hc$
    - iii. Construct *overhead* for pixels with  $Z$  intensity, append to  $payload$ :  
 $payload[hc:PF] = overhead[0:0: \log_2(M \times N)]$
    - iv. Shift the histogram  $cover\_hist[0,255]$  towards right or left by 1:  
**if**  $(P < Z)$  then shift  $cover\_hist[P + 1, Z - 1]$  towards right  
**if**  $(P > Z)$  then shift  $cover\_hist[Z + 1, P]$  towards left
    - v. Scan the  $cover$  for hiding  $payload[0:PF]$  into it:  
**for** each pixel in the  $cover$ :  
**if**  $(P < Z)$  then  
**if** pixel is  $P$  then      *#hide the bit from payload*  
**if**  $payload$  bit is 1 then increment the pixel intensity by 1  
**elseif**  $(P > Z)$  then  
**if** pixel is  $P - 1$  then      *#hide the bit from payload*  
**if**  $payload$  bit is 1 then increment the pixel intensity by 1
2. **Return**  $stego$       *# hiding updates the cover as stego*

Figure 3.7: Algorithm for two layer embedding

**3.5. Robustness to Steganalysis**

The security of the scheme is evaluated by conducting steganalysis on widely used datasets such as SIPI [36], watermarking images [40], and BOSSbase [41, 42]. These datasets include diverse image types including dark, bright, reduced colour, textured, smooth, and sharp-edged images, featuring aerial views, buildings, humans, animals, and objects. Statistical analysis results are presented in Table 3.4 for SIPI database and Table 3.5 for watermarking database.

Table 3.4: Statistical steganalysis on SIPI database

Image	Image Analysis		Histogram Analysis			RS Analysis
	SSIM	NCC	Mean Ratio	Std. Ratio	Entropy $\Delta$	RS Ratio
1	0.999	0.999	0.996	0.983	0.004	0.995
2	0.997	0.999	0.995	0.971	0.004	0.990
3	0.997	0.999	0.997	0.962	0.004	0.997
4	0.998	0.999	0.993	0.970	0.005	0.988
5	0.996	0.999	0.992	0.979	0.006	0.996
6	0.997	0.999	0.996	0.968	0.006	0.988
7	0.995	0.999	0.996	0.973	0.010	0.987
8	0.995	0.999	0.994	0.981	0.012	0.995
9	0.997	0.999	0.993	0.979	0.053	0.972
10	0.953	0.999	0.975	0.995	0.048	0.988
11	0.994	0.999	0.990	0.994	0.081	0.993
12	0.910	0.999	0.975	0.995	0.090	0.974

Imperceptibility is evaluated by computing SSIM and Normalized Cross-Correlation (NCC) between cover and stego using equations (3.3) and (3.4) respectively, where  $\mu$  and  $\sigma$  are mean and standard deviation, and  $C1, C2, C3$  are constants to prevent the division by zero. The values approaching 1 confirm high similarity between original and stego images.

$$SSIM = \frac{2\mu_c\mu_s + C1}{\mu_c^2 + \mu_s^2 + C1} \cdot \frac{2\sigma_c\sigma_s + C2}{\sigma_c^2 + \sigma_s^2 + C2} \cdot \frac{\sigma_{cs} + C3}{\sigma_c\sigma_s + C3} \quad (3.3)$$

$$NCC = \frac{\sum_{i=1}^M \sum_{j=1}^N (S(i,j) \times C(i,j))}{\sum_{i=1}^M \sum_{j=1}^N S(i,j)^2} \quad (3.4)$$

Histogram analysis shows a mean and a standard deviation ratio for cover to stego near 1, with minimal changes in brightness and contrast due to embedding. Additionally, negligible changes in histogram entropy after embedding suggest negligible alteration to the histogram. RS steganalysis [43] shows that the regular-singular group ratio remains similar for cover and stego images, ensuring robustness against detection.

Table 3.5 Statistical steganalysis on Petitcolas database

Image	Image Analysis		Histogram Analysis			RS Analysis
	SSIM	NCC	Mean Ratio	Std. Ratio	Entropy $\Delta$	RS Ratio
1	0.998	0.999	0.996	0.968	0.004	0.995
2	0.998	0.999	0.999	0.984	0.005	0.987
3	0.996	0.999	0.997	0.962	0.006	0.997
4	0.997	0.999	0.993	0.975	0.009	0.993
5	0.999	0.999	0.991	0.998	0.001	0.978
6	0.995	0.999	0.985	0.990	0.009	0.999
7	0.999	0.999	0.985	0.991	0.008	0.950
8	0.993	0.998	0.994	0.952	0.014	0.994
9	0.998	0.999	0.984	0.996	0.014	0.968
10	0.993	0.997	0.991	0.951	0.016	0.995
11	0.999	0.999	0.988	0.991	0.022	0.983
12	0.994	0.999	0.982	0.988	0.026	0.962

The BOSSbase dataset, consisting of 10,000 grayscale images, was used to generate cover-stego pairs with the this method. Classification accuracy – using both the Spatial Rich Model (SRM) [44] and the Steganalysis Residual Network (SRNet) [45] – ranged between 49% and 51%, – a result comparable to random guessing, demonstrating the method’s robustness against machine-learning and deep-learning based steganalysis.

### 3.6. Performance Comparison

Reversibility is validated by computing PSNR and SSIM between original and reconstructed cover, with all experiments yielding  $PSNR = \infty$  and  $SSIM = 1$ , confirming reconstruction of cover. Empirical observations comparing the new method with prior methods are presented in Tables from 3.6 to 3.8. Table 3.6 compares the capacity and PSNR with methods [8], [9] where the new method offers up to 82% higher capacity. Table 3.7 shows that the adaptive histogram shifting yields improved PSNR values relative to methods reported in [8] [12].



Table 3.6: Performance comparison on hiding capacity and PSNR

Image Name	Capacity				PSNR		
	[8]	[9]	Proposed	% ↑ over [9]	[8]	[9]	Proposed
Baboon	5412	5892	10308	74.94	48.20	48.35	42.27
Pepper	5449	9499	10377	09.24	48.20	48.06	42.26
Lena	5460	9571	11938	24.73	48.20	47.30	42.55
Sailboat	7301	9039	14355	58.81	48.20	51.50	42.26
Fishing Boat	7301	12018	21891	82.15	48.10	47.85	42.61
Jet	16171	24421	30427	24.59	48.30	48.54	45.45
Airplane	59979	99099	113233	14.26	48.70	48.80	42.45

Table 3.7: Comparison of adaptive histogram shifting for Lena image

Method	Payload	Embedding	Pixel moves	PSNR
[8]	1000bits	Full Image	79482	49.88
[12]	1000bits	Block1	16681	60.38
		Block2	23588	58.22
		Block3	11336	62.05
		Block4	26892	57.73
Proposed	1000bits	Block1	6544	80.18
		Block2	6864	79.77

Table 3.8 provides a comparative summary of PSNR, selected peak-zero pairs, and resistance to steganalysis against methods [8, 9, 10, 11].

Table 3.8: Comparison with prior methods

Hiding Method	PSNR	Peak-Zero pairs	Resistance to analysis
Ni. et al. [8]	>48	2	High
Fallahpour [9]	>48	8-32	Low
Z. Pan et al. [10]	>48	Peak unaltered	Not secure
Murthy et al. [11]	>48	256	Not secure
Proposed	>42	4	High

While the PSNR of two-layer embedding is comparatively lower, it consistently exceeds the acceptable threshold of  $40dB$ .

Figure 3.8 depicts that new method achieves higher embedding rates than methods presented in [8, 9, 11] for Baboon, Pepper, Boat, and Jet images, measured in bits per pixel ( $bpp$ ) with embedding rates observed from  $0.02bpp$  to  $0.75bpp$ .

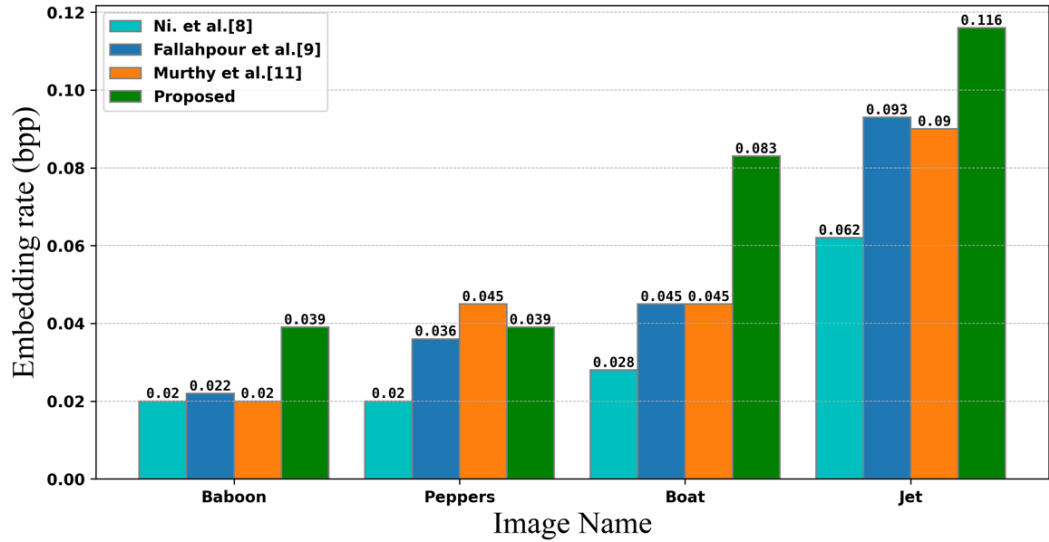


Figure 3.8: Embedding rates for selected standard images

### 3.7. Conclusion

The experiments are conducted using Python in Jupyter Notebook on MacBook Air with an Apple M1 chip and 8GB of RAM. Each embedding layer in the multi-layer embedding requires approximately 0.15 to 0.20 seconds of processing time and 2MB of memory for  $512 \times 512$  images. The scheme's simplicity, efficiency, high embedding capacity, and robustness to steganalysis make it well-suited for applications such as medical imaging, copyright protection, lossless image compression, cropping, etc. The scheme can be directly applicable to encrypted images when lightweight encryption methods such as affine ciphers, permutation-based (transposition) techniques, or substitution ciphers are used as these encryption methods preserve the overall histogram structure required for histogram shifting and making space for data embedding. Since video files consist of a sequence of images as frames, the scheme can be extended to videos by embedding data into individual frames. Applying the same embedding technique across multiple frames significantly enhances the overall embedding capacity.

## CHAPTER 4

### INTRODUCTION TO QUANTUM COMPUTING

#### 4.1. Background

A quantum computer is a machine that can receive inputs in a coherent superposition of multiple states, process these states and compute results into a superposition of possible outcomes. The idea of quantum computing was introduced by Richard Feynman in 1982 to address the limitations of existing computers in simulating quantum systems [46]. He suggested that a machine utilizing quantum bits (qubits) can simulate these systems more efficiently (with exponential speedup) than classical systems.

In 1985 David Deutsch proposed the concept of Quantum Turing Machine capable of simulating any physical system. He introduced the Deutsch algorithm, demonstrating that certain problems could be solved faster on quantum computers than classical computers [47]. The Deutsch-Jozsa algorithm was later developed as a generalization for  $n$ -bit inputs. It offers an exponential speedup over classical algorithms for determining whether a Boolean function is constant or balanced, under the promise that it is one of the two [48].

In 1994, Peter Shor published a quantum algorithm [49], and provided a detailed theoretical analysis of the algorithm [50], fundamentally challenging the security of the systems over the Internet. It showed that two problems considered hard for classical computers — integer factorization and the discrete logarithm problem — could be efficiently solved using a quantum computer. Since these problems underpin many cryptographic protocols, Shor's algorithm marked a major breakthrough, highlighting the practical impact of quantum computing in number theory, cryptography, and information security.

In 1995, Lov Grover demonstrated [51] that quantum computers could accelerate the searching in unstructured database, providing quadratic speed-up over classical

methods. Its broad applicability to search-based tasks generated significant interest in its potential.

Experimental progress began in 1998 with successful implementation of Deutsch-Jozsa algorithm using Nuclear Magnetic Resonance (NMR) [52]. In 2001, IBM researchers achieved a significant breakthrough by demonstrating the first experimental realization of Shor's algorithm on a 7-qubit NMR machine [53] by factoring 15, a milestone in quantum computing.

In 2012, John Preskill investigated the entanglement as an underlying principle that can be harnessed for computational advantage to surpass the capabilities of classical systems [54]. In 2018, Preskill came with the concept of the Noisy Intermediate-Scale Quantum (NISQ) devices [55], emphasizing challenges and potential of quantum devices with 50-100 qubits. These near-term processors, despite being affected by noise and imperfections, are capable of addressing practical problems.

In 2019, Google showcased quantum supremacy using its Sycamore processor, which contains 53 superconducting qubits. The processor accomplished a computational task in 200 seconds, a task that would have taken 10,000 years on classical supercomputers [56]. Although errors in the hardware led to accurate outputs only once in every 500 runs, repeating the experiment millions of times in just a few minutes gave statistically meaningful results.

IBM Quantum has pioneered quantum computing advancements [57], making quantum hardware accessible via IBM Cloud since 2016. The 127-qubit IBM Eagle processor is available for access since 2021. In 2023, IBM introduced Condor, a 1,121-qubit superconducting quantum processor. IBM's Qiskit SDK stands as the most widely adopted quantum development platform, enabling seamless execution on IBM's QPUs through Qiskit Runtime service. The release of Qiskit SDK 1.x in 2024, a comprehensive full-stack solution, marked a significant milestone. It now encompasses middleware software and services for developing and optimizing quantum circuits, and executing the developed circuits on IBM's state-of-the-art quantum systems.

Quantum computing relies on quantum arithmetic operations, ranging from addition to modular exponentiation, which require additional memory to store intermediate results. The construction of quantum networks with optimal auxiliary memory usage was first explored in 1995 [58]. The Quantum Fourier Transform (QFT) offers a unique approach to performing arithmetic operations on quantum computers. Thomas G. Draper introduced a QFT-based method for quantum addition, reducing the number of qubits needed by eliminating temporary carry bits [59].

QFT-based circuits have since been developed for various operations, including controlled weighted sums for computing the inner product of two data vectors [60], modular and non-modular arithmetic with signed integers [61], and quantum comparators [62]. Ripple-carry adders were introduced in 2004 as a foundational design for quantum addition circuits [63].

Toffoli-based adder circuits are developed using controlled-NOT gates and Toffoli gates. In [64], it was shown that five two-qubit gates are required to simulate a Toffoli gate. Additionally, a transformation of QFT addition circuits into Toffoli-based adders was presented in [65] offering an alternative approach to quantum arithmetic.

Quantum Image Processing (QIP) is an emerging field that involves performing image processing operations on quantum images, with the potential to achieve exponential computational speedup over classical methods [66]. Quantum algorithms designed for edge detection can achieve significant computational advantage [67]. QIP is built upon three key elements: encoding of images in quantum states, application of quantum algorithms on quantum images, and quantum measurement. To facilitate quantum computation for imaging tasks, various encoding techniques have been developed to map classical image data onto quantum states [68].

The Flexible Representation of Quantum Images (FRQI) is the first image encoding framework representing grayscale images using a single-qubit for intensity encoding and multi-qubit for positional encoding [69]. The pixel positions are mapped to a superposition of states, while intensities are encoded as quantum rotation angles. While FRQI offers a compact and simple image representation, it has notable drawbacks. Encoding intensities as rotation angles introduces computational

complexity, as retrieval often requires inverse trigonometric operations. The probabilistic nature of measurements and limited precision due to angle quantization reduce accuracy, making FRQI less suitable for high-fidelity image processing tasks. To overcome these limitations, researchers have developed improved models that enhance efficiency and reduce circuit depth [70]. Expanding on the FRQI framework, the RGB Multichannel Quantum Image (MCQI) representation is introduced to encode colour images by separately representing red, green, and blue channels using quantum states [71]. A quantum algorithm for scrambling of the images using MCQI model is reported [72], highlighting its potential for secure image processing and encryption.

NEQR is a framework [73] that represents both pixel positions and intensities using multiple qubits, offering improvements over FRQI. Since it directly maps intensity values to computational basis states, it allows for lossless image reconstruction, making it suitable for quantum image steganography, quantum machine learning, and other operations on quantum images. Several image processing techniques have been developed based on the NEQR, including methods for line detection [74], grayscale-to-binary image conversion [75], and guided quantum filtering [76]. The Novel Quantum Representation of Color Digital Images (NCQI) builds upon NEQR by using 24 qubits to represent Red, Green, and Blue channels [77] for colour images. NEQR based steganographic techniques are reported in [78, 79, 80], quantum image encryption techniques are proposed in [81, 82, 83], and image scrambling techniques are presented in [84, 85].

## **4.2. Principles of Quantum Mechanics**

The fundamental principles of quantum mechanics are superposition, entanglement, decoherence, and interference, offering profound insights into the behaviour of matter and energy at sub-atomic level. Quantum computers utilize these foundational quantum phenomena to perform calculations in a probabilistic and quantum-mechanical manner. When fully developed, they will be capable of solving highly complex problems at speeds exponentially greater than today's classical computers, offering unprecedented computational power.

### 4.2.1. Superposition

Quantum mechanical systems do manifest all the possible states that they can assume at the same time, such as electrons or photons can be in a combination of states. This phenomenon is called superposition. A quantum state in superposition represents a linear combination of multiple states where the combination is a new, valid state. A single-qubit quantum system is mathematically expressed by equation (4.1):

$$|\Psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle \quad (4.1)$$

Here the coefficients  $\alpha_0$  and  $\alpha_1$  are complex numbers known as probability amplitudes associated with basis states  $|0\rangle$  and  $|1\rangle$ , respectively.

When a measurement is made, the superposition reduces to a single, definite state, with basis states probabilities given by  $|\alpha_0|^2$  and  $|\alpha_1|^2$ . These probabilities always sum to one, satisfying the condition  $|\alpha_0|^2 + |\alpha_1|^2 = 1$ .

A two-qubit system is in a superposition of the four basis states, with probability amplitudes being  $\alpha_{00}$ ,  $\alpha_{01}$ ,  $\alpha_{10}$ , and  $\alpha_{11}$  respectively, and  $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$ .

Ket notation for a superposition state in this system is given by equation(4.2):

$$|\Psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \quad (4.2)$$

Or by vector notation presented in equation (4.3):

$$\begin{bmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{bmatrix} \quad (4.3)$$

Similarly, an  $n$ -qubits system can be in a superposition of  $2^n$  states ranging from  $|000\dots 0\rangle$  to  $|111\dots 1\rangle$ . A quantum computer can process all these states simultaneously, making computations faster.

