

# **Analysis and Design of GPGPU-based Secure Visual Secret Sharing (VSS) Schemes**

Thesis

Submitted in partial fulfilment of the requirements for the degree of

**DOCTOR OF PHILOSOPHY**

*by*

**RAVIRAJA HOLLA M**



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

SURATHKAL, MANGALORE - 575 025

August, 2022



## **DECLARATION**

*by the Ph.D. Research Scholar*

I hereby declare that the Research Thesis entitled **ANALYSIS AND DESIGN OF GPGPU-BASED SECURE VISUAL SECRET SHARING SCHEMES** which is being submitted to the **National Institute of Technology Karnataka, Surathkal** in partial fulfilment of the requirements for the award of the Degree of **Doctor of Philosophy** in Department of Computer Science and Engineering is a bonafide report of the research work carried out by me. The material contained in this Research Thesis has not been submitted to any University or Institution for the award of any degree.



Raviraja Holla M, 177128 177CO502

Department of Computer Science and Engineering

Place: NITK, Surathkal.

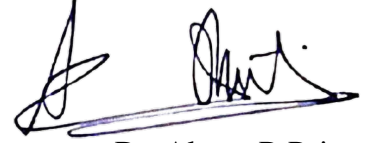
Date: August 23, 2022





## CERTIFICATE

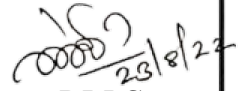
This is to certify that the Research Thesis entitled **ANALYSIS AND DESIGN OF GPGPU-BASED SECURE VISUAL SECRET SHARING SCHEMES** submitted by **RAVIRAJA HOLLA M** (Register Number: 177128 177CO502) as the record of the research work carried out by him, is accepted as the Research Thesis submission in partial fulfillment of the requirements for the award of degree of **Doctor of Philosophy**.



Dr. Alwyn R Pais

Research Guide

(Signature with Date and Seal)

  
Chairman DRPC

(Signature with Date and Seal)

डॉ. अल्पजोशी / डॉ. अल्पजोशी / डॉ. अल्पजोशी  
संगणक विज्ञान एवं अभियांत्रिकी विभाग

एन आइ टी के - सुरथकल

श्रीनिवासनगर - ५७५ ०२५

Chairman

BUGC / BPEE / DRPC

Dept. of Computer Science & Engineering

NITK, Surathkal

Srinivasnagar - 575 025



## **ACKNOWLEDGEMENTS**

Foremost, I would like to express my sincere gratitude to my Supervisor Dr. Alwyn R Pais, Associate Professor in the Department of Computer Science & Engineering, for his continuous encouragement, patience, motivation, enthusiasm, and immense knowledge. His guidance and insightful comments helped me in all the time of research and writing of this thesis. This investigation would not have been possible without his wholeheartedness, criticism, and advice.

My sincere thanks go to research progress committee members Dr. Basavaraj Talawar, Assistant Professor in Department of Computer Science & Engineering, and Dr. Nagendrappa H, Assistant Professor in Electrical & Electronics Engineering, for their continuous valuable suggestions and recommendations.

I am also grateful to all office staff members of the Computer Science & Engineering Department for their generous support throughout this work. My appreciation extends to my friends Nikhil, Srinivasa, Alok, and Apoorva for helping me and even hear my problem.

I would like to especially acknowledge Dr. G.K Prabhu, President, Manipal University Jaipur, Dr. Anila Rana, Director, MIT, and Dr. Smitha N Pai, HOD, I&CT Department, MIT, Manipal. I must also thank my colleagues Dr. Balachandra, Dr. Sucheta Kolekar and Dr. Veena Maiya at MIT for their help during my study.

Most importantly, I would like to thank my family for their love and support for this work and throughout my life.

Raviraja Holla M



## ABSTRACT

Visual Secret Sharing (VSS) stands for sharing confidential image data across the participants as shares. As each share is not self-sufficient to compromise the secrecy, it is also called a shadow. Only the authorized participants can successfully recover the secret information with their shadows based on the type of VSS schemes. Recent advancements in VSS schemes have reduced the computation cost of encryption and decryption of the images. Still, real-life applications can not utilize these techniques with resource-constrained devices as they are computationally intensive. Another well-identified problem with VSS modalities is that they result in the distorted quality of the recovered image. The solution to improve the contrast of the decrypted image demand additional extensive processing. Therefore an ideal remedy is to exploit the advancements in hardware like GPGPU to propose novel parallel VSS schemes.

The existing VSS schemes reconstruct the original secret image as a halftone image with only a 50% contrast. The Randomized Visual Secret Sharing (RVSS) scheme overcomes this disadvantage by achieving contrast of 70% to 90% for noise-like and 70% to 80% for meaningful shares (Mhala et al. 2017). But RVSS is computationally expensive. As a remedy we presented a GPGPU-based RVSS (GRVSS) technique that harness the high-performance computing power of the General-Purpose Computation on Graphics Processing Unit (GPGPU) for the data-parallel tasks in the RVSS. The presented GRVSS achieved a speedup range of  $1.6\times$  to  $1.8\times$  for four shares and  $2.63\times$  to  $3\times$  for eight shares, for different image sizes.

To further enhance the visual quality of the retrieved image, we presented an effective secret image sharing with super-resolution that leverage the deep learning super-resolution technique. The presented model outperformed the benchmark model in many-objective parameters with the recovered secret image contrast of 96.6% - 99.8% for noise-like and 64% - 80 % for meaningful shares. We used GPGPU to conserve the training time of the neural network model to achieve a speedup of  $1.92\times$  over the counterpart CPU super-resolution.

Also, we presented an effective VSS model with super-resolution utilizing a Convolution Neural Network (CNN) intending to increase both the contrast of the recovered image and the speedup. The objective quality assessment proved that the presented model produces a high-quality reconstructed image with the contrast of 97.3% - 99.7% for noise-like and 78.4% - 89.7 % for meaningful shares having the Structural Similarity Index (SSIM) of 89% - 99.8% for the noise-like shares and 71.6% - 90% for the meaningful shares. The presented technique achieved an average speedup of  $800\times$  in comparison with the sequential model.

The application of VSS schemes to medical images poses additional challenges due to their inherent poor quality. Also, medical imaging demands real-time VSS schemes to save the life of patients. We extended these concepts to propose a Medical Image Secret Sharing (MISS) model that deems fit for the Computer-Aided Diagnostic (CAD) tools. The presented fused feature extractor with a random forest classifier demonstrated that our VSS scheme reconstructs the secret medical images suitable for CAD tools. The result analysis confirmed the high-performance of the MISS with a 99.3% contrast and a 98% SSIM of the reconstructed image. MISS achieved an average speedup of  $800\times$  in comparison with the sequential model. We used cluster sizes of 300, 500, and 1000 to obtain the MISS model's CAD performance measures such as accuracy, sensitivity, specificity, precision, and F-measure using a presented fused feature extractor and a random forest classifier. The achieved precisions of 99.45%, 99.83%, and 100% for 300, 500, and 1000 cluster sizes prove the presented model's suitability for the CAD systems.

**Keywords:** Visual Secret Sharing, GPGPU, Discrete Cosine Transform, Super-resolution.

# CONTENTS

<b>List of Figures</b>	<b>x</b>
<b>List of Tables</b>	<b>xii</b>
<b>List of Abbreviations</b>	<b>xiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Visual Secret Sharing (VSS)	2
1.2 Few Applications of VSS	3
1.3 GPGPU VSS	4
1.4 Super Resolution (SR) Reconstruction	4
1.5 Motivation	5
1.6 Thesis Contributions	6
1.7 Thesis Organization	8
<b>2 Literature Review</b>	<b>11</b>
2.1 Visual Secret Sharing	11
2.2 VSS taxonomy	12
2.3 Feature enhancements in size-invariant VSS schemes	16
2.4 Super-Resolution (SR) Image Reconstruction	25
2.4.1 Observation model of a real imaging system	26
2.4.2 SR Techniques	27
2.4.3 Challenges in SR	29
2.5 Heterogeneous Computing (HC)	30
2.6 Problem Description	31
2.7 Problem Statement	31
2.8 Objectives	31
2.9 Summary	32

<b>3</b>	<b>GPGPU-based Randomized VSS</b>	<b>33</b>
3.1	Random Grid-based VSS for Grayscale and Colour Images . . . . .	35
3.1.1	Random Grid-Based VSS for Grayscale and Colour Images . . .	37
3.1.2	Parallelization through GPGPU . . . . .	38
3.1.3	Random Grid-based model on a Many-core System . . . . .	38
3.1.4	Experimental Results . . . . .	42
3.2	GPGPU-based Randomized Visual Secret Sharing (GRVSS) . . . . .	49
3.2.1	RVSS Technique . . . . .	50
3.2.2	GRVSS Technique . . . . .	53
3.2.3	Computational Complexity of the RVSS and GRVSS model . . .	57
3.2.4	Experimental Results . . . . .	59
3.3	Summary . . . . .	67
<b>4</b>	<b>VSS using Quantum logic and GPGPU based EDNN Super Resolution</b>	<b>69</b>
4.1	Comparison of existing super-resolution models . . . . .	71
4.2	Presented Methodology . . . . .	73
4.2.1	Half-toning using EDVT . . . . .	74
4.2.2	Shares generation using quantum logic coding . . . . .	76
4.2.3	Embedding share and recover the secret image . . . . .	79
4.2.4	Enthalpy based DNN (EDNN) . . . . .	80
4.3	Results and Discussion . . . . .	83
4.3.1	Performance Analysis . . . . .	87
4.3.2	Effectiveness of super-resolution . . . . .	89
4.3.3	Multiple Scales based PSNR and Execution Time . . . . .	91
4.4	Summary . . . . .	93
<b>5</b>	<b>GPGPU VSS by contrast-adaptive CNN super-resolution</b>	<b>95</b>
5.1	GPGPU VSS . . . . .	99
5.1.1	Halftoning using GPGPU error diffusion . . . . .	102
5.1.2	Creating shadows using GPGPU . . . . .	102
5.1.3	Information embedding using Discrete Wavelet Transform (DWT)	104
5.1.4	Secret image reconstruction . . . . .	106



5.1.5	Contrast-adaptive Convolution Neural Network (CCNN) . . . . .	108
5.2	EXPERIMENTAL RESULTS . . . . .	109
5.2.1	Mean Squared Error ( <i>MSE</i> ) . . . . .	114
5.2.2	Peak Signal to Noise Ratio ( <i>PSNR</i> ) . . . . .	115
5.2.3	Normalized Cross Correlation ( <i>NCC</i> ) . . . . .	115
5.2.4	Normalized Absolute Error ( <i>NAE</i> ) . . . . .	116
5.2.5	Structural Similarity Index ( <i>SSIM</i> ) . . . . .	117
5.2.6	Speedup . . . . .	118
5.3	Summary . . . . .	118
<b>6</b>	<b>Medical image secret sharing using super-resolution for CAD systems</b>	<b>121</b>
6.1	VSS schemes applied to medical images . . . . .	122
6.2	MISS Methodology . . . . .	126
6.2.1	Error diffusion halftoning using GPU . . . . .	127
6.2.2	Generating shadow images using GPU . . . . .	129
6.2.3	Information embedding using Discrete Wavelet Transform (DWT)	129
6.2.4	Reconstruction of the secret image . . . . .	133
6.2.5	Convolution Neural Network (CNN) . . . . .	135
6.3	Experimental Results . . . . .	137
6.3.1	Objective performance analysis of MISS . . . . .	137
6.3.2	Random forest classifier for Computer Aided Diagnosis (CAD)	138
6.3.3	Speedup . . . . .	142
6.4	Summary . . . . .	142
<b>7</b>	<b>Conclusions and Future Scope</b>	<b>145</b>
	<b>Bibliography</b>	<b>148</b>
	<b>Research Outcomes</b>	<b>165</b>



## LIST OF FIGURES

2.1	VSS taxonomy. . . . .	13
2.2	Size-variant VSS scheme. . . . .	14
2.3	Size-invariant VSS scheme. . . . .	15
2.4	Image observation model. . . . .	26
3.1	Block diagram of the presented system. . . . .	39
3.2	Execution times for generating halftone images. . . . .	43
3.3	Execution times for generating shares. . . . .	43
3.4	Total execution times. . . . .	44
3.5	(a) Sample grayscale test image Lenna ( $256 \times 256$ ) (b) Halftone image ( $256 \times 256$ ) (c) Master share ( $256 \times 256$ ) (d) Encoded share ( $256 \times 256$ ) (e) Reconstructed image ( $256 \times 256$ ). . . . .	44
3.6	(a) Sample colour test image Lenna ( $256 \times 256$ ) (b) C component ( $256 \times$ $256$ ) (c) M component ( $256 \times 256$ ) (d) Y component ( $256 \times 256$ ) (e) Halftone of C component (f) Halftone of M component (g) Halftone of Y component. . . . .	45
3.7	(a) Master grid for C halftone (b) Encoded grid for C halftone (c) Stack- ing of C grids (d) Master grid for M halftone (e) Encoded grid for M halftone (f) Stacking of M grids (g) Master grid for Y halftone (h) En- coded grid for Y halftone (i) Stacking of Y grids (j) Recovered image. Note: Size of all images is ( $256 \times 256$ ). . . . .	46
3.8	Speedup in generating halftone images. . . . .	47
3.9	Speedup in generating share images. . . . .	47
3.10	Total speedup in generating halftone and share images. . . . .	48

3.11	The standard quantization table (Bovik 2009).	51
3.12	The defined sets and locations for embedding the data.	52
3.13	The sample test images used for experimentation (Grayscale images) (a) Girl ( $256 \times 256$ ) (b) Harry ( $400 \times 400$ ) (c) Carcinoma in situ ( $760 \times 570$ ) and (d) Invasive carcinoma ( $2048 \times 1536$ ).	59
3.14	The sample test images used for experimentation (Colour images) (a) Girl ( $256 \times 256$ ) (b) Harry ( $400 \times 400$ ) (c) Carcinoma in situ ( $760 \times 570$ ) and (d) Invasive carcinoma ( $2048 \times 1536$ ).	59
3.15	Total execution time of RVSS and GRVSS for 4 shares (Grayscale images).	64
3.16	Total execution time of RVSS and GRVSS for 8 shares (Grayscale Images)	64
3.17	Total execution time of RVSS and GRVSS for 4 shares (Colour Images)	65
3.18	Total execution time of RVSS and GRVSS for 8 shares (Colour Images)	65
4.1	Flow diagram of the presented methodology.	74
4.2	Flow chart for EDVT Color half-toning.	76
4.3	Flow chart for Share generation using quantum logic coding.	77
4.4	Overview of the low-resolution image formation.	81
4.5	Architecture of the EDNN.	82
4.6	Test input images.	84
4.7	Halftone images.	84
4.8	Share images of halftone images.	85
4.9	Embedded images.	85
4.10	Extracted images.	86
4.11	Reconstructed images.	86
4.12	Final super resolution images.	86
4.13	Time overlapping architecture of the implementation.	86
4.14	Color test images of size ( $256 \times 256$ ).	87
4.15	<i>Accuracy</i> of the presented model with and without super-resolution.	90
4.16	<i>MSE</i> of the presented model with and without super-resolution.	90
4.17	<i>SNR</i> of the presented model with and without super-resolution.	91

4.18	Comparison analysis of <i>PSNR</i> and time in Scale $\times 2$ . . . . .	91
4.19	Comparison analysis of <i>PSNR</i> and time in Scale $\times 3$ . . . . .	92
4.20	Comparison analysis of <i>PSNR</i> and time in Scale $\times 4$ . . . . .	92
5.1	Block diagram of the presented methodology. . . . .	100
5.2	Architecture of the presented CCNN. . . . .	108
5.3	Test input images (a) Baboon-color image, (b) Barbara-grayscale image, and (c) Lena-color image. . . . .	110
5.4	Halftone images (a) Baboon-color image, (b) Barbara-grayscale image, and (c) Lena-color image. . . . .	110
5.5	Shadow (Share) images of halftone images (a) Baboon-color image, (b) Barbara-grayscale image, and (c) Lena-color image. . . . .	111
5.6	Embedded images (a) Baboon-color image, (b) Barbara-grayscale image, and (c) Lena-color image. . . . .	111
5.7	Extracted images (a) Baboon-color image, (b) Barbara-grayscale image, and (c) Lena-color image. . . . .	112
5.8	Reconstructed images after extracting data (a) Baboon-color image, (b) Barbara-grayscale image, and (c) Lena-color image. . . . .	112
5.9	Final reconstructed images after super resolution (a) Baboon-color image, (b) Barbara-grayscale image, and (c) Lena-color image. . . . .	112
5.10	PSNR convergence curve of CCNN for noise-like shares. . . . .	113
5.11	The receiver operating characteristic (ROC) curve. . . . .	114
5.12	Test colour images (a) Female ( $256 \times 256$ ) (b) Couple ( $256 \times 256$ ) (c) House ( $256 \times 256$ ), and (d) Tree ( $256 \times 256$ ). . . . .	114
5.13	Test grayscale images (a) Airplane ( $256 \times 256$ ) (b) Clock ( $256 \times 256$ ) (c) Couple ( $512 \times 512$ ), and (d) Man ( $1024 \times 1024$ ). . . . .	114
6.1	Block diagram of the MISS methodology. . . . .	126
6.2	Floyd-Steinberg error diffusion. . . . .	128
6.3	Breast cancer biopsy sample images (a) normal, (b) carcinoma in situ, and (c) invasive. . . . .	128

6.4	Breast cancer biopsy halftone sample images (a) normal, (b) carcinoma in situ, and (c) invasive. . . . .	128
6.5	Share image of breast cancer biopsy sample images (a) normal, (b) carcinoma in situ, and (c) invasive. . . . .	129
6.6	DWT transformation (2 level). . . . .	132
6.7	DWT transformation (2 level) (Dhage et al. 2015). . . . .	132
6.8	Embedded image of breast cancer biopsy sample images (a) normal, (b) carcinoma in situ, and (c) invasive. . . . .	133
6.9	Extracted image of breast cancer biopsy sample images (a) normal, (b) carcinoma in situ, and (c) invasive. . . . .	133
6.10	Recovered image of breast cancer biopsy sample images (a) normal, (b) carcinoma in situ, and (c) invasive. . . . .	135
6.11	MISS system: Final SR recovered image of breast cancer biopsy sample images (a) normal, (b) carcinoma in situ, and (c) invasive. . . . .	135
6.12	Layered CNN architecture of the proposed SR model. . . . .	135
6.13	Convolution computation with a filter (con 2021). . . . .	136
6.14	RGB histogram of the input secret image (rescaled to the size $64 \times 64$ ) analysis. . . . .	138
6.15	RGB histogram of the recovered secret image (rescaled to the size $64 \times 64$ .) analysis . . . . .	138

## LIST OF TABLES

2.1	The basis matrices for $n = 2$ . . . . .	14
2.2	The sharing matrices used in Hou and Quan (2011) . . . . .	18
2.3	The characteristics of various VSS schemes . . . . .	22
2.4	Future trends in multi-core and many-core systems (Mittal and Vetter 2015). . . . .	30
3.1	PARAM Shavak supercomputer environment. . . . .	44
3.2	Generated basis matrices for $n = 4$ . . . . .	51
3.3	Selected set size for embedding. . . . .	52
3.4	Hardware specification. . . . .	60
3.5	Execution time to generate 4 shares. . . . .	60
3.6	Execution time to generate 8 shares. . . . .	61
3.7	Execution time to embed data in 4 shares. . . . .	61
3.8	Execution time to embed data in 8 shares. . . . .	62
3.9	Execution time to restore image data in 4 shares. . . . .	62
3.10	Execution time to restore image data in 8 shares. . . . .	63
3.11	Speedup of GRVSS with 4 shares in generic architecture. . . . .	66
3.12	Speedup of GRVSS with 8 shares in generic architecture. . . . .	66
3.13	Speedup of GRVSS with 4 shares in PARAM Shavak. . . . .	66
3.14	Speedup of GRVSS with 8 shares in PARAM Shavak. . . . .	67
4.1	Comparison of existing super-resolution models. . . . .	72
4.2	Generated basis matrix. . . . .	78
4.3	$MSE$ esteems for different test images. . . . .	88
4.4	Comparison analysis in terms of execution speed. . . . .	88

4.5	<i>NCC</i> esteems for test images. . . . .	88
4.6	<i>NAE</i> esteems for the test images. . . . .	88
5.1	Generated basis matrices . . . . .	104
5.2	<i>CNN</i> summary utilized by the proposed system. . . . .	113
5.3	<i>MSE</i> comparison. . . . .	115
5.4	<i>PSNR</i> comparison. . . . .	116
5.5	<i>NCC</i> comparison. . . . .	116
5.6	<i>NAE</i> comparison. . . . .	117
5.7	<i>SSIM</i> comparison. . . . .	117
5.8	Comparison of execution times (in seconds) and speedups. . . . .	118
6.1	Description of symbols . . . . .	125
6.2	Five basis matrices to create four shadow images. . . . .	132
6.3	<i>CNN</i> layer summary of the proposed SR model. . . . .	136
6.4	Objective analysis of MISS with RVSS and IRVSS. . . . .	138
6.5	Accuracy, sensitivity, and specificity of MISS for the CAD system. . . . .	139
6.6	Precision, recall, and F-measure of MISS for the CAD system. . . . .	139
6.7	MISS performance against benchmark deep learning networks. . . . .	140
6.8	Speedups of the MISS over the sequential model. . . . .	142
6.9	MISS Speedup against benchmark GPU-based models. . . . .	142



## LIST OF ABBREVIATIONS

<b><u>Abbreviations</u></b>	<b><u>Expansion</u></b>
BPVSS	Block-based Progressive Visual Secret Sharing
CAD	Computer-Aided Diagnostic
CNN	Convolution Neural Network
CCNN	Contrast-adaptive Convolution Neural Network
CUDA	Compute Unified Device Architecture
DCT	Discrete Cosine Transformation
DWT	Discrete Wavelet Transformation
EDNN	Enthalpy based Deep Neural Network
EDVT	Error Diffusion with Varying Thresholds
ESPCNN	Efficient Sub-Pixel Convolutional Neural Network
GPU	Graphics Processing Unit
GPGPU	General-Purpose Graphics Processing Unit
GRVSS	GPGPU based Randomized Visual Secret Sharing
HC	Heterogeneous Computing
HR	High-Resolution
LR	Low-Resolution
MISS	Medical Image Secret Sharing
MSE	Mean Squared Error
NAE	Normalized Absolute Error
NCC	Normalized Cross Correlation
NPCR	Number of Pixels Change Rate
POCS	Projection onto Convex Sets
PSNR	Peak Signal to Noise Ratio

<b><u>Abbreviations</u></b>	<b>Expansion</b>
RG	Random Grid
ROC	Receiver Operating Characteristic
RVSS	Randomized Visual Secret Sharing
SISR	Single Image Super-Resolution
SR	Super-Resolution
SRCNN	Super-Resolution Convolution Neural Network
SSIM	Structural Similarity Index Measure
UACI	Unified Average Changing Intensity
VC	Visual Cryptography
VSS	Visual Secret Sharing

# CHAPTER 1

## INTRODUCTION

Internet is essentially a global linking of diverse computer networks into a single virtual network. During its inception, the Internet had a small manageable trustworthy research community as its clientele. Security was not a primary consideration in the development of Internet protocols. Within almost five decades from its inception, the Internet has evolved with an explosive growth with its technologies with open wireless communication paths. Internet of computers became the Internet of Things (IoT) with immensely diverse devices and users. Rapid software development cycles evolved to meet this vast, diverse community. All these growths intuitively leave behind many vulnerabilities.

The overall growth of the Internet has a significant impact on society. With the various information available with a click, the Internet's data has become a vital resource. An indispensable resource is always susceptible to threats. Information remains significant as long as it is safe and trustworthy. Therefore information security brings value to the information available on the Internet. In social media, colored images draw much attention because human brains respond to images and colors quickly compared to other types of information. The visual stimuli are much more memorable and processed in mind than linear linguistic details. The importance of visual information has led to the emerging field of visual secret sharing.

### **1.1 VISUAL SECRET SHARING (VSS)**

Visual Secret Sharing (VSS), in general, is a Visual Cryptography (VC) technique to address one or more of the security concerns of the digital images. This technique encrypts the secret image into several shares. Individual shares are not sufficient to restore the private image enabling safe communication of these shares over the public Internet channel. The combination of a prescribed number of shares can reconstruct an image approximate to the secret image. This technique differs from traditional cryptography schemes, mainly because the encryption is less-computational, and decryption is computation-less. This difference is evident due to the secret images' size to be encrypted and decrypted in VSS. The assumption here is that the shares are photocopied on to several transparencies. When these transparencies stacked on each other in a particular manner, the human visual system perceives sufficient secret image. It is in this context the shares are also called transparencies.

More the randomness in any cryptography algorithms more is the security. But cryptography schemes with complete randomness does not allow for decryption to succeed. The same is the case in VSS. Therefore, most VSS schemes allow randomness to a certain extent so that the restored image is approximately equal to the original secret image. The literature has shown an increasing number of novel protection methods for securing images over the Internet. These secure image sharing techniques overcome the traditional cryptographic approach, providing new solutions to develop new and secure imaging applications.

VSS schemes have many applications. The speed and the quality of the images are the real concerns expressed in the VSS literature. However, there is little work in this direction. With advancements in hardware technology, algorithms need not starve with the slow and sparse computing resources. So, designing VSS algorithms that exploit parallel heterogeneous computing resources such as General Purpose Graphics Processing Units (GPGPUs) is an amicable solution to meet the speed requirement. The super-resolution techniques improve the contrast of the images. If these techniques also leverage GPGPU heterogeneous computing resources, then the speed augments the quality gain. The following sections introduce applications of VSS, GPGPU VSS, and

Super-resolution, respectively.