### 4.2.2. Entanglement

Entanglement represents the intrinsic non-locality of quantum mechanics, manifesting when the state of two or more subsystems in a quantum system cannot be decomposed into individual local states of the subsystems. In an entangled system, the state of each

individual particle cannot be defined on its own without considering the states of the other particles. Although the system as a whole is in a definite state, the individual components do not possess well-defined separate states. For instance, if two particles are generated with a total spin of zero, measuring the spin of one particle immediately determines that the spin of the second particle will be opposite of the spin of first particle, even if the two particles are far apart.

Two entangled qubits are correlated and the measurement outcome of one of the qubits instantly determines the measurement outcome of another qubit. This phenomenon is explained by Bell states, which can be represented as follows by equation (4.4):

$$|\Phi^\pm\rangle = \frac{|00\rangle \pm |11\rangle}{\sqrt{2}}, \quad |\Psi^\pm\rangle = \frac{|01\rangle \pm |10\rangle}{\sqrt{2}} \quad (4.4)$$

Entanglement gives quantum computing an advantage over classical computing by enabling faster algorithms, enhancing secure communication, and aiding in development of fault-tolerant quantum systems.

#### 4.2.3. Decoherence

Decoherence is a key phenomenon in which fragile superpositions in a quantum system break down, transforming from quantum behaviour to classical behaviour. This happens when the system interacts with its surroundings or undergoes measurement. Since the system behaves classically after decoherence, this is important for allowing quantum states to produce observable outcomes and interface with classical systems.

However, environmental interactions that induce decoherence lead to disruption of quantum superposition or entangled states, introducing errors and information loss in qubits. The timescale of decoherence depends on the underlying qubit technology and the degree of isolation from the external environment. More isolated systems exhibit longer coherence times. Mitigating decoherence is a major research challenge, and maintaining quantum coherence is essential for performing accurate and reliable computations.



#### 4.2.4. Interference

In a quantum system, particles are represented as probability waves that indicate a range of possible positions. These probability waves can interact with one another, resulting in an interference pattern that influences the likelihood of different measurement outcomes. Specifically, when these waves reinforce each other, the phenomenon is referred to as constructive interference, leading to an increased probability of certain outcomes. Conversely, when the waves cancel each other out, this is termed destructive interference, resulting in a decreased likelihood of particular measurement results.

Interference enables quantum algorithms to manipulate the probabilities associated with various possible outcomes, guiding the system toward the correct solution. Constructive interference enhances the likelihood of measuring the correct outcome, while destructive interference effectively reduces the probability of incorrect outcomes. This manipulation of interference is essential for implementation of many quantum algorithms, as it enhances both the efficiency and precision of the algorithms.

#### 4.3. Quantum Logic Gates

A qubit is a fundamental unit of quantum information and computation, governed by quantum mechanical principles like superposition, entanglement, and interference. A qubit can exist in a superposition of two basis states of  $|0\rangle$  and  $|1\rangle$  and can also become entangled with other qubits. A single-qubit quantum system is defined as shown by equation (4.5):

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (4.5)$$

A qubit in superposition has some probability of being  $|0\rangle$  and some probability of being  $|1\rangle$ , and this can be represented by equation (4.6) :

$$|\Psi\rangle = a|0\rangle + b|1\rangle = \begin{bmatrix} a \\ b \end{bmatrix} \quad (4.6)$$

where  $a$  and  $b$  are complex numbers representing the amplitudes of component states in the superposition state  $|\Psi\rangle$ , with  $|a|^2 + |b|^2 = 1$ .

Quantum logic gates [86] are the operations used to control and change the states of qubits, similar to how classical gates (AND, OR, and NOT) are used to manipulate bits. Key Characteristics of quantum gates include:

- **Gate Size:** Quantum gates act on single or multiple qubits, enabling complex operations and interactions between qubits.
- **Unitary:** Quantum gates are inherently reversible operators. An operator  $U$  is unitary if the inverse of the operator is equivalent to its conjugate transpose, meaning  $U^{-1} = U^\dagger$ . For an  $n$ -qubit quantum gate, the matrix size is  $2^n \times 2^n$ .
- **Superposition:** Gates can operate on the qubits which are in superposition and can process all possible states simultaneously.
- **Entanglement:** Some gates create entanglement, establishing required correlations between qubits.

#### 4.3.1. Single-Qubit Gates

Single-qubit gates operate on individual qubits. The Hadamard gate ( $H$ ) is essential for creating superposition, converting a qubit from standard basis states into an equal superposition of both, and can also reverse this transformation.

The Pauli gates form a core set, represented by Pauli matrices, and perform operations like flipping, rotating, or inverting qubits. The set includes the  $X$ ,  $Y$ ,  $Z$  gates, each corresponding to a rotation of the qubit around respective axes of the Bloch sphere by  $\pi$  radian. The  $X$  gate flips the qubit between  $|0\rangle$  and  $|1\rangle$ . The  $Y$  gate performs both a bit and a phase flip on the qubit. The  $Z$  gate does not change the state  $|0\rangle$  while it flips the phase of the  $|1\rangle$ . The unitary matrices for Hadamard gate and Pauli gates are represented by equation (4.7):

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad (4.7)$$

The phase shift is a set of gates that transform the basis state  $|1\rangle$  to  $e^{i\varphi}|1\rangle$ , leaving the measurement probabilities unchanged. Special phase shift gates include the  $S$  and the  $T$  gates, applying a phase shift of  $\pi/2$  and  $\pi/4$ , respectively, to state  $|1\rangle$ .

The unitary matrices for these gates are shown in equation (4.8):

$$Ph(\varphi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix}, \quad S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} \quad (4.8)$$

Rotation gates are the most commonly used parametric gates. They apply a rotation around one of the axes on the Bloch sphere (X, Y, or Z), with the rotation angle serving as the parameter. These gates perform operations allowing for more flexibility and control in quantum algorithms, especially in quantum machine learning, variational quantum algorithms, and quantum optimization. The unitary matrices for these gates are shown in the equations (4.9), (4.10) and (4.11), respectively :

$$R_x(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix} \quad (4.9)$$

$$R_y(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix} \quad (4.10)$$

$$R_z(\theta) = \begin{bmatrix} e^{-\frac{i\theta}{2}} & 0 \\ 0 & e^{\frac{i\theta}{2}} \end{bmatrix} \quad (4.11)$$

#### 4.3.2. Multi-Qubit Gates

Multi-qubit gates operate on multiple qubits simultaneously, harnessing the complete potential of qubit interactions using entanglement. The commonly used two qubit gates are SWAP, CNOT, CZ and CP, shown in Table 4.1 with their unitary matrices and outputs of their application on the basis states  $|00\rangle$ ,  $|01\rangle$ ,  $|10\rangle$ , and  $|11\rangle$ .

The SWAP gate interchanges states of two qubits. The CNOT gate operates by using the first qubit as control and second qubit as target. The target qubit is flipped only if the control qubit is  $|1\rangle$ . The CZ gate applies a phase flip to the target qubit under the same condition. The CP gate is a generalization of CZ gate, introducing a phase shift of  $e^{i\varphi}$  on the target qubit when the control qubit is in the  $|1\rangle$  state. These gates are used for performing controlled quantum phase rotations and are fundamental to interference-based quantum algorithms.

Table 4.1: Two-Qubit gates and their unitary matrices

Gate	Unitary Matrix	Application on Basis States
SWAP	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{aligned}  00\rangle &\xrightarrow{SWAP}  00\rangle, &  01\rangle &\xrightarrow{SWAP}  10\rangle \\  10\rangle &\xrightarrow{SWAP}  01\rangle, &  11\rangle &\xrightarrow{SWAP}  11\rangle \end{aligned}$
CNOT	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	$\begin{aligned}  00\rangle &\xrightarrow{CNOT}  00\rangle, &  01\rangle &\xrightarrow{CNOT}  01\rangle \\  10\rangle &\xrightarrow{CNOT}  11\rangle, &  11\rangle &\xrightarrow{CNOT}  10\rangle \end{aligned}$
CZ	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$	$\begin{aligned}  00\rangle &\xrightarrow{CZ}  00\rangle, &  01\rangle &\xrightarrow{CZ}  01\rangle \\  10\rangle &\xrightarrow{CZ}  10\rangle, &  11\rangle &\xrightarrow{CZ} - 11\rangle \end{aligned}$
CP	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\varphi} \end{bmatrix}$	$\begin{aligned}  00\rangle &\xrightarrow{CP}  00\rangle, &  01\rangle &\xrightarrow{CP}  01\rangle \\  10\rangle &\xrightarrow{CP}  10\rangle, &  11\rangle &\xrightarrow{CP} e^{i\varphi} 11\rangle \end{aligned}$

The Toffoli or CCNOT is a three-qubit gate. It flips the third qubit, target qubit, if the first two qubits, known as control qubits, are  $|1\rangle$ . It is a reversible counterpart to the classical AND gate when considering the third output, making it an important gate for reversible and quantum computing. The unitary matrix and its application on basis states is shown by equation (4.10):

$$Toffoli = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (4.10)$$

$$\begin{aligned} |000\rangle &\xrightarrow{Toffoli} |000\rangle, & |001\rangle &\xrightarrow{Toffoli} |001\rangle \\ |010\rangle &\xrightarrow{Toffoli} |010\rangle, & |011\rangle &\xrightarrow{Toffoli} |011\rangle \\ |100\rangle &\xrightarrow{Toffoli} |100\rangle, & |101\rangle &\xrightarrow{Toffoli} |101\rangle \\ |110\rangle &\xrightarrow{Toffoli} |111\rangle, & |111\rangle &\xrightarrow{Toffoli} |110\rangle \end{aligned}$$

### 4.3.3. Universal Quantum Gates

Any quantum computation can be implemented by a universal set of quantum gates. Several examples of these sets are presented in Table 4.2. Clifford gates preserve the Pauli operators ( $X, Y, Z$ ) by mapping to other Pauli operators. Pauli matrices are all involutions ( $X^2 = Y^2 = Z^2 = I$ ), and the relations between them are shown by equation (4.11). Well-known Clifford gates are Pauli ( $X, Y, Z$ ), Hadamard, CNOT and SWAP gate.

$$\begin{aligned} XY &= iZ; & YX &= -iZ; \\ YZ &= iX; & ZY &= -iX; \\ ZX &= iY; & XZ &= -iY \end{aligned} \quad (4.11)$$

Table 4.2: Unitary sets of Quantum gates

Sr. No.	Universal Gate Set	Common Examples
1.	Clifford with T	$\{ \text{Clifford Gates}, T \}$
2.	Single qubit gates with CNOT	$\{ H, S, T, CNOT \}$
3.	Rotation based gates with CNOT	$\{ R_x(\alpha), R_y(\beta), CNOT \}$
		$\{ R_y(\beta), R_z(\gamma), CNOT \}$
		$\{ R_x(\alpha), R_z(\gamma), CNOT \}$

### 4.4. Quantum Circuits

Quantum algorithms are implemented using quantum circuits. A quantum circuit consists of a sequence of quantum gates applied to qubits to carry out the desired computations. It is represented as a diagram where qubits are shown as horizontal lines, and quantum gates are placed along these lines. The circuit execution sequence is from left to right, with each gate applying a specific operation to the input qubits.

Since quantum operations must be reversible, classical logic gates are adapted into reversible quantum circuits by including extra qubits to store output information. The circuits implementing the logic of AND, OR, and XOR gates for all possible input combinations are shown in Figure 4.1.

The circuit structure changes with the input to preserve reversibility and accurately reflect the corresponding classical logic functionality. All the quantum circuits were developed using Qiskit's QuantumCircuit module.

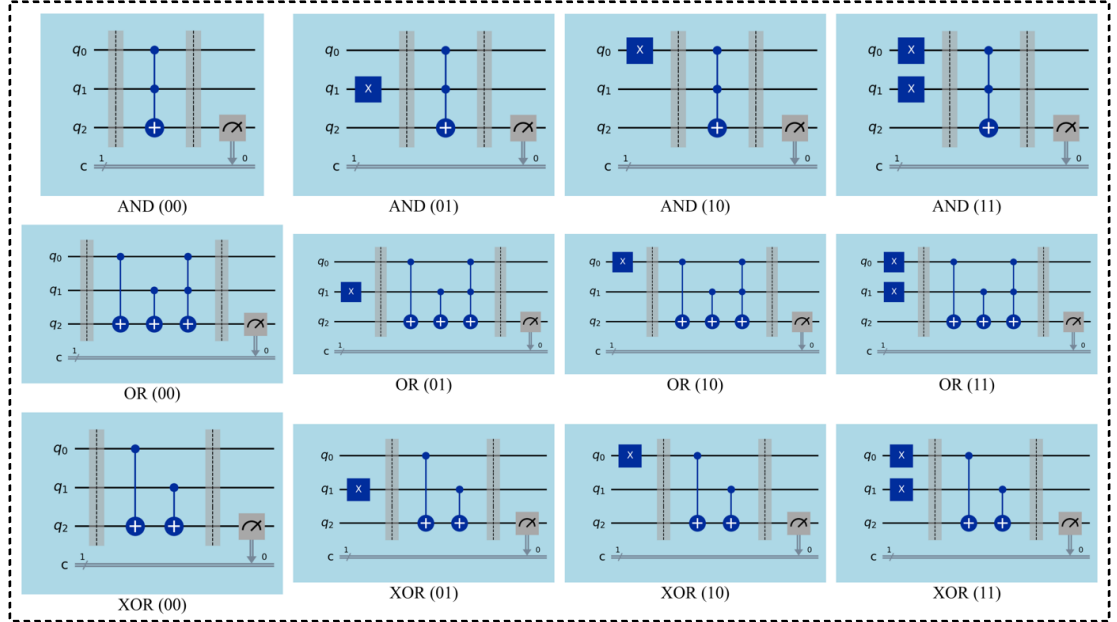


Figure 4.1: Quantum circuits for classical gates

## 4.5. Experimental Methodology

The experiments in this study were conducted using Qiskit, a Python-based SDK for developing and simulating quantum circuits, and the IBM Quantum platform for executing them on IBM quantum hardware. The detailed step-by-step procedure is described below:

### 4.5.1. Qiskit Installation

To begin, Qiskit was installed. Python and pip, the Python package manager, were pre-installed. A virtual environment was set up to isolate the dependencies of Qiskit and then it was installed using: “*pip install qiskit*”. To enable execution on QPUs, the IBM Runtime module was installed using: “*pip install qiskit-ibm-runtime*”. Detailed installation instructions can be found in [87].

#### 4.5.2. Quantum Circuit Creation

Quantum circuits were developed by using Qiskit's *QuantumCircuit* module. The basic steps for development of quantum circuits are as below:

- Initialization of quantum and classical registers.
- Construction of the quantum circuit.
- Addition of gates to the circuit for carrying out the computations.
- Measurements to map quantum states to classical bits, facilitating result extraction.

#### 4.5.3. Ideal Simulation

To debug the circuits and verify the correctness of the implementation, the Basic Simulator was used. This is a local noiseless simulator, supporting up to 24 qubits. The steps for simulations are as follows:

- Import the simulator: `"from qiskit.providers.basic_provider import BasicSimulator"`
- Initialize the backend: `"backend=BasicSimulator()"`
- Execute the circuit : `"job=backend.run(circuit)"`
- Retrieve the results: `"results=job.result()"`

#### 4.5.4. Noisy Simulation

To emulate real hardware, IBM's noisy quantum simulators were used. These simulators incorporate noise models to predict performance on actual quantum devices. The steps include:

- Import the simulator: `"from qiskit.providers.fake_provider import GenericBackendV2"`
- Initialize the backend: `"backend=GenericBackendV2(num_of_qubits)"`
- Transpile the circuit:  
`"from qiskit import transpile"`  
`"circuit1=transpile(circuit, backend)"`
- Execute the circuit: `"job=backend.run(circuit1)"`
- Retrieve the results: `"results=job.result()"`

#### 4.5.5. Execution on Quantum Hardware

Quantum circuits were executed on IBM Quantum physical devices to assess performance in real-world conditions, including noise, decoherence, and scalability challenges. The procedure included:

a) User Account Creation:

Users register on the IBM Quantum platform and receive an API token for authentication. User under the Open Plan are granted access to QPUs for up to 10 minutes per month.

b) Service and Backend Initialization:

The `QiskitRuntimeService` class was used to interface with IBM Quantum services. The backends that can be accessed under the Open Plan are “*ibm\_brisbane*”, “*ibm\_kyiv*”, and “*ibm\_sherbrooke*”. The python code for this is as follows:

```
“from qiskit_ibm_runtime import QiskitRuntimeService”  
“service=QiskitRuntimeService (channel="ibm_quantum", token="API token)”  
“backend=service.backend("ibm_brisbane)”
```

c) Circuit Transpilation:

The quantum circuit was transpiled to the optimized and compatible circuit for the selected backend:

```
“qc_transpiled=transpile(qc, backend, optimization_level = 3)”
```

d) Job Submission:

The transpiled circuit was submitted to the backend using the `SamplerV2` primitive, which generates a unique job ID for future reference:

```
“from qiskit_ibm_runtime import SamplerV2”  
“sampler=SamplerV2(backend)”  
“job_qc=sampler.run(qc_transpiled, shots=1000)”  
“job_id=job_qc.job_id()”
```



e) Result Extraction and Post-Processing:

After job execution, the results were retrieved and measurement counts were visualized as histograms:

```
“result=job_qc.result()”  
“data1=result[0].data”  
“counts=data1.creg.get_counts()”  
“plot_histogram(counts)”
```

The methodology presents the process of designing, validating, and executing circuits using Qiskit.

# CHAPTER 5

## QUANTUM ARITHMETIC OPERATIONS

Quantum arithmetic operations are essential for performing computations. These operations are implemented using quantum algorithms with reversible and unitary gates that preserve quantum information. Addition forms the core of all arithmetic operations because the operations of subtraction, multiplication, exponentiation, and division can be decomposed to repeated forms of addition. Efficient realization of quantum addition circuits is therefore fundamental, serving as the building block for higher-level quantum arithmetic.

### 5.1. Quantum Circuits for Half Adder and Full Adder

A half adder circuit performs the addition of two single-bit numbers and outputs a sum bit and a carry bit. By combining two half adders, a full adder is formed, which adds three binary inputs (two data bits and one carry-in bit) and generates a sum and a carry-out bit. Quantum circuits for these operations are illustrated in Figure 5.1.