## 1.2 FEW APPLICATIONS OF VSS

- **Watermarking Applications:** The copyright of digital media such as images, text, music, and movie can be protected by watermarking. VSS-based watermarking schemes assume three types of members namely owners, attackers, and an arbitrator. Owners seek copyright protection of the cover images they own. Attackers are the adversaries who may misuse the cover images. An arbitrator arbitrates any dispute on the ownership of the cover images. Such VSS schemes use an embedding algorithm to generate watermarking cover images ( $W$ ), a secret share ( $S$ ) and chaotically encoded key images ( $K$ ). Here the VSS is used to generate  $S$ . Owners publish the  $W$  and the watermark. However, owners register  $S$  to the arbitrator secretly. Each owner gets a key from the set  $K$ . After the  $W$  gets published, if the attackers misuse them, the legitimate owners approach the arbitrator with their keys as pieces of evidence to claim their copyright. Then the arbitrator extracts a watermark from the misused image,  $S$ , and  $K$  together. The arbitrator decides whether the ownership claims of the misused images stand legitimate by comparing the extracted and the original watermarks. The merits of applying VSS in watermarking include large embedding capacity, high security, and sharing between multiple users.
- **Resolution Variant VSS:** Resolution Variant VSS techniques use a single share generated by the VSS to recover secret information at multiple resolutions or zooms. The share is embedded within only the censored content of the images rather than the whole image. When an image with varying zooms superimposed with the same share, secrets revealed also vary to extract any specific censored information such as vehicle registration numbers from particular images.
- **Secured online transactions:** For the secured online transaction, financial institutions issue the numbered collection of shares to the customers. Each share contains a VSS pattern embedded in it. The customer sends the transaction information to the server. But the server does not commit the transaction immediately

to defeat the consequence of clandestine manipulation if any. Instead, it sends the gist of the received transaction detail and a transaction number to the customer but encoded by the VSS scheme so that a man-in-the-middle can not manipulate it. It also sends a share number that the customer must stack on the encoded image to decode. The decoding is possible only when the customer stacks the corresponding share on the encoded image. The server executes the transaction only when it receives the correct transaction number.

### 1.3 GPGPU VSS

The images are considered the massive mid-size data on the Internet. The image in the computer is a two-dimensional array of discrete values. Visual secret shares also being two-dimensional data, constructing them intuitively amenable to massive data parallelism. Depending on the design, there is a possibility of task parallelism. CPUs optimized for task parallelism and GPGPUs optimized for data parallelism, exploiting one or both of these parallelisms in VSS schemes protect the security and quality in real-time. The scalability of the GPGPU-based VSS schemes offers these advantages at a reasonable cost. The GPGPU VSS schemes create either a single share block-wise or many shares in parallel. They yield better speedup and resource utilization when the number of shares and the image size is more.

### 1.4 SUPER RESOLUTION (SR) RECONSTRUCTION

Most of the digital imaging applications require High-resolution (HR) images. If the pixel density within the image is high, it is said to be the HR image. The details in HR images are great compared to the Low Resolution (LR) images. The applications of the HR image is prominent in high-definition television (HDTV), military imaging, medical imaging, remote sensing, satellite imaging, and underwater imaging. It has been well established that integrating both the hardware and software capabilities can produce the required HR image more economically. The HR images can be effectively produced using digital image processing algorithms. Image interpolation uses a single LR image to obtain the HR image. On the other hand, image super-resolution reconstruction uses multiple degraded LR image observations of the same scene to generate a single HR

image.

## 1.5 MOTIVATION

The field of visual secret sharing (VSS) has emerged as a digital cryptography technique distinct from the conventional approaches. This field has evolved notably in recent years. Still, this field has many challenges and opportunities mentioned below.

- The principle of less computation for massive pixel processing during the VSS inception is intuitive under the limited execution resources. Later emerged VSS approaches were computationally expensive and can leverage advancements in latency optimized multi-core systems, to increase their execution speed. But the computational complexity disqualifies these VSS schemes for real-time applications. Therefore, the need of the hour is to investigate VSS approaches that meet real-time demand.
- In addition, the processing power on a single multi-core system has made a paradigm shift to the throughput optimized many-core GPGPU. To the best of our knowledge, there are no VSS techniques that can exploit the data-parallel tasks in them to delegate to the GPGPU during run-time. GPGPUs are evolving with increased optimizations and execution resources. Therefore, the GPGPU-based VSS schemes offer scalable solutions to meet real-time demand.
- The quality of the reconstructed image is another bottleneck in the VSS schemes. Any effort to improve the quality intensifies computational cost worsening the VSS scheme's applicability to real-time applications. Therefore VSS schemes in improving the contrast, in turn, motivate to utilize GPGPU power to meet such matured demands.
- The evolving super-resolution techniques aim to improve the contrast quality of the input image. The use of deep neural networks for super-resolution has proved the phenomenal performance of the output image quality. Very few VSS schemes took advantage of super-resolution in improving the quality of the recovered image. Moreover, computationally intensive super-resolution further motivates to

utilize GPGPU to improve the speedup while improving the quality of the recovered image.

- Medical images pose additional challenges when VSS approaches are applied to secure them. Their inherent poor quality further degrades the quality of the reconstructed secret image. In addition, their health-criticality demands real-time computational VSS schemes for security reasons.

This dissertation considers these issues in the VSS field to present novel GPGPU-based VSS schemes.

## 1.6 THESIS CONTRIBUTIONS

For improving the visual reconstruction quality of the secret and the execution time, this thesis presents various GPGPU based VSS schemes. These schemes proved significant high performance over the existing state-of-the-art approaches.

1. VSS in Shyu (2009) extended the concept of random grid VSS introduced by Kafri and Keren (1987) to the colour images. However, the scheme is not suitable for real-time applications due to the massive pixel processing. Therefore we presented a random grid-based model to improve the speedup by exploiting the computational power of GPGPU (Holla and Pais 2021c). The experimental results of the presented random grid-based VSS approach proved its efficiency over the traditional random-grid method with a considerably increased speedup as the secret image size increases. This scheme outperforms with a speedup of  $2688\times$  compared to the the conventional method.

The existing VSS schemes reconstruct the original secret image as a halftone image with only a 50% contrast. A technique, namely, the Randomized Visual Secret Sharing (RVSS), overcomes the disadvantages of existing VSS schemes by recovering the secret image with a contrast of 70% to 80% for noise-like shares and 70% to 90% for meaningful shares (Mhala et al. 2017). But RVSS is computationally expensive. We presented a GPGPU based Randomized Visual Secret Sharing (GRVSS) technique (Holla et al. 2020) that leverages data parallelism



in the RVSS pipeline. GRVSS achieved a speedup in the range of  $1.6\times$  to  $1.8\times$  with four shares and  $2.63\times$  to  $3\times$  with eight shares for the test images of distinct sizes while retaining the contrast of RVSS.

2. We presented an effective VSS with super-resolution utilizing quantum logic and enthalpy-based adaptive deep neural network, to further enhance recovered image quality with reduced time complexity of deep learning (Holla and Pais 2021b). When compared with the RVSS and other benchmarks, the presented work is proved proficient with the empirical values, say Mean Squared Error (MSE), and Normalized Absolute Error (NAE). For the noise-like shares and meaningful shares, the average MSE of the presented work is 0.002 and 0.184 and the average NAE is 0.1217 and 0.4863. We achieved recovered image contrasts of 96.6% to 99.8% for noise-like shares and 64% to 80% for meaningful shares. Moreover, the GPGPU model's deep learning time is almost half the sequential learning times with a speedup of  $1.92\times$ .
3. We presented an effective secret image sharing model with super-resolution utilizing a Contrast-adaptive Convolution Neural Network (CCNN) (Holla and Pais 2021a). This model exploits the computational power of the GPGPU to achieve a better quality of the recovered image and speedup. The objective quality assessment proved that this model produced a high-quality reconstructed image with the recovered image contrasts of 97.3% to 99.7% and 78.4% to 89.7% having the Structural Similarity Index Measure (SSIM) of 89% to 99.8% and 71.6% to 90% for the noise-like shares and the meaningful shares respectively. The presented technique demonstrated a speedup of  $800\times$  in comparison with the sequential model.
4. We presented a novel GPGPU-based MISS for the histopathological medical images to achieve a high-resolution recovered image in real-time. A Convolution Neural Network (CNN) for super-resolution produces a high-contrast reconstructed image. We evaluated the presented model using standard objective assessment parameters and the Computer-Aided Diagnosis (CAD) systems. The

result analysis confirmed the high-performance of the MISS with a 99.3% contrast and a 98% SSIM of the reconstructed image. We achieved a categorization precision that fits the CAD systems. We used cluster sizes of 300, 500, and 1000 to obtain the MISS model's CAD performance measures accuracy, sensitivity, specificity, precision, and F-measure using a novel fused feature extractor and a random forest classifier. We achieved a maximum accuracy of 99.71% for the cluster size of 300, 100% sensitivity (Recall) for cluster sizes 500 and 1000, 100% specificity for the cluster sizes of 300 and 1000, 100% precision for a cluster size of 1000, and 100% F-measure for a cluster size of 1000. Also, we attained an overall speedup of  $800\times$  over the sequential model, and 98.2% SSIM.

### 1.7 THESIS ORGANIZATION

The rest of the thesis is organized as follows:

- **Chapter 2: Literature review:** The survey on various existing VSS techniques is discussed in Chapter 2.
- **Chapter 3: GPGPU-based Randomized VSS:** This chapter presents two GPGPU-based randomized VSS schemes for grayscale and colour images with improved performance.
- **Chapter 4: VSS using quantum logic and GPGPU-based EDNN super-resolution:** This chapter presents an effective secret image sharing using quantum logic and GPGPU based EDNN Super-Resolution.
- **Chapter 5: GPGPU VSS by contrast-adaptive CNN super-resolution:** This chapter presents an efficient GPGPU VSS by contrast-adaptive Convolutional Neural Network (CNN)-based Super-Resolution.
- **Chapter 6: Medical image secret sharing using super-resolution for CAD systems:** This chapter presents high-performance medical image secret sharing scheme using super-resolution for the CAD systems.

- **Chapter 7: Conclusions and future scope:** This chapter concludes all the presented techniques of this thesis and provides some recommendations for the future research directions.



## **CHAPTER 2**

### **LITERATURE REVIEW**

There exists substantial work in the field of VSS schemes. In this chapter, we survey different VSS techniques with their salient features. The implications of these feature enhancements demand intensive computations with low power consumptions to fit those techniques for real-time and light weight applications.

This chapter is organized as follows. Section 2.1 begins with VSS. A brief taxonomy of VSS schemes and two fundamental VSS methods are discussed in section 2.2. The feature enhancements in size-invariant VSS schemes are presented in section 2.3. This section also gives insight into the necessity of state-of-the-art GPGPU-based VSS schemes and GPGPU-based VSS schemes particular to the medical images. Section 2.4 covers super-resolution image reconstruction techniques and challenges therein. The evolution in multi-core and many-core systems together constituting heterogeneous computing is explained in section 2.5. Sections 2.6 and 2.7 give problem description and problem statement. Thesis objectives are listed in section 2.8. Section 2.9 summarizes this chapter.

#### **2.1 VISUAL SECRET SHARING**

The explosive growth in the Internet of Things (IoT) continuously demanding rapid software development cycles. It is natural that such rapid advancements in technology compromise security requirements. The information produced remains significant only when it is safe and trustworthy. Therefore information security brings value to the

information available on the Internet. In the context of social media, coloured images draw much attention because human brains respond to images and colours quickly in comparison to other types of information. The importance of visual information has manifested in the emerging Visual Secret Sharing.

The encryption is the process of maintaining the confidentiality of information among authorized users. The VSS is an image encryption technique that focuses solely on encrypting image information. The Visual Cryptography (VC) is another nomenclature to VSS. The encrypted images are called shares or transparencies. Each share is not self-sufficient to reveal information. This property enables safe share transmission over public communication channels. The shares are distributed to the authorized users during decryption. A group of a pre-qualified number of transparencies is sufficient to obtain the secret image. The quality of this secret is usually approximate to the original confidential image. The image becomes transparent to the human visual system when shares stacked over each other. That is the reason the term transparency used to refer to shares.

The research in VSS started with a fundamental objective of bringing computationally efficient image encryption and decryption methods. This efficiency need is intuitive as a massive amount of data processed in an image. The investigations on novel VSS approach led to their refined features. More the randomness in any cryptography algorithms more is the security. But cryptography schemes with complete randomness does not allow for decryption to succeed. The same is the case in VSS. Therefore more controlled randomness in such schemes proved the restored image approximately equal to the original secret image. The literature has shown an increasing number of novel protection methods for securing images over the Internet. These secure image sharing techniques overcome the traditional cryptographic approach, providing new solutions for the development of new and secure imaging applications.

### 2.2 VSS TAXONOMY

The VSS schemes differ primarily concerning the following features:

- Number and type of input images to be encrypted

- Number, size, and type of shares generated
- The way the shares stacked during decryption

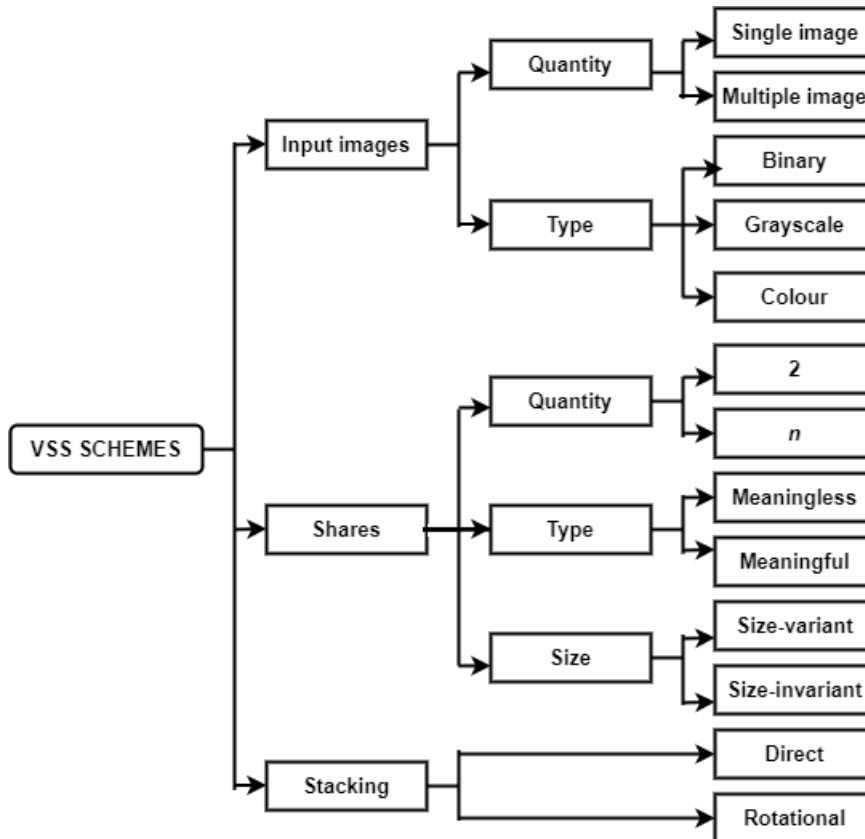


Figure 2.1: VSS taxonomy.

Figure 2.1 shows the taxonomy of VSS schemes based on the above features. These classes are not necessarily exclusive in the sense that a given VSS technique may belong to more than one class. For example, a VSS scheme designed for a colour image can secure a grayscale image with equal potential. As this classification has emerged consequent to the VSS featural refinements, the VSS schemes with advanced features in a particular category generally belong to the other classes that existed before. We now briefly focus on only two types of VSS schemes based on the size of the generated shares relative to the original secret image.

**1. Size-variant VSS schemes:** The history of size-variant VSS dates back to 1994 with the technique proposed by researchers Naor and Shamir (1994) with an objective of minimum computation during encryption and no calculation for its decryption. For

Table 2.1: The basis matrices for  $n = 2$

$$M^0 = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \quad M^1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

a given secret image  $I$ , two shares say  $S_0$  and  $S_1$ , are needed. Each share corresponds to one participant. Because  $I$  contains transparent and opaque pixels, two  $(2 \times 2)$  basis matrices  $M^0$  and  $M^1$  generated as shown in Table 2.1. A given white pixel of  $I$  is shared in the respective positions in  $S_0$  and  $S_1$  by selecting randomly one row of  $M^0$ . Similarly, for the opaque pixel in  $I$ , a randomly selected row from  $M^1$  is encoded in the corresponding positions of  $S_0$  and  $S_1$ . The ratio of a given pixel in the secret image to the corresponding shared pixel in each share is  $1 : 2$ . This pixel expansion makes each share double the width of the secret image, thereby degrading the contrast of the obtained image by at least 50%.

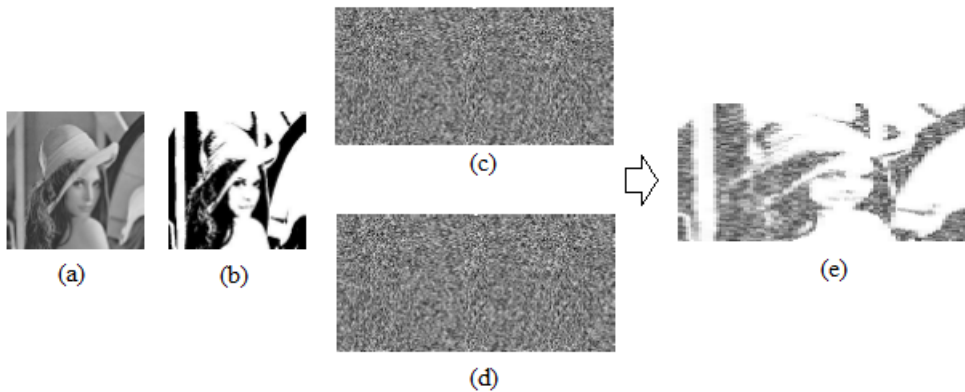


Figure 2.2: Size-variant VSS scheme.

Note: (a) A grayscale input image Lenna ( $256 \times 256$ ) (b) Halftone image ( $256 \times 256$ ) (c) Share 1 ( $256 \times 512$ ) (d) Share 2 ( $256 \times 512$ ) (e) Recovered image ( $256 \times 512$ ).

The general case of this approach can generate shares of any size larger than the original confidential image. Such schemes are called size-variant VSS schemes. The overlaying of shares reconstruct the secret image. The  $k$  out of  $n$  VSS approaches are less restrictive with a minimum  $k$  out of  $n$  generated shares required to decipher the secrecy (Naor and Shamir 1994). The basis matrices and the pixel multiplicity amounts to extra storage cost. The transmission of expanded shares occupies significant channel bandwidth while maintaining confidentiality. Besides the design of basis matrices takes



considerable effort (Chen and Tsao 2011a). Figure 2.2 visualizes size-variant VSS technique for the grayscale image. Figure 2.2 (a) is the secret image. This image is usually converted to a halftone image to save space and computation. Figure 2.2 (b) is the corresponding halftone image. Figure 2.2 (c) and Figure 2.2 (d) the share images generated as an outcome of the share generation phase, assuming two authorized users. The size of these two shares is double the width of the secret image. The decryption is achieved with one share kept over the other to reveal the secret image as shown in Figure 2.2 (e). The width of the reconstructed image is also double the secret image width.

**2. Size-invariant VSS schemes:** Much earlier seminal work on size-invariant VSS scheme by Kafri and Keren (1987) has evolved over the last decade. The underlying encryption algorithms introduced in Kafri and Keren (1987) uses two random grids G1 and G2. Random grids are matrices containing 0s and 1s. The random grid G1 is a matrix having a 50% probability of 0s and 1s. This matrix is the master-random grid as its pixel values used to set G2 based on the corresponding pixel value in secret image  $I$ . G2 is the encoded random grid. The number of different master grids for a matrix of  $N$  pixels is  $2^N$ . These random grids themselves are the shares in random-grid size-invariant VSS schemes. The secret image revealed when these two random grids overlaid. Otherwise, the human eye cannot identify dissimilarity between the master and the encoded random grids. Figure 2.3 illustrates a size-invariant VSS scheme where the size of the shares (Figure 2.3 (c) and Figure 2.3 (d)) and the reconstructed image (Figure 2.3 (e)) is equal to the size of the secret image. There are three variants of the

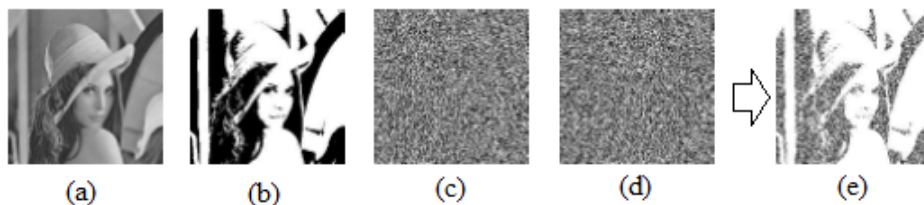


Figure 2.3: Size-invariant VSS scheme.

Note: (a) A grayscale input image Lenna ( $256 \times 256$ ) (b) Halftone image ( $256 \times 256$ ) (c) Share 1 ( $256 \times 256$ ) (d) Share 2 ( $256 \times 256$ ) (e) Recovered image ( $256 \times 256$ ).

algorithm proposed by Kafri and Keren (1987). The excerpts of these three algorithms

are shown in equations 2.1, 2.2, and 2.3.

$\forall [i,j] (0 \leq i < \text{height of the image} \ \& \ 0 \leq j < \text{width of the image})$

$$G2[i, j] = \begin{cases} G1[i, j] & I[i, j] = 0 \\ \overline{G1[i, j]} & \text{otherwise} \end{cases} \quad (2.1)$$

$$G2[i, j] = \begin{cases} G1[i, j] & I[i, j] = 0 \\ \text{random}(0 \text{ or } 1) & \text{otherwise} \end{cases} \quad (2.2)$$

$$G2[i, j] = \begin{cases} \text{random}(0 \text{ or } 1) & I[i, j] = 0 \\ G1[i, j] & \text{otherwise} \end{cases} \quad (2.3)$$

The size-invariant VSS schemes are preferred over the size-variant VSS schemes as they reduce computational, storage, and communication costs in real-world applications. In the next section, we discuss the feature enhancements in size-invariant VSS schemes over time. These enhancements demand additional computational resources to make such VSS schemes suitable for time-critical applications.

### 2.3 FEATURE ENHANCEMENTS IN SIZE-INVARIANT VSS SCHEMES

The work in Shyu (2009) is a variant of random grid technique proposed by Kafri and Keren (1987). The random grid technique is extended to encrypt the colour images. The input grayscale image is converted to a halftone image, and then the random grid technique is applied directly to this halftone image. The brighter and darker area in the input grey image results in a sparse and darker area in the converted halftone image. Halftone equivalent saves memory with quality (Lou et al. 2011). It means that the converted halftone image looks pretty similar to the grayscale image. There are many techniques to get a halftone image from a grayscale image. The error diffusion technique is used in Shyu (2009) to convert grayscale to a halftone image.

For the colour images, there are two colour models based on the sources of colours. If the colours originated from the mix light sources, the additive colour model is used. If the colours are from natural colourants like paints, pigments, or dyes, a subtractive colour model is used. This technique considers the subtractive colour model with the assumption that colourants are painted on transparencies. There are three primary inde-

pendent colours in the subtractive model. They are Cyan (C), Magenta (M) and Yellow (Y). Other colours can be considered the linear combination of these primary colours. Alternatively, it is possible to decompose a pixel to its equivalent three monochromatic coloured-grey level C, M, and Y components. These three images converted to coloured halftone images. Three sub-master random grids generated as done with binary halftone image. Three sub-encoded random grids generated using the technique presented in Kafri and Keren (1987). Then the corresponding colour components of sub-random grids are mixed to obtain the two coloured master and encoded random grids. When these two random grids are superimposed, the colour secret image gets revealed.

The scheme in Lou et al. (2011) uses two halftone cover images and the secret image as input and produces two shares. The secret image is affixed in two halftone cover images. This method assumes each of three images as  $2 \times 2$  blocks, which do not overlap. This scheme swaps the pixels based on a specific condition. The block size of  $2 \times 2$  is convenient to ensure exchange between adjacent pixels. A larger block size increases the distance between the pixels to be exchanged and may lead to distortion. Also, larger blocks make abundant possibilities of swapping among the pixels and makes computation expensive. The blocks in the two cover images are modified based on the respective block of the secret image. Processing all the blocks results in two shares.

The notion here is that the dark (white) block of the secret image makes the corresponding stacked block almost dark (white). It dispenses the modified pixels to the two share images. This scheme also allows to implant another secret information in addition to the secret information related to the secret image. This provision fulfilled another concern of security called authenticity. Recovering the secret image is possible once the shares are overlaid. The authentication secret information can be recovered by shifting one share to the right by a predefined amount, say half the width of the secret image, over the other share. This scheme assumes that the height of the secret image, two cover images and the authentication secret information is the same. The difference lies in the width of the authentication image, which is equal to half of the secret image. This technique is applicable to the colour images.

Hou and Quan (2011) presented a progressive VC which recovers the secret image

Table 2.2: The sharing matrices used in Hou and Quan (2011)

$$M^0 = \begin{bmatrix} 1 & 1 & \cdots & 1 & 1 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \quad M^1 = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

gradually by superimposing more and more shares. Less number of shares reveal only a skeleton of the secret image. The detail of the produced secret image increases progressively as more shares are superimposed. This scheme uses  $n \times n$  sharing matrices  $M^0$  and  $M^1$ , for white and black pixels respectively, as shown in the Table 2.2. Each row of the matrix is a sharing method. Every participant is assigned (0 for white and 1 for black) the corresponding column.