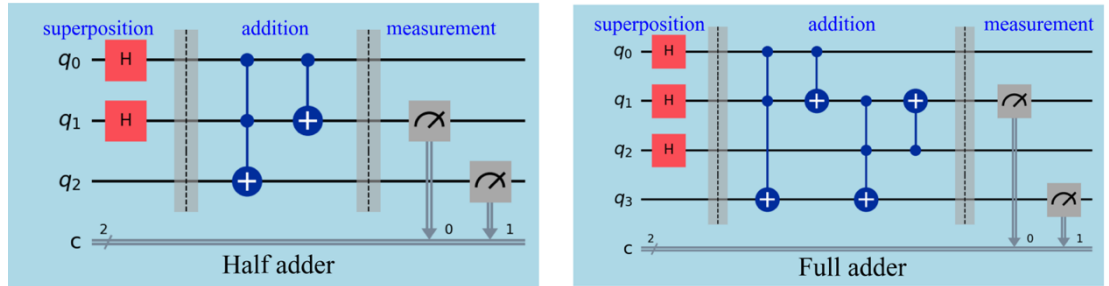


Figure 5.1: Quantum circuits for half adder and full adder

### 5.2. Quantum Circuits for Binary Adder

A binary adder refers to a circuit that adds multiple bits, extending the full adder concept. Adding two  $n$ -bit numbers requires  $n$  full adders. In classical computing, binary adders are built using irreversible logic gates, whereas quantum computation requires all operations to be unitary and thus reversible. Quantum adders are generally classified into two main types: based on Toffoli gates and based on the QFT.

### 5.2.1. Toffoli-Based Adder

Toffoli-based quantum adders simulate classical binary addition using reversible quantum logic gates such as the Toffoli (CCNOT), CNOT, and NOT gates. The adders were developed for adding 1-bit to 6-bit numbers. Figure 5.2 shows a circuit for adding two 3-bit numbers ( $7+7$ ).

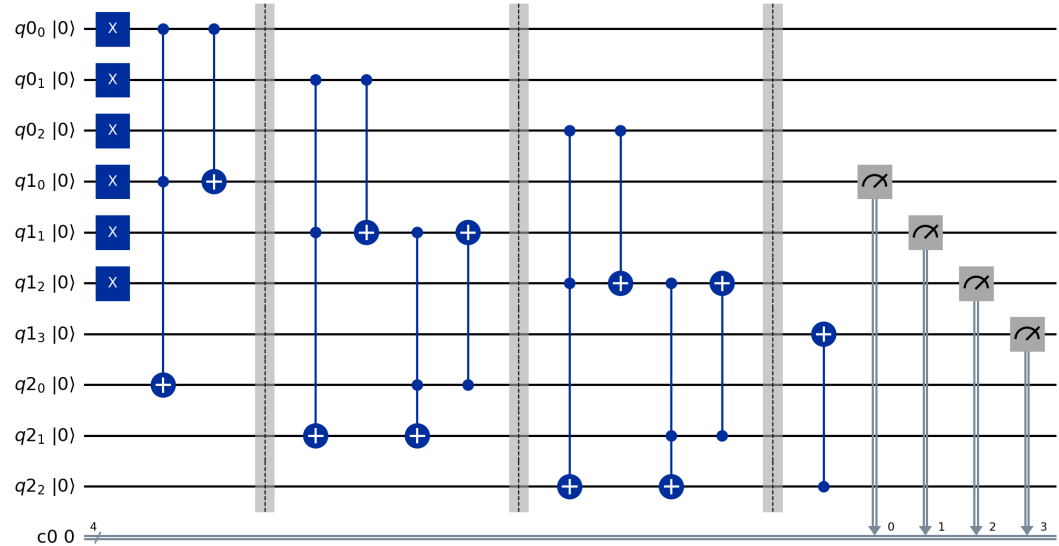


Figure 5.2: Toffoli-based adder circuit for  $7+7$

### 5.2.2. QFT-Based Adder

QFT-based adders implement binary addition through the Quantum Fourier Transform, which enables operations in the phase domain. QFT-based adder encodes numbers in quantum phase space and performs addition by applying controlled phase rotations. After addition, the inverse QFT is applied to revert the result back to the computational basis. Circuits were developed for adding 1-bit to 6-bit numbers. Figure 5.3 depicts the circuit for adding two 2-bit numbers ( $3+3$ ).

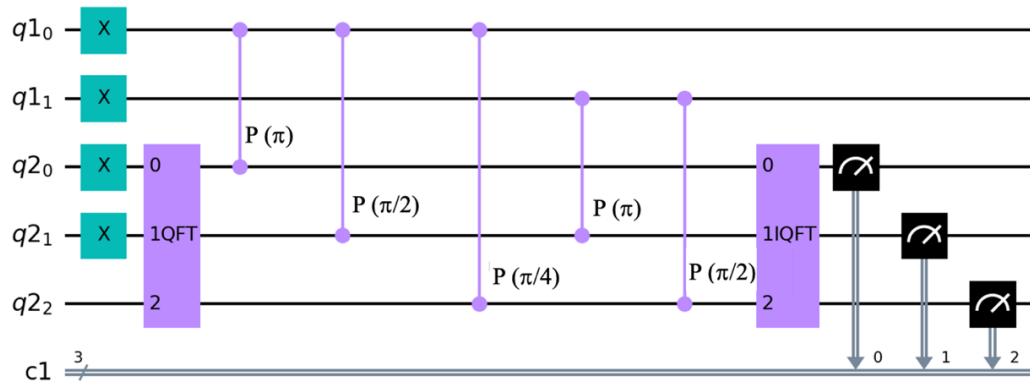


Figure 5.3: QFT-based adder circuit for  $3+3$

### 5.3. Quantum Comparator Circuit

Comparator circuits for 1-bit to 6-bit numbers were developed. The circuits for 2-bit, 3-bit, and 4-bit comparators are shown by Figures 5.4 (a), (b) and (c), respectively.

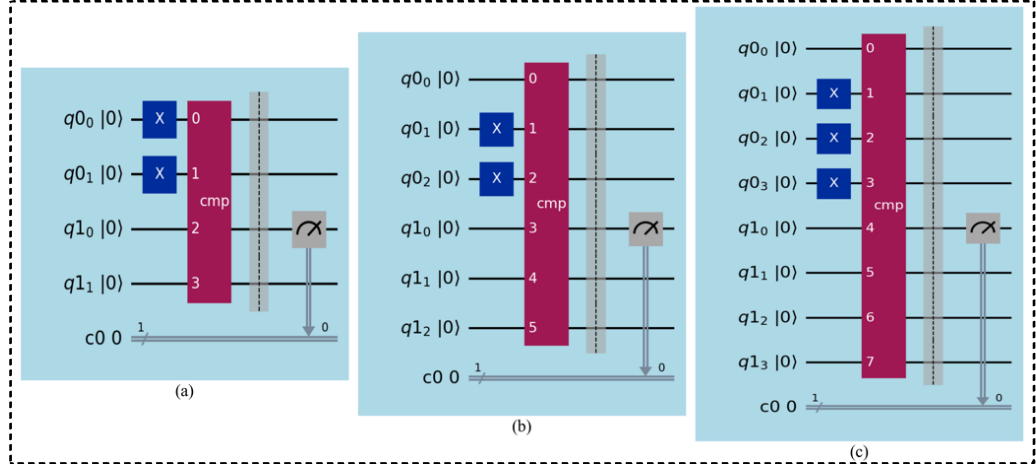


Figure 5.4: Comparator circuits for (a) 2bit (b) 3bit (c) 4bit

### 5.4. Circuit Transpilation

The circuits are transpiled to match the topology of a quantum simulator or hardware. Figure 5.5(a) is the transpiled circuit for Toffoli adder for 1+1, and Figure 5.5(b) is the transpiled circuit for QFT adder for 1+1. Transpilation increases the circuit depth.

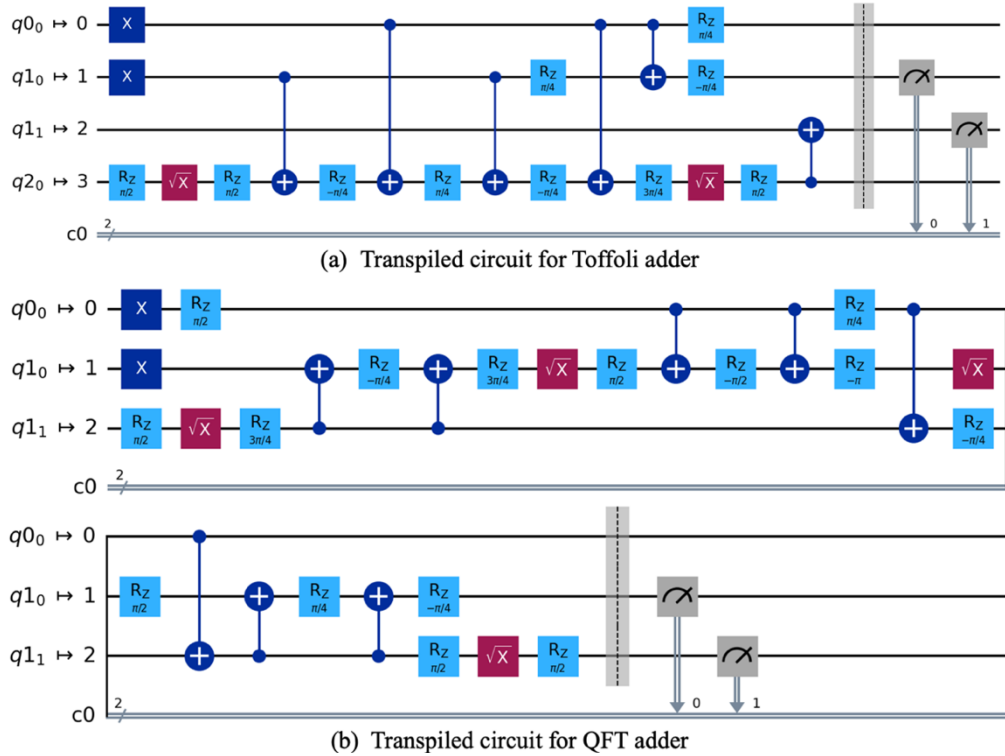


Figure 5.5: Transpiled circuits for Toffoli and QFT adders for 1+1

## 5.5. Experimental Results

The quantum circuits were initially simulated on IBM's *BasicSimulator* to verify the correctness of implementation. Then, simulations were performed on the noisy backend to emulate the quantum hardware noise. Finally, the circuits were executed on IBM's 127-qubit Eagle quantum processor, using the *ibm\_brisbane* and *ibm\_kyiv* backends via the *SamplerV2* primitive in Qiskit Runtime. Each circuit was executed for 1000 shots. The results are visualized using histograms, where X-axis represents possible outcomes, and the Y-axis represents frequencies associated with outcomes.

### 5.5.1. Half Adder and Full Adder Results

Figure 5.6(a) illustrates the results for half-adder, where the sum and carry are measured with inputs in a maximal superposition of states. During ideal simulation, all possible outcomes are observed except the outcome 11. This aligns with the expected outcomes for all inputs:  $0 + 0 = 00$ ,  $0 + 1 = 01$ ,  $1 + 0 = 01$ ,  $1 + 1 = 10$ . The small frequency of measuring 11 in the noisy simulation and execution on actual hardware, is attributed to the noise introduced by decoherence.

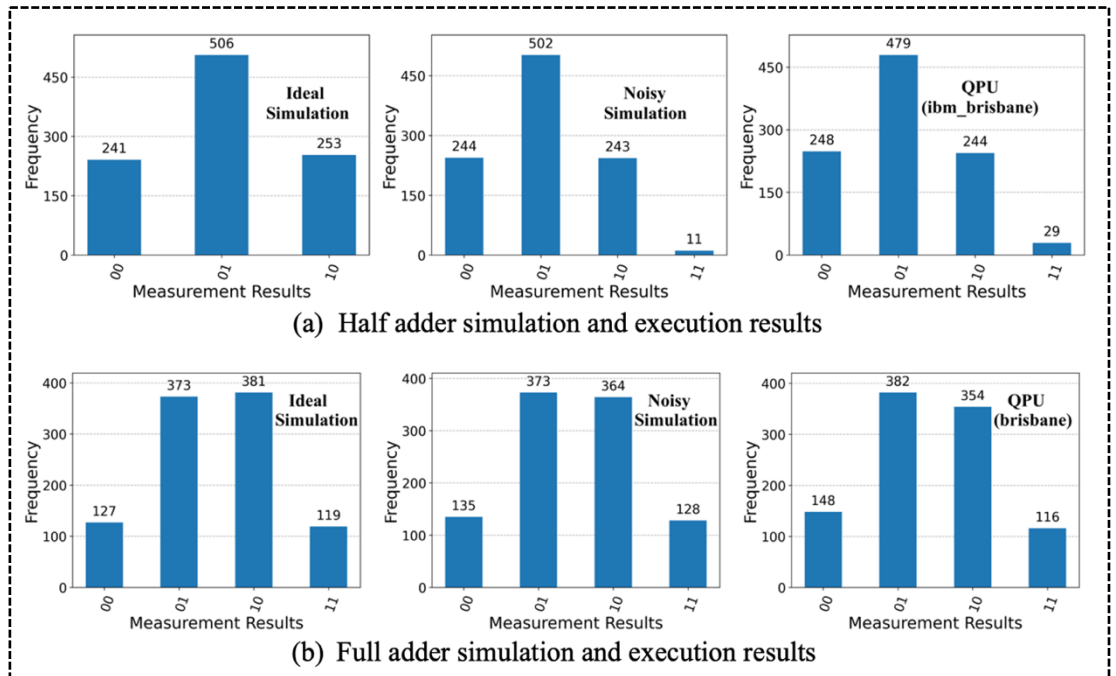


Figure 5.6: Half adder and full adder results on simulator and QPU

Figure 5.6 (b) presents the results for full-adder. The results on ideal simulator align precisely with theory. The results from noisy simulator incorporate quantum noise models to simulate the effects of decoherence and gate imperfections. The experimental results on the QPU via *ibm\_brisbane* are comparable to noisy simulator results. It is noted that the results on QPU are reliable.

### 5.5.2. Toffoli-Based Adder Results

Figure 5.7 presents the measurement results for the Toffoli-based adder circuit executed on noisy simulator for input sizes ranging from 1-bit to 6-bits, with corresponding output sizes ranging from 2-bit to 7-bits. Subfigures (a) to (f) illustrate the results for input pairs:  $(1 + 1)$ ,  $(3 + 3)$ ,  $(7 + 7)$ ,  $(15 + 15)$ ,  $(31 + 31)$ , and  $(63 + 63)$ , respectively.

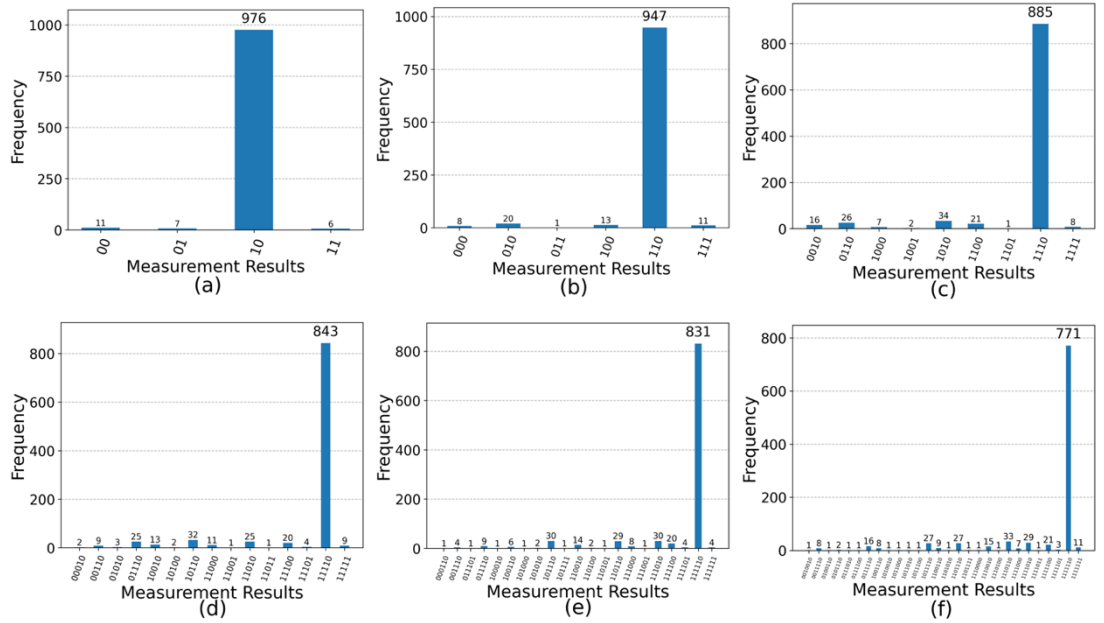


Figure 5.7: Noisy simulator results for Toffoli-based adder

Figure 5.8 presents the results of executing the same circuit on IBM's QPU via *ibm\_brisbane*. A comparison of results on noisy simulator and QPU shows that the noise increases significantly with input size on the QPU compared to the simulator.