Each pixel on the shares has only  $\frac{1}{n}$  chance to occur as black irrespective of whether it is on account of white or black pixels of the secret image. A black pixel of the secret image shares a black on different shares at different positions. It increases the chance of being black for the black area of the secret image as the stacking gradually results in more and more shares. If the first row of  $M^0$  is chosen for the white pixel of the secret image, then a black occupy the same position in all shares. Thus the chance that the black for the white pixel of the secret image is  $\frac{1}{n}$ . It improves the contrast of the shared pixel when more and more shares are stacked. Thus the secret content is revealed progressively. Each column in  $M^0$  and  $M^1$  contains only one 1. Hence the chance of being a 1 on each share is  $\frac{1}{n}$ . This value is regardless of whether it is due to the white or black pixel of the secret image. It helps in achieving confidentiality of the secret image in the generated shares. If the number of shares stacked is  $k$ , then chances of a black for a white pixel of the secret image is  $\frac{1}{n}$ . The chances of a black for a black pixel increases to  $\frac{k}{n}$ . It results in a contrast of  $\frac{(k-1)}{n}$  on the stacked image. The contrast between white and black increases as the number of shares stacked increased. Hence the content gets revealed progressively. This scheme (Hou and Quan 2011) improves contrast to  $\frac{(n-1)}{n}$  when all shares are superimposed. This technique does not expand the pixels. Thus the scheme meets security, quality and space concerns.

As per Chen and Tsao (2011b), the scheme proposed by Kafri and Keren (1987) is not participant friendly. It means that created transparencies are meaningless. The data in shares is huge, and hence users feel that it is difficult to manage. On the other hand, there is progress in the direction of visual cryptography that generates meaningful shares. These meaningful shares in the form of some shape or information make the shares easily managed. These methods can also be called friendly schemes. Most of these techniques are not space-efficient as they generate large shares. But the concept of meaningful shares does not ally directly with the random grid schemes, as the random grid is based on unexpanded shares. Therefore Chen and Tsao (2011b) designed a novel scheme to yield a friendly random-grid based scheme. This scheme is user-friendly without pixel expansion. The procedure here uses input images, namely a logo image  $L$ , the secret image  $I$ , and two random grids ( $R1$  and  $R2$ ) all equal in size.  $R1$  and  $R2$  are encoded based on  $I$  and  $L$ . Stacking  $R1$  and  $R2$  recovers  $I$ . It distinguishes different light transmissions on a random-grid based on the pixel values of the logo image. A varying-parameter can adjust the visual quality between meaningful random-grids and superimposed results. Hence this scheme is friendlier for the dealer.

Lee and Chiu (2011) proposed the creation of meaningful shadows for general access structures with no pixel multiplicity. There are two stages in this scheme. Firstly, the construction of meaningless shares based on a specified access structure with an optimization strategy. Then a cover image is added to each shadow directly to obtain meaningful shadows using a marking algorithm. The contrast of the shadows and the resulting image depends on the ratio of the incremental pixel density of the cover image. The contrasts of the shares and the recovered image are mutually orthogonal. An increase in one results in a decrease in the contrast of the other. Lee and Chiu (2011) scheme addresses most of the drawbacks of the earlier works. The contrast of the recovered image in this scheme is below 50%.

The observations in Hou et al. (2013b) about the previous VSS are as follows. Most of the conventional random grid approaches yield poor contrast because they are probabilistic. Also, they produce meaningless shares. The transmission of these meaningless shares may create suspicion of an adversary that these shares may contain secret infor-

mation. This situation may invite the adversary to put effort into revealing rather than preventing not to disclose the secrecy. Few restrictions in the scheme presented in Chen and Tsao (2011b) are the pixels for the shadow images chosen from both confidential image and the protection-image. Hence, the resulting image and also the shadows themselves having low quality. This approach is restrictive in the sense that only one protection image is used. Also, the lack of dark pixels may lead to recovery impossible. The corresponding colour pixels in two shares were complementary to each other. The approach, when used with two cover images, turned out to be unnatural.

The research in this direction in Hou et al. (2013b) led to the capability of generating not only noise-like shares but also, the meaningful shares. The approach is based on the random-grid and also, non-expanded VSS. The spread of dark pixels on both the shadows and the reconstructed image were analyzed. Then, a pixel allocation was accomplished with a probabilistic approach. The contrast reached was the theoretical maximum with the flexibility to adjust it. The light and dark contrast in the image produced with these two different probabilities show a black and white pattern. Also, it provides flexibility to adjust the contrast. So the contrast on the share images and the stack image can reach the theoretical maximum.

A versatile scheme, the Block-Based Progressive Visual Secret Sharing (BPVSS) technique, was presented in Hou et al. (2013a). This scheme works like a jig-saw puzzle. This method applies to both grayscale and colour images with better contrast. However, it suffers from the problems like recovery of multi-tone secret image as the monotone image. Also, BPVSS recovers images with maximum contrast of 50% for grayscale and colour images. To overcome the issues of BPVSS, a novel Randomized Visual Secret Sharing (RVSS) scheme was presented in Mhala et al. (2017). This scheme combines BPVSS for generating secret shares and reversible DCT data embedding technique. Security being the underlying theme of cryptography, this scheme is more secure and systematic due to the randomness in its share generation and image reconstruction phases. The four phases of the RVSS scheme are 1) Share generation, 2) Data hiding into shares, 3) Extract hidden data & restore shares, and 4) Image reconstruction. These four steps reformulated into primary two phases as 1) Sender block

and 2) Receiver block. The sender module consists of block-wise share generation followed by, data hiding into the shares. The receiver module consists of the inverse process to recover the secret image. In the inverse process, embedded data is extracted and coefficients of all shares are restored. Using these parameters, the original image is reconstructed and extracted data is added to improve the quality of the image.

Table 2.3 shows the features offered by a few VSS schemes to secure a single secret image based on four primary criteria explained below.

1. **Input image feature:** Input secret image to the VSS scheme may be a halftone equivalent of the grayscale image or a colour image. Lou et al. (2011) (column labelled VSS 5 in table 2.3) proposed a single secret image scheme that uses two cover images along with an authentication image. Such schemes obviously are core-intensive in nature as there are more images to be processed. The scheme proposed by Mhala et al. (2017) (column labelled VSS 9 in Table 2.3) is a single secret image sharing without any cover images or authentication images. Still, it is computationally intensive due to the inherent computational tasks to provide secret sharing. VSS schemes to secure grayscale input images relatively consume less core and storage compared to the VSS schemes meant for both grayscale and colour images.
2. **Share image features:** The shares are generated from the corresponding halftone images. Table 2.3 shows some schemes that can generate only two shares and other flexible schemes generating  $n$  shares, where  $n$  is the number of participants. The computational complexity of the flexible schemes reaches a prohibitive level as the number of participants increase.
3. **Secret image reconstruction features:** Few decryption procedures require a minimum threshold of  $k$  ( $k \leq n$ ) shares stacked to reveal the secret image. Such schemes are referred to as  $(k, n)$  schemes. Otherwise, either 2 or all  $n$  shares need to be superposed on each other leading to  $(2, 2)$ ,  $(2, n)$ , or  $(n, n)$  schemes. The shares generated look either noise like or meaningful. The reconstruction in many cases reveals the secret image if all shares are overlaid otherwise nothing can be

Table 2.3: The characteristics of various VSS schemes

VSS Schemes	VSS 1	VSS 2	VSS 3	VSS 4	VSS 5	VSS 6	VSS 7	VSS 8	VSS 9
<b>1. Input image features</b>									
# of input images	1 secret	1 secret	1 secret	1 secret	1 secret 2 cover	1 secret	1 secret 1 authentication	1 secret 2 cover	1 secret
Secret image type	Halftone	Halftone/ Color	Halftone	Halftone/ Color	Halftone/ Color	Halftone	Halftone/ Color	Halftone/ Color	Halftone/ Color
<b>2. Share image features</b>									
# of share images	$n$	$n$	2	$n$	2	$n$	2	2	$n$
Share size variant?	Yes	No	No	No	No	No	No	No	No
Share type	Noise-like	Noise-like	Noise-like	Noise-like	Meaningful	Noise-like	Meaningful	Noise-like/ Meaningful	Noise-like/ Meaningful
<b>3. Secret image reconstruction features</b>									
# of shares stacked	$k \leq n$	$k \leq n$	2	$n$	2	$k \leq n$	2	2	$n$
All/Nothing(AN) or Progressive (P) ?	AN	AN	AN	AN	AN	P	AN	AN	P
Contrast degradation?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
<b>4. GPGPU-based ?</b>	No	No	No	No	No	No	No	No	No

**VSS schemes:** VSS 1-(Naor and Shamir 1994), VSS 2-(Chen and Tsao 2011a), VSS 3-(Kafri and Kerem 1987), VSS 4-(Shyu 2009), VSS 5-(Lou et al. 2011), VSS 6-(Hou and Quan 2011), VSS 7-(Chen and Tsao 2011b), VSS 8-(Hou et al. 2013b), VSS 9-(Mhala et al. 2017).



discerned. Such schemes are called All/Nothing schemes. In addition to All/Nothing schemes, Table 2.3 also shows  $k$  threshold VSS schemes that reveal the secret image only when minimum  $k$  ( $k \leq n$ ) shares are stacked over each other. Another category of VSS schemes namely progressive VSS schemes reconstructs the secret image progressively as the shares are stacked. It is nevertheless noteworthy that the contrast of the reconstructed image in all these categories is low despite the continual effort in improving it. However, any effort to improve the quality of the recovered image leads to additional computation.

4. **GPGPU-based:** As evident from Table 2.3, despite VSS schemes increasingly demanding more computations for the reasons discussed above, no VSS scheme hitherto utilized emerging GPGPU computational resources to meet those demands. In other words, GPGPU technological advances pave the way for inventions in GPGPU-based VSS schemes.

As stated in criteria 3 above, the contrast row in Table 2.3 indicates that there is still scope for improving the contrast of the restored secret image. The use of evolving SR techniques is handy in restoring an HR image from diverse LR image sources. Though there are many SR reconstruction techniques in the literature, none of them has been implemented in GPGPU. Also, as mentioned in criteria 4 above, there is no work found in the GPGPU implementation of VSS. However, all those schemes have expressed the need for improving the speed during secret sharing.

Despite the continued advancements in the GPGPU parallelism that the qualified VSS schemes can leverage, there is no work on this endeavour. The four recent review literature published in reputable journals, say, Punithavathi and Geetha (2017), Sharma et al. (2018), Chanu and Neelima (2019), and Ibrahim et al. (2021), complement the dearth of GPGPU-based VSS schemes to date. These findings also pave the way for understanding the rationale, challenges and opportunities to exploit GPGPU in the VSS domain a priori.

On the other hand, Medical image secret sharing (MISS) is an emerging VSS field to address the performance challenges in sharing medical images, such as efficiency

and effectiveness. Medical imaging systems produce images conveying information about the structure and functioning of the human organs. These medical images pose several challenges to automate their analysis owing to their low quality. The trade-offs in the design and manufacturing of such medical equipment lead to compromise the quality degradation of the images they produce, with an absence of useful information (Qiu et al. 2021). In the thirst to improve the medical images' effectiveness, medical-practitioners started accepting the concept of super-resolution (SR) assisting medical diagnosis (Deeba et al. 2020). The biomedical advancements necessitated the communication of medical images over the public channels. Therefore, recent investigations started focusing on the privacy and security of medical images (Pandey et al. 2020).

In addition to work on GPGPU-based VSS schemes being scarce, there is little work in the field of MISS exclusively designed and applied to secure the medical images. A recent review article by Zhang et al. (2020) on medical image confidentiality (or security) technology complements this fact. The authors collected the recent five years 123 papers from reputable academic sources to build a coherent taxonomy. Among the five classes they identified, MISS, being a sub-category of type medical image security algorithms, occupied only 3% of the literature. Further, they classified the MISS into 1) Shamir's MISS schemes (Marwan et al. 2019; Sah et al. 2018) and 2) Visual Cryptography based MISS schemes (Bakshi and Patel 2019; Kanso and Ghebleh 2018) with only two references per category.

Also, SR techniques and the VSS schemes cannot be used together in real-time applications unless we leverage additional computational resources. On the other hand, both SR and VSS techniques intrinsically operate on massive data of matrices representing images. The heterogeneous GPGPU computing offers an improved speed up by exploiting such data-level parallelism. In this way GPGPU SR and VSS techniques enable them to be useful in time and quality-critical applications. In this context, we discuss SR techniques and heterogeneous computing in the following sections.

## 2.4 SUPER-RESOLUTION (SR) IMAGE RECONSTRUCTION

The scope of this section is limited to the spatial resolution defined by the pixel density. Image resolution gives the details of the image; the higher the resolution, the more image details. Digital imaging applications use high-resolution images for further investigations. They also appeal better to the human visual system. Hence high-resolution images find significance in both human and machine perception. The limitations of image acquisition systems and image processing devices constrain spatial resolution. As an alternative, to overcome those limitations by constructing high-resolution devices is also prohibitively expensive. It is impractical to expect such high-resolution devices in most real applications. Sometimes physical constraints also limit the device resolution used in applications.

Super resolution is a way to address these problems. It accepts the image degradations and post-processes the captured images to reconstruct better quality equivalent images. SR uses signal processing techniques to reconstruct the image. Therefore SR is a technique to trade-off computational cost with the hardware cost. Super-Resolution (SR) refers to techniques for reconstructing high-resolution (HR) images from several observed low-resolution (LR) images. While doing so, SR increases the high-frequency components and alleviates the degradations. The notion in SR is that a HR image can be constructed by combining non-redundant information available with many LR images. The SR is more flexibly constrained as it rely on many LR images rather than a single image.

The non-redundant information in LR images is introduced due to the subpixel shifts between them. The uncontrolled motion between image acquisition system and the scene produces such subpixel shifts. E.g., the satellite imaging system moves in its orbit round the earth with specific speed. The SR is possible only if there exists a sub-pixel shifts. Therefore SR can be considered the techniques that better conditions ill-posed up-sampling problem. The SR techniques are applied in Medical imaging (Cristani et al. 2004; Kennedy et al. 2006; Maintz and Viergever 1998; Malczewski and Stasinski 2008; Peled and Yeshurun 2001), surveillance video to freeze a frame and zoom region of interest (ROI) (Li et al. 2008; Lin et al. 2005), Video standard

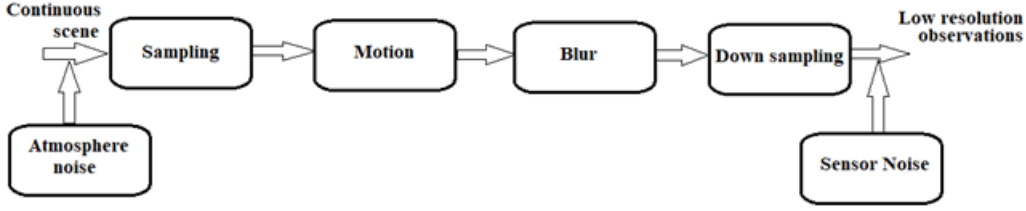


Figure 2.4: Image observation model.

conversion, and remote sensing (Joshi et al. 2005).

### 2.4.1 Observation model of a real imaging system

The real imaging system is always not perfect. As a result, they always produce decimated image (Park et al. 2003). Figure 2.4 depicts a model presented in Farsiu et al. (2004a). Several factors in the low resolution imaging system pipeline ultimately produce noisy, decimated, and blurred image. This model can be formulated into a linear system as follows. Let  $X$  denote the HR image desired, and  $Y_k$  be the  $k^{th}$  LR observation from the camera. Assume the camera captures  $Q$  LR frames of  $X$ , where the LR observations are related with the HR scene  $X$  by the equation 2.4.

$$Y_k = D_k H_k F_k X + V_k \quad (1 \leq k \leq Q) \quad (2.4)$$

where  $F_k$  encodes the motion information for the  $k^{th}$  frame,  $H_k$  models the blurring effects,  $D_k$  is the down-sampling operator, and  $V_k$  is the noise term.

A large linear system by rearranging these linear equations is shown below.

$$\begin{bmatrix} Y_1 \\ \vdots \\ Y_k \end{bmatrix} = \begin{bmatrix} D_1 H_1 F_1 \\ \vdots \\ D_k H_k F_k \end{bmatrix} \mathbf{X} + \mathbf{V} \quad (2.5)$$

The matrices  $D_k$ ,  $H_k$ ,  $F_k$ , or  $M$  are very sparse, and this linear system is typically ill-posed. Also, these matrices need to be estimated in real imaging systems, from the available LR observations. As a result the problem turns out to be not well conditioned. Therefore HR requires proper prior regularization. Many SR techniques try to model these degradations fully or partially.

### 2.4.2 SR Techniques

1. **Frequency Domain SR:** Tsai (1984) stated that the HR image is analogous to the multiple shifted low-resolution images in the frequency domain. The assumption here is that the shift and aliasing properties have properties of Continuous and Discrete Fourier Transforms. A set of linear equations to solve  $\underline{X}$  is given by equation 2.6.

$$\underline{Y} = \phi \underline{X} \quad (2.6)$$

where  $Q$  is the number of shifted images,  $\underline{Y}$  is a  $Q \times 1$  column vector with the  $k^{th}$  element being the Discrete Fourier Transformation (DFT) coefficient  $Y_k[r1, r2]$ ,  $\phi$  is a  $Q \times N_1 N_2$  matrix relating  $\underline{X}$  and  $\underline{Y}$ ,  $\underline{X}$  is a  $N_1 N_2 \times 1$  vector, and  $N_1$  and  $N_2$  representing sampled LR images. Then inverse DFT is used to reconstruct the image. Though there are many variants of the scheme, they did not go beyond initially presented technique in Tsai (1984). Though these techniques are computationally efficient, they have limitations in handling complicated image distortion models. They use many image priors for regularization. Therefore spatial domain SR models predominated over the other categories.

2. **Non-Iterative SR:** In non-Iterative SR, a sparse linear model in the spatial domain (shown in equation 2.7) relates the HR image and the LR images. It uses many estimators for the SR reconstruction.

$$Y_k = D_k H F_k X + V_k = D_k F_k Z \quad (1 \leq k \leq Q) \quad (2.7)$$

where,  $H$  denotes the Linear Spatial Invariant (LSI) assuming  $H_k$  is same for all  $Q$  LR frames,  $F_k$  models simple motion with  $H$  and  $F_k$  commuted (Elad and Hel-Or 2001; Farsiu et al. 2004b). Then non-iterative SR reconstructs the image with a non-iterative forward interpolation and restoration approach containing the following three stages. 1) LR image registration, 2) Obtaining  $Z$  using non-uniform interpolation, and 3) Getting  $X$  by deblurring and noise removal. Non-Iterative method has the following demerits. It does not provide an estimation optimality guarantee. The errors during initial registration can propagate to the subsequent processing stages. It uses suboptimal interpolation as it does not assume the blurring and noise effects. The approach requires a remedy to minimize aliasing

without proper regularization using HR image prior.

3. **Statistical SR:** Statistical SR approaches relate the SR reconstruction steps stochastically toward optimal reconstruction. The stochastic variables represent the HR image and motions among low-resolution inputs. Assuming motion vector  $v$  and blurring kernel  $h$  defines the degradation matrix  $M(v, h)$ , a Bayesian framework can define the SR reconstruction (Hardie et al. 1997). The Maximum a Posteriori (MAP) for SR is given by the equation 2.8.

$$X = \arg \min_x \{ \|\underline{Y} - MX\|^2 + \lambda F(X) \} \quad (2.8)$$

where  $\lambda$  absorbs the noise variance and  $F(X)$  is a non-negative function. Bayesian framework approaches deprecate unknowns and are hence optimal. But these SR models are computationally tractable only in simple cases with simple image priors or registration estimates. Therefore computation is also a primary concern when applying these approaches to complex use-cases.

4. **Example-Based SR:** Example-based methods use sampling of other images in developing the prior. A category of this method directly uses the examples (Freeman et al. 2002) by maintaining two sets of training patches namely, HR image samples  $\{(x_i)_{i=1}^n\}$  and LR image samples  $\{(y_i)_{i=1}^n\}$ . The observation model (equation 2.9) connects each pair  $(x_i, y_i)$ .

$$(y_i) = DH(x_i) + v \quad (2.9)$$

This model can predict the HR image for the target image in a patch-based manner. This approach performs well with the available inadequate observations, optimal patch size for the specific target image, and proper choice of database. They are computationally complex models.

5. **Projection onto Convex Sets (POCS) SR:** The POCS methods use multiple constraining convex sets. The image needed is a point within such sets. Defining the convex sets with various constraints or priors is flexible. However, POCS is computationally expensive and slow in converging. POCS methods do not guarantee a unique solution using an initial guess (Elad and Feuer 1997). These techniques allow priors on system blurs and the motion parameters. They do not estimate reg-

istration parameters and the HR image together. The solution is to use a hybrid technique that can combine a stochastic with the POCS.

### 2.4.3 Challenges in SR

- **Image Registration:** Image registration fuses complementary spatial samplings of the HR image. This problem is ill-posed particularly in the SR setting because aliasing is more with many LR observations. Registration errors will increase as the resolution of the observations goes down. These errors create annoying visual quality. In traditional SR reconstruction, image registration is a separate pre-step to HR image estimation. So quality of reconstructed HR image heavily relies on the image registration accuracy. There are many image registration techniques in the literature (Brown 1992; Zitova and Flusser 2003). However, there is interdependency between LR image registration and the HR image estimation (Robinson and Milanfar 2006). HR image estimation takes the advantage of sub-pixel motion estimation. On the contrary, accurate motion estimation is possible with high-quality HR estimation. This cues that both LR registration and HR image reconstruction can be addressed together in the ambit of SR reconstruction problem. But the limitation of many of these approaches demand computational efficiency.
- **Computational feasibility:** SR reconstruction is difficult due to the intensive computation with a large number of unknowns. This requires expensive matrix manipulations. The efficiency makes the algorithms deem fit to any real time applications. Even in the control applications, demand for real time is intuitive with the users tuning some parameters. The scheme in Hardie (2007) claims that the algorithm can be applied in real time with global translation model. However, non-translation models require intensive computation. This invites massive parallel computing. Other variants require precise and massively computational image registration. Implication of parallel computing, e.g., GPU, and hardware implementations on SR techniques is an interesting phenomena to study further.
- **Sensitivity to outliers:** SR techniques must be robust so that they take care of

sensitivity to the outliers. This sensitivity is unwarranted in many applications (Pham et al. 2006). There are few works found in this direction (Protter and Elad 2009). Existing algorithms assume some toy data to show their improvements.

- **Performance yardsticks:** The performance analysis of SR reconstruction is done with simple Mean Squared Error (MSE). But MSE is not sufficient for the good analysis (Yang et al. 2008). It is not necessarily the case that estimation with good MSE always yield a better visual quality. There are distinct proposals in the literature. They suggest the future to progress in this direction. The performance even warrants benchmark and realistic datasets for analysis.

## 2.5 HETEROGENEOUS COMPUTING (HC)

The involvement of both Central Processing Unit (CPU) and the General Purpose Graphics Processing Unit (GPGPU or in short GPU) in a computing is called heterogeneous computing. This can be also be considered as collaborative technique. The term accelerator also refers to GPU with regard to its purpose in computation. In this collaboration, work need to be divided for CPU and GPU. The CPU also gains performance with this work distribution.

Table 2.4: Future trends in multi-core and many-core systems (Mittal and Vetter 2015).

Hardware Attributes	Multi-core system	Many-core system
1. Number of transistors	Crosses 10 Billion	8 Billion
2. Number of cores	Crosses 60	3072
3. Size of LLC	96 MB	2048 KB
4. 3D Integrated Circuit	Exists	Exists
5. Inter-connectivity bandwidth	5 to 12 times bandwidth compared to PCIe-Gen3 by NVLink.	

There are two types of architectures supporting heterogeneous computing. They are fused architecture where CPU-GPU are integrated in the same chip (Yang et al. 2012) and discrete architecture in which CPU and GPU are separate chips connected by Peripheral Component Interconnect Express (PCIe). In the context of HC, CPU is the host and GPU is called the device. There is a rapid evolution that has already taken place in the architectural features of both CPU and GPU. Table 2.4 reflects these trends.



This evolution will continue further in future. This is indeed a motivation to explore possibilities of exploiting performance through this heterogeneous computing.

According to the view expressed by Wang et al. (2018), reformulation of any sequential algorithms in the emerging GPGPU technological advancements has been the recent research trend. This parallel transformation adds equal value to the existing VSS schemes like the completely innovative parallel-VSS schemes.

## **2.6 PROBLEM DESCRIPTION**

Visual Secret Sharing (VSS) is an approach to secure secret digital images in the form of encrypted shares. The VSS schemes are core-intensive and hence demand more execution time making them quite unsuitable for real-time applications. Also, the randomness introduced to ensure robust security of the VSS schemes results in a distorted recovered image disqualifying them for utility in critical healthcare systems. These challenges and the opportunities in the era of VSS schemes attracted many researchers to come out with different techniques. However, to the best of our knowledge, there is no literature in the area of GPGPU-based VSS and GPGPU-based super-resolution. Leveraging GPGPU parallel computation in the VSS schemes achieves high performance to fulfil these two demands. Therefore there is a need to analyze and design novel GPGPU-based VSS schemes.

## **2.7 PROBLEM STATEMENT**

Analysis and Design of GPGPU Visual Secret Sharing security schemes with improved resolution and speed.

## **2.8 OBJECTIVES**

The primary aim of this work is to propose GPGPU-based VSS models. The objectives are as follows:

1. Propose a GPGPU-based Randomized Visual Secret Sharing technique for real-time applications.
2. Propose a GPGPU-based Visual Secret Sharing scheme with Discrete Wavelet

Transformation (DWT)-data embedding and super-resolution.

3. Propose a GPGPU-based Convolution Neural Network (CNN) technique for Visual Secret Sharing.
4. Design and apply GPGPU-based CNN technique for medical images.

### 2.9 SUMMARY

In this chapter, we have surveyed the existing VSS schemes and their features. The VSS field is maturing and contributing increasingly more features. However, the study showed that there is no canonical VSS technique that meets all matured demands. However, the supplementary methods can improve the quality of the reconstructed image at a considerable computational cost. Borrowing super-resolution techniques helps to enhance the contrast of the recovered image. However, the choice or creation of a super-resolution method brings a significant challenge. The recent propensity among the research community towards exploiting the GPGPUs power to speed up the computation through throughput optimization in other domains motivated us to investigate novel GPGPU based VSS, and SR approaches.

This chapter formulates the problem statement and also provides the objectives achieved by this thesis. We discuss our presented GPGPU VSS techniques and their performance evaluations against the benchmark VSS schemes in the following chapters.

## CHAPTER 3

### GPGPU-BASED RANDOMIZED VSS

Visual Secret Sharing (VSS) is a type of image cryptography to secure secret visual information using encrypted shares. The decryption in the VSS is done by the Human Visual System (HVS). Based on the decryption of the secret image, the VSS is categorized primarily into the following three types namely:- 1) Traditional Visual Secret Sharing (TVSS) schemes, 2) Random Grid Visual Secret Sharing (RGVSS) schemes, and 3) Progressive Visual Secret Sharing (PVSS) schemes.

1) Traditional VSS schemes:- In TVSS, to recover a secret image, minimal  $k$  shares out of  $n$  shares need to be stacked together. The term Visual Cryptography (VC) was founded by Naor and Shamir (1994). Basis matrices are used to veil the original pixels of the secret images. Generating uniform shares for each participant and the pixel expansion are the drawbacks of this model (Hou and Quan 2011). Ateniese et al. (1996) invented a general VSS known as General Access Structure (GAS). In this paper, researchers divided users into two sets as qualified and forbidden. According to the proposed technique, the eligible users are only allowed to recover the secret image, whereas prohibited users cannot reveal any part of the secrecy. Ateniese et al. (2001) extended VSS work, by proposing a scheme to generate meaningful shares, known as the Extended Visual Cryptography Scheme (EVCS). These shares are distinct from the noise-like shares. They contain the cover image other than the secrecy imposed on the generated shares.

A deterministic extended VC scheme for binary images is proposed in Kanakkath

et al. (2019). This model focuses on reducing storage space requirements by minimizing average pixel multiplicity. There are many VSS methods applicable to both grayscale and colour images (Bassirian et al. 2019; Blundo et al. 2000; Chen and Tsao 2011a; MacPherson 2002). A quantum Secret Sharing (QSS) for GASs is constructed with only 7-qubits to improve efficiency by reducing computation. As the VC application increases, there is a need to focus on efficiency.

2) Random-Grid Visual Secret Sharing (RGVSS) schemes:- Much earlier seminal work on encryption of images proposed in Kafri and Keren (1987) has evolved over the last decade. The basic encryption algorithms introduced in Kafri and Keren (1987) used two random-grids. In this approach, an encoded random-grid is obtained using a master random-grid containing a 50% probability of bits and an input image. The secret image gets revealed when these two random-grids overlaid. Otherwise, the human eye cannot identify dissimilarity between the master and the encoded random grids. The work proposed in Shyu (2009) extended the technique in Kafri and Keren (1987) to encrypt the colour images. The essential features of VSS schemes include non-pixel expansion, no dependency on the codebook, applicability to the colour images, and the threshold generality. All these essential features characterize the VSS scheme proposed in Yan et al. (2018). In addition to these, this technique enhanced the recreated image quality.

On the one hand, the purpose of the random-grid crypto-system is to minimize computation. This purpose has an underlying assumption that its application is in the computation-restricted environment (Yan et al. 2020a). The mobile and computation-less authentication applications were considered the two general instances of the computation restricted environment. However, this intuition turns out to be less significant because the environment is advanced now. The mobile devices are no more computation-restricted as they are empowered with the heterogeneous computing capability. To take advantage of this capability, it is necessary to look at VC from the efficiency angle. The random-grid model is seeing progress in providing security to multiple images with lesser shares, with the aim of reducing the communication cost (Huang and Juan 2020).

3) Block-based Progressive Visual Secret Sharing (BPVSS) Schemes:- A BPVSS scheme was proposed by Hou et al. (2013a). This method works like a jig-saw puzzle. It works even on colour images. However, it suffers from problems like the recovery of the multitone secret image as the monotone image. Also, the BPVSS recreates the image with the utmost 50% contrast. To overcome the issues of BPVSS, a novel Randomized Visual Secret Sharing (RVSS) scheme is invented by Mhala et al. (2017). This technique combines BPVSS for generating secret shares and reversible DCT data embedding technique. But, the input size in RVSS is not only dependent on the input image size, but also, on the number of users. Moreover, the RVSS is computationally intensive not only while encrypting but also during decryption. These features of the RVSS require greater efficiency for its usefulness in real-time applications. However, the privileged aspect of the RVSS scheme is that the phases in it are intuitively amenable to the data parallelism at different scales. These varying scales of data parallelism at different stages let to use the parallel computing platform to improve the efficiency of the RVSS. Improving the efficiency of the RVSS by using an advanced data-parallel approach is the novelty in the proposed GRVSS. In Mhala and Pais (2019a), the researchers augmented RVSS with a super-resolution model to enhance the revealed image details. The RVSS is extended to 3-dimensional hyperspectral image sharing in Srujana et al. (2020). In such applications, efficiency becomes paramount.

In this chapter, we present two GPGPU-based VSS techniques for real-time applications. Section 3.1 explains the random grid-based GPGPU VSS technique, and section 3.2 discusses the GPGPU-based randomized progressive VSS scheme for grayscale and colour images. Section 3.3 summarizes this chapter.

### **3.1 RANDOM GRID-BASED VSS FOR GRAYSCALE AND COLOUR IMAGES**

In the days when the cryptography started, there was no more attention to the future utility. There was a belief that more security should rely on more computation. Now the information is not just letters. Picture, sound, and video are all accepted as information. The availability of such information elsewhere should be made safely available in real-time. The image is precious medium-size information. Conventional cryptography of

extensive computation may not be more effective in achieving efficiency. Therefore image cryptography started with an objective of little processing during encryption for medium-sized information.

Image security models have new characteristics and capabilities (Kabirirad and Es-lami 2019). The Visual Cryptography that Naor and Shamir (1994) engendered is continuously appealing to researchers. This field is now with a large corpus of literature in the research (D'Arco and De Prisco 2016). The variants of this scheme cost additional storage overhead for the basis matrices and the larger shares. The dealers also transmit shadows. Hence they consume additional traffic bandwidth. Moreover, this scheme requires cumbersome effort in designing appropriate basis matrices (Chen and Tsao 2011a). Besides, the image encryption proposed in Kafri and Keren (1987) has attracted many of the researchers in the last two decades. The inventions based on this method are generally called Random Grid(RG)-based techniques. The investigators came to the view that the RG models remedy the drawbacks of Naor and Shamir (1994). The first refinement in Shyu (2009) is an extension of the technique in Kafri and Keren (1987) to the colour images.

But innovations in image encryption, while effective, do not place much emphasis on efficiency even with multi-core Central Processing Units (CPUs). These CPU-based models do not fit for real-time applications. The sequential model even leads to resource under-utilization. The computing technology has evolved from multi-core to many-core processing units. These many-core processing units are throughput optimized devices in contrast with the latency optimized multi-core processing units. Although the processing elements in a many-core device are simple, their massive quantity is conducive to the data-parallel tasks. Besides, each such processing unit has the multi-task capability. GPGPUs are the data-parallel many-core systems optimized for throughput.

This section presents a reformulated Random Grid (RG) image encryption proposed in Shyu (2009) to leverage the computing power of GPGPU. Such reformulations are novel approaches in research (Wang et al. 2018). Security brings value to the information—the faster the safety, the higher the value of it.

### 3.1.1 Random Grid-Based VSS for Grayscale and Colour Images

This section explains the primary random grid for the grayscale and colour images. Three basic grayscale random grid models were presented in Kafri and Keren (1987) which generate an encrypted or encoded random grid  $E_2$  pixels using a master random grid  $E_1$  based on the pixel values in the binary converted plain-image  $I$ . Equations 3.1, 3.2, and 3.3, represent these three models. The  $[p,q]$  in each equation indicates the pixel position. The  $E_1$  is preset to have a 50% probability of zeros and ones.  $E_1$  and  $E_2$  when overlaid on each other, reveals the plain-image. Otherwise, one cannot discern any secrecy from grids.

$$E_2[p, q] = \begin{cases} E_1[p, q] & I[p, q] = 0 \\ \overline{E_1[p, q]} & otherwise \end{cases} \quad (3.1)$$

$$E_2[p, q] = \begin{cases} E_1[p, q] & I[p, q] = 0 \\ random(0 \text{ or } 1) & otherwise \end{cases} \quad (3.2)$$

$$E_2[p, q] = \begin{cases} random(0 \text{ or } 1) & I[p, q] = 0 \\ E_1[p, q] & otherwise \end{cases} \quad (3.3)$$

Shyu (2009) extended the RG model proposed in Kafri and Keren (1987) for colour images. There are three primary independent colours in the subtractive colour model. They are Cyan (C), Magenta (M) and Yellow (Y). Other colours can be considered the linear combination of these basic colours. Alternatively, a pixel in the given image decomposed to its equivalent three monochromatic coloured-grey levels. These three coloured-grey level images converted to coloured halftone images. Halftone equivalent of the grayscale saves memory with quality (Lou et al. 2011). Three sub-master random grids generated. Three sub-encoded random grids are generated using a technique specified in Kafri and Keren (1987). Then the corresponding colour components of sub-random grids are mixed to obtain the two coloured master and encoded random grids. When these two random grids are superimposed, the colour secret image gets revealed.

#### 3.1.2 Parallelization through GPGPU

The CUDA programming model (Zeller 2011) for GPGPUs has the following hierarchical structure of threads: multiple GPU threads are grouped to form a thread block and multiple thread blocks comprise a grid. A thread block is allocated to one Streaming Multiprocessor (SM). The threads in a thread block share the resources (e.g., shared memory) of the SM. Each thread block in a grid is allocated to each SM in a round-robin fashion, and the number of thread blocks (grid dimension) and number of threads in a block (block dimension) are configured at each GPU kernel launch. The basic parallelization strategy is to assign each independently computable output element to a thread.

Threads can be configured in one-dimensional, two-dimensional or three-dimensional blocks and grids. Few values each thread knows include block identifier (ID), thread identifier, and the block size. CUDA Built-in variables `blockBlockId`, `BlockDim` and `threadId` represent these values. The `blockIdx.x` is a built-in variable that returns the block ID in the x-axis. The `threadIdx.x` is a built-in variable that return the thread ID in the x-axis in a particular block. The `blockDim.x` is a built-in variable that return the block dimension in the x-axis (i.e., the number of threads in a block in the x-axis). We use these three CUDA primitives in all parallel Algorithms. Each thread computes a global identifier using these three primitives to access GPGPU global memory.

#### 3.1.3 Random Grid-based model on a Many-core System

Figure 3.1 shows the block diagram of the presented many-core random-grid model. The input to the model is a colour or a grayscale image. The halftoning block transforms it into the corresponding halftone image. The encryption block generated encoded-grid using a master-grid based on the pixel values of the halftone image. As the master-grid is preset to contain 50% randomness of 0's and 1's, the produced encoded-grid using it also introduces randomness. This randomness increases the security of the system. Decryption requires the stacking of the master and the encoded grids to reveal the secret image. The halftoning and the encryption blocks utilize a many-core system to exploit the data-parallel tasks resulting in an improved speed. The generated encoded-grid



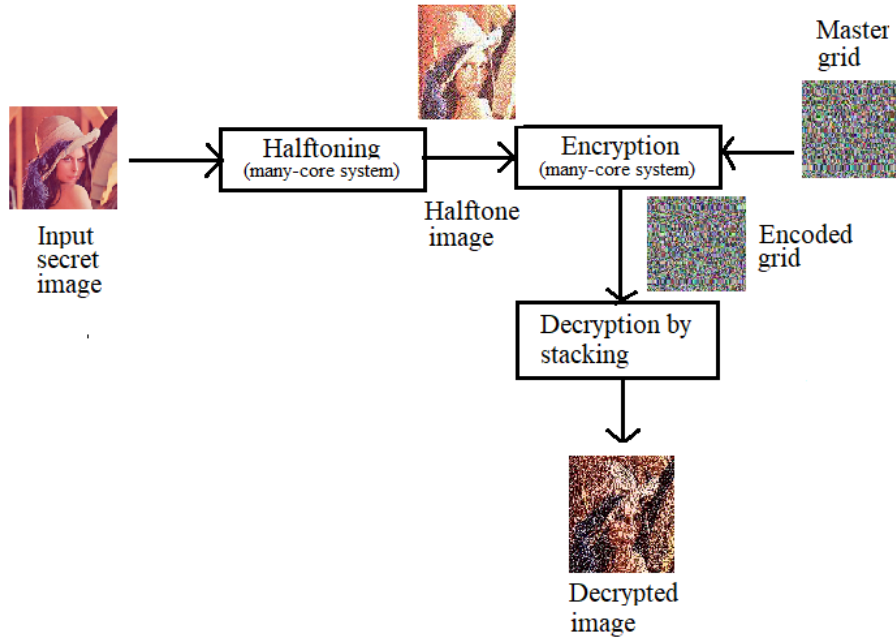


Figure 3.1: Block diagram of the presented system.

resulting from encryption is a share and hence share generating process also refers to the encryption.

Algorithm 3.1 on a host launches Algorithm 3.2 and Algorithm 3.3 to perform halftoning and share generation on a many-core system. A host module is for the multi-core system to execute. This module acts as a staging area where the necessary allocations, data transfers between multi-core and many-core memories, thread creations and kernel invocation are done. The CUDA API is used in the host module to perform such tasks. The function *cudaMemcpyToSymbol()* transfers the data from the CPU host memory to the CUDA constant memory in GPGPU. When *cudaMalloc()* gets executed, memory is allocated in GPGPU global memory. *cudaMemcpy()* is used to transfer data between CPU and GPGPU memories. Finally, *cudaFree()* releases the allocated memory in GPGPU. A device module in C language intended to execute on a many-core system. It is also called a kernel. It also uses CUDA primitives to obtain global unique thread indices. The global device memory is accessed using these indices. The number of pixels to be processed is high. So, the kernel is launched with multiple blocks and threads.

Algorithm 3.1 is the main host module. It re-launches the same kernels shown by

---

**Algorithm 3.1:** Allocate memory, perform data transfer, create threads, and launch kernel to generate a halftone image (**Host module**)

---

**Input:** The secret grayscale image  $I1$  and colour image  $I2$  of size  $(h \times w)$ .

**Output:** Reconstructed grayscale image  $R1$  and colour image  $R2$  of size  $(h \times w)$

- 1 Declare device constant memory using `__constant__float r[4]`.
  - 2 Copy 7/16, 5/16, 3/16, 1/16 to the constant memory `r[4]` using `cudaMemcpyToSymbol()`.
  - 3 Use `cudaMalloc()` to allocate device memory for input images  $I1$  and  $I2$ , halftone images  $HG, HC, HM, HY$ , and the share images  $SG1, SG2, SC1, SC2, SM1, SM2, SY1, SY2$ .
  - 4 Initialize  $SG1, SC1, SM1$ , and  $SY1$  with 50% probability of bits.
  - 5 Set `timer = 0` and Use `cudaMemcpy()` to transfer  $I1$  to device memory  $d_I1$ .
  - 6 Launch `Halftone_kernel(d_I1, HG, width, height)` to generate halftone image with the number of threads equal to the `height`.
  - 7 Use `cudaMemcpy()` to transfer  $HG$  to CPU memory.
  - 8 Store `timer` value and Set `timer = 0`.
  - 9 Launch `Shares_kernel(HG, SG1, SG2, size)` with the number of threads equal to the `size`.
  - 10 Use `cudaMemcpy()` to transfer  $SG1$  and  $SG2$  to CPU memory.
  - 11 Store `timer` value.
  - 12 Stack  $SG1, SG2$  to output reconstructed grayscale image  $R1$ .
  - 13 Set `timer = 0` and Decompose  $I2$  into  $I2c, I2m$ , and  $I2y$ .
  - 14 Store `timer` value and Set `timer = 0`.
  - 15 Use `cudaMemcpy()` to transfer  $I2c, I2m$ , and  $I2y$  to device memory  $d_I2c, d_I2m$ , and  $d_I2y$  respectively.
  - 16 Launch `Halftone_kernel(d_I2c, HC, width, height)`, `Halftone_kernel(d_I2m, HM, width, height)` and `Halftone_kernel(d_I2y, HY, width, height)` to generate halftone image with the number of threads equal to the `height`.
  - 17 Use `cudaMemcpy()` to transfer  $HC, HM$ , and  $HY$  to CPU memory.
  - 18 Store `timer` value and Set `timer = 0`.
  - 19 Launch `Shares_kernel(HC, SC1, SC2, size)`, `Shares_kernel(HM, SM1, SM2, size)` and `Shares_kernel(HY, SY1, SY2, size)` with the number of threads equal to the `size`.
  - 20 Use `cudaMemcpy()` to transfer  $SC1, SC2, SM1, SM2, SY1$ , and  $SY2$  to CPU memory and Store `timer` value.
  - 21 Stack  $SC1, SC2, SM1, SM2, SY1, SY2$  to output reconstructed image  $R2$
  - 22 Use `cudaFree()` function to free all allocated device memory.
  - 23 End
-

---

**Algorithm 3.2:** *Halftone\_kernel*( $I, H, width, height$ ): Generate halftone image in device memory (**Device module**)

---

**Input:** The secret grayscale image  $I$ , Device allocated halftone image  $H$ ,  $height, width$ .

**Output:** Computed halftone image  $H$ .

```

// r[4]- Error diffusion constants already in GPU
// constant memory for aggressive broadcasting of
// read-only data
// assume err[height×width] = 0.
1 tid = blockIdx.x × blockDim.x + threadIdx.x
2 if (t < height) then
3   For j ∈ [0, ..., width - 1] do:
4     if (I[tid × width + j] + err[tid × width + j] < 128)
5       H[tid × width + j] = 0
6     else
7       H[tid × width + j] = 255
8     diff = I[tid × width + j] + err[tid × width + j] - H[id × width + j]
9     if (j + 1 < width)
10      err[tid × w + j + 1] = err[tid × w + j + 1] + diff × r[0]
11     if (tid < height - 1)
12      err[tid × width + j + 1] = err[tid × width + j + 1] + diff × r[1]
13     if (tid < height - 1 and j - 1 ≥ 0)
14      err[tid × width + j] = err[tid × width + j] + diff × r[2]
15     if (tid < height - 1 and j + 1 < width)
16      err[tid × width + j + 2] = err[tid × width + j + 2] + diff × r[3]
17 end
18 Return H

```

---



---

**Algorithm 3.3:** *Shares\_kernel*( $H, G1, G2, size$ ): Generating shares for the halftone colour images(**Device module**)

---

**Input:** Halftone images  $H$ , Master-grid  $G1$ , Allocated encoded grid  $G2$ , and  $size$ .

**Output:** Encoded grid  $G2$ .

```

1 tid = blockIdx.x × blockDim.x + threadIdx.x
2 while (tid < size)
3   if (H[tid] = 0)
4     G2[tid] = G1[tid]
5   else
6     G2[tid] =  $\overline{G1[tid]}$ 
7   tid = tid + blockDim.x × gridDim.x
8 Return G2 device global memory.

```

---

Algorithm 3.2 and 3.3 for the grayscale and the colour images. These kernel invocations are asynchronous and independent. CUDA constant memory initialized in step 2 of Algorithm 3.1 with the constants that are re-used by all threads. This memory facilitates efficient accessing by appropriate caching and broadcasting. The grayscale image gets processed from steps 3 to 12. Step 6 launches the kernel for generating the halftone image. It creates the number of threads equal to the height of the image. The rationale is to process each row of the image with a thread. Step 9 launches share generating kernel with the number of threads equal to the size of the image. The timer records execution times in these two stages separately. The colour image is handled from step 13 to step 21. The colour image is decomposed into C, M, Y components in step 13. Each component is converted to halftone image by reusing the same kernel in step 16. Step 19 re-invokes the kernel used for generating shares. The allocated device memory is released in step 22.

Algorithm 3.2 generates a halftone image. The loop in step 2, provided for a thread to iterate through each row of the image. It uses an error diffusion technique to generate halftone images. The process of generating shares given in Algorithm 3.3. Each thread processes a pixel by device memory read and writes. Step 1 in Algorithm 3.2 and Algorithm 3.3 generates global thread indices. In Algorithms 3.2 and 3.3, the statement  $tid = blockIdx.x \times blockDim.x + threadIdx.x$  maps successive memory locations of GPGPU global memory to the successive threads. *BlockId*, *BlockDim* and *threadId* are the CUDA built-in variables. For example, *blockIdx.x* is a built-in variable that returns the block ID in the x-axis. *threadIdx.x* is a built-in variable that return the thread ID in the x-axis in a particular block. *blockDim.x* is a built-in variable that return the "block dimension" in the x-axis (i.e., the number of threads in a block in the x-axis). We explained parallelization through GPGPU in section 3.1.2.

#### 3.1.4 Experimental Results

##### Performance of Sequential Random Grid-Based VSS

The performance aspects of sequential (2, 2) random grid-based VC is studied using a system with dual-core Intel Core i3-2370M Processor and 4GB RAM. The model is tested with the grayscales and the colour images of different sizes.

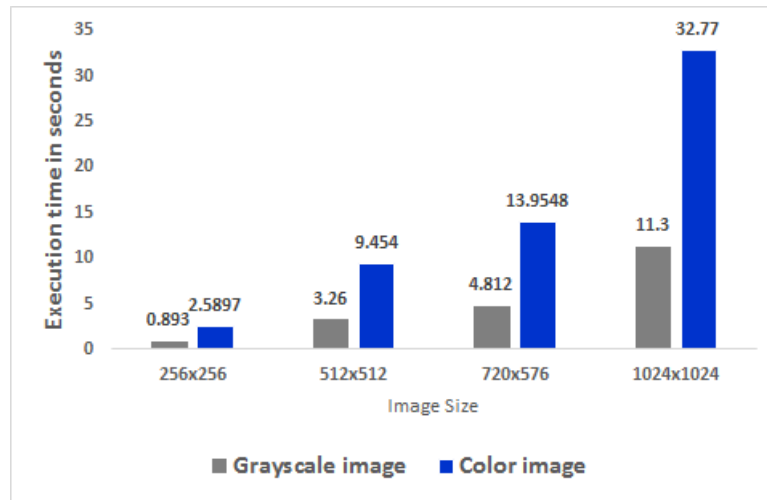


Figure 3.2: Execution times for generating halftone images.

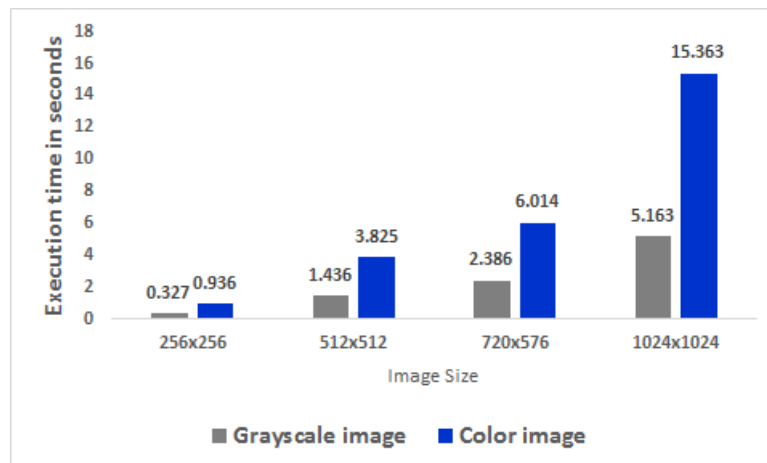


Figure 3.3: Execution times for generating shares.

Figure 3.2 depicts the execution times for generating the halftone images for grayscale and colour images. As evident from Figure 3.2, the execution time in producing the halftone images for the corresponding colour images increases drastically with the image size. The processing time to generate shares for grayscale and colour images is shown in Figure 3.3. Again, the time for creating shares for colour images reaches prohibitive with the increase in image size. In summary, the total execution time for the VC of colour images is computationally expensive as evident from Figure 3.4.

The total time includes halftone and shares generating time typical to both grayscale and colour images. These two components consume more execution time. For the grayscale images, these two components take 70% and 30% of the overall execution

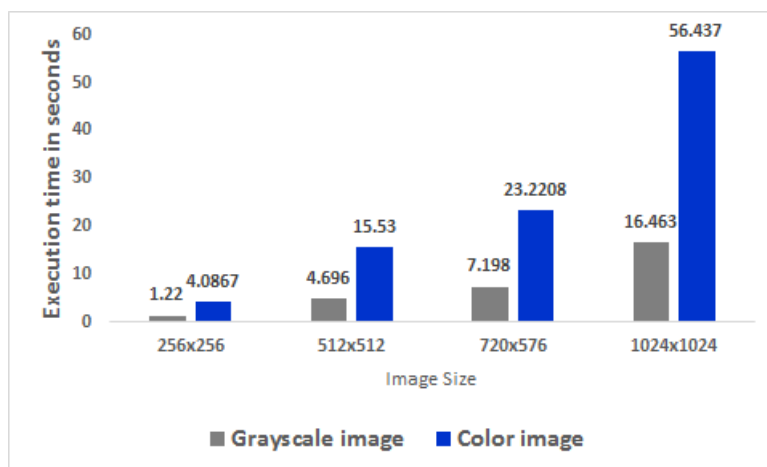


Figure 3.4: Total execution times.