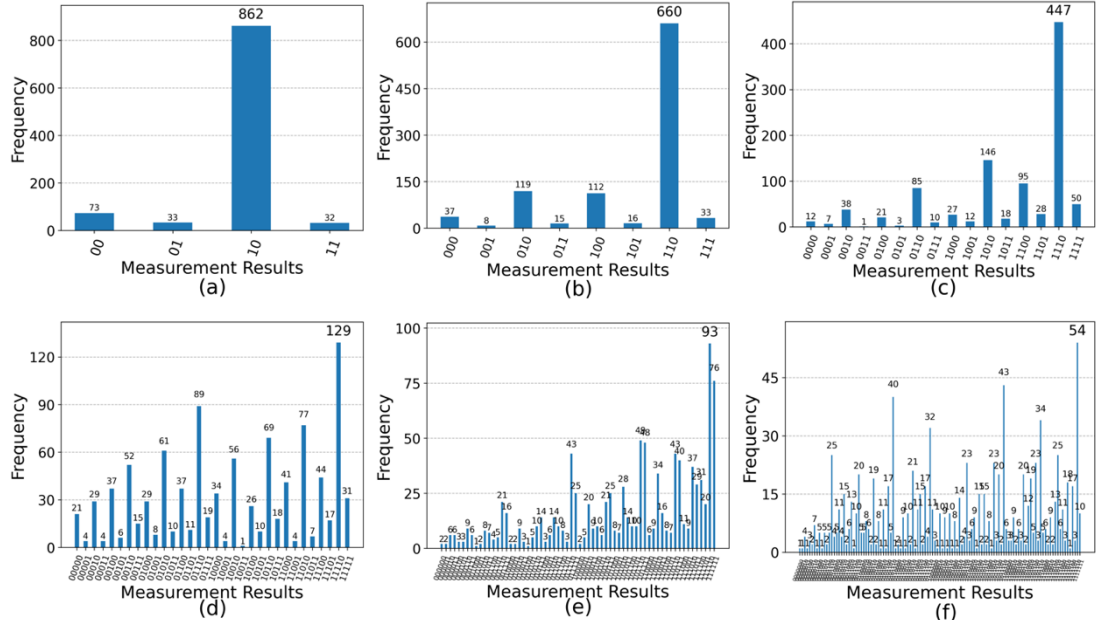


Figure 5.8: QPU results for Toffoli-based Adder

### 5.5.3. QFT-Based Adder Results

Figures 5.9 and 5.10 present the results for QFT-based adder on noisy simulator and IBM QPU respectively, using the same input cases as in Figures 5.7 and 5.8. The noisy simulator consistently yields correct outputs with highest frequency, demonstrating stable performance (Figure 5.9).

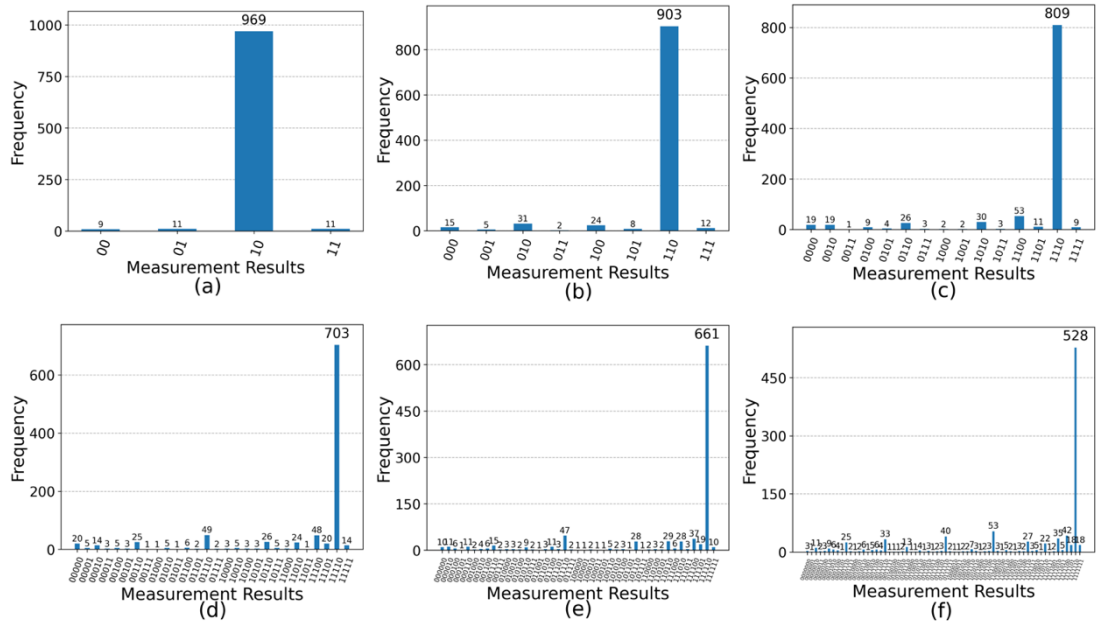


Figure 5.9: Noisy simulator results for QFT-based adder

However, on QPU via `ibm_brisbane`, the noise impact increases with input size, where beyond 2-bit inputs (subfigures (c) to (f) in Figure 5.10), the results become unreliable. These observations are attributed to quantum decoherence and hardware imperfections.

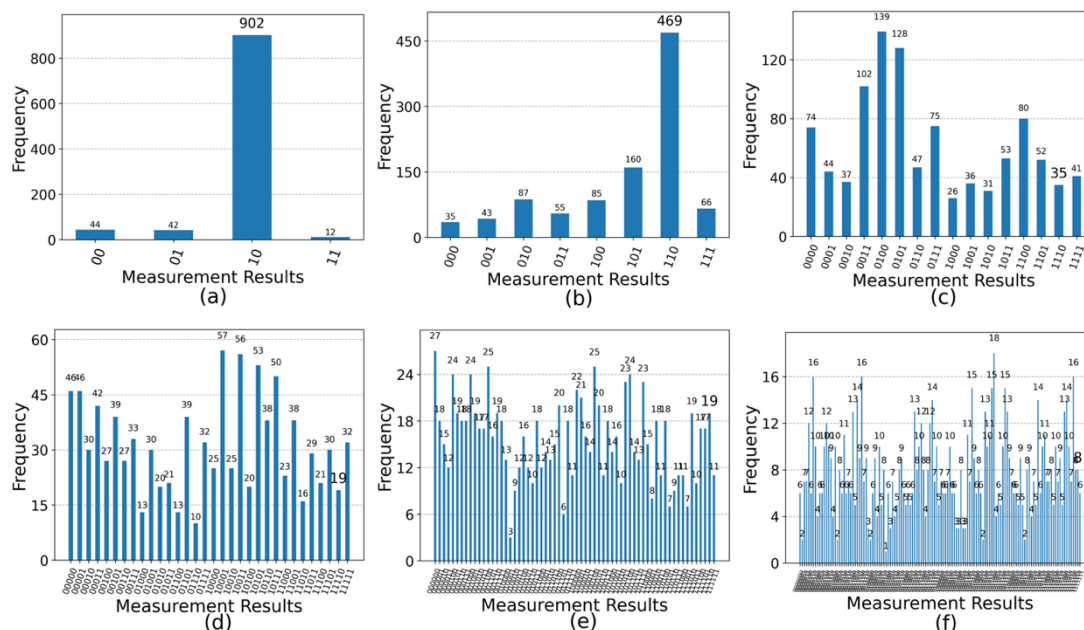


Figure 5.10: OPU results for OFT-based adder

#### 5.5.4. Quantum Comparator Results

Table 5.1 presents the frequency of correct outcomes over 1000 executions of the circuit for quantum comparator on IBM’s noisy simulator and QPU (ibm\_brisbane) for input sizes ranging from 1-bit to 6-bit numbers. Figure 5.11 visualizes accuracy across both the backends, showing a decline on the QPU as input size grows, especially beyond 3 bits, due to quantum noise and hardware limitations.

Table 5.1: Quantum comparator results

Input size(#bits)	1	2	3	4	5	6
Simulator	997	990	988	979	978	953
QPU	985	905	877	539	534	479



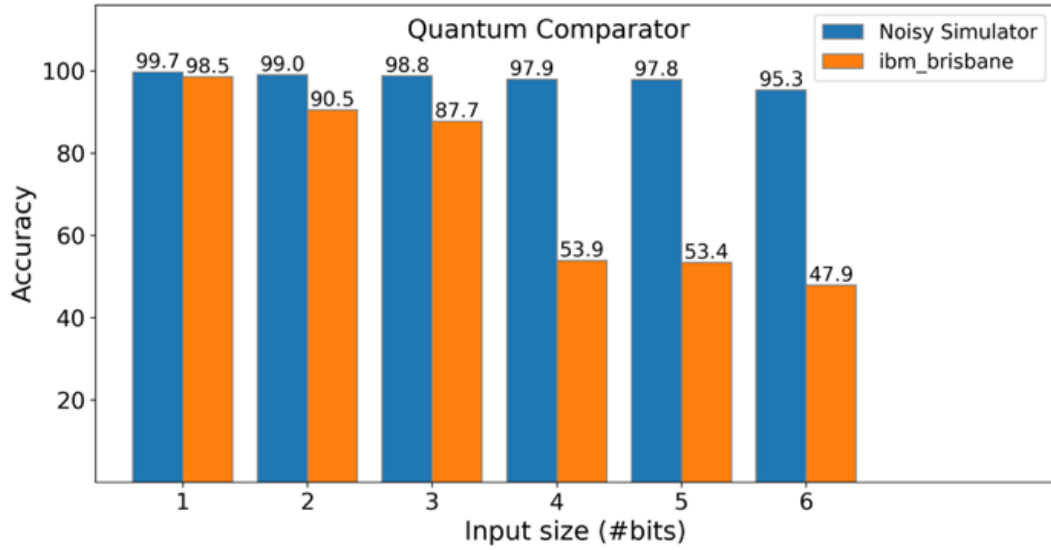


Figure 5.11: Quantum comparator accuracy on simulator and QPU

### 5.6. Comparison of Toffoli and QFT adders

Table 5.2 presents the accuracy of both the adders on the noisy simulator, Eagle processor accessed via *ibm\_brisbane*, and accessed via *ibm\_kyiv*. The observations presented in the table demonstrate that the Toffoli-based adder outperforms the QFT-based adder across all simulations. On QPUs, the Toffoli-based adder shows correct outcomes with higher frequency among all results for input sizes up to 5-bit numbers. In contrast, for QFT-based adder, this trend is observed only up to 2-bit inputs. The bar graph of Figure 5.12 shows the higher accuracy of Toffoli adder.

Table 5.2: Noisy simulator and QPU results for Toffoli and QFT adders

Input size (#bits)	Noisy Simulator		QPU (ibm_brisbane)		QPU (ibm_kyiv)	
	Toffoli adder	QFT adder	Toffoli adder	QFT adder	Toffoli adder	QFT adder
1	976	969	862	902	927	842
2	947	903	660	469	700	539
3	885	809	447	35	370	82
4	843	703	129	19	312	49
5	831	661	93	19	143	15
6	771	528	54	8	62	9

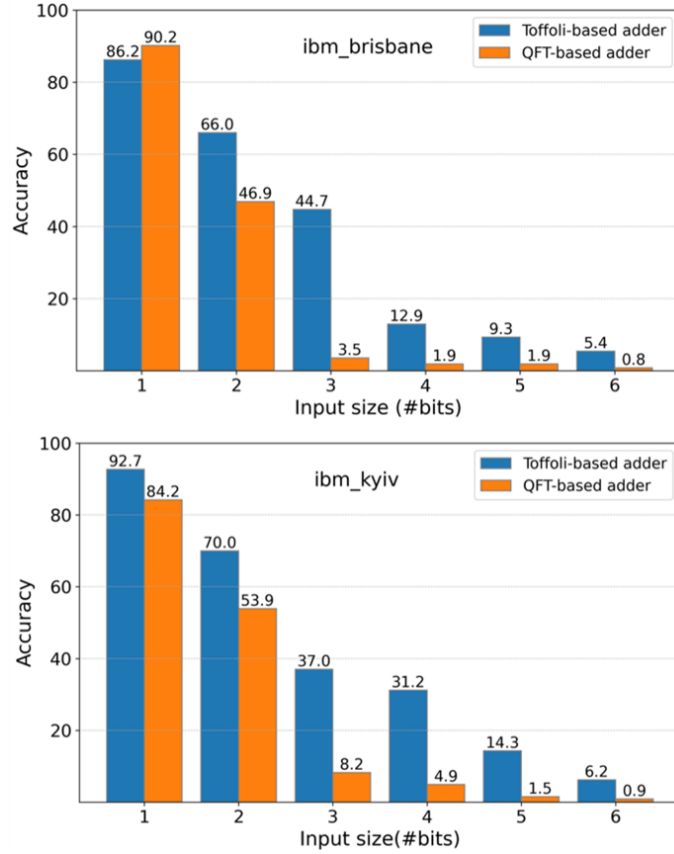


Figure 5.12: Accuracy comparison of Toffoli- and QFT-based adders on QPU

## 5.7. CONCLUSION

The performance of quantum arithmetic operations was evaluated on IBM QPUs, analysing the impact of circuit depth and noise on computational accuracy. Results showed that circuit depths and noise levels increase with input size, resulting in frequent incorrect outcomes. For  $n$ -bit numbers, the Toffoli-based circuit requires  $3n + 1$  qubits, the QFT-based adder requires  $2n + 1$  qubits, and the comparator circuit needs  $2n$  qubits. The Toffoli-based adder achieves lower circuit depth than QFT-based adder, as the latter introduces additional complexity by incorporating the QFT at the beginning and its inverse at the end of computations. Therefore, the Toffoli-based adders demonstrated higher accuracy for input sizes up to 5 bits, while the QFT-based adders maintained accuracy only for inputs up to 2 bits. Since increased circuit depth leads to execution times longer than the qubit coherence times and hence introducing errors. These findings highlight scalability challenges in quantum arithmetic on current quantum hardware and the need for noise-mitigation strategies to enhance computational reliability on quantum computers.

## CHAPTER 6

### OPTIMIZATION AND PARALLEL IMPLEMENTATION OF NEQR

#### 6.1. Quantum Image Representation

Quantum Image Representation (QIR) relies on fundamental principles of superposition and entanglement. NEQR (Novel Enhanced Quantum Representation) is a method to represent digital images on a quantum computer. In NEQR, a binary representation of pixel positions and intensities is stored in the computational basis states of quantum bits, enabling efficient image processing operations. This representation is suitable for reversible operations and is useful for tasks like edge detection, encryption, and watermarking. Table 6.1 summarizes the quantum gates for NEQR implementation. It presents the gate types, symbols, Qiskit methods, and their unitary matrices, essential for encoding classical images into quantum images.

Table 6.1: Quantum gates for NEQR-based image representation

Name	Type	Circuit Symbol	Qiskit method	Unitary
Hadamard (H)	Single-qubit	$q \text{ --- } \boxed{H} \text{ ---}$	qc.h(q)	$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
NOT (X)	Single-qubit	$q \text{ --- } \boxed{X} \text{ ---}$	qc.x(q)	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
CNOT (CX)	Two-qubit	$q_0 \text{ --- } \bullet \text{ ---} \oplus \text{ ---} q_1$	qc.cx(q0, q1)	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
TOFFOLI (CCX)	Three-qubit	$q_0 \text{ --- } \bullet \text{ ---} \bullet \text{ ---} \oplus \text{ ---} q_2$ $q_1 \text{ --- } \bullet \text{ ---}$	qc.ccx(q0, q1, q2)	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$
Measure	Variable-size	$q \text{ --- } \boxed{\text{Measure}} \text{ ---} \begin{matrix} 1 \\ 0 \end{matrix}$ $c$	qc.measure(q, c)	(Collapses quantum state to classical bit)

An image is mathematically described as a function  $f(X, Y)$ , where  $X$  and  $Y$  are spatial coordinates. The value of function at each coordinate  $(X, Y)$ , represents image intensity, commonly denoted as  $I(X, Y)$ . The NEQR method uses two qubit sequences for encoding digital image pixels into quantum states, one for encoding the spatial coordinates or pixel positions, and the other for encoding the intensity values. Both the sequences are encoded in computational basis states.

The number of qubits is determined by both the image size and the range of gray levels present in the image. For instance, for an image of size  $2^n \times 2^n$ ,  $2n$  qubits are allocated for pixel position encoding, with  $n$  for rows and  $n$  for columns. If the image contains  $2^m$  gray levels, then  $m$  qubits are required for intensity encoding, resulting in a total of  $2n + m$  qubits. To preserve the pixel-to-intensity mapping the two qubit sequences encoding pixel positions and intensity values are entangled.

Figure 6.1 displays a  $2 \times 2$  grayscale image, where each pixel is annotated with its intensity in both decimal and 8-bit binary formats, and position in binary format. The corresponding quantum state representation, denoted as  $|QI\rangle$  is also presented in the figure. The representation encodes the image using position and intensity qubits using the NEQR model.

$50_{10}$ $00110010_2$ pos = 00	$100_{10}$ $01100100_2$ pos = 01
$150_{10}$ $10010110_2$ pos = 10	$200_{10}$ $11001000_2$ pos = 11

$$\begin{aligned}
|QI\rangle &= \frac{1}{2} (|50_{10}\rangle \otimes |00\rangle + |100_{10}\rangle \otimes |01\rangle \\
&\quad + |150_{10}\rangle \otimes |10\rangle + |200_{10}\rangle \otimes |11\rangle) \\
&= \frac{1}{2} (|00110010_2\rangle \otimes |00\rangle + |01100100_2\rangle \otimes |01\rangle \\
&\quad + |10010110_2\rangle \otimes |10\rangle + |11001000_2\rangle \otimes |11\rangle)
\end{aligned}$$

Figure 6.1: A  $2 \times 2$  image and its quantum representation

The quantum state of an image of size  $2^n \times 2^n$  with 256 intensity levels is given by equation (6.1):

$$\begin{aligned}
|QI\rangle &= \frac{1}{2^n} \sum_{X=0}^{2^n-1} \sum_{Y=0}^{2^n-1} |I(X, Y)\rangle |XY\rangle \\
&= \frac{1}{2^n} \sum_{X=0}^{2^n-1} \sum_{Y=0}^{2^n-1} \bigotimes_{k=0}^7 |I_{XY}^k\rangle |XY\rangle
\end{aligned} \tag{6.1}$$

## 6.2. Algorithm for Circuit Development

Algorithm presented in Figure 6.2 outlines the procedure for constructing a quantum circuit that transforms a classical image into its quantum counterpart using the NEQR method on the IBM Qiskit framework. The algorithm provides a step-by-step method for mapping classical image pixel data onto quantum registers, namely position register and intensity register. The position register is initialized into a superposition state, allowing simultaneous representation of all the pixel locations. The position register is then entangled with the intensity register, which encodes pixel intensity values into quantum states. The entanglement links each pixel's position with its corresponding intensity, enabling a correct quantum image representation and facilitating efficient operations on the quantum image.