Table 3.1: PARAM Shavak supercomputer environment.

Attributes	Multi-core	Many-core
	Intel(R) Xeon(R)- E5-2670	Tesla K40c
Processing elements	2 CPUs each having 12 cores	2880 cores
Memory	8 TB	12 GB
Cache	30720 KB	L1-64 KB, L2-1.5 MB
Clock Speed	2.30 GHz	745 MHz

time. In colour images, they are 61% and 25%, respectively. These percentages remain constant independent of image sizes. The time to decompose the colour image into its CMY components is an additional time component only for the colour images. Moreover, this time component occupies only 14% of the total execution time, irrespective of the image size. We optimize the process of creating halftone and share images using GPGPU.

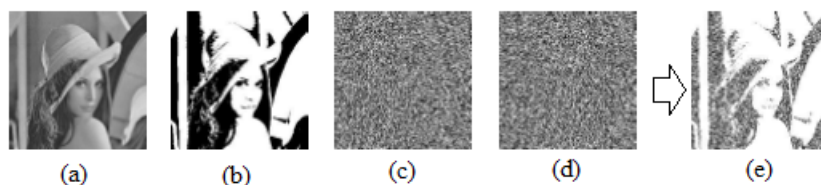


Figure 3.5: (a) Sample grayscale test image Lenna ( $256 \times 256$ ) (b) Halftone image ( $256 \times 256$ ) (c) Master share ( $256 \times 256$ ) (d) Encoded share ( $256 \times 256$ ) (e) Reconstructed image ( $256 \times 256$ ).



Figure 3.6: (a) Sample colour test image Lenna ( $256 \times 256$ ) (b) C component ( $256 \times 256$ ) (c) M component ( $256 \times 256$ ) (d) Y component ( $256 \times 256$ ) (e) Halftone of C component (f) Halftone of M component (g) Halftone of Y component.

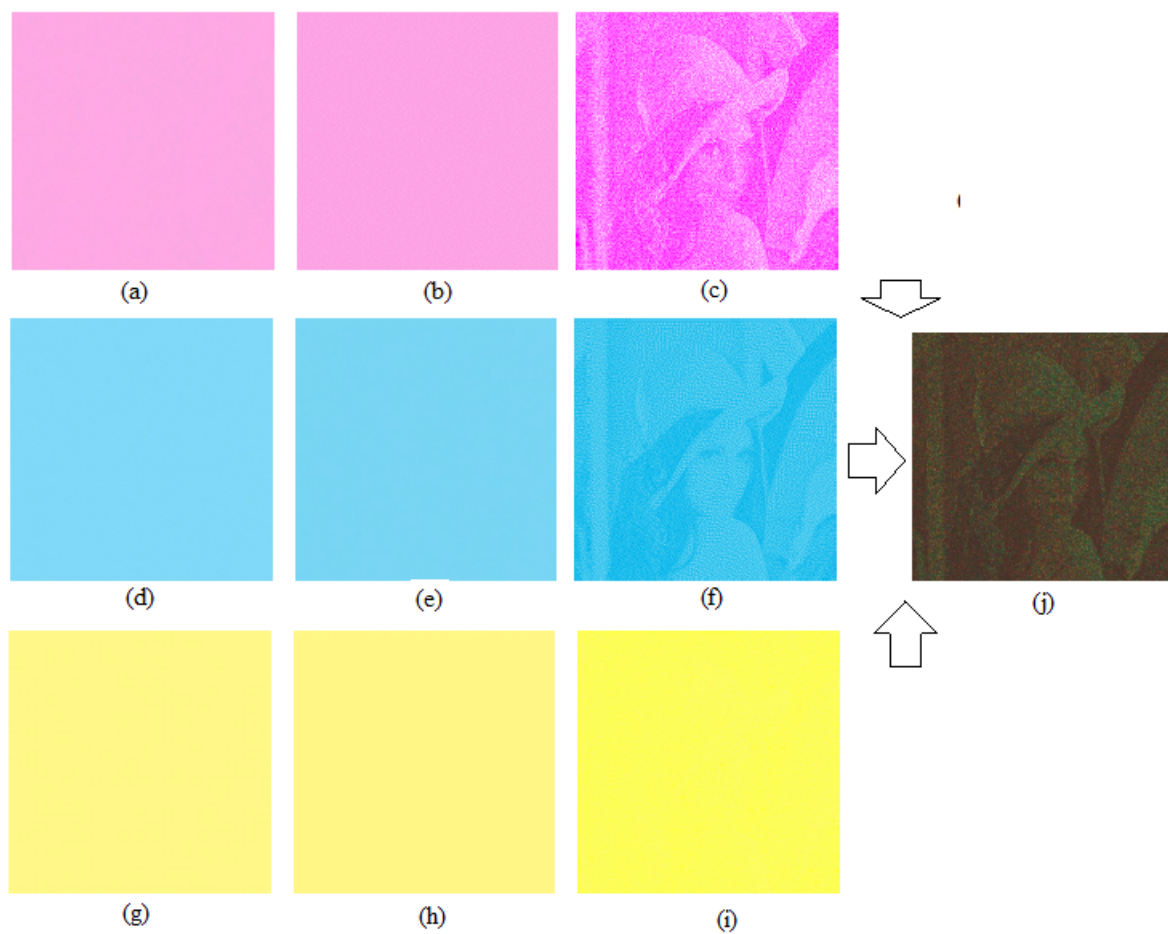


Figure 3.7: (a) Master grid for C halftone (b) Encoded grid for C halftone (c) Stacking of C grids (d) Master grid for M halftone (e) Encoded grid for M halftone (f) Stacking of M grids (g) Master grid for Y halftone (h) Encoded grid for Y halftone (i) Stacking of Y grids (j) Recovered image.

Note: Size of all images is  $(256 \times 256)$ .



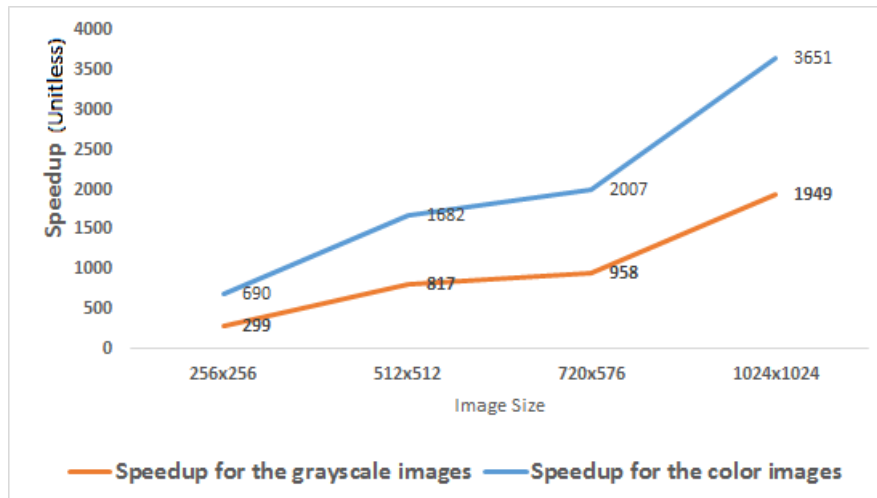


Figure 3.8: Speedup in generating halftone images.

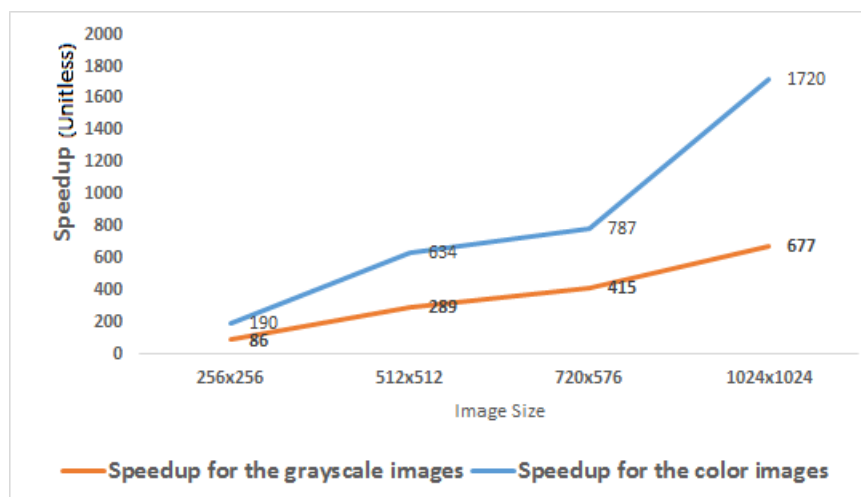


Figure 3.9: Speedup in generating share images.

### Performance of GPGPU Random Grid-Based VSS

The presented scheme is implemented in CUDA with OpenCV, executed on a multi-core PARAM Shavak supercomputer with the Intel(R) Xeon(R) -E5-2670 CPU. This supercomputer contains a many-core Nvidia Tesla K40c GPGPU. Table 3.1 shows the hardware configuration. Many grayscale and colour images with various resolutions taken as test samples. Figure 3.5 (a) shows an input grayscale image. Figure 3.5 (b) shows the corresponding halftone image. Figure 3.5 (c) and Figure 3.5 (d) are the random master grid and the encoded random grid, respectively. They do not reveal the secrecy. Upon stacking these two grids, the resultant secret image is in Figure 3.5

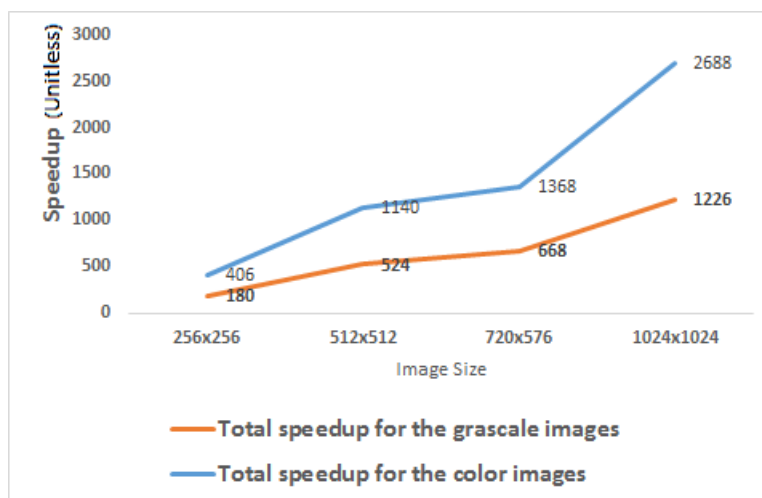


Figure 3.10: Total speedup in generating halftone and share images.

(e). Similarly, Figure 3.6 (a) shows a secret colour image. The derived C, M, and Y decomposed images displayed in Figure 3.6 (b), Figure 3.6 (c), and Figure 3.6 (d), respectively. Figure 3.6 (e), Figure 3.6 (f), and Figure 3.6 (g) are their corresponding halftone images. Figure 3.7 (a), Figure 3.7 (b), Figure 3.7 (d), Figure 3.7 (e), and Figure 3.7 (g), Figure 3.7 (h) respectively, show the generated two shares corresponding to each halftone image. Figure 3.7 (c), Figure 3.7 (f), and Figure 3.7(i) are the images upon stacking their respective shares. The overlaying of all shares reveal the secret image shown in Figure 3.7 (j).

Figure 3.8 depicts the increase in these speedups as the image size grows. Figure 3.9 shows the speedup in generating shares for the grayscale and colour images. The total speedup considering both the execution times for generating halftone and colour images in sequential and the presented techniques is displayed in Figure 3.10. The execution time for generating the halftone images is considerably greater than that of generating the shares in a multi-core random grid method. The number of memory accesses and computations is higher in obtaining the halftone images than the shares. Accordingly, the speedup achieved is also more in generating the halftone images than in producing the shares. The experimental results reveal that the speedups achieved increase considerably as the image size grows. The performance of the presented many-core random grid method is  $406\times$  to  $2688\times$  better than the random grid in a multi-core system.

The GPGPU-based random grid VSS scheme presented in this section takes advantage of inherent data-parallel tasks of the original random grid VSS scheme. In the next section, we explain another randomized VSS scheme which is characterized by a mix of task and data-parallelism. Then we present a GPGPU-based randomized VSS scheme and analyze the performance implications.

### 3.2 GPGPU-BASED RANDOMIZED VISUAL SECRET SHARING (GRVSS)

Visual Secret Sharing (VSS) is a type of image cryptography to secure secret visual information using encrypted shares. The decryption in the VSS is done by the Human Visual System (HVS). There are many visual cryptography techniques (Blundo et al. (2000), MacPherson (2002), (Zhou et al. 2006), Hou (2003)) to securely transmit visual media. The VSS suffers from common problems such as:-

- The majority of the techniques recover secret images with poor contrast. The maximum contrast achievable by VSS techniques is 50% (Mhala et al. 2017).
- The existing VSS schemes mainly convert grayscale and colour image into monotone images using halftone techniques. The recovered image in VSS is always a monotone image.

Recently, the Block-based Progressive Visual Secret Sharing (BPVSS) scheme is presented by Hou *et al.*(Hou et al. 2013a). The contrast achieved in this method is 50%. Also, the BPVSS recovers the secret image in the form of a halftone image. Randomized Visual Secret Scheme (RVSS) (Mhala et al. 2017) overcomes the problem of the BPVSS scheme. RVSS recovers the secret image using a data embedding technique with a contrast of 70-80% for noise-like shares and 70-90% for meaningful shares. The RVSS is a combination of multiple tasks and is expensive due to many high computation tasks. The overall complexity of RVSS is equal to the product of the number of participants and the size of the image. When the number of participants or the size of the image increases the RVSS becomes computationally expensive for real-time applications. Few such applications include biometric identification security, bar code protection, online evaluation, and e-voting, anti-forgery, and commodity tracing (Yan

et al. 2020a).

A recent study in the field of parallel computing has shown that it is desirable to process large size image using Nvidia Graphics Processing Unit (GPU) to achieve faster computation (Nvi 2020). The use of Nvidia GPU and Compute Unified Device Architecture (CUDA) improves processing throughput. Recently, the researchers have emphasized the need for efficiency in the VSS field (Kabirirad and Eslami 2019; Yan and Lu 2019). This section explains the presented efficient GPGPU-based Randomized Visual Secret Sharing (GRVSS) scheme to make the RVSS suitable for real-time applications.

#### 3.2.1 RVSS Technique

For convenience, the existing RVSS method is reformulated into the forward phase and the inverse phase. The forward phase consists of block-wise share generation followed by, data hiding into the shares. The inverse phase recreates the original image. In the inverse process, embedded data is extracted, coefficients of all shares are restored, the secret image is unveiled, the image is added with the data, and post-processing is done to remove the noise. These phases are explained below.

#### Forward Phase in RVSS

This step uses BPVSS scheme presented in Hou et al. (2001). The number of shares equal to the number of stakeholders is created. Let  $n$  be the number of stakeholders and  $I$  be the image having size  $W \times H$ . The BPVSS converts the original multitone image  $I$  into a halftone image. The half-tone version of the secret image  $I_{halftone}$  is separated into  $n$  non-overlapping blocks, each of size  $\frac{W \times H}{n}$ . The size of the encrypted transparencies is equal to the size of the secret image  $I$ . The BPVSS uses basis matrices to create such transparencies. Total  $n + 1$  basis matrices say  $B^0, B^1, \dots, B^n$ , of size  $2 \times n$  are required to generate  $n$  shares.

Table 3.2 shows the generated basis matrices for four shares. In the embedding step, the shares generated using BPVSS are transformed into the frequency domain using the Discrete Cosine Transformation (DCT) by  $8 \times 8$  blocks (using equation 3.4). The idea

Table 3.2: Generated basis matrices for  $n = 4$ .

$B^0 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$	$B^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$
$B^2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}$	$B^3 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$
$B^4 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$	

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Figure 3.11: The standard quantization table (Bovik 2009).

in Gujjunoori and Amberker (2013b) and Gujjunoori and Amberker (2013a) to embed data into videos based on the ceaseless zeros is customized for grayscale and colour images in Mhala et al. (2017). A standard quantization table in Figure 3.11, crafted to make high-frequency DCT coefficients to zero, is used to enable more data embedding in the mid frequencies. The technique in Chang et al. (2007) is used to embed data (using equation 3.6) into a share in the middle frequency. The share-pixel is embedded with ancillary information such that the original pixel value is restored in the inverse phase.

$$F_{u,v} = \frac{C(u) \times C(v)}{4} \sum_{s=0}^7 \sum_{t=0}^7 B_i^i(s, t) \times \bar{f}(s, t, u, v) \quad (3.4)$$

where  $0 \leq u, v \leq 7$  and

$$\bar{f}(s, t, u, v) = \cos \frac{(2s+1)u\pi}{16} \cos \frac{(2t+1)v\pi}{16},$$

$$C(e) = \begin{cases} \frac{1}{\sqrt{2}} & e = 0 \\ 1 & e > 0 \end{cases}.$$

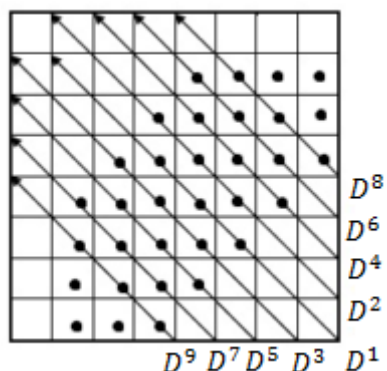


Figure 3.12: The defined sets and locations for embedding the data.

Further processing of each block uses nine sets of  $D_k$  ( $1 \leq k \leq 9$ ) defined per block as shown in the Figure 3.12 and also, Table 3.3, containing the size of the each defined set.

Table 3.3: Selected set size for embedding.

<b>k</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>
<b>K(k)</b>	7	7	7	6	6	5	5	4	4

$$A = Amb(x) = \begin{cases} 0 & \text{if } x = 0 \\ x + 256 & \text{if } x \geq 1 \\ x - 256 & \text{Otherwise,} \end{cases} \quad (3.5)$$

$$E = Embed(A, s) = \begin{cases} s & \text{if } A = 0 \\ A & \text{Otherwise,} \end{cases} \quad (3.6)$$

$$D = Extract(x) = \begin{cases} x & \text{if } 0 \leq x \leq 255 \\ 0 & \text{Otherwise,} \end{cases} \quad (3.7)$$

$$D = Restore(x) = \begin{cases} 0 & \text{if } 0 \leq x \leq 255, \\ x - 256 & \text{if } x \geq 256, \\ x + 256 & \text{Otherwise,} \end{cases} \quad (3.8)$$

The function  $Amb$  in equation 3.5 is used to disambiguate later the coefficient and the

embedded data.

### **Inverse Phase in RVSS**

After embedding data, the shares are transmitted to the participants. These shares are divided into  $(8 \times 8)$  blocks. The inverse process extracts the data from these blocks. The *Extract* (Equation 3.7) is the function to extract the embedded data from the received share. This extract results in two sets of values  $E_{min}$  and  $E_{max}$  from even and odd share blocks respectively. These two sets are used to generate  $E_{final}$ . The function *Restore* restores the original coefficient value as in equation 3.8. Each block is de-quantized by multiplying the standard table shown in Figure 3.11 and then inverse DCT is applied to restore the original shares. Then random values in the range minimum to the maximum are embedded in the middle and its adjacent location, as shown in Figure 3.12. The shares when stacked recreates the original halftone image. However, the original image has values ranging from 0 to 255. So to estimate pixel values from the halftone image, the inverse halftone technique (Xiong et al. 1999) is used.  $E_{final}$  values are added at the respective locations of the halftone image to improve contrast. Then post-processing is done by applying the Wiener filter to alleviate noise effect (Mhala et al. 2017).

### **3.2.2 GRVSS Technique**

This section explains the parallel approaches used for better performance of the RVSS scheme using two subsections, namely 1) Forward Phase in GRVSS and 2) Inverse Phase in GRVSS.

#### **Forward Phase in GRVSS**

The presented technique uses Nvidia GPU and the CUDA parallel computing platform. Generating shares from the original image block by block based on the randomness with the basis matrices is an embarrassingly data-parallel workload. A CUDA program has a host code to run on the CPU and two kernels, one to generate the shares and another kernel to embed data into the shares, to run on the GPU. The parallel CUDA threads organized into a grid of thread blocks. Also the basis matrices  $B_i$  ( $0 \leq i \leq n$ ) and the array  $k$  containing the sizes of sets  $D_k$  ( $1 \leq k \leq 9$ ) are placed in the CUDA

---

**Algorithm 3.4:** Generation of shares using BPVSS. //Host module

---

- Input:** 1. The halftone image  $I_{halftone}$  having size  $(W \times H)$   
 2. Number of participants  $n$   
 3. The  $n + 1$  basis matrices each of size  $2 \times n$   
 4. Array  $k$
- Output:**  $n$  shares each of size  $W \times H$
- 1 Allocate device constant memory space and write basis matrices and the array  $k$  into the constant memory
  - 2 Allocate device global memory for the image and the shares.
  - 3 Copy the image to the allocated device global memory.
  - 4 Compute Grid and Block dimension and size.
  - 5 Invoke *Kernel module1-share generate* with the requisite configuration parameters.
  - 6 Copy the device global memory shares to host memory and then store the shares into images in the host.
  - 7 Compute Grid and Block dimension and size.
  - 8 Invoke *Kernel module2-data embed* with requisite configuration parameters.
  - 9 Free the allocated memory.
  - 10 End
- 

constant memory as they are just read. For a given image, as the number of users or shares increases, the number of blocks increases. But the number of 'read' operations per block for each thread remains the same. However, for each thread, the number of reads from the basis matrices and the writes to the shares increase.

The global input index is a function of the thread block index (blockIdx.x), the total number of threads per block (blockDim.x), and a unique thread index within a block (threadIdx.x). The share generating stage with rich data-parallel workload can be considered a fine-grained parallel than the other modules of GRVSS and hence takes the best advantage of the GPU architecture. The data embedding module uses fixed  $8 \times 8$  blocks, and hence the threads are organized into  $8 \times 8$  blocks. Most of them in each block count the ceaseless zeros and also minimum and maximum of set  $D_k$  within the block. Those threads in the block embed either minimum or maximum value depending on whether they belong to even shares or odd shares. This workload can be considered to be a bit task-level parallelism, however rich with data reads and writes per thread. The parallel algorithm is formulated in Algorithm 3.4 with its kernel modules in Algorithm 3.5 and Algorithm 3.6.



---

**Algorithm 3.5: Kernel module1-share generate**

---

- Input:**
1. The halftone image  $I_{halftone}$  having size  $W \times H$  in device global memory
  2. Size of the image in total number of pixels from host module
  3. Device allocated  $n$  shares in device global memory
  4. The  $n + 1$  basis matrices each of size  $(2 \times n)$  in constant memory
- Output:** Updated  $n$  shares each size  $W \times H$  in device global memory
- 1 Logically treat  $I$  into  $n$ , non-overlapping blocks, each of size  $\frac{(W \times H)}{n}$
  - 2 Make each thread create a unique global index
  - 3 Per-non-overlapping block, do in parallel.
    - 4 a. Generate random values  $r = 0$  or  $1$ ;
    - 5 b. Use each thread to read basis matrix elements from
    - 6 constant memory based on  $r$  and image pixel value and
    - 7 then write to the corresponding pixel of each share
  - 8 Return shares.
- 

---

**Algorithm 3.6: Kernel module2-data embed**

---

- Input:**
1. The halftone image  $I_{halftone}$  having size  $W \times H$  in device global memory
  2. Size of the image in total number of pixels from host module
  3. Device allocated  $n$  shares in device global memory
  4. Array  $k$  in constant memory
- Output:** Embedded  $n$  shares each size  $(W \times H)$  in device global memory
- 1 Logically treat  $I$  and each of the  $n$  shares into  $8 \times 8$  blocks.
  - 2 Make each thread create a unique global index
  - 3 Per-  $8 \times 8$  blocks, do in parallel.
    - 4 a. Find the DCT of the blocks of shares and quantize
    - 5 b. Use corresponding threads to compute ceaseless zero in
    - 6 each set  $D_k$  ( $1 \leq k \leq 9$ ) and fix the positions to embed
    - 7 using array  $k$ .
    - 8 c. Use the corresponding threads to compute minimum
    - 9 and the maximum of each set  $D_k$  ( $1 \leq k \leq 9$ ) in the
    - 10 image blocks.
    - 11 d. For even share blocks, embed minimum values and for
    - 12 odd shares, embed maximum values at their fixed
    - 13 positions, after resolving ambiguous condition using
    - 14  $Amb$  function
  - 15 Return shares.
-

Algorithm 3.4 takes halftone image, basis matrices shown in Table 3.2 and array  $k$  shown in Table 3.3 as input. Basis matrices and array  $k$  are stored in GPGPU constant memory as the threads only read these values. The "configuration parameters" specify the dimension and quantity of threads before we launch kernels given in Algorithm 3.5 and Algorithm 3.6. In these two kernels, threads have their own identity. Threads using this identity or using local variables making use of this identity are arranged to point to their memory locations. Algorithm 3.5 takes halftone image as its input along with image size. The read-only basis matrices are accessible by all threads as they are in constant memory. Then shares are generated in the GPGPU memory that are later transferred to CPU memory in the host module. The host module in Algorithm 3.4 invokes Algorithm 3.6 to embed image information into the shares. Therefore, Algorithm 3.6 uses halftone image, generated shares as input. All threads can read array  $k$  stored in the constant memory. In this way, data embedding is accomplished.

#### **Inverse Phase in GRVSS**

The inverse process uses fixed  $8 \times 8$  blocks and hence the threads are organized into  $8 \times 8$  blocks. Most of the threads of each block involved in the task of counting ceaseless zeros for the nine sets per block. The extracted  $E_{min}$  and  $E_{max}$  per block are used to obtain  $E_{final}$  values. After restoring the original coefficients in the blocks, de-quantizing requires multiplication by a standard table shown in Figure 3.11. This computationally rich multiplication is a fine-grained data-parallel task that takes the best advantage of the GPU architecture. The standard table kept in constant memory is used for the read-only operation. CUDA constant memory adds performance gain for the read-only operations by suitably caching the data, optimizing the read-only operations, and broadcasting the data to many threads at a time.

Then the inverse DCT is applied to reproduce the initial share values. Once the embedded values are extracted the secret image is unveiled by the overlaid shares. The inverse halftoning is then applied to this image. Then the pixel at the desired location is replaced with  $E_{final}$  values to get the final original image. The parallel algorithm is formulated in Algorithm 3.7 along with its kernel module in Algorithm 3.8. Algorithm 3.7

is the CPU host module to do the inverse process to extract the embedded information and to restore the shares. The reconstructed image is the output. Algorithm 3.8 operates on the embedded shares made available by its host Algorithm 3.7. The reference table  $k$  is accessible by all threads. The reconstructed image generated by this module is stored in GPGPU memory. Algorithm 3.7 then copies the reconstructed image to the CPU memory.

---

**Algorithm 3.7:** Extract-restore-reconstruct-post-process. //Host module

---

**Input:** 1.  $n$  embedded shares

2. Array  $k$

**Output:** Reconstructed image  $I$  having size  $W \times H$

- 1 Allocate device constant memory space; write array  $k$  and standard table into the constant memory.
  - 2 Allocate device global memory for the embedded shares and the image output.
  - 3 Copy embedded shares to the allocated device global memory.
  - 4 Compute Grid and Block dimension and size.
  - 5 Invoke *Kernel module3-extract-restore-reconstruct* with the requisite configuration parameters.
  - 6 Copy the device global memory image  $I$  to host memory.
  - 7 Free the allocated memory.
  - 8 Apply Wiener filter post process to  $I$ .
  - 9 End
- 

### 3.2.3 Computational Complexity of the RVSS and GRVSS model

This section compares the time complexity of the the RVSS and the GRVSS model. The RVSS is a combination of many subtasks such as 1) Share generation, 2) Data embedding 3) Extraction of data and 4) Secret image recovery. For an image of size  $W \times H$ , the RVSS algorithm takes  $O(n.W.H)$  time which is of the order of the polynomial, where  $W$  is the width and  $H$  is the height of an image, and  $n$  is the number of shares ( $n \ll (W \text{ or } H)$ ). If  $W = H$ , then each stage time complexity is  $O(n.W^2)$ .

In GRVSS, each stage is parallelized using GPU. For a square image of width  $W$ , a total of  $W^2$  threads launched. So, the complexity of the GRVSS model is  $O(\frac{W^2}{\text{total number of threads}})$  =  $O(1)$ . The  $(8 \times 8)$  combination of threads is used for the computation.

---

**Algorithm 3.8:** *Kernel module3-extract-restore-reconstruct.*

---

**Input:** 1. Device allocated  $n$  embedded shares in device global memory  
2. Array  $k$  and standard table in constant memory  
3. Size of the image in total number of pixels from host module

**Output:** The halftone image  $I$  having size  $W \times H$  in device global memory

- 1 Logically divide  $n$  embedded shares into  $8 \times 8$  blocks.
  - 2 Make each thread create a unique global index
  - 3 Using thread index, for all  $8 \times 8$  blocks in each share, do in parallel.
    - 4 a. Compute ceaseless zero count  $z_k$  ( $1 \leq k \leq 9$ ) for
    - 5 each of the nine sets,  $D_k$  ( $1 \leq k \leq 9$ ) in every block.
    - 6 b. If  $z_k \leq (\frac{(K(k))}{2}) - 1$ , then the middle embedded
    - 7 coefficient is  $x = D_k(k, \frac{(K(k))}{2})$
    - 8 c. Call Extract(x) to extract the data at that location. Let
    - 9 the extracted elements from even shares be  $E_{min}$  and
    - 10 odd shares be  $E_{max}$  from even and odd share blocks
    - 11 respectively. These two sets are used to generate
    - 12  $E_{final}$
    - 13 d. Call Restore(x) to restore the original coefficients in
    - 14 blocks ;
    - 15 e. De-quantize each block by multiplying standard
    - 16 table and then apply inverse DCT to restore the original
    - 17 share blocks
    - 18 f. Embed random values in the range  $E_{min}$  and  $E_{max}$
    - 19 at the respective middle and its adjacent locations
    - 20 g. Stack blocks to reconstruct  $I_{halftone}$  blocks
    - 21 h. Apply inverse halftone technique to the block
    - 22 i. Add  $E_{final}$  values at the respective locations to get
    - 23 image  $I$
  - 24 Return  $I$
-

### 3.2.4 Experimental Results

The experiment is performed on the image dataset available at SIP (2020) and BPV (2020). This dataset contains a total of 44 images, out of which 16 are colour images, and 28 are grayscale and two binary images. Also, the high dimension medical image (breast cancer) dataset used is publicly available at SIP (2020). The medical dataset contains histopathological images for four classes, namely 1) normal, 2) benign, 3) in situ carcinoma and 4) invasive carcinoma. Due to space restriction, only four images are shown.

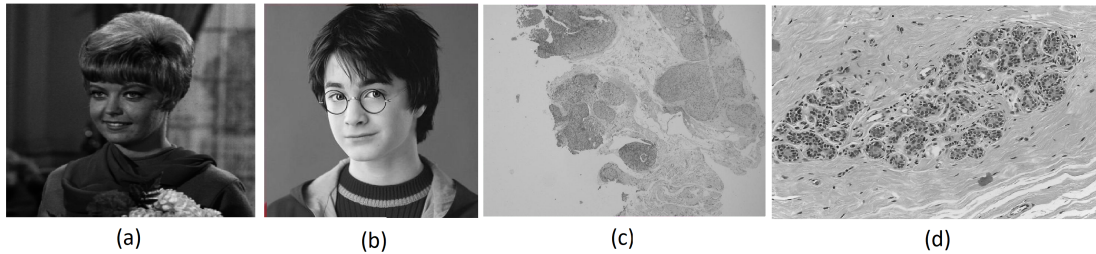


Figure 3.13: The sample test images used for experimentation (Grayscale images) (a) Girl ( $256 \times 256$ ) (b) Harry ( $400 \times 400$ ) (c) Carcinoma in situ ( $760 \times 570$ ) and (d) Invasive carcinoma ( $2048 \times 1536$ ).

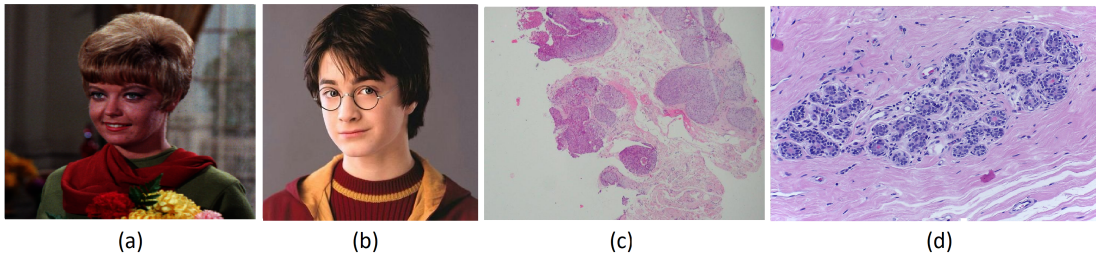


Figure 3.14: The sample test images used for experimentation (Colour images) (a) Girl ( $256 \times 256$ ) (b) Harry ( $400 \times 400$ ) (c) Carcinoma in situ ( $760 \times 570$ ) and (d) Invasive carcinoma ( $2048 \times 1536$ ).

The sample test images used for experimentation are shown in Figure 3.13 and Figure 3.14 respectively. The experiments conducted in the hardware environment specified in Table 3.4. The RVSS implemented using python and executed on HP ProBook 440 G3 with the Processor Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz and PARAM Shavak with Single Nvidia Tesla K40 GPGPU Card. The parallel GRVSS is implemented with PyCUDA and OpenCV to run on the same PARAM Shavak with Single

Table 3.4: Hardware specification.

Systems		Processor	Clock Speed	Cache memory	No. of cores	Memory bandwidth
HP ProBook 440 G3		Intel Core i5 6200U	2.3 GHz	3 MB	2	34.1 GB/s
PARAM Shavak	CPU	Intel(R) Xeon(R)-E5-2670	2.30GHz	30720 KB	12	68 GB/s
PARAM Shavak	GPU	Tesla K40c	745 MHz	64KB(L1), 1.5 MB(L2)	2880	288 GB/s

Table 3.5: Execution time to generate 4 shares.

Images	RVSS		GRVSS
	HP ProBook 440	PARAM Shavak	PARAM Shavak
Grayscale images (time in seconds)			
Fig. 3.13 (a)	0.082828	0.079917	0.002251
Fig. 3.13 (b)	0.202979	0.196942	0.003099
Fig. 3.13 (c)	0.563112	0.457839	0.008473
Fig. 3.13 (d)	3.95069	3.75696	0.042993
Colour images (time in seconds)			
Fig. 3.14 (a)	0.207056	0.215497	0.005043
Fig. 3.14 (b)	0.529693	0.518914	0.009065
Fig. 3.14 (c)	1.726191	1.152122	0.020341
Fig. 3.14 (d)	11.00877	9.352378	0.084462

Nvidia K40 GPGPU Card. PyCUDA allows accessing CUDA parallel computation API from Python.

GRVSS is tested for secret sharing having 4 and 8 shares. The performance of the GRVSS scheme is compared with the RVSS in the generic and supercomputer in terms of execution time for the forward phase and the inverse phase.

1) Forward Phase:- The first step in this phase is to create shares. The execution time for 4 shares is given in Table 3.5. We observe from Table 3.5 that, RVSS takes 0.082828 sec. to generate shares for Figure3.13 (a). Similarly, the execution time for

Table 3.6: Execution time to generate 8 shares.

Images	RVSS		GRVSS
	HP ProBook 440	PARAM Shavak	PARAM Shavak
Grayscale images (time in seconds)			
Fig. 3.13 (a)	0.150003	0.134155	0.002706
Fig. 3.13 (b)	0.367747	0.31162	0.005889
Fig. 3.13 (c)	1.01768	0.79076	0.013928
Fig. 3.13 (d)	7.094723	5.503569	0.051467
Colour images (time in seconds)			
Fig. 3.14 (a)	0.375074	0.387212	0.005864
Fig. 3.14 (b)	1.247678	0.70037	0.015067
Fig. 3.14 (c)	2.880003	2.029443	0.027422
Fig. 3.14 (d)	20.51858	13.88113	0.125745

Table 3.7: Execution time to embed data in 4 shares.

Images	RVSS		GRVSS
	HP ProBook 440	PARAM Shavak	PARAM Shavak
Grayscale images (time in seconds)			
Fig. 3.13 (a)	0.389095	0.372835	0.295054
Fig. 3.13 (b)	0.969431	0.947873	0.667831
Fig. 3.13 (c)	2.526547	2.405026	1.81665
Fig. 3.13 (d)	17.5835	16.89468	12.28516
Colour images (time in seconds)			
Fig. 3.14 (a)	0.938663	1.130506	0.748454
Fig. 3.14 (b)	2.39211	4.63249	1.832753
Fig. 3.14 (c)	7.787275	5.789142	5.171561
Fig. 3.14 (d)	51.48547	41.38304	36.08632

share generation on the PARAM Shavak is 0.079917 sec. for Figure 3.13 (a). The execution time to produce shares using the GRVSS is 0.002251 sec. for Figure 3.13 (a). It is evident from the Table 3.5 that the GRVSS parallel strategy provides the speedup in the range 36.79 to 91.89 for the generation of shares over the RVSS scheme. This steady increase in speedup is achieved due to the parallelization of the RVSS into  $W \times H$  threads, where  $W$  is the width and  $H$  is the height of an image. Similarly, Table 3.6 shows the execution time required for the generation of shares for 8 participants. The

Table 3.8: Execution time to embed data in 8 shares.

Images	RVSS		GRVSS
	HP ProBook 440	PARAM Shavak	PARAM Shavak
Grayscale images (time in seconds)			
Fig. 3.13 (a)	0.60006	0.552055	0.284124
Fig. 3.13 (b)	1.450145	1.431405	0.668762
Fig. 3.13 (c)	3.800501	3.602954	1.757695
Fig. 3.13 (d)	27.25289	26.32582	12.17883
Colour images (time in seconds)			
Fig. 3.14 (a)	1.499876	1.755487	0.750124
Fig. 3.14 (b)	3.62508	3.467873	1.823029
Fig. 3.14 (c)	10.87648	9.360216	5.007975
Fig. 3.14 (d)	97.82241	66.09241	36.04342

Table 3.9: Execution time to restore image data in 4 shares.

Images	RVSS		GRVSS
	HP ProBook 440	PARAM Shavak	PARAM Shavak
Grayscale images (time in seconds)			
Fig. 3.13 (a)	0.287543	0.587135	0.317787
Fig. 3.13 (b)	0.707097	1.138186	0.674423
Fig. 3.13 (c)	1.966796	3.32698	1.83106
Fig. 3.13 (d)	13.70468	21.57173	12.36353
Colour images (time in seconds)			
Fig. 3.14 (a)	0.735198	1.493223	0.774911
Fig. 3.14 (b)	1.853896	3.124777	1.956826
Fig. 3.14 (c)	5.495038	7.935168	5.258796
Fig. 3.14 (d)	36.9388	58.70312	31.56124

generic RVSS takes 0.15003 sec. and PARAM Shavak 0.134155 sec. to generate shares for Figure 3.13 (a) whereas, the GRVSS takes 0.002706 sec. to generate shares for 8 participants. Table 3.5 and Table 3.6 reflects that as the number of shares increases the speedup for the generation of shares also increases.

The next step in the forward phase is to embed data into the shares. This step is computationally expensive in the encryption of shares. Its complexity in RVSS explained in section 4. Execution time for embedding data shown in Table 3.7 and Table 3.8 for 4



Table 3.10: Execution time to restore image data in 8 shares.

Images	RVSS		GRVSS
	HP ProBook 440	PARAM Shavak	PARAM Shavak
Grayscale images (time in seconds)			
Fig. 3.13 (a)	0.499951	0.872016	0.305582
Fig. 3.13 (b)	1.200092	2.606813	0.676089
Fig. 3.13 (c)	3.39689	5.791631	1.818659
Fig. 3.13 (d)	23.2996	42.51863	12.5263
Colour images (time in seconds)			
Fig. 3.14 (a)	1.375381	2.873913	0.773387
Fig. 3.14 (b)	3.00276	5.994911	1.905694
Fig. 3.14 (c)	10.62611	14.60596	5.243884
Fig. 3.14 (d)	70.10889	102.7191	35.91416

and 8 shares respectively. The RVSS in generic computing takes 0.389095 sec. and in PARAM Shavak takes 0.372835 sec. execution time to complete the data embed phase for Figure 3.13 (a) whereas, the GRVSS takes 0.295954 sec. execution time. From Table 3.7, it may be observed that as the size of the image increases, the GRVSS achieves the steady increase in speedup over the RVSS and PARAM Shavak.

2) Inverse Phase:- This phase is a combination of extraction of data, restoration of secret image, and reconstruction of the secret image using extracted data followed by post-processing step. Table 3.9 and Table 3.10 shows the execution time required for the inverse process for 4 shares and 8 shares respectively. From Table 3.9 and Table 3.10, as the number of shares increases the performance of the GRVSS system also improves in terms of speedup. Alternatively, the GRVSS is scalable for large instances.

From the Table 3.11, for Figure 3.13 (a), the forward phase achieves the speedup of 1.587336 and the inverse phase achieves the speedup of 0.904829. The inverse phase achieves less speedup only for a small number of shares whereas, as shown in Table 3.12 for the same image in Figure 3.13 (a) for the generation of 8 shares, the GRVSS scheme achieves the speedup of 1.636062. GRVSS provides better speedups for the generation of more than 4 shares. It is evident from the Table 3.11 and Table 3.12 that

### 3. GPGPU-based Randomized VSS

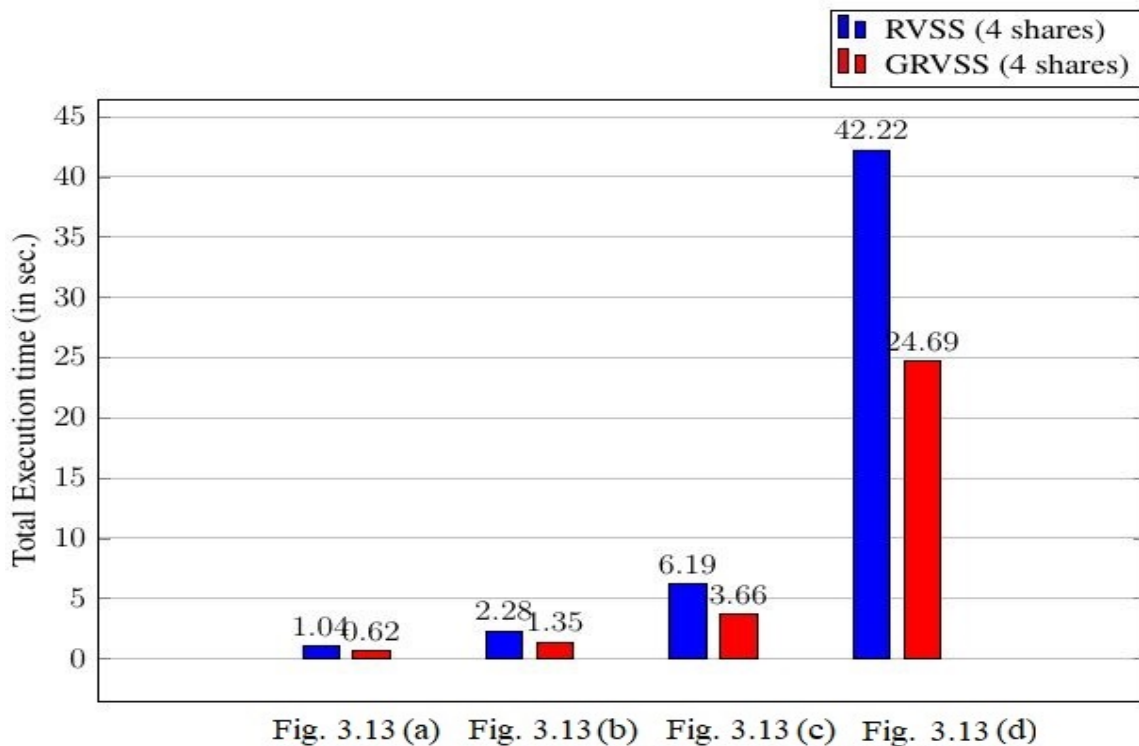


Figure 3.15: Total execution time of RVSS and GRVSS for 4 shares (Grayscale images).

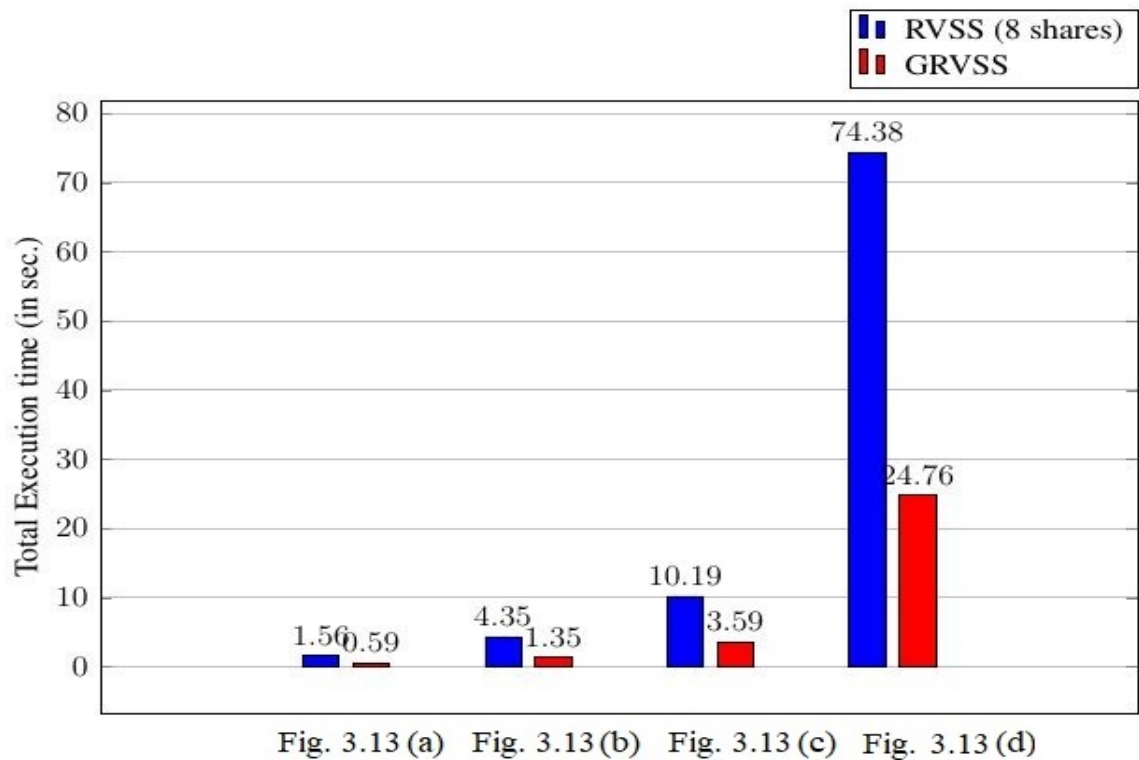


Figure 3.16: Total execution time of RVSS and GRVSS for 8 shares (Grayscale Images)

### 3.2. GPGPU-based Randomized Visual Secret Sharing (GRVSS)

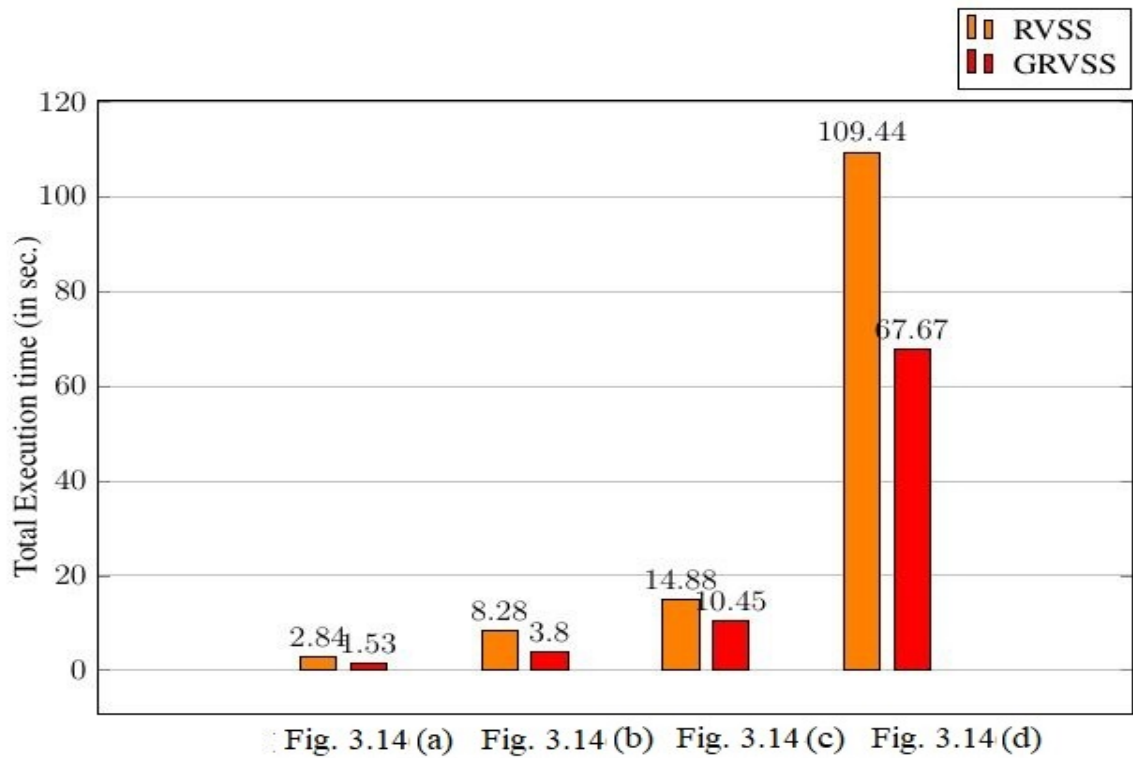


Figure 3.17: Total execution time of RVSS and GRVSS for 4 shares (Colour Images)

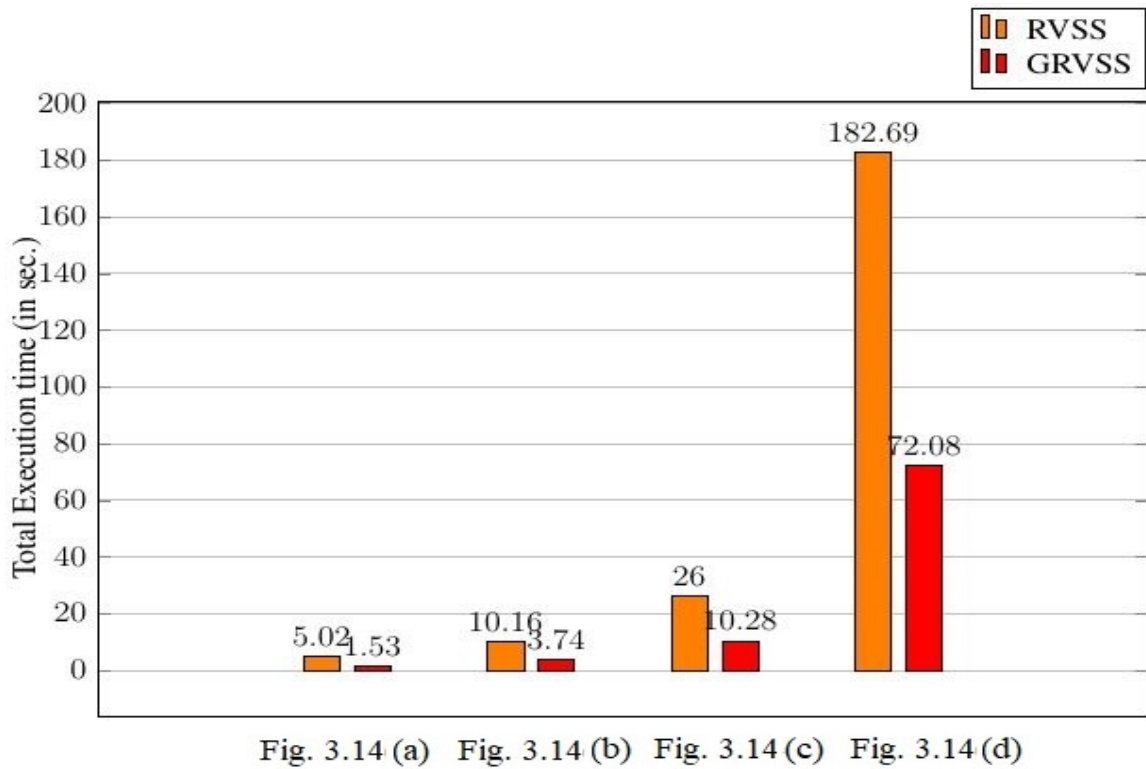


Figure 3.18: Total execution time of RVSS and GRVSS for 8 shares (Colour Images)

Table 3.11: Speedup of GRVSS with 4 shares in generic architecture.