<b>Algorithm:</b> Quantum Image Representation using NEQR		
<b>Input:</b>	<i>Image img</i> with size $2^n \times 2^n$	
<b>Output:</b>	<i>neqr_circuit</i>	# Quantum circuit
<b>Step 1:</b> Create two quantum registers for encoding pixel position and intensity		
	<i>pos_qr</i> = <i>QuantumRegister</i> ( $2n$ , 'pos')	# Position register
	<i>intensity_qr</i> = <i>QuantumRegister</i> (8, 'I')	# Intensity register
The registers are initialized with computational basis state $ 0\rangle$ .		
<b>Step 2:</b> Create a quantum circuit using the quantum registers:		
	<i>neqr_circuit</i> = <i>QuantumCircuit</i> ( <i>pos_qr</i> , <i>intensity_qr</i> )	
<b>Step 3:</b> Create superposition of pixel positions by applying Hadamard gate to position register		
	<i>neqr_circuit.h</i> ( <i>pos_qr</i> )	#Applies H to all qubits of <i>pos_qr</i>
<b>Step 4:</b> Create Entanglement of pixel positions with the corresponding intensities:		
	<i>for r in range</i> ( $2^n$ ):	# Traverse image rows
	<i>for c in range</i> ( $2^n$ ):	# Traverse image columns
	<i>pos_bin</i> = <i>binary</i> ( $r \times 2^n + c$ )	# Convert 2D to 1D and get its binary
	<i>intensity</i> = <i>img</i> ( <i>r</i> , <i>c</i> )	# Retrieve intensity at pos ( <i>r</i> , <i>c</i> )
	<i>intensity_bin</i> = <i>binary</i> ( <i>intensity</i> )	# Convert intensity to binary
	<i>for b in range</i> (8):	# Traverse intensity bits, if it is 1
	<i>if intensity_bin</i> [ <i>b</i> ] == 1:	# apply MCX to intensity qubit
	<i>neqr_circuit.MCX</i> ( <i>pos_qr</i> , <i>intensity_qr</i> [ <i>b</i> ])	# controlled by position qubit
<b>Step 5:</b> Return the quantum circuit <i>neqr_circuit</i>		

Figure 6.2: Algorithm for quantum image representation using NEQR

## 6.3. Optimization of MCX Gate Decomposition

In NEQR, the position register functions as the control, and intensity register as the target or working register. Entanglement is established from control register to target

register, binding pixel positions to their corresponding intensity values. This is implemented using multi-controlled X (MCX) gates, which require decomposition into elementary gates — specifically Toffoli (CCNOT) gates — for compatibility with current quantum hardware. Therefore, a  $k$ -controlled X ( $k$ -CNOT) gate, where  $k$  denotes the number of qubits in position register, is decomposed into multiple Toffoli gates to enable practical realization of the circuit on existing quantum processors.

### 6.3.1. Decomposition Algorithm

The decomposition method proposed in [73] requires  $4k - 8$  Toffoli gates with a circuit depth of  $4k - 8$ . The method is optimized reducing the number of Toffoli gates to  $2k - 3$ , achieving a circuit depth of  $2k - 3$ . In quantum computing, the circuit depth serves a role analogous to time complexity in classical computing, hence a reduction in circuit depth enhances the computational efficiency. The algorithm for decomposition is presented by Figure 6.3.

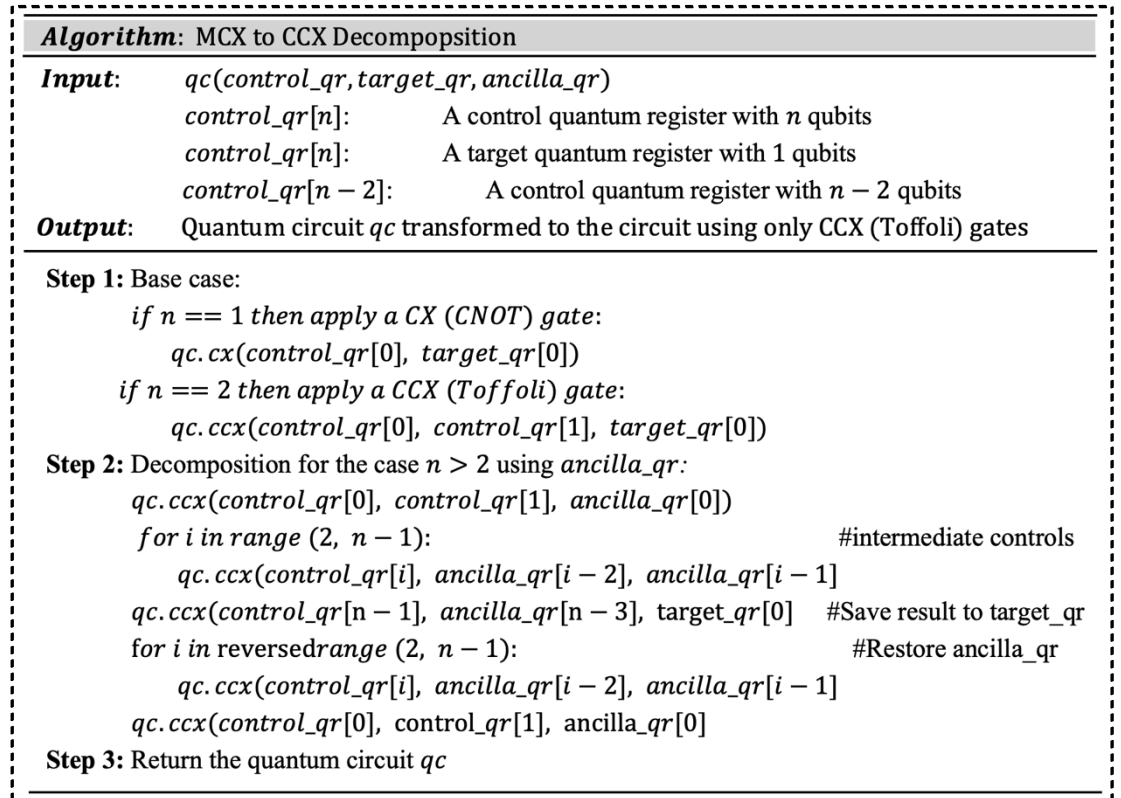


Figure 6.3: Algorithm for MCX to CCX decomposition

Illustrative examples of 3-controlled and 4-controlled X gate decompositions using the algorithm are presented by Figure 6.4(a) and Figure 6.4(b) , respectively.

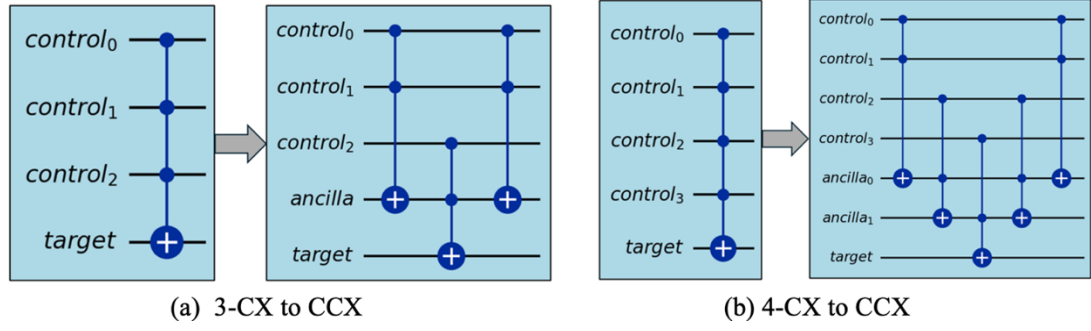


Figure 6.4: Optimized decomposition of 3-CX and 4-CX Gates

### 6.3.2. Comparative Analysis

The decomposition of a 4-controlled X (4-CX) gate into multiple Toffoli (CCX) gates, as described in [73], has been redrawn and is presented in Figure 6.5 (a) for comparison. The corresponding optimized version of this decomposed circuit is shown in Figure 6.5 (b). A comparison of these two circuits demonstrates a significant reduction in circuit depth, with the original circuit having a depth of 8, whereas the optimized circuit having a reduced depth of 5.

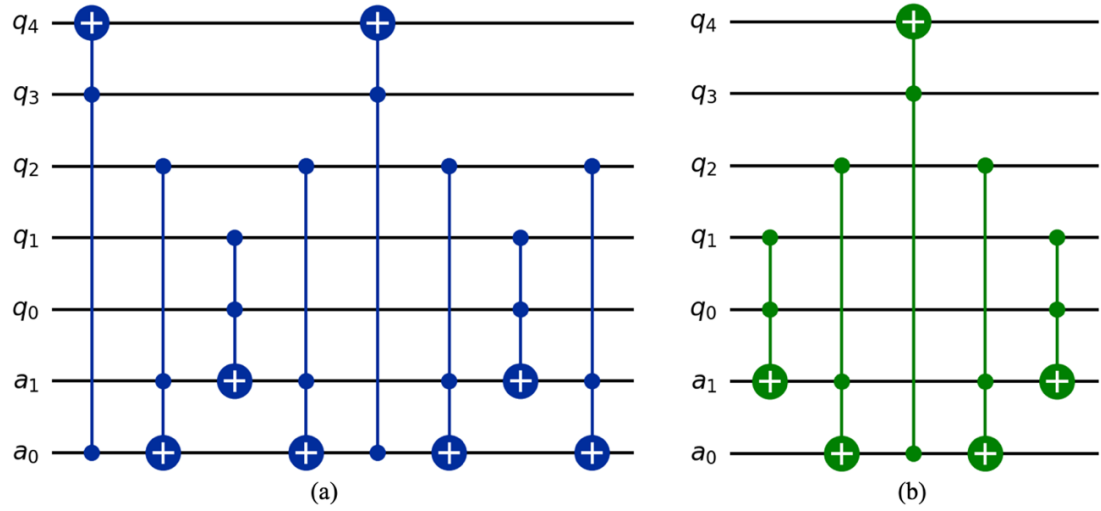


Figure 6.5: (a) Original circuit (b) Optimized circuit

Figure 6.6 presents the comparative analysis of circuit depths between the original and optimized methods using line graphs. The X-axis denotes the number of control qubits, and the Y-axis represents the circuit depth after decomposition. The blue line illustrates the depth associated with the original method, and the green line depicts the depth associated with optimized method. As the number of control qubits increases, the optimized method consistently achieves a lower circuit depth, highlighting its

scalability and efficiency. This reduction in depth contributes to faster execution and improved resilience against quantum decoherence, thereby enhancing overall computational reliability.

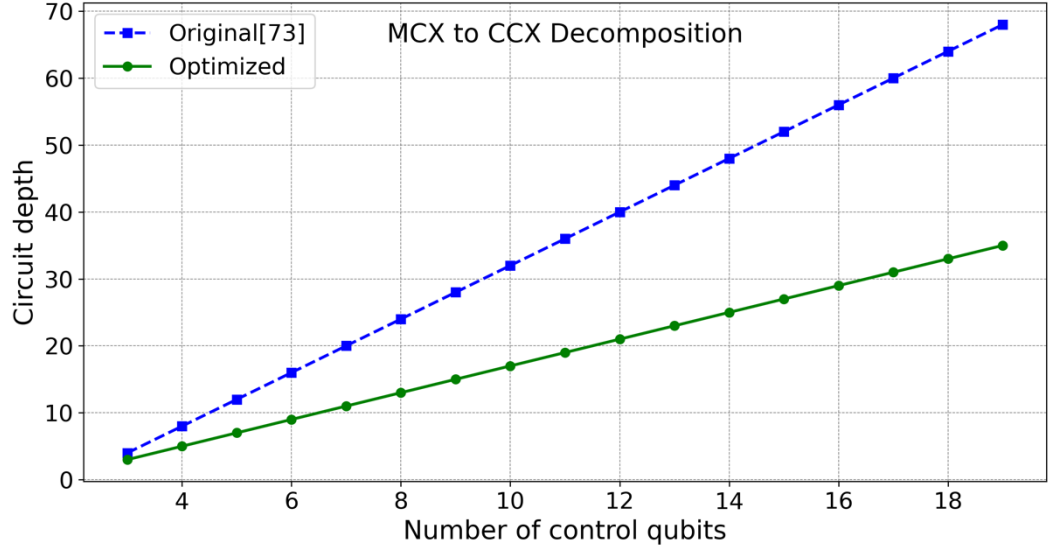


Figure 6.6: Comparison of circuit depths in MCX decomposition

#### 6.4. NEQR Circuit for a $2 \times 2$ Image

Figure 6.7 shows the optimized NEQR circuit for quantum image representation of the image of Figure 6.1. Since the image size is  $2 \times 2$ , the position register utilizes 2 qubits to encode the spatial coordinates. The intensity values range from 0 to 255, requiring 8 qubits for the intensity register.

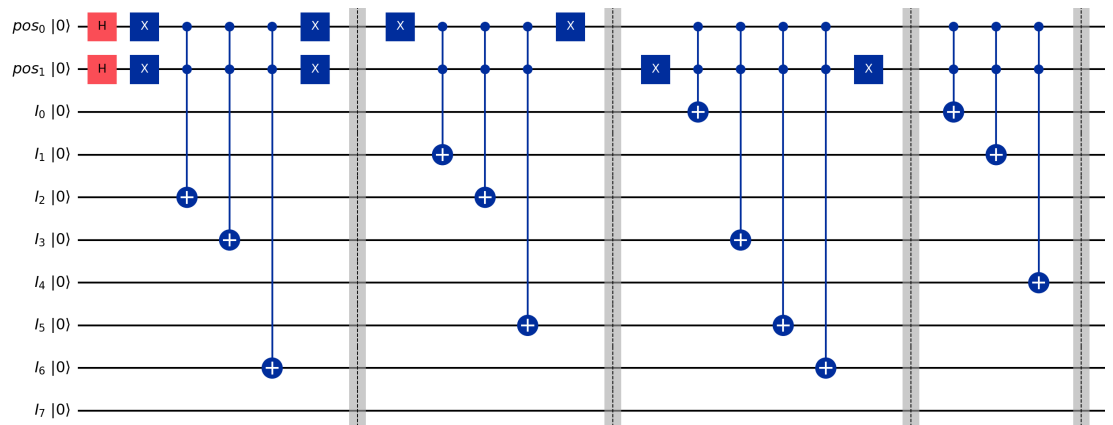


Figure 6.7: NEQR circuit for image of Figure 6.1

Therefore, a total of 10 qubits are required to encode the image into quantum form. The circuit depth, which varies based on image content, is 20 for this particular



example. The encoding of each of the four pixels is delineated by barriers shown by gray lines for improving understandability of the intensity encoding process.

## 6.5. Parallel Bit-Plane NEQR

The Parallel Bit-Plane NEQR is an extension of the sequential NEQR representation to enhance the efficiency of encoding classical images into quantum states. A bit-plane refers to a binary layer extracted from the binary representation of pixel intensities, where each intensity, typically represented using 8 bits, contributes one bit to each of the eight bit-planes. The parallel model decomposes the image into its constituent bit-planes, with each plane independently and concurrently encoded using a dedicated quantum circuit. Parallelizing the encoding process reduces the circuit depth compared to the sequential NEQR approach. The reduction in depth not only improves the execution speed on quantum hardware but also minimizes quantum noise due to shorter coherence windows. The use of separate quantum circuits for each bit-plane enhances scalability and allows more flexible manipulation of image features at different intensity and significance levels, thereby supporting advanced quantum image processing such as filtering, compression, and segmentation.

### 6.5.1. Quantum Representation

For quantum representation of an image of size  $2 \times 2$ , pixel positions  $(X, Y)$  require 2 qubits, representing four position states as below:

$$|XY\rangle = |00\rangle, |01\rangle, |10\rangle, |11\rangle$$

The intensity of these pixels is represented by 8 qubits, corresponding to 8-bit binary value of the intensity. In the parallel bit-plane NEQR, each of these intensity bits is entangled with a dedicated position register, allowing for parallel construction of quantum circuit for each bit-plane. The composite quantum state for the entire  $2 \times 2$  image is expressed by equation (6.2):

$$= \bigotimes_{k=0}^7 \left( \frac{1}{2} \sum_{X=0}^1 \sum_{Y=0}^1 |I_{XY}^k\rangle |XY\rangle_k \right) \quad (6.2)$$

Expanding the formulation to represent individual pixels along with their corresponding 8-bit intensity values, the quantum state can be expressed by equation (6.3) as follows:

$$= \frac{1}{2} \left( \bigotimes_{k=0}^7 |I_{XY}^k\rangle |00\rangle_k + \bigotimes_{k=0}^7 |I_{XY}^k\rangle |01\rangle_k + \bigotimes_{k=0}^7 |I_{XY}^k\rangle |10\rangle_k + \bigotimes_{k=0}^7 |I_{XY}^k\rangle |11\rangle_k \right) \quad (6.3)$$

where

- $|XY\rangle_k$  represents the 2-qubit pixel encoding for the  $k^{th}$  intensity qubit
- $|I_{XY}^k\rangle$  is the  $k^{th}$  qubit of the intensity value at pixel  $(X, Y)$

Thus, the quantum state for an individual bit-plane for a  $2^n \times 2^n$  image can be defined as equation (6.4):

$$|QI^k\rangle = \frac{1}{2^n} \sum_{X=0}^{2^n-1} \sum_{Y=0}^{2^n-1} |I_{XY}^k\rangle |XY\rangle_k \quad (6.4)$$

The full quantum state, representing all bit-planes of the image, is expressed by equation (6.5):

$$\begin{aligned} |QI_{bit-plane}\rangle &= \bigotimes_{k=0}^7 |QI^k\rangle \\ |QI_{bit-plane}\rangle &= \bigotimes_{k=0}^7 \left( \frac{1}{2^n} \sum_{X=0}^{2^n-1} \sum_{Y=0}^{2^n-1} |I_{XY}^k\rangle |XY\rangle_k \right) \end{aligned} \quad (6.5)$$

### 6.5.2. Circuit for a $2 \times 2$ Image

Figure 6.8 presents the quantum circuit for the  $2 \times 2$  image depicted in Figure 6.1, using the parallel bit-plane NEQR. Pixel positions are encoded using a 2-qubit register. As the image contains 8 bit-planes, the circuit requires 8 separate position registers – one for each bit-plane – resulting in a total of 16 position qubits. Including the 8 qubits required for intensity encoding, the complete quantum image representation requires 24 qubits. Compared to sequential NEQR, the circuit depth is reduced from 20 to 11, demonstrating improved efficiency. The encoding of individual pixels is delineated by barriers (dotted gray lines) in the circuit diagram shown in the Figure 6.8.