Phases	Grayscale images			
	Fig. 3.13 (a)	Fig. 3.13 (b)	Fig. 3.13 (c)	Fig. 3.13 (d)
Forward Phase	1.587336	1.74744	1.69285	1.746749
Inverse Phase	0.904829	1.048447	1.07413	1.108476
Total	1.234719	1.397036	1.382987	1.427155
Phases	Colour images			
	Fig. 3.14 (a)	Fig. 3.14 (b)	Fig. 3.14 (c)	Fig. 3.14 (d)
Forward Phase	1.520536	1.586369	1.832366	1.727755
Inverse Phase	0.948752	0.9474	1.044923	1.170385
Total	1.230638	1.257212	1.436125	1.468036

Table 3.12: Speedup of GRVSS with 8 shares in generic architecture.

Phases	Grayscale images			
	Fig. 3.13 (a)	Fig. 3.13 (b)	Fig. 3.13 (c)	Fig. 3.13 (d)
Forward Phase	2.615009	2.694567	2.719642	2.808404
Inverse Phase	1.636062	1.77505	1.867799	1.860054
Total	2.110042	2.234319	2.288141	2.32856
Phases	Colour images			
	Fig. 3.14 (a)	Fig. 3.14 (b)	Fig. 3.14 (c)	Fig. 3.14 (d)
Forward Phase	2.480132	2.650981	2.731956	3.271875
Inverse Phase	1.778386	1.575678	2.026382	1.952124
Total	2.125268	2.103622	2.372013	2.614334

Table 3.13: Speedup of GRVSS with 4 shares in PARAM Shavak.

Phases	Grayscale images			
	Fig. 3.13 (a)	Fig. 3.13 (b)	Fig. 3.13 (c)	Fig. 3.13 (d)
Forward Phase	1.522854	1.706311	1.568587	1.675161
Inverse Phase	1.847572	1.687645	1.81697	1.744787
Total	1.69062	1.696953	1.69298	1.710024
Phases	Colour images			
	Fig. 3.14 (a)	Fig. 3.14 (b)	Fig. 3.14 (c)	Fig. 3.14 (d)
Forward Phase	1.786342	2.796913	1.33694	1.402663
Inverse Phase	1.926961	1.59686	1.508932	1.859975
Total	1.857636	2.17872	1.423487	1.615758

Table 3.14: Speedup of GRVSS with 8 shares in PARAM Shavak.

Phases	Grayscale images			
	Fig. 3.13 (a)	Fig. 3.13 (b)	Fig. 3.13 (c)	Fig. 3.13 (d)
Forward Phase	2.392393	2.583595	2.48005	2.602503
Inverse Phase	2.853624	3.855725	3.184561	3.394349
Total	2.630308	3.220337	2.836921	3.00316
Phases	Colour images			
	Fig. 3.14 (a)	Fig. 3.14 (b)	Fig. 3.14 (c)	Fig. 3.14 (d)
Forward Phase	2.834304	2.267696	2.261919	2.211097
Inverse Phase	3.716009	3.145789	2.785332	2.860128
Total	3.280171	2.71467	2.528934	2.534465

the GRVSS scheme achieves the total speedup in the range 1.23 to 1.42 for grey images and total speedup in the range 1.23 to 1.47 for colour images. For colour images, the total speedup is better because of more threads created for colour images. It indicates that the GRVSS achieves better speedups for the images with more number of shares. Similarly, the GRVSS scheme provides better speedup over the PARAM Shavak implementation. The speedup achieved by the GRVSS over the PARAM Shavak is given in Table 3.13 and Table 3.14. The total execution time for grayscale images for 4 and 8 participants is shown in Figure 3.15 and Figure 3.16 respectively. Similarly, the total execution times of RVSS and GRVSS for 4 and 8 shares for colour images are given in Figure 3.17 and Figure 3.18 respectively. The experimental results show that the GRVSS is efficient than RVSS.

### 3.3 SUMMARY

The presented random grid-based GPGPU VSS technique brings the efficiency considerations to the conventional VC to qualify it to the real-time applications. Less computation is very characteristic of any VC. However, due to the massive data to be processed, sequential VCs cannot be used in real-time applications. This work is significant in that it optimizes two stages of conventional random grid-based (2, 2) VC. The reformulation of a solution to fit into a many-core system is a novel approach in the era of evolving many-core systems. The experimental results proved the efficiency

of the presented scheme over the traditional (2, 2) random-grid method. The improved speedup is significant with the increased image resolutions. This method outperforms  $3651\times$  and  $1720\times$  in generating halftone and share images, respectively, for a colour image size of  $1024 \times 1024$ . For this image, the total performance gain is  $2688\times$  more over the conventional method.

We presented another efficient GPGPU based RVSS scheme for sharing images. This model achieves a speedup in the range of  $1.23\times$  to  $1.42\times$  for 4 shares and a speedup range of  $2.11\times$  to  $2.32\times$  for 8 shares in comparison with the generic RVSS. Also, the speedup in the range of  $1.6\times$  to  $1.8\times$  for 4 shares and  $2.63\times$  to  $3\times$  for 8 shares achieved in comparison with the PARAM Shavak supercomputer. In other words, the speedup of the GRVSS is more than RVSS in the generic machine. As the input image size and the number of shares increase, there could be a substantial improvement in the dynamic task-level parallelism for the RVSS with PARAM Shavak than in generic architecture. The GRVSS in PARAM Shavak adds potential fine-grained data-level parallelism to improve the speedup. As the number of shares increases the speedup of GRVSS also increases due to the dynamic allocation of threads and latency tolerance therein to improve the throughput.

Among the two VSS schemes presented in this chapter, the GPGPU-based random-grid VSS scheme has resulted in a significant speedup than the GPGPU-based randomized VSS scheme. This difference is intuitive in that each thread in the GPGPU-based random-grid VSS has less computational overhead than the GPGPU-based randomized VSS scheme. Also, the GPGPU-based random-grid VSS scheme contains purely data-parallel tasks that the GPGPU cores can exploit more efficiently. However, the speedup of the GPGPU-based random-grid VSS scheme is significant as the number of shares increases.

## CHAPTER 4

### VSS USING QUANTUM LOGIC AND GPGPU BASED EDNN SUPER RESOLUTION

Visual Secret Sharing (VSS) is a mechanism used for the secure transmission of secret images. VSS is a significant innovative area in communication and information security (Padiya et al. 2015) like transmitting confidential words, image concealing, approval, and distinguishing proof. The fundamental attraction of VC is the decryption performed without any calculations (Al-Khalid et al. 2017). Securing secrets with this system permits additional confidence and lessens system undependability (Hou et al. 2018; Monoth 2019). The downside of traditional VC systems is that the source image itself gets encrypted with a reversible key. This key is used again for decryption. The visual secret sharing scheme emerged as an alternative mechanism to secure images by encoding the secret images using different shares distributed to connecting individuals (Hou and Quan 2011; Ulutas 2010; Yan et al. 2018). A qualified number of individuals can have the option to retrieve the secret image using their shares (Bharanivendhan and Amitha 2014; Jesalkumari and Sedamkar 2013).

The visual secret sharing (VSS) shares the secret image among the clients (Hou et al. 2013a). Methodical creation of shares facilitates the original image recovery with trivial computation costs. Throughout the years, various researchers have presented VSS systems that restore the secret image with 50% and 25% similarity for noise-like and meaningful shares (Mhala and Pais 2019b). In the designed model, we reduced the computation time of VSS using the CUDA platform. Parallel VSS scheme will attain a

speedup for its usefulness in various real-time applications that require image security (Mayya and Nayak 2017).

Enhancing the contrast of the recreated secret image and lessening the size extension have been the focal points of VSS in the recent past. VSS with little or no size development is regularly picked since it forces lesser troubles on handling and stockpiling (Liu et al. 2012). Various techniques were presented for size invariant VSS comprising as; random grid (RG) calculations (De Prisco and De Santis 2014; Yan et al. 2016), block encoding calculations, and Extended visual cryptography scheme (EVCS) (Yamaguchi 2014) for general access structures (GAS). Other approaches to ensure the size invariance of shares include the adaptive area augmenting XOR-based VC, XOR-based visual cryptography (XVCS), Random visual cryptography strategy, and XOR-based broadened visual cryptography (Naphade and Khobaragade 2016).

At present, there are numerous algorithms for super-resolution (SR). In the investigation of interpolation dependent image super-resolution reconstruction methodologies, the most typical interpolation techniques comprise the nearest neighbour interpolation, the linear interpolation, the bicubic interpolation, and the spline interpolation. The utmost standard dictionary-based technique is sparse encoding, which looks for a meagre portrayal of LR input picture squares and afterward utilizes the coefficients of this inadequate portrayal to create HR yield. The LapSRN model (Lai et al. 2018) depends on the pyramid calculation.

The important contributions of this work are summarized below.

- The secret input image is changed into a halftone image format by utilizing Error diffusion with varying thresholds (EDVT) method for further effective processing.
- DWT is utilized for the effective encoding of shares. Also at the receiver side, the encoded image is reconstructed using XOR operation.
- Enthalpy based adaptive deep neural network (EDNN) is designed with the General Purpose Graphic Processing Unit (GPGPU) to enhance the resolution of the



reconstructed images and to lessen the time complexity of deep learning.

#### 4.1 COMPARISON OF EXISTING SUPER-RESOLUTION MODELS

He et al. (2018) presented deep neural systems with numerous responsive fields to build infrared pictures' spatial goal by a considerable scope factor ( $\times 8$ ). Rather than recreating an HR image from its LR rendition utilizing a solitary complex profound system, the methodology's critical thought is to set up a mid-point (scale  $\times 2$ ) between scale  $\times 1$  and  $\times 8$  with the end goal that lost data can be separated into two segments. Lost data inside every part contains comparative examples in this way can be all the more precisely recuperated in any event, utilizing a lucid, deep model. Then two successive deep neural models with various responsive fields trained together through a multi-scale loss function.

Qin et al. (2020) presented a novel multi-scale feature fusion residual network (MSFFRN) to make use of image characteristics of Single Image Super-Resolution (SISR). In light of the remaining learning, the investigators presented a multi-scale feature fusion residual block (MSFFRB) with numerous interlaced ways to identify and intertwine picture highlights at various scales adaptively. This model also uses the yields of each MSFFRB and the shallow highlights to combine general characteristics to achieve the HR image.

Qiu et al. (2020) invented a medical image super-resolution that used a sub-pixel layer of an efficient sub-pixel convolution neural network (ESPCN) with the three hidden layers of the SR convolution neural network (SRCNN) to improve the efficiency. The cascaded kernels improved the speed and quality of the image.

Cao et al. (2019) established a trade-off between speed and the contrast in SISR using a simple yet efficient neural model compared to the conventional complex models. Restructuring the residual blocks and eliminating the connection between the layers resulted in the simplicity of the model. This method also brought flexibility of adaptively adjusting the high and low contrast regions of the reconstructed image in this method.

Deka et al. (2020) investigated a novel SISR plan to improve spatial goal of Diffusion-Weighted (DW) and Spectroscopic Magnetic Resonance (MRS) pictures. It depends on

Table 4.1: Comparison of existing super-resolution models.

References	Contribution	Merits	Limitations
(He et al. 2018)	Streamed Deep Networks with many receptive fields for Infrared Image Super-Resolution	Attains better reconstruction accuracy utilizing suggestively less parameters	Need reduction in computational cost and Optimize the number of parameters.
(Qin et al. 2020)	Multiple-scale feature fusion residual framework for SISR	Attains improved accuracy, speed, and graphic quality	Computational cost
(Qiu et al. 2020)	Super-resolution rebuilding of knee magnetic resonance imaging based on deep learning	Better reconstruction speed and reconstruction quality	Difference concerning effectiveness of the reconstructed edge and the HR image with a scope to improve the speed using many-core systems.
(Cao et al. 2019)	Improved speed and quality SISR using energy-enhanced deep neural model	Enhances SISR accuracy eliminating redundant work loads	Trade-off between the speed and precision with the simplicity of the neural model.
(Deka et al. 2020)	DW and spectroscopic MRI super-resolution by sparse depictions	Uses accelerator to get computationally efficient results and empirically effective	Computationally efficient in the clinical application.
(Yuan et al. 2019)	Fast SISR method through CUDA	Method gets a significant efficiency with the collaborative computing.	The parallel part constrained the speed as the feature map extraction is not parallelized. The precision restricted to the limited training set though there is scope for using other transformations and multiple-type filters.

fix astute meager remaking of HR patches from LR highlight patches. Remaking exploits the sparsity of MR image as well as use the non-neighbourhood self-closeness of patches of the information LR image as earlier information. At long last, the presented calculation is additionally actualized utilizing the GPGPU-based equal equipment alongside successive executions so as to feature its potential for genuine clinical applications.

In the SR model in Yuan et al. (2019), the training set is expanded by rotating the LR image obviating the need for external data set for training. This model takes advantage of multiple-scale Gaussian filters to produce multiple-view feature maps. The nonlocal central tendency applied to the LR patches, which act as internal training data, inferred the HR patches. The method proved considerable improvement in speed and visual quality, utilizing the CUDA GPGPU platform. Table 4.1 shows the merits and limitations of the investigations discussed in this section.

## 4.2 PRESENTED METHODOLOGY

In this scheme, an input secret image is initially converted into a halftone image utilizing EDVT. By then, shares are produced utilizing quantum logic coding and in the embedding phase, DWT is utilized to embed shares. The XOR operation is utilized to recreate the images from embedded shares. It results in low-resolution images. Lastly, EDNN is designed with the General Purpose Graphic Processing Unit (GPGPU) to enhance the resolution of reconstructed images. The flow diagram of the presented technique is given in Figure 4.1. It consists of the following phases:

1. Sender phase: Tasks involved in this phase are as follows: (a) Half-toning using Error diffusion with varying thresholds (EDVT), (b) Shares generation using quantum logic coding, and (c) Embedding share.
2. Reconstruction phase: Tasks involved in this phase are as follows: (a) Reconstruction of the image by logical XOR and (b) contrast enhancement of the reconstructed image using EDNN.

The details of these phases is described below.

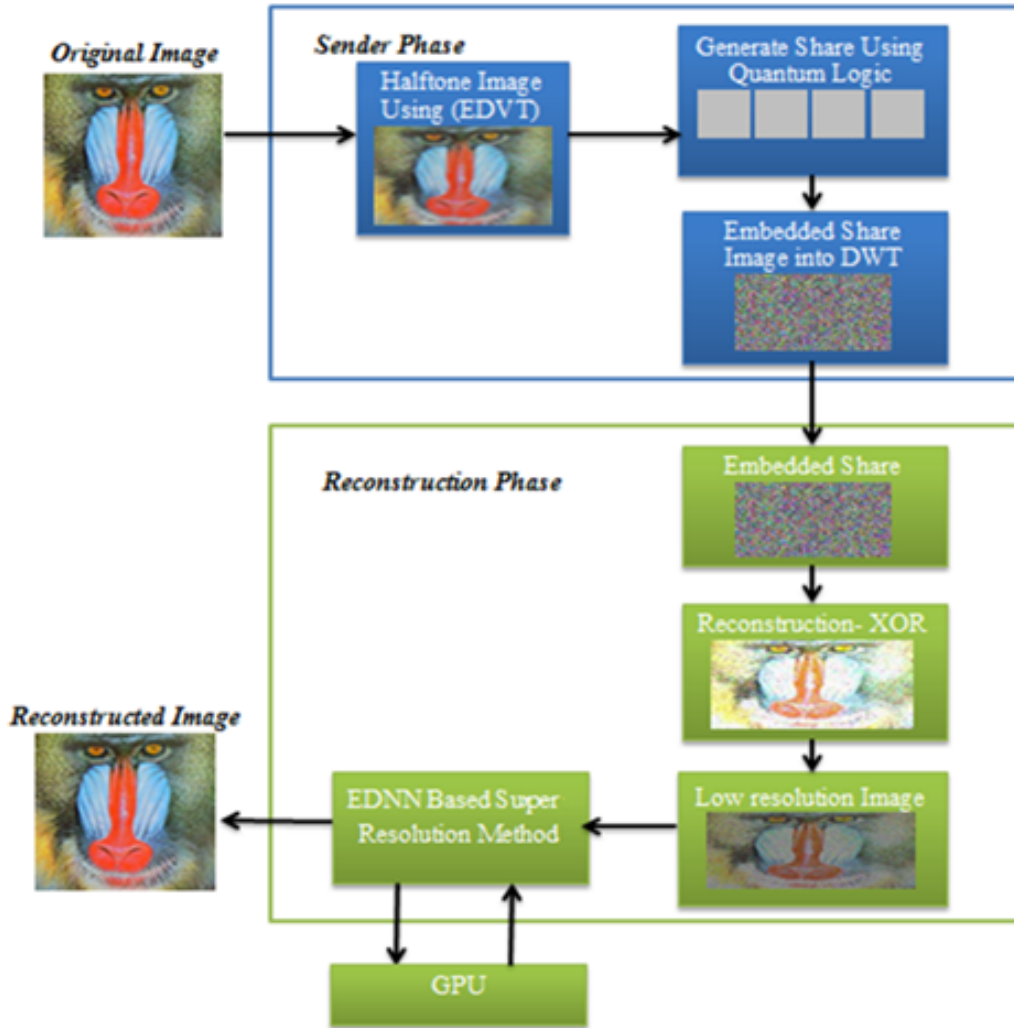


Figure 4.1: Flow diagram of the presented methodology.

#### 4.2.1 Half-toning using EDVT

The input image is converted into a halftone image by using EDVT. Error diffusion provides halftone shares with enhanced image quality. Algorithm 4.1 portrays the EDVT based colour halftoning in secret images. Figure 4.2 shows the flowchart for EDVT colour halftoning. The details of this process are as follows.

**Step 1:** The initial requirement for applying EDVT is to separate the red (R), green (G), blue (B) channels from the input image (Srividhya et al. 2019).

**Step 2:** The EDVT halftoning is then used independently for all channels. Consider  $m' \times m'$  denotes the window size and  $M' \times N'$  denotes the image size. The mean of the pixels in the window is determined and entered in the dithered threshold array with

**Algorithm 4.1:** EDVT color halftoning.

---

```

Input: Color image  $S'$  of size  $(M' \times N')$ , dither array size  $(m' \times m')$ ,
        normalized factor  $(n')$ .
Output: Halftone image  $(B)$  of size  $(M' \times N')$ .
// Separate the color image into 3 channels and
// follow this step for each channel.
// Compute threshold for each dither array of  $S'$ .
1 for  $i = 1:M'/m'$  do
2   for  $j = 1:N'/m'$  do
3      $w = S'((m'(i-1)+1) : m'i, (m'(j-1)+1) : m'j);$ 
4      $DT(i, j) = \text{mean}(w);$ 
5   end
6 end
7 for  $i = 1:M'$  do
8   for  $j = 1:N'$  do
9     // Compute threshold  $T(i, j)$  for each pixel of the
10    image.
11    if  $DT[i/m', j/m'] \leq 127$  then
12       $T(i, j) = 127 + \frac{DT[i/m', j/m']}{n'}$  else
13       $T(i, j) = 127 - \frac{DT[i/m', j/m']}{n'}$ 
14    end
15    // Compare color image pixel against the
16    threshold
17    if  $S'(i, j) \geq T(i, j)$  then
18       $B(i, j) = 1$  else
19       $B(i, j) = 0$ 
20    end
21    // Diffuse the quantization error to the nearby
22    pixels
23     $\phi(i, j) = S'(i, j) - B(i, j);$ 
24     $S'(i+x, j+y) = S'(i, j) + \phi_{x,y}\phi(i, j);$ 
25    //  $\phi_{x,y}$  are coefficients of error diffusion
26    filter
27    // Concatenate 3 channels to obtain an EDVT
28    color halftone image.
29  end
30 end

```

---

the size of  $M'/m' \times N'/m'$ .

**Step 3:** The mean is normalized using a factor  $n'$ , where  $n' = 1, 2, 3, \dots, 255$ .

**Step 4:** The addition or the subtraction of the respective normalized mean value in the

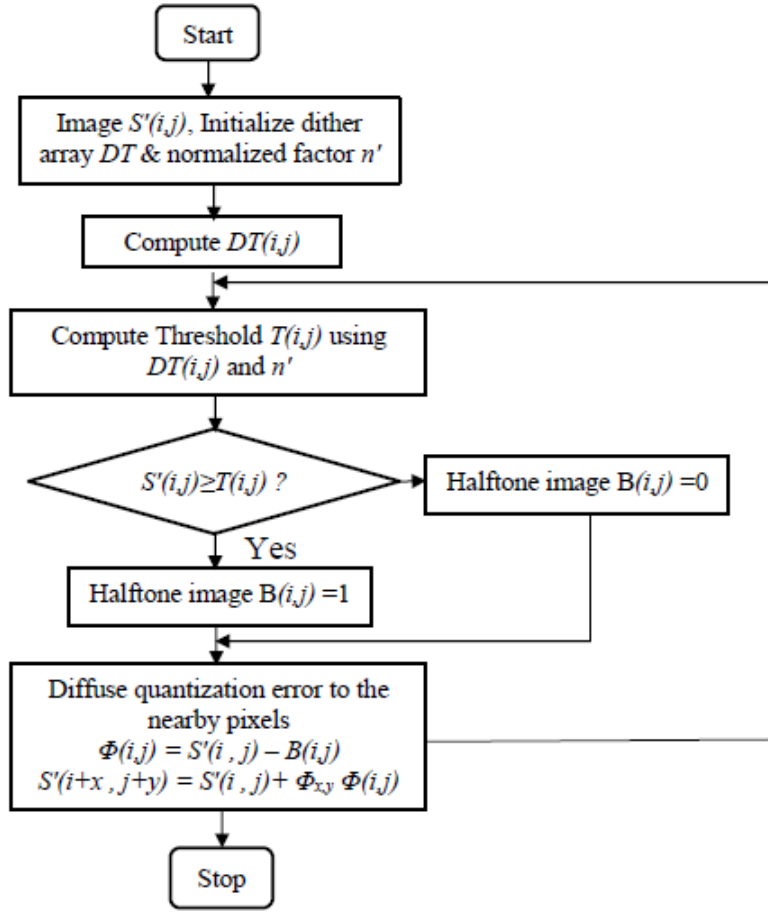


Figure 4.2: Flow chart for EDVT Color half-toning.

threshold array 127 gives the final threshold for every square of pixels in the input image  $S(i, j)$ .

**Step 5:** We assign a white pixel to the halftone image if image pixel  $S(i, j)$  is greater than the final threshold. Otherwise, we assign a black pixel. Now we diffuse quantization error to the surrounding pixels.

**Step 6:** Finally, we overlap the RGB channels to obtain a halftone image.

The resulting EDVT halftoned image has enhanced contrast when compared to the original image.

#### 4.2.2 Shares generation using quantum logic coding

The image is encoded in  $n'$  shares and the message is revealed by assembling of those  $n'$  shares. The quantum conditions of subsystems just as composite system are  $|\psi_A\rangle$ ,  $|\psi_B\rangle$  and  $|\psi_A\rangle \otimes |\psi_B\rangle$  respectively. In the event that two subsystems entangled one another,

the relationship is depicted in equation (4.1).

$$\psi_{AB} \neq |\psi_A\rangle \otimes |\psi_B\rangle \quad (4.1)$$

There is a composite quantum bit  $|\psi_{AB}^+\rangle$ , which is a quantum entangled state. It must satisfy the underneath condition in equation (4.2).

$$|\psi_{AB}^+\rangle = \frac{1}{\sqrt{2}} \{ |0_A\rangle \otimes |1_B\rangle + |1_A\rangle \otimes |0_B\rangle \} \quad (4.2)$$

When the,  $|\psi_A\rangle = |0\rangle$  the state of  $|\psi_B\rangle$  is the inverse  $|1\rangle$  and vice versa. Nonetheless, when  $|\psi_A\rangle$  a collapsed to Eigen state  $|1\rangle$  by measurement,  $|\psi_B\rangle$  unavoidably collapses to opposite Eigen state  $|0\rangle$  and vice versa. In like manner, the halftone image is coded into a number of shares in the share generation phase. Share generation algorithm using quantum logic coding is given in Algorithm 4.2. Also, the flow chart for share generation using quantum logic coding is given in Figure 4.3.