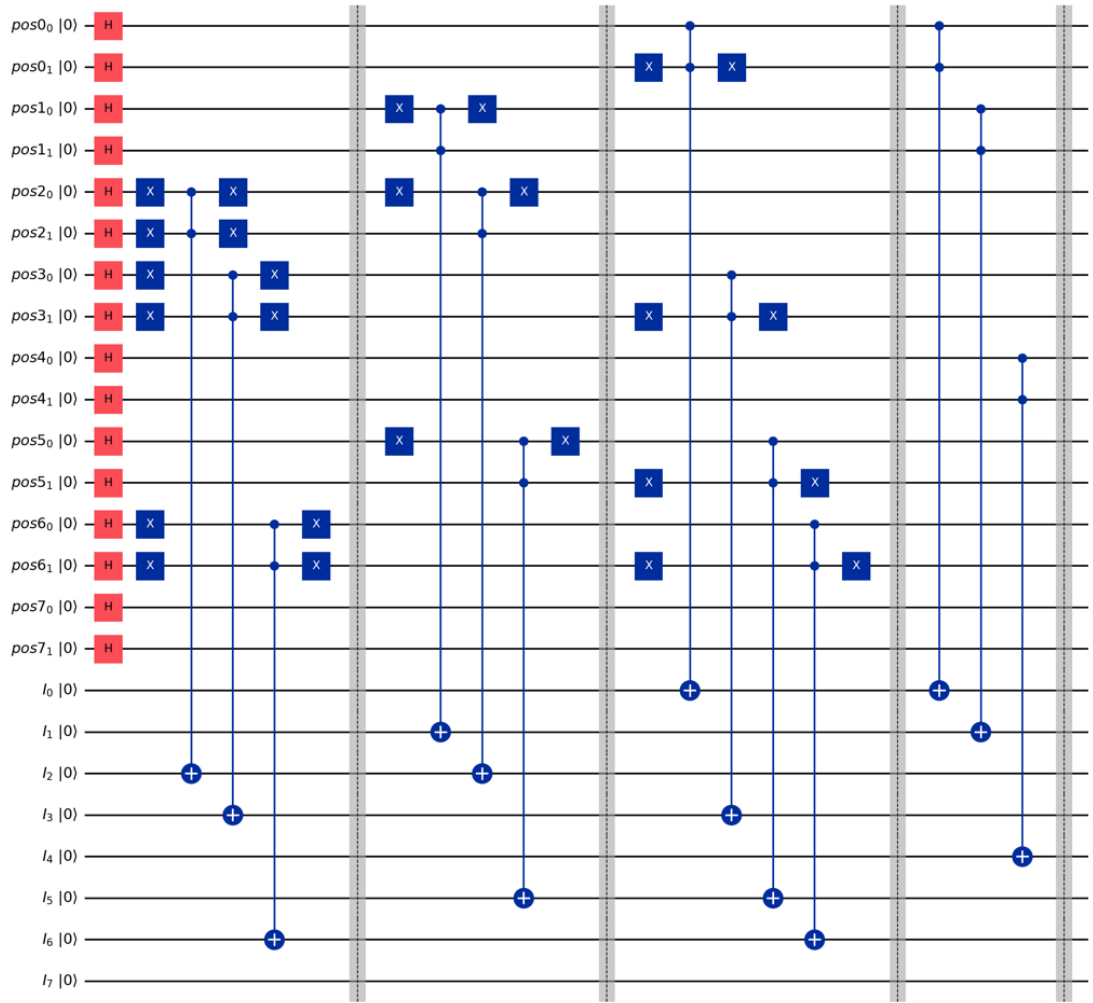


Figure 6.8: Parallel Bit-Plane NEQR circuit for image of Figure 6.1

### 6.5.3. Circuit Complexity Analysis

In quantum computing, circuit depth is analogous to time complexity in classical computing, while circuit width (the number of qubits used) is analogous to space complexity. The original NEQR has a low space complexity of only  $2n + m$  qubits for an image of size  $2^n \times 2^n$  with  $2^m$  intensity levels. The time complexity or circuit depth of NEQR grows as  $O(8 * 2^{2n})$ . Given that circuit depth has a direct impact on quantum decoherence and the fidelity of computation, minimizing it is essential for practical implementations on NISQ devices.

The parallel Bit-Plane NEQR reduces the circuit depth (i.e., time complexity) over sequential NEQR by enabling simultaneous encoding and processing of bit-planes. Although it requires higher number of qubits,  $16n + m$ , compared to the sequential

NEQR. This trade-off aligns with current trends in quantum hardware development, where increasing the number of qubits is generally more feasible than extending coherence times to support deeper circuits. Recent advances in superconducting and ion-trap technologies have enabled scalable architectures with hundreds of qubits, yet coherence time remains a critical bottleneck due to environmental noise and physical limitations of quantum systems.

## 6.6. Experimental Results

### 6.6.1. MNIST Dataset

The MNIST dataset [88], shown by Figure 6.9, comprises thousands of  $28 \times 28$  images representing handwritten digits from 0 to 9. It serves as a standard benchmark for evaluating and training models in image classification and pattern recognition tasks. This dataset is selected due to its small size ( $28 \times 28$ ) and grayscale nature, requiring less number of qubits and lower circuit depths enabling efficient encoding. Although larger images can also be encoded by dividing the image into smaller subblocks and then encoding each block separately. The MNIST dataset can be simulated without such division.

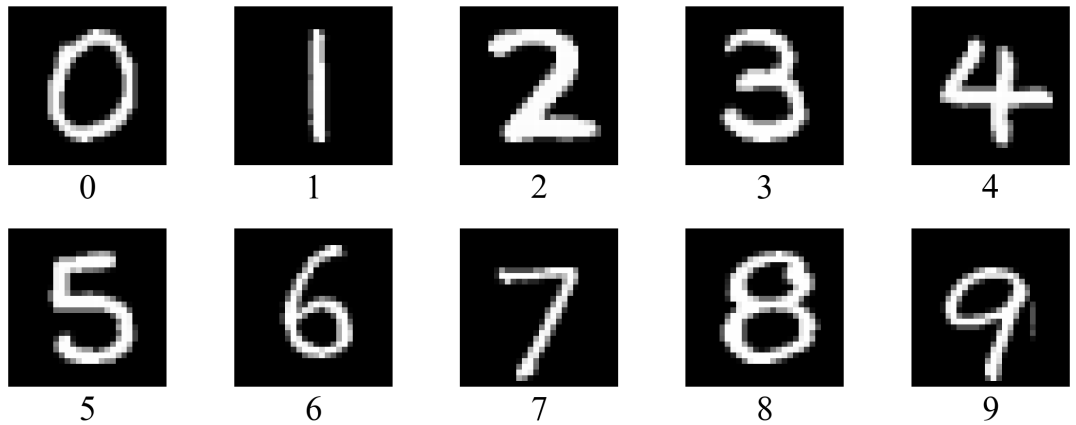


Figure 6.9: Sample MNIST dataset

### 6.6.2. Basic Operations on NEQR Images

The optimized NEQR method was implemented and applied on original MNIST dataset. The original images in this dataset were encoded and image operations were applied to the encoded images, with an exponential speedup over classical image processing. Basic image operations such as image negation, rotation, and intensity

superposition are applied on the images as illustrated by Figure 6.10. For image negation, a NOT gate (X) is applied to the intensity qubits, inverting pixel intensity values. For image rotation, the corresponding qubits in the row and column quantum registers are swapped using SWAP gate, followed by the application of NOT gate to the column register. Similarly, for intensity superposition, Hadamard gate (H) is applied to the intensity register, obfuscating the image, and preventing its direct retrieval.

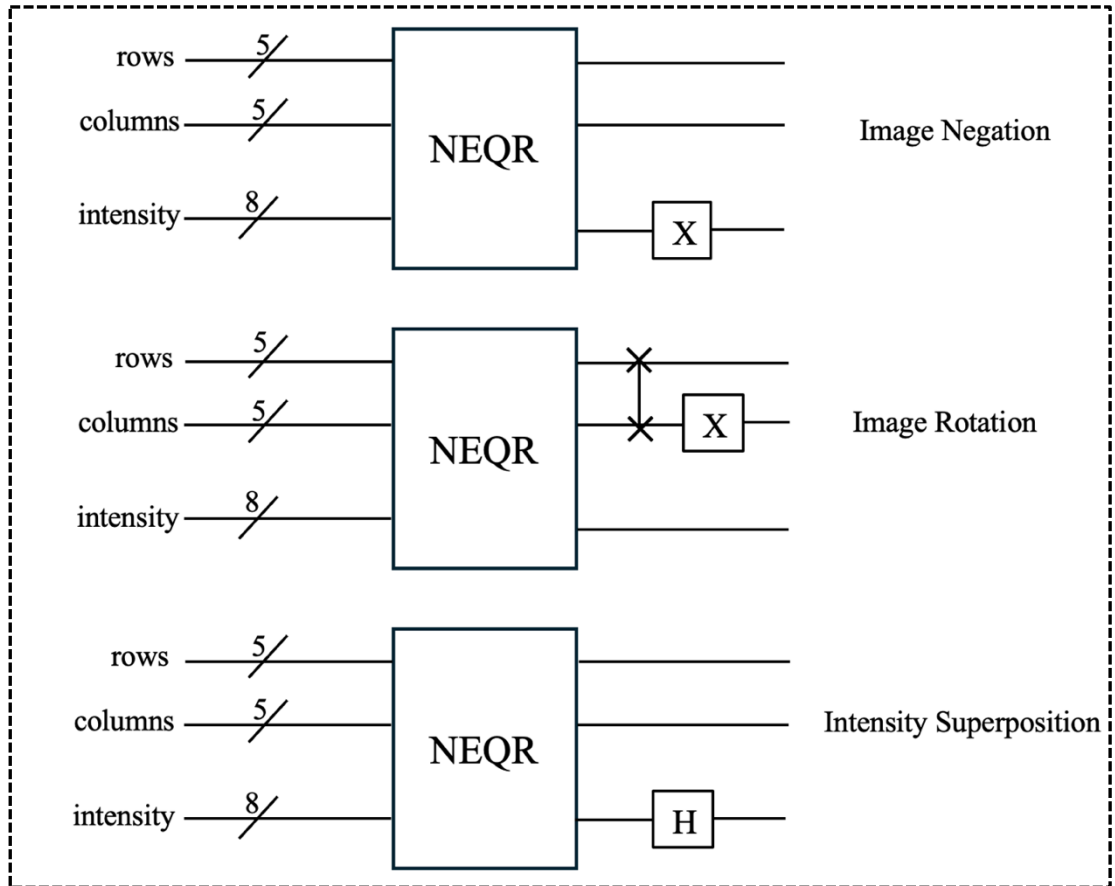


Figure 6.10: Circuits for basic operations on QIR of MNIST images

Quantum images must be measured to observe the results of processing operations, as they exist in superposition and require collapse into a classical form. This step is crucial for extracting meaningful information and visualizing transformations.

The measured outcomes, shown in Figure 6.11, illustrate the effects of operations like image negation, rotation, and intensity superposition. In classical computing, the time complexity for image processing operations on an  $2^n \times 2^n$  image is  $O(2^n \times 2^n)$ . On quantum computers, this complexity is reduced to  $O(1)$  because all pixels exist in

superposition, allowing simultaneous manipulation of all intensity values in a single operation, regardless of image size.

These operations on quantum images were performed using an ideal quantum simulator provided by IBM Qiskit's *AerSimulator* backend. The simulator was initialized with the '*matrix\_product\_state*' method, which enables efficient simulation of quantum circuits up to 63 qubits, when the circuit exhibits limited entanglement. The backend is as follows:

```
from qiskit_aer import AerSimulator
backend = AerSimulator(method='matrix_product_state')
```

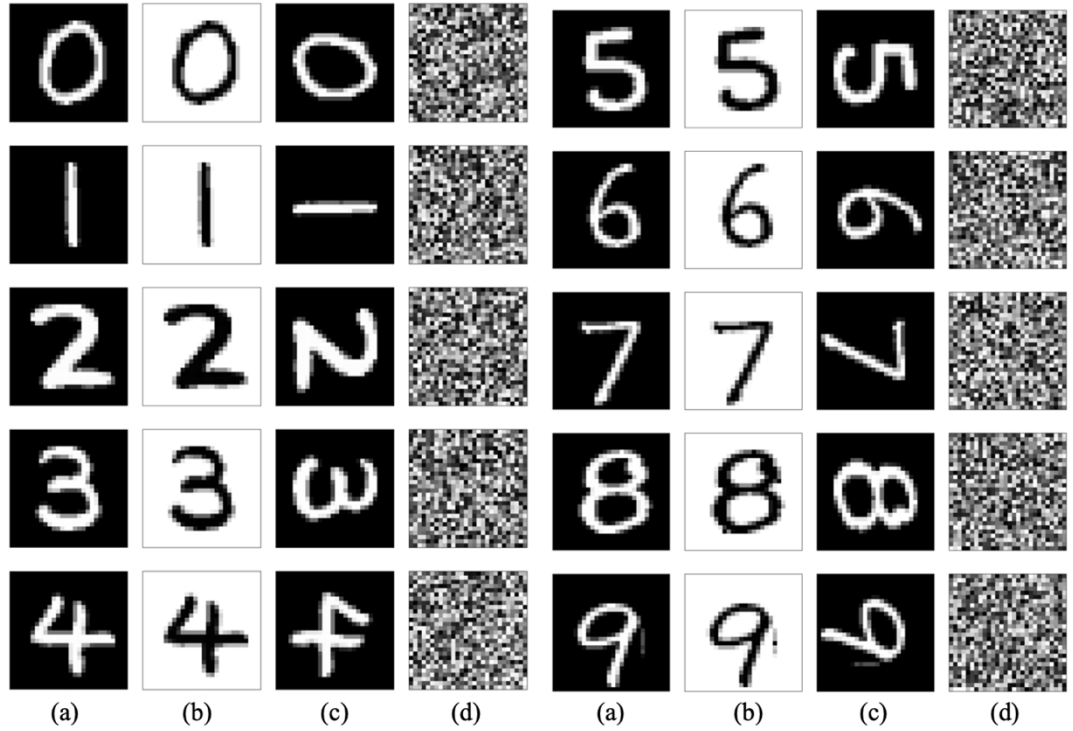


Figure 6.11: (a) Original (b) Negation (c) Rotation (d) Superposition

### 6.6.3. Comparison of Sequential and Parallel NEQR

Both sequential and parallel bit-plane NEQR were implemented and simulated on MNIST dataset – both in ideal (noiseless) and noisy environments – and executed on quantum hardware. The images were down sampled to smaller sizes such as  $16 \times 16$ ,  $8 \times 8$  and  $4 \times 4$  for conducting performance comparisons with different resolutions. Abstract circuit depths and simulation times were calculated using an ideal simulator,

transpiled circuit depths were obtained on a noisy simulator. Execution times and circuit depths were measured on the IBM Eagle quantum processor.

#### ▪ **Ideal Simulation**

Ideal simulation isolates algorithmic performance from hardware-induced noise, enabling an evaluation of theoretical designs. Its primary objective is to verify implementation correctness and facilitate algorithmic analysis by observing theoretical principles in a controlled, noise-free environment.

Circuits for both the original sequential NEQR and the Parallel Bit-Plane NEQR method are developed and simulated on MNIST dataset to compare their circuit depths. The correctness of the implementation is validated by quantum encoding of an image, followed by quantum measurement and classical post-processing, for lossless reconstruction. To ensure reliable reconstruction, the number of shots per simulation should exceed the total number of pixels. In this study for ideal simulations, a shot count equal to eight times the image size was used to achieve lossless reconstruction.

Post-processing involves extracting pixel intensity values from the measured intensity qubits and pixel position values from the position qubits. The image is then reconstructed by mapping the extracted intensities to their corresponding positions in the classical image array. Under ideal simulation conditions, a correctly implemented circuit yields a MSE of zero between original and reconstructed images. It was observed that both the sequential and parallel methods achieved an MSE of zero for images with resolutions of  $4 \times 4$ ,  $8 \times 8$ , and  $16 \times 16$ , thereby confirming the correctness of the implementations.

Table 6.2 reports the circuit depths for these implementations for images with resolutions of  $4 \times 4$ ,  $8 \times 8$  and  $16 \times 16$ , with comparative analysis shown by Figure 6.12. The results show that parallel NEQR achieves lower circuit depths across all images when compared to sequential NEQR. The gap between the two models widens with increase in image size.

Table 6.2: Circuit depths for sequential and parallel NEQR for MNIST digits

MNSIT Digits	4 × 4 Images		8 × 8 Images		16 × 16 Images	
	Sequential	Parallel	Sequential	Parallel	Sequential	Parallel
0	215	89	1404	435	5937	1842
1	188	54	1020	349	3456	1009
2	330	99	1361	404	7835	1983
3	260	92	1470	437	6490	1763
4	235	75	1398	464	5765	1646
5	315	87	1279	433	6271	1759
6	215	73	1182	415	4696	1434
7	225	82	1101	380	4467	1342
8	315	99	1457	433	6921	1939
9	220	75	1226	413	4945	1505

Table 6.3 reports the simulation times for the MNIST images with resolutions of  $4 \times 4$ ,  $8 \times 8$  and  $16 \times 16$  with 128, 512, 2048 shots respectively.

Table 6.3: Simulation time for sequential and parallel NEQR on ideal simulator

MNSIT Digits	4 × 4 (128 shots) (milliseconds)		8 × 8 (512 shots) (milliseconds)		16 × 16 (2048 shots) (seconds)	
	Sequential	Parallel	Sequential	Parallel	Sequential	Parallel
0	73	85	796	394	24.10	2.17
1	56	90	394	343	04.11	1.60
2	90	93	634	377	33.80	2.37
3	79	99	756	399	26.70	2.15
4	62	88	690	381	18.40	2.11
5	75	97	672	336	24.50	2.22
6	72	99	619	339	13.80	1.95
7	50	86	507	355	10.10	2.06
8	84	94	957	393	33.40	2.19
9	62	89	671	369	14.50	2.01



The results demonstrate that the performance gain is modest for low-resolution images (e.g., at  $4 \times 4$ , simulation times are slightly higher for the parallel model due to parallel overhead), the benefits become significant with image size. For  $8 \times 8$  images, the parallel model consistently outperforms the sequential counterpart, reducing simulation times by over 40–50% on average. Most notably, for  $16 \times 16$  images, the parallel approach achieves more than a tenfold reduction in simulation time, lowering execution from over 20 seconds in the sequential model to under 2.5 seconds.

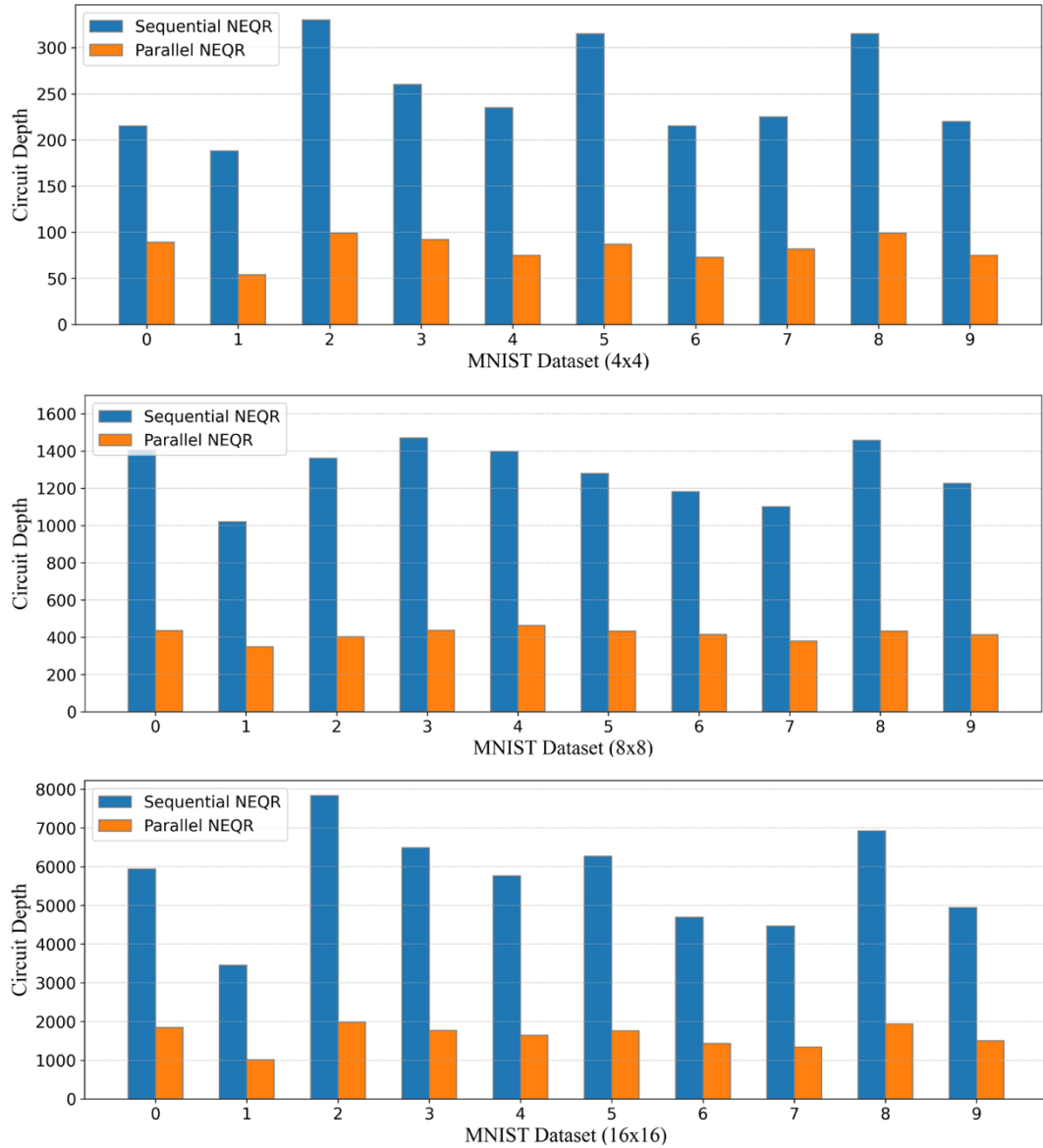


Figure 6.12: Circuit depth comparison for ideal simulation

### ▪ Noisy Simulation

For noisy simulation, the implementations were simulated using IBM Qiskit's Fake127QPulseV1 backend, which is part of the FakeBackendV2 series. It emulates the noise of a 127-qubit IBM quantum processor. The circuits were transpiled to match the backend's qubit layout and error model, using a fixed seed (42) to ensure reproducibility. Transpilation, similar to classical compilation, increases circuit depth. The final simulations were executed using the AerSimulator:

```
from qiskit_aer import AerSimulator
from qiskit.providers.fake_provider import Fake127QPulseV1
fake_backend = Fake127QPulseV1()
backend = AerSimulator.from_backend(fake_backend)
qc_transpiled = transpile(neqr_circuit, backend, seed_transpiler=42)
depth=qc_transpiled.depth()
```

Table 6.4 reports the circuit depths on noisy simulator for both sequential and parallel NEQR implementations, with circuit depths shown by bar graphs in Figure 6.13.

Table 6.4: Transpiled circuit depths for sequential and parallel NEQR on noisy simulator

MNSIT Digits	4 × 4 Images		8 × 8 Images		16 × 16 Images	
	Sequential	Parallel	Sequential	Parallel	Sequential	Parallel
0	3508	1517	28079	10230	108401	39467
1	3005	1353	17623	7505	62033	25034
2	5554	1824	24138	9876	145050	41967
3	4249	1930	25482	11699	115305	43963
4	3913	1496	24176	10635	105477	40473
5	5334	1615	22673	9795	113426	42025
6	3523	1472	20770	8223	84302	33735
7	3688	1283	18547	9083	78647	31721
8	5312	1742	26133	10640	124394	44304
9	3413	1470	21673	8964	89627	38982

A comparison of the transpiled circuit depths for sequential and parallel NEQR implementations demonstrates that the parallel Bit-Plane NEQR model achieves significantly lower circuit depths. On average, the parallel model reduces the circuit depth by approximately 50%–70%, with the improvement becoming more substantial as image resolution increases.

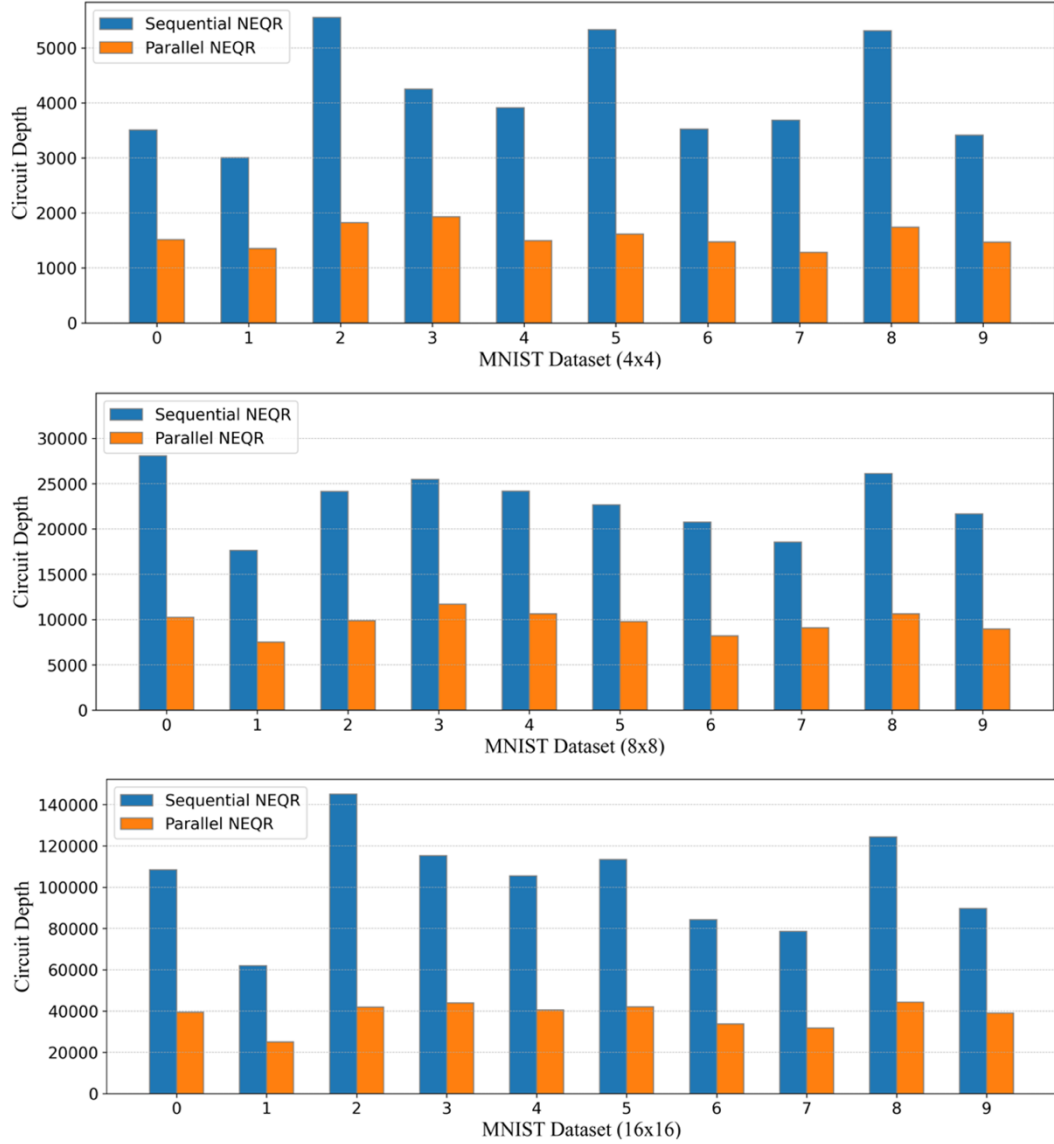


Figure 6.13: Circuit depth comparison for noisy simulation

#### ▪ Eagle Processor Results

To compare the performance of both the sequential and parallel models on quantum hardware, the implementations were executed on Eagle processor via IBM Quantum platform. The QiskitRuntimeService was used to interface with IBM Quantum

services using API token provided during user registration. The backend “ibm\_brisbane” was selected for circuit execution. The python code for backend initialization is as follows:

```
from qiskit_ibm_runtime import QiskitRuntimeService
service=QiskitRuntimeService(channel="ibm_quantum", token="API token")
backend = service.backend("ibm_brisbane")
```

The quantum circuit was transpiled to the optimized circuit compatible to the backend:

```
qc_transpiled=transpile(qc, backend, optimization_level = 3)
depth=qc_transpiled.depth()
```

The transpiled circuit was submitted to the backend using the SamplerV2 primitive, which generates a unique job ID for future reference:

```
from qiskit_ibm_runtime import SamplerV2
sampler=SamplerV2(backend)
job_qc=sampler.run([qc_transpiled],shots=4096)
job_id=job_qc.job_id()
```

The circuits were applied to a subset of  $8 \times 8$  MNIST digit images (digits 0–6). Circuit depths and execution times are presented in Table 6.5.

Table 6.5: Observations for Sequential and Parallel NEQR on QPU

MNSIT Digits ( $8 \times 8$ )	Transpiled Circuit depth		Depth reduction (%)	Execution time (seconds)		Time reduction (%)
	Sequential	Parallel		Sequential	Parallel	
0	66361	21181	68	49	18	63
1	44039	15332	65	33	13	61
2	64018	22209	65	48	19	60
3	66076	23314	65	50	19	62
4	68515	23364	66	52	20	62
5	54518	23155	58	41	19	54
6	58139	22562	61	42	19	55

The results demonstrate that the parallel Bit-Plane NEQR model significantly reduces circuit depth and execution time compared to the sequential NEQR model across all tested digits. The reduction in circuit depths is observed from 58% to 68% , and the reduction in execution time is observed from 54% to 63%.

#### **6.6.4. Quantum Image Obfuscation**

Quantum image obfuscation offers significantly enhanced security compared to classical image obfuscation by leveraging key quantum principles of superposition, entanglement, and no-cloning theorem. Unlike classical methods like pixel scrambling or standard encryption, NEQR-based obfuscation generates a quantum state with exponentially larger state space, thereby making unauthorized reconstruction highly challenging. The use of unitary quantum gates ensures that the process remains fully reversible, allowing accurate de-obfuscation when the appropriate inverse operations are applied. In this study, the Hadamard operation is used as the core mechanism. Despite its simplicity, the Hadamard gate serves as a powerful tool for obfuscation. When applied to qubits encoding pixel positions or intensity values, it transforms computational basis states into uniform superpositions, dispersing image information and effectively concealing both position and intensity-related features.

Figure 6.12 presents the block diagram of quantum image obfuscation and recovery. The process starts with encoding a  $128 \times 128$  grayscale image into a quantum state using the parallel-bit-plane NEQR representation. Due to resource constraints – hardware as well as simulator constraints – it is not feasible to process the entire image simultaneously. Therefore, the image is divided into smaller blocks ( $16 \times 16$ ), and quantum operations are applied independently to each block. The resulting blocks are then recombined to reconstruct the complete obfuscated image. To perform obfuscation, Hadamard gate is applied to all intensity qubits, converting their computational basis states into uniform superpositions and thereby scrambling the image information in the quantum domain. The obfuscated quantum state can then be transmitted securely. At receiver, the original image is recovered by applying Hadamard transformation to the intensity qubits followed by quantum measurement. The symmetric use of Hadamard operations ensures a reversible and lossless recovery of the original image.

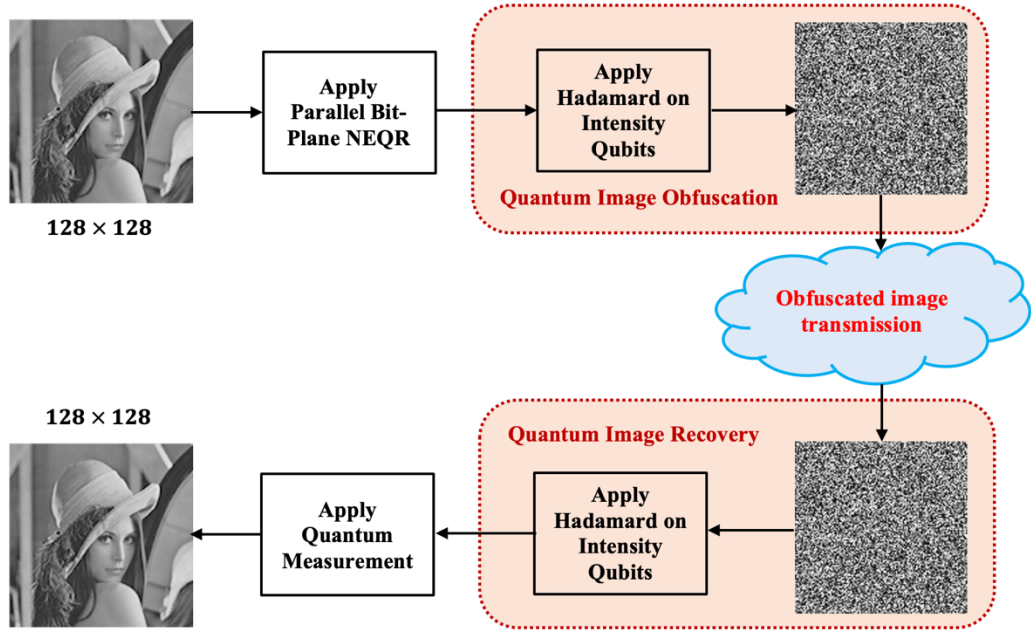


Figure 6.14: Quantum image obfuscation and recovery

## 6.7. Conclusion

In this work, the NEQR was first optimized by reducing the circuit depth for decomposition of multi-controlled quantum gates. Then, to further improve efficiency, a parallel version of NEQR was implemented that operates on the bit-planes of an image reducing the circuit depth. Experimental results showed that the parallel NEQR reduces circuit depth and execution time, thereby minimizing quantum errors associated with decoherence.

Quantum image processing demonstrates exponential speedup over classical approaches for fundamental operations. For instance, image negation is achieved by applying a NOT gate to the intensity qubits, requiring only a circuit depth of 1. Image rotation is accomplished by swapping the row and column qubits, followed by a NOT gate on the column register, resulting in a circuit depth of only 2. Similarly, quantum image obfuscation is implementable with a circuit depth of 1. Quantum circuits applied to NEQR encoded images enable advanced tasks such as quantum steganography, steganalysis, and image encryption using lightweight schemes with enhanced efficiency and security compared to classical methods. Additionally, quantum techniques for edge detection and filtering can be leveraged to improve feature extraction, facilitating more effective quantum image classification.

## REFERENCES

- [1] A. K. Sahu and G. Swain, "A Review on LSB Substitution and PVD Based Image Steganography Techniques," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 2, no. 3, p. 712, 2016.
- [2] D. Kumar, V. K. Sudha, N. Manikandan, and others, "Efficient three layer secured adaptive video steganography method using chaotic dynamic systems," *Sci Rep*, vol. 14, p. 18301, 2024.
- [3] S. Kumar, A. Gupta, and G. S. Walia, "Reversible data hiding: A contemporary survey of state-of-the-art, opportunities and challenges," *Applied Intelligence*, vol. 52, pp. 7373–7406, 2021.
- [4] J. M. Barton, "Method and apparatus for embedding authentication information within digital data," Jul. 1997.
- [5] M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Reversible Data Hiding," in *Proceedings of the International Conference on Image Processing (ICIP)*, Rochester, NY, USA, 2002.
- [6] M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Lossless generalized-LSB data embedding," *IEEE Transactions on Image Processing*, vol. 14, no. 2, pp. 253–266, 2005.
- [7] K. F. Rafat and S. M. Sajjad, "Advancing Reversible LSB Steganography: Addressing Imperfections and Embracing Pioneering Techniques for Enhanced Security," *IEEE Access*, vol. 12, pp. 143434–143457, 2024.
- [8] Z. Ni, Y. Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 3, pp. 354–362, 2006.
- [9] M. Fallahpour and M. H. Sedaaghi, "High capacity lossless data hiding based on histogram modification," *IEICE Electronics Express*, vol. 4, no. 7, pp. 205–210, Apr. 2007.
- [10] Z. Pan, S. Hu, X. Ma, and others, "Reversible data hiding based on local histogram shifting with multilayer embedding," *J Vis Commun Image Represent*, vol. 31, 2015.

- [11] K. S. R. Murthy and M. V Manikandan, "A Block-wise Histogram Shifting based Reversible Data Hiding Scheme with Overflow Handling," in *11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Kharagpur, India, 2020.
- [12] J. Hao, P. Ping, X. Peng, and others, "Reversible data hiding scheme based on image partitioning and histogram shifting," in *IEEE 8th International Conference on Big Data Computing Service and Applications*, Newark, CA, USA, 2022, pp. 27–34.
- [13] S. K. Khudhair, M. Sahu, R. K. R., and others, "Secure Reversible Data Hiding Using Block-Wise Histogram Shifting," *Electronics (Basel)*, vol. 12, p. 1222, 2023.
- [14] C. C. Chang, W. L. Tai, and K. N. Chen, "Lossless Data Hiding Based on Histogram Modification for Image Authentication," in *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, 2008, pp. 506–511.
- [15] K. H. Jung and K. Y. Yoo, "Data Hiding Method using image interpolation," *Comput Stand Interfaces*, vol. 31, no. 2, pp. 465–470, 2009.
- [16] C. F. Lee and Y. L. Huang, "An efficient image interpolation increasing payload in reversible data hiding," *Expert Syst Appl*, vol. 39, no. 8, pp. 6712–6719, 2012.
- [17] H. Ye, K. Su, X. Cheng, and others, "Research on reversible image steganography of encrypted image based on image interpolation and difference histogram shift," *IET Image Process*, vol. 16, pp. 1959–1972, 2022.
- [18] A. Benhfid, E. B. Ameer, and Y. Taouil, "Reversible steganographic method based on interpolation by bivariate linear box-spline on the three directional mesh," *Journal of King Saud University – Computer and Information Sciences*, vol. 32, pp. 850–859, 2020.
- [19] F. S. Hassan and A. Gutub, "Novel embedding secrecy within images utilizing an improved interpolation-based reversible data hiding scheme," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 5, pp. 2017–2030, 2022.
- [20] R. Punia, A. Malik, Aruna, and S. Singh, "Innovative image interpolation based reversible data hiding for secure communication," *Discover Internet of Things*, 2023.



- [21] J. Tian, "Reversible Data Embedding Using a Difference Expansion," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 890–896, 2003.
- [22] A. M. Alattar, "Reversible watermark using the difference expansion of a generalized integer transform," *IEEE Transactions on Image Processing*, vol. 13, no. 8, pp. 1147–1156, 2004.
- [23] D. M. Thodi and J. J. Rodríguez, "Expansion Embedding Techniques for Reversible Watermarking," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 721–730, 2007.
- [24] A. Arham and O. S. Riza, "Reversible Data Hiding Using Hybrid Method of Difference Expansion on Medical Image," *JITEKI*, vol. 6, no. 1, pp. 11–19, 2020.
- [25] C. Chang, Y. Huang, and T. Lu, "A difference expansion based reversible information hiding scheme with high stego image visual quality," *Multimed Tools Appl*, 2017.
- [26] A. Arham and H. A. Nugroho, "Enhanced reversible data hiding using difference expansion and modulus function with selective bit blocks in images," *Cybersecurity*, vol. 7, p. 61, 2024.
- [27] S. Kumar, G. S. Walia, and A. Gupta, "An efficient separable reversible data hiding method based on bit pair difference at pixel level," *The Imaging Science Journal*, pp. 1–14, 2024, doi: 10.1080/13682199.2024.2430874.
- [28] M. Fallahpour, "Reversible image data hiding based on gradient adjusted prediction," *IEICE Electronics Express*, vol. 5, no. 20, pp. 870–876, 2008.
- [29] J. Wang, J. Ni, and X. Zhang, "Rate and Distortion Optimization for Reversible Data Hiding Using Multiple Histogram Shifting," *IEEE Trans Cybern*, vol. 47, no. 2, pp. 315–326, 2017.
- [30] S. Kim, X. Qu, V. Sachnev, and others, "Skewed histogram shifting for reversible data hiding using a pair of extreme predictions," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 11, pp. 3236–3246, 2019.
- [31] J. Wang, X. N. J. Chen, N. Mao, and others, "Multiple histograms-based reversible data hiding: Framework and realization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 8, pp. 2313–2328, 2020.

- [32] M. Xiao, X. Li, Y. Zhao, and others, “A Novel Reversible Data Hiding Scheme Based on Pixel-Residual Histogram,” *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 19, no. 1, 2023.
- [33] L. Wang, Y. Huang, Y. Zhang, and others, “An effective steganalysis algorithm for histogram-shifting based reversible data hiding,” *Computers, Materials & Continua*, vol. 64, no. 1, 2020.
- [34] L. Dong, J. Zhou, W. Sun, and others, “First steps toward concealing the traces left by reversible image data hiding,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 5, pp. 951–955, 2020.
- [35] H. Shi, B. Hu, J. Geng, and others, “Trace Concealment Histogram-Shifting-Based Reversible Data Hiding with Improved Skipping Embedding and High-Precision Edge Predictor,” *Mathematics*, vol. 10, p. 4249, 2022.
- [36] USC Signal and I. P. Institute, “The USC-SIPI Image Database,” 1997.
- [37] S. N. V. J. D. Kosuru, A. Pradhan, K. A. Basith, and others, “Digital Image Steganography With Error Correction on Extracted Data,” *IEEE Access*, vol. 11, pp. 80945–80957, 2023.
- [38] L. Y. Por, D. Beh, T. F. Ang, and others, “An enhanced mechanism for image steganography using sequential colour cycle Algorithm,” *International Arab Journal of Information Technology*, vol. 10, 2013.
- [39] R. Al-Jawry and H. Alshrebaty, “High PSNR Using Fibonacci Sequences in Classical Cryptography and Steganography Using LSB,” *International Journal of Intelligent Engineering and Systems*, 2023.
- [40] F. Petitcolas, “Watermarking Image Database.”
- [41] P. Bas, T. Filler, and T. Pevný, “Break Our Steganographic System: The Ins and Outs of Organizing BOSS,” in *International Workshop on Information Hiding*, Springer, 2011, pp. 59–70.
- [42] Kaggle, “BOSSBase & BOWS2 Dataset,” 2023.
- [43] J. Fridrich, M. Goljan, and R. Du, “Reliable detection of LSB steganography in color and grayscale images,” in *Proceedings of 2001 Workshop on Multimedia and Security*, 2001, pp. 27–30.
- [44] P. Wang, Z. Wei, and L. Xiao, “Pure spatial rich model features for digital image steganalysis,” *Multimed Tools Appl*, vol. 75, 2016.

- [45] M. Boroumand, M. Chen, and J. Fridrich, “Deep Residual Network for Steganalysis of Digital Images,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1181–1193, May 2019.
- [46] R. P. Feynman, “Simulating physics with computers,” *International Journal of Theoretical Physics*, vol. 21, no. 6, pp. 467–488, 1982, doi: 10.1007/BF02650179.
- [47] D. Deutsch, “Quantum theory, the Church–Turing principle and the universal quantum computer,” *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 400, no. 1818, pp. 97–117, 1985, doi: 10.1098/rspa.1985.0070.
- [48] D. Deutsch and R. Jozsa, “Rapid solution of problems by quantum computation,” *Proc R Soc Lond A Math Phys Sci*, vol. 439, no. 1907, pp. 553–558, 1992, doi: 10.1098/rspa.1992.0167.
- [49] P. W. Shor, “Algorithms for quantum computation: discrete logarithms and factoring,” in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, Santa Fe, NM, USA: IEEE, 1994, pp. 124–134. doi: 10.1109/SFCS.1994.365700.
- [50] P. W. Shor, “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer,” *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997, doi: 10.1137/S0097539795293172.
- [51] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*, ACM, 1996, pp. 212–219. doi: 10.1145/237814.237866.
- [52] J. A. Jones and M. Mosca, “Implementation of a quantum algorithm on a nuclear magnetic resonance quantum computer,” *J Chem Phys*, vol. 109, no. 5, pp. 1648–1653, 1998, doi: 10.1063/1.476739.
- [53] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, “Experimental realization of Shor’s quantum factoring algorithm using nuclear magnetic resonance,” *Nature*, vol. 414, pp. 883–887, 2001, doi: 10.1038/414883a.
- [54] J. Preskill, “Quantum computing and the entanglement frontier,” 2012. doi: 10.48550/arXiv.1203.5813.
- [55] J. Preskill, “Quantum computing in the NISQ era and beyond,” *Quantum*, vol. 2, p. 79, 2018, doi: 10.22331/q-2018-08-06-79.

- [56] F. Arute and others, “Quantum supremacy using a programmable superconducting processor,” *Nature*, vol. 574, pp. 505–510, 2019, doi: 10.1038/s41586-019-1666-5.
- [57] M. AbuGhanem, “IBM quantum computers: evolution, performance, and future directions,” *Journal of Supercomputing*, vol. 81, p. 687, 2025, doi: 10.1007/s11227-025-07047-7.
- [58] V. Vedral, A. Barenco, and A. Ekert, “Quantum networks for elementary arithmetic operations,” *Phys Rev A (Coll Park)*, vol. 54, no. 1, pp. 147–153, 1995, doi: 10.1103/PhysRevA.54.147.
- [59] T. Draper, “Addition on a quantum computer,” 2000. doi: 10.48550/arXiv.quant-ph/0008033.
- [60] L. Ruiz-Perez and J. C. Garcia-Escartin, “Quantum arithmetic with the quantum Fourier transform,” *Quantum Inf Process*, vol. 16, p. 152, 2017, doi: 10.1007/s11128-017-1603-1.
- [61] E. Şahin, “Quantum arithmetic operations based on quantum Fourier transform on signed integers,” *International Journal of Quantum Information*, vol. 18, no. 6, 2020, doi: 10.1142/S0219749920500355.
- [62] Y. Yuan, C. Wang, B. Wang, Z.-Y. Chen, Y. Wu, and G.-P. Guo, “An improved QFT-based quantum comparator and extended modular arithmetic using one ancilla qubit,” *New J Phys*, vol. 25, 2023, doi: 10.1088/1367-2630/acfd52.
- [63] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, “A new quantum ripple-carry addition circuit,” 2004. doi: 10.48550/arXiv.quant-ph/0410184.
- [64] N. Yu, R. Duan, and M. Ying, “Five two-qubit gates are necessary for implementing Toffoli gate,” *Phys Rev A (Coll Park)*, vol. 88, p. 010304(R), 2013, doi: 10.1103/PhysRevA.88.010304.
- [65] A. Paler, “Quantum Fourier addition simplified to Toffoli addition,” *Phys Rev A (Coll Park)*, vol. 106, p. 42444, 2022, doi: 10.1103/PhysRevA.106.042444.
- [66] Z. Wang, M. Xu, and Y. Zhang, “Review of Quantum Image Processing,” *Archives of Computational Methods in Engineering*, vol. 29, no. 2, pp. 737–761, 2022, doi: 10.1007/s11831-021-09599-2.
- [67] X. W. Yao, H. Wang, Z. Liao, and others, “Quantum image processing and its application to edge detection: theory and experiment,” *Phys Rev X*, vol. 7, p. 31041, 2017, doi: 10.1103/PhysRevX.7.031041.

- [68] J. Su, X. Guo, C. Liu, and others, “A New Trend of Quantum Image Representations,” *IEEE Access*, vol. 8, pp. 214520–214537, 2020, doi: 10.1109/ACCESS.2020.3039996.
- [69] P. Q. Le, F. Dong, and K. Hirota, “A flexible representation of quantum images for polynomial preparation, image compression, and processing operations,” *Quantum Inf Process*, vol. 10, no. 1, pp. 63–84, 2011, doi: 10.1007/s11128-010-0177-y.
- [70] A. Geng, A. Moghiseh, C. Redenbach, and others, “Improved FRQI on superconducting processors and its restrictions in the NISQ era,” *Quantum Inf Process*, vol. 22, p. 104, 2023, doi: 10.1007/s11128-023-03838-0.
- [71] B. Sun, A. Iliyasu, F. Yan, and others, “An RGB Multi-Channel Representation for Images on Quantum Computers,” *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 17, no. 3, pp. 404–417, 2013, doi: 10.20965/jaciii.2013.p0404.
- [72] F. Yan, Y. Guo, A. Iliyasu, and others, “Multi-Channel Quantum Image Scrambling,” *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 20, pp. 163–170, 2016, doi: 10.20965/jaciii.2016.p0163.
- [73] Y. Zhang, K. Lu, Y. Gao, and others, “NEQR: a novel enhanced quantum representation of digital images,” *Quantum Inf Process*, vol. 12, pp. 2833–2860, 2013, doi: 10.1007/s11128-013-0567-z.
- [74] T. Li, P. Zhao, Y. Zhou, and others, “Quantum Image Processing Algorithm Using Line Detection Mask Based on NEQR,” *Entropy*, vol. 25, no. 5, p. 738, 2023, doi: 10.3390/e25050738.
- [75] S. Du, K. Luo, Y. Zhi, and others, “Binarization of grayscale quantum image denoted with novel enhanced quantum representations,” *Results Phys*, vol. 39, 2022, doi: 10.1016/j.rinp.2022.105710.
- [76] J. Mu, X. Li, X. Zhang, and others, “Quantum implementation of the classical guided image filtering algorithm,” *Sci Rep*, vol. 15, p. 493, 2025, doi: 10.1038/s41598-024-84211-8.
- [77] J. Sang, S. Wang, and Q. Li, “A novel quantum representation of color digital images,” *Quantum Inf Process*, vol. 16, p. 42, 2017, doi: 10.1007/s11128-016-1463-0.

- [78] Z. Qu, Z. Cheng, and X. Wang, “Matrix Coding-Based Quantum Image Steganography Algorithm,” *IEEE Access*, vol. 7, pp. 35684–35698, 2019, doi: 10.1109/ACCESS.2019.2894295.
- [79] G. Luo, R. G. Zhou, and W. Hu, “Efficient quantum steganography scheme using inverted pattern approach,” *Quantum Inf Process*, vol. 18, p. 222, 2019, doi: 10.1007/s11128-019-2341-3.
- [80] S. Zhao, F. Yan, K. Chen, and others, “Interpolation-Based High Capacity Quantum Image Steganography,” *International Journal of Theoretical Physics*, vol. 60, pp. 3722–3743, 2021, doi: 10.1007/s10773-021-04891-0.
- [81] X. Liu, D. Xiao, and Y. Xiang, “Quantum Image Encryption Using Intra and Inter Bit Permutation Based on Logistic Map,” *IEEE Access*, vol. 7, pp. 6937–6946, 2019, doi: 10.1109/ACCESS.2018.2889896.
- [82] V. Verma and S. Kumar, “Quantum image encryption algorithm based on 3D-BNM chaotic map,” *Nonlinear Dyn*, vol. 113, pp. 3829–3855, 2025, doi: 10.1007/s11071-024-10403-6.
- [83] T. S. Gururaja and P. Pravinkumar, “Enhanced quantum image encryption using DNA-QTRNG and Sattolo-RQFT shuffling,” *Journal of Supercomputing*, vol. 81, p. 667, 2025, doi: 10.1007/s11227-025-07085-1.
- [84] S. Heidari, M. Vafaei, M. Houshmand, and others, “A dual quantum image scrambling method,” *Quantum Inf Process*, vol. 18, p. 9, 2019, doi: 10.1007/s11128-018-2122-4.
- [85] V. K. Sharma, P. C. Sharma, H. Goud, and others, “Hilbert quantum image scrambling and graph signal processing-based image steganography,” *Multimed Tools Appl*, vol. 81, pp. 17817–17830, 2022, doi: 10.1007/s11042-022-12426-w.
- [86] C. P. Williams, “Quantum gates,” in *Explorations in Quantum Computing*, in Texts in Computer Science. , London: Springer, 2011. doi: 10.1007/978-1-84628-887-6\_2.
- [87] IBM Quantum Documentation, “Install Qiskit,” 2025.
- [88] L. Deng, “The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web],” *IEEE Signal Process Mag*, vol. 29, no. 6, pp. 141–142, 2012, doi: 10.1109/MSP.2012.2211477.



**DELHI TECHNOLOGICAL UNIVERSITY**  
(Formerly Delhi College of Engineering)  
Shahbad Daulatpur, Main Bawana Road, Delhi-110042

### **PLAGIARISM VERIFICATION**

Title of Thesis : Enhancement of Reversible Image Steganography and  
Optimization of Quantum Image Representation using  
the NEQR Model

Total Pages : 99

Name of Scholar : Sumitra Singh

Supervisor : Professor Dinesh Kumar Vishwakarma

Department : Information Technology

This is to report that the above thesis was scanned for similarity detection. Process and outcome is given below:

Software used: **TURNITIN**, Similarity Index: **10%**, Total Word Count: **20,764**

Date: 29/05/2025

**Candidate's signature**

**Signature of supervisor(s)**

# Sumitra Singh

## final-thesis-plag-23ITY26.docx

 Delhi Technological University

### Document Details

Submission ID

trn:oid::27535:98300560

Submission Date

May 29, 2025, 10:52 AM GMT+5:30

Download Date

May 29, 2025, 10:59 AM GMT+5:30

File Name

final-thesis-plag-23ITY26.docx

File Size

12.0 MB

99 Pages

20,764 Words

117,548 Characters







# 10% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.




## Filtered from the Report

- Bibliography

## Match Groups

-  **226** Not Cited or Quoted 10%  
Matches with neither in-text citation nor quotation marks
-  **2** Missing Quotations 0%  
Matches that are still very similar to source material
-  **2** Missing Citation 0%  
Matches that have quotation marks, but no in-text citation
-  **0** Cited and Quoted 0%  
Matches with in-text citation present, but no quotation marks

## Top Sources

- 6%  Internet sources
- 7%  Publications
- 7%  Submitted works (Student Papers)

## Integrity Flags

### 0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

## PUBLICATIONS

S. No.	Paper	Status
1.	<p>Title: Analysis of Reversible Steganography in Image Histograms</p> <p>Authors: Sumitra Singh, Dinesh Kumar Vishwakarma</p> <p>Conference: International Conference on Innovations in Intelligence Systems: Advancements in Computing, Communications, and Cyber-Security, (ISAC3-2025), 25-26 July 2025</p>	Accepted
2.	<p>Title: Efficient Quantum Image Representation via Optimized NEQR: Experiments on MNIST Dataset</p> <p>Authors : Sumitra Singh, Dinesh Kumar Vishwakarma</p> <p>Conference: Emerging Trends in Defence Technology (ETDT-2025), 20-22 August 2025</p>	Accepted
3.	<p>Title: Performance Analysis of Quantum Arithmetic Circuits on IBM Eagle Processor</p> <p>Authors: Sumitra Singh, Dinesh Kumar Vishwakarma</p> <p>Journal: Defence Science Journal</p>	Under Review
4.	<p>Title: Enhancing Reversible Steganography in Image Histograms via Two-Layer Embedding</p> <p>Authors: Sumitra Singh, Dinesh Kumar Vishwakarma</p> <p>Journal: Multidimensional Systems and Signal Processing</p>	Communicated
5.	<p>Title: Parallel Bit-Plane NEQR for Efficient Quantum Image Representation</p> <p>Authors: Sumitra Singh, Dinesh Kumar Vishwakarma</p> <p>Journal: Journal of Supercomputing</p>	Communicated