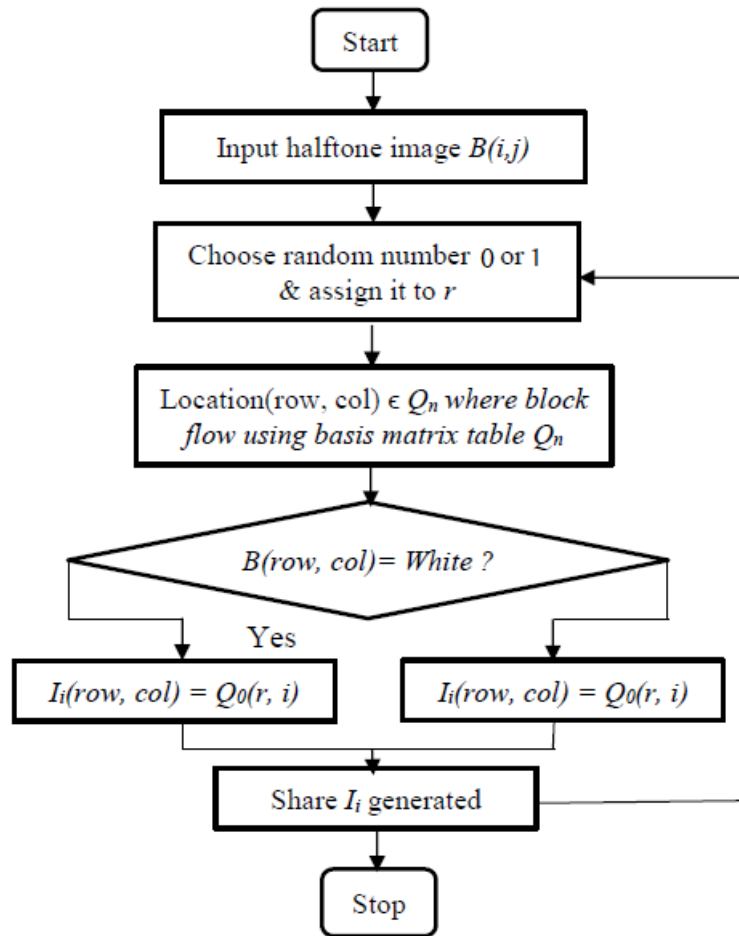


Figure 4.3: Flow chart for Share generation using quantum logic coding.

---

**Algorithm 4.2:** Share generation using quantum logic coding.

---

**Input:** Halftone image  $B$  of size  $(M' \times N')$   
 Number of participants  $n'$ .  
 The  $n' + 1$  number of basis matrices  $Q_0, Q_1, \dots, Q_{n'}$ .  
**Output:**  $n'$  shares:  $I_i$ , where  $i = 1, 2, \dots, n'$ .

```

1 for row = 1:w do
2   for col = 1:h do
3     Location  $(row, col) \in Q_n$  where, block flow using Table 6.2.
4     Choose random number 0 or 1 and assign it to  $r$ ;
5     for k = 1:n' do
6       if  $B(row, col)$  is White then
7          $I_k(row, col) \leftarrow Q_0(r, k)$  else
8          $I_k(row, col) \leftarrow Q_{n'}(r, k)$ 
9       end
10    end
11  end
12 end
13 end
    
```

---

Share generation algorithm fundamentally accepts premise lattices for creating shares with gatherings of zeros and ones. To produce  $n'$  shares,  $n' + 1$  basis matrices with an aspect of  $2 \times n'$  are formed. The premise matrix  $Q_0$  is intended for a white pixel and  $Q_1, \dots, Q_n$  are intended for a black pixel. Here,  $Q_0$  comprises two rows and  $n$  columns ( $n' = 4$ ). The generated base matrix, dependent on condition 4.2, is given in Table 4.2.

Table 4.2: Generated basis matrix.

---


$$Q_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad Q_1 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$Q_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad Q_3 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

$$Q_4 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$


---



### 4.2.3 Embedding share and recover the secret image

The presented model uses the DWT reversible data embedding approach to embedding secret image information into shares by changing the image to the frequency domain. Equation (4.3) represents DWT shifting and scaling of mother wavelet  $\tilde{\psi}_{s,\tau}$  by a power of two.

$$\tilde{\psi}_{s,\tau}(t) = \frac{1}{\sqrt{2^s}} \tilde{\psi} \left( \frac{t - r \times 2^s}{2^s} \right) \quad (4.3)$$

Where  $s$  is the scale factor and  $r$  is the shift parameter. The information  $x(n')$  is separated into four sub-bands as coarse information, and three detail information. Consider,  $\hat{h}_0(k')$  and  $\hat{h}_1(k')$  are low-pass and high-pass filters. The four sub-bands are recursively integrated to obtain the original signal  $x(n')$  with the filters satisfying the equations in 4.4, 4.5, and 4.6.

$$\hat{h}_1(k) = (-1)^k \hat{h}_0(\hat{l} + k) \quad (4.4)$$

$$\hat{h}_0(k) = \hat{h}_0(\hat{l} + 1 - k) \quad (4.5)$$

$$\hat{h}_1(k) = (-1)^k \hat{h}_0(\hat{l} + 1 - k) \quad (4.6)$$

Here,  $\hat{l}$  is the length of filters and  $n = 1, 2, \dots, \hat{l}$ . Equation 4.7 computes the embed coefficient  $I'(i, j)$  of the selected halftone image block  $B(i, j)$ .

$$I'(i, j) = B(i, j) + \tilde{\varphi}(w - 1) \quad (4.7)$$

Where  $w$  is the watermark bit, and  $\tilde{\varphi}$  has the embedding strength coefficient that controls the embed strength.

The data embedding steps are defined as follows

**Step 1:**

An original image with pixel dimensions of  $256 \times 256$  is first selected.

**Step 2:**

A shared image with pixel dimensions  $64 \times 64$  is used as an embed. The embed image is then converted into binary share.

**Step 3:**

- Three levels of wavelet decomposition are made for the original cover image with the based filters  $\hat{h}_1(k)$  and  $\hat{h}_0(k)$ . DWT processes the image by splitting it into four non-overlapping multi-resolution sub-bands:  $LL$ ,  $LH$ ,  $HL$ , and  $HH$ .

- The sub-bands  $LH$ ,  $HL$ , and  $HH$  signify the fine-scale of DWT, while the sub-band  $LL$  signifies the coarse-scale of DWT coefficients.
- For each of the further wavelet decomposition level, the  $LL$  sub-band of the previous level is used as input.
- Lastly, we obtain four sub-bands of three levels, namely,  $LL3(eB3)$ ,  $LH3(eC3)$ ,  $HH3(eE3)$ , and  $HL3(eH3)$  each of which is  $64 \times 64$  pixels.

**Step 4:**

The three wavelet sub-band coefficients  $eC3$ ,  $eE3$ , and  $eH3$  with pixel sizes of  $64 \times 64$  are split into non-overlapping small blocks with pixel sizes of  $4 \times 4$ . This technique produces  $16 \times 16$  blocks for each coefficient. This fine-scale DWT coefficients, renders the watermark imperceptible to the human eye.

**Step 5:**

Share blocks are embedded into the  $eC3$ ,  $eE3$ , and  $eH3$  blocks by the subsequent embedding formula (4.7).

**Step 6:**

Later, the image can be reconstructed from these DWT coefficients. This rebuilding procedure is called the inverse DWT (IDWT). Finally, the XOR operation of the recreated shares results in the low-resolution image as shown in Figure 4.4. This low-resolution image is given as input to the EDNN to obtain the super-resolution reconstructed image.

#### 4.2.4 Enthalpy based DNN (EDNN)

The presented model uses EDNN for super-resolution to enhance the resolution of the reconstructed image. The use of GPGPU reduces the time complexity in deep learning. Enthalpy based DNN demonstrated considerably adaptive to the resolutions during training in contrast to the traditional neural networks with a capability to create high-resolution images. EDNN comprises layers such as convolution, enthalpy based normalization, pooling, and a fully connected layer. Figure 4.5 depicts the structure of the EDNN.

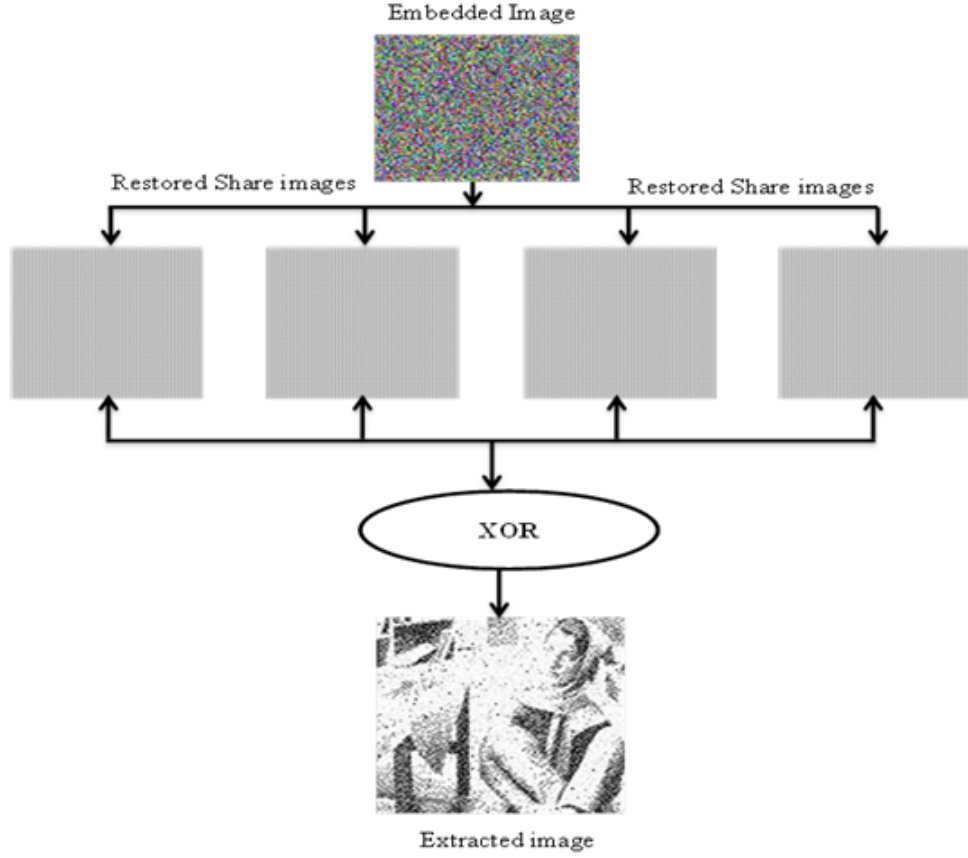


Figure 4.4: Overview of the low-resolution image formation.

The overall quality of the EDNN depends upon the weights and biases of the preceding layers in the framework. The conditions (4.8) and (4.9) show how the coefficients are updated independently in all layers of EDNN.

$$\Delta W_l = -\frac{x\lambda}{r}W_n - \frac{x}{N_t}\frac{\partial C}{\partial W_n} + m\Delta W_n(t) \quad (4.8)$$

$$\Delta B_n = -\frac{x}{n}\frac{\partial C}{\partial B_n} + m\Delta B_n(t) \quad (4.9)$$

Where,  $W_n$  signifies the weight,  $B_n$  signifies the bias,  $n$  signifies the layer number,  $\lambda$  denotes the regularization parameter,  $x$  signifies the learning rate,  $N_t$  signifies the total number of training samples,  $m$  signifies the momentum,  $t$  signifies the apprising phase and  $C$  signifies the cost function. We discuss the layers of EDNN below.

1. **Convolutional layer:** The convolution layer uses multiple learned weight matrices named filters to glide across input features by a stride, as shown in equation (4.10). The layer repeats this procedure to cover the entire image for extract-

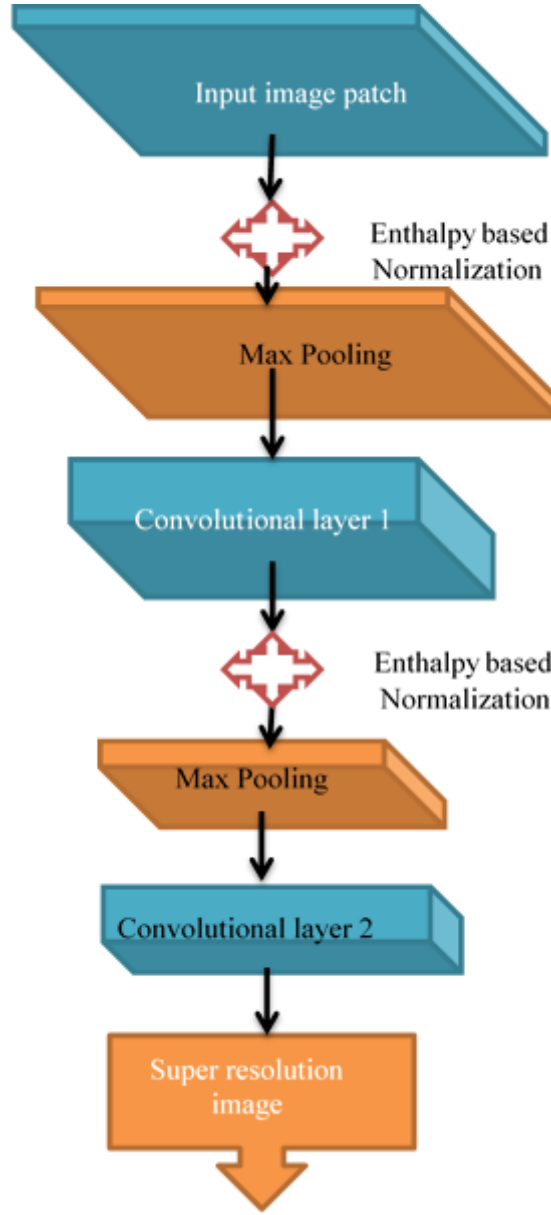


Figure 4.5: Architecture of the EDNN.

ing features. This layer also contains non-linear activation function to replace all negative values with zero.

$$C_k = \sum_{m=0}^{M-1} y_n \hat{h}_{k-n} \quad (4.10)$$

Where,  $y_n$  denotes recreated low resolution images,  $\hat{h}$  denotes the filter, and  $M$  signifies the quantity of components in  $y$  and the output vector is  $C_k$ .

2. **Enthalpy based normalization layer:** The enthalpy based normalization is used

to mitigate the over-fittings in layers. Enthalpy is calculated using equation (4.11).

$$\hat{E}_y = N' + S' \quad (4.11)$$

Where  $S'$  is given by equation (4.12).

$$S' = (s_1 \times s_2 \times s_3 \times \dots \times s_k) \quad (4.12)$$

Where,  $\hat{E}_y$  denotes the enthalpy of the system,  $N'$  signifies the quantity of resultant windows,  $s_k$  signifies the size of convolution layer result, here  $k = 1, 2, \dots, n$ .

**Normalization:** Normalization performs the linear transformation of the data to fit appropriate into a specific range. We used Z-score normalization in condition (4.13).

$$Z_{norm} = \frac{\hat{E}_y - \mu}{\sigma} \quad (4.13)$$

Here,  $Z_{norm}$  signifies the normalized output,  $\hat{E}_y$  signifies the enthalpy esteem,  $\mu$  and  $\sigma$  signifies the mean and standard deviation of the esteems in the output image.

3. **Pooling layer:** The pooling layer between convolution layers is also called the downsampling layer. This layer diminishes the spatial volume of the input image, computation, and overfitting.
4. **A fully connected layer:** A fully connected layer links each neuron in the preceding layer to every neuron in the next layer. We used the non-linear soft-max function given in the equation (4.14).

$$p_i = \frac{e^{y_i}}{\sum_{i=1}^k e^{y_i}} \quad (4.14)$$

Where,  $y$  denotes the resultant reconstructed image.

### 4.3 RESULTS AND DISCUSSION

We implemented the presented model in MATLAB and assessed the outcomes on the desktop with a GPGPU. Figure 4.6 depicts the sample Baboon, Barbara and Lena input images. The sample secret input images are converted into a halftone images by uti-

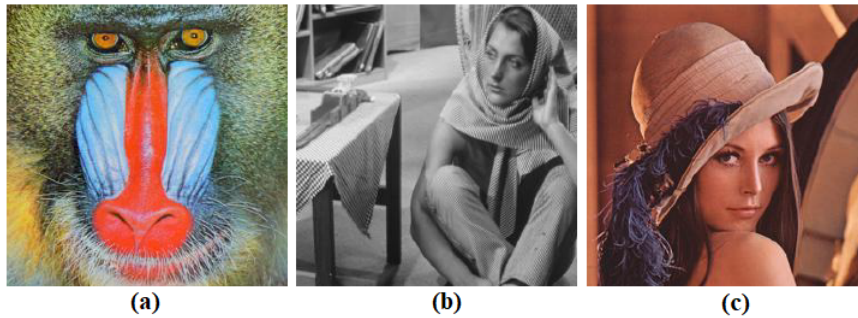


Figure 4.6: Test input images.

lizing EDVT technique as shown in figure 4.7 . The halftone image is converted into

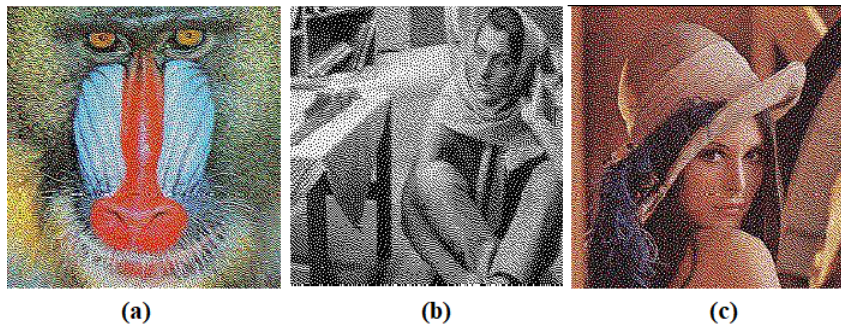


Figure 4.7: Halftone images.

several shares. The share generation phase generates shares using basis matrices. Figure 4.8 (a), (b), (c) depicts the generated shares. In embedding stage, discrete wavelet transform is utilized for encoding the generated shares. The embedded images of generated shares are depicted in Figure 4.9. The images are extracted from the embedded shares. The extracted images are the resultant of the embedded shares and the extracted sample images are depicted in Figure 4.10. The XOR operation of the shares results in the low-resolution reconstructed secret image as shown in Figure 4.11. EDNN is finally designed with the GPGPU to enhance the resolution of the reconstructed images and to lessen the time complexity of deep learning. This results in super-resolution of images. The final super-resolution output images are depicted in Figure 4.12. EDNN relies upon the execution on CPU with GPGPU. The idea keeping the GPGPU constantly occupied while the CPU would possess time for different tasks as exposed in Figure 4.13.

We also used datasets available at SIP (2020) and BPV (2020) to compare the investigation outcomes of the presented system with the RVSS (Mhala et al. 2017) and



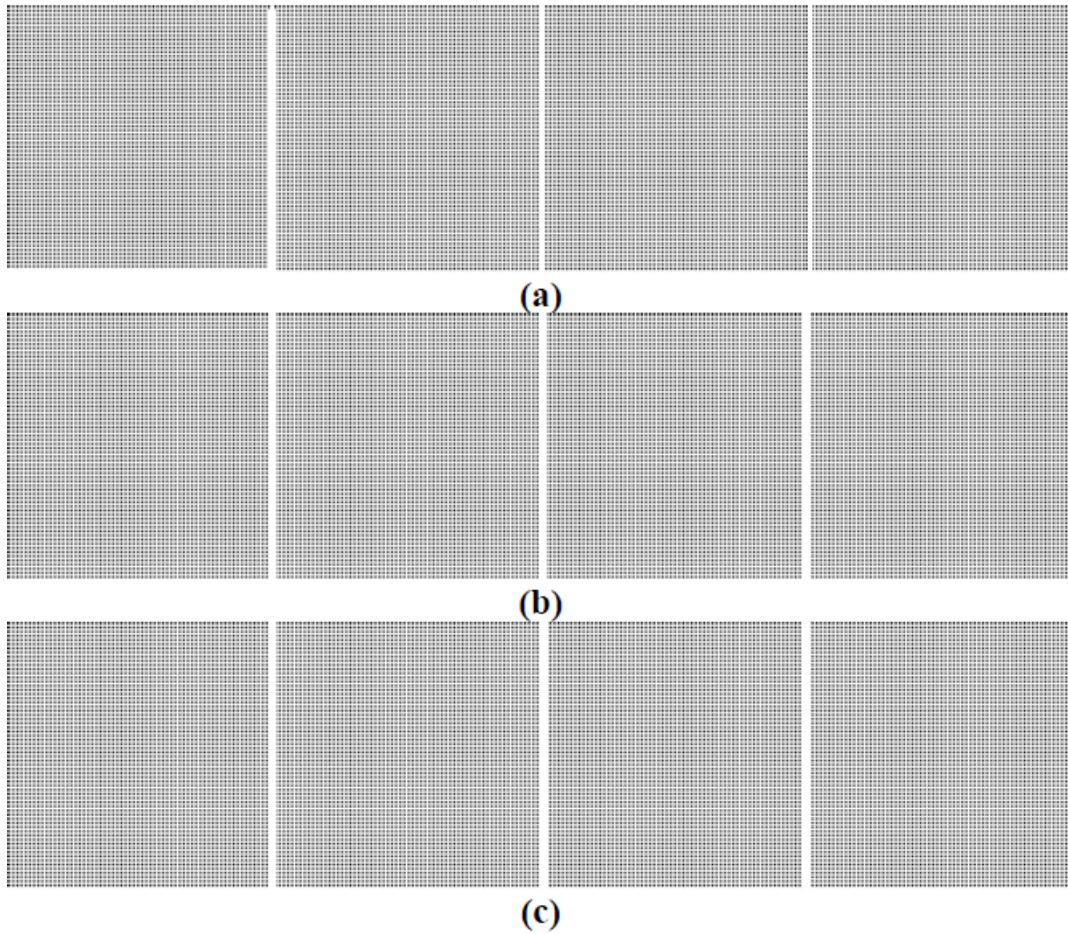


Figure 4.8: Share images of halftone images.

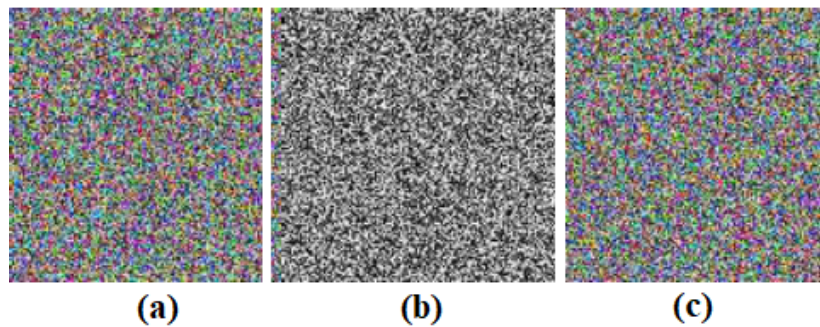


Figure 4.9: Embedded images.

BPVSS (Hou et al. 2013a) schemes. Figure 4.14 shows three test images chosen to discuss the presented model's performance with the RVSS and BPVSS models. From Table 4.3, it is evident that the presented model gives the Mean Square Error (MSE) values of the retrieved secret images close to zero. The House image gives the greatest estimation of 0.037 for meaningful shares and the smallest error estimation of 0.0014.

4. VSS using Quantum logic and GPGPU based EDNN Super Resolution

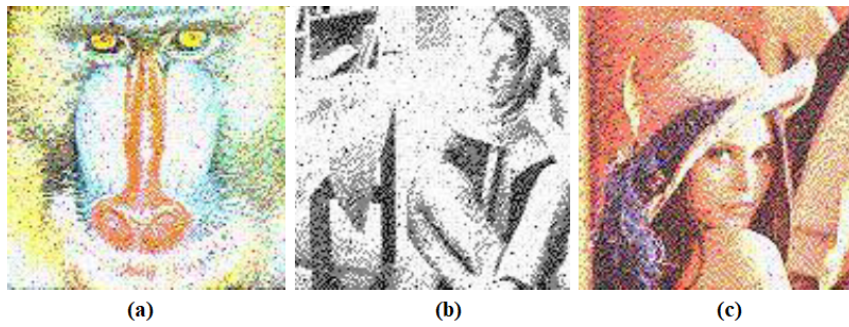


Figure 4.10: Extracted images.

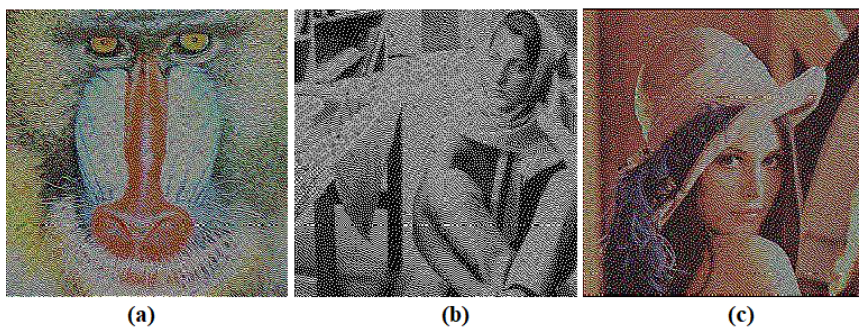


Figure 4.11: Reconstructed images.



Figure 4.12: Final super resolution images.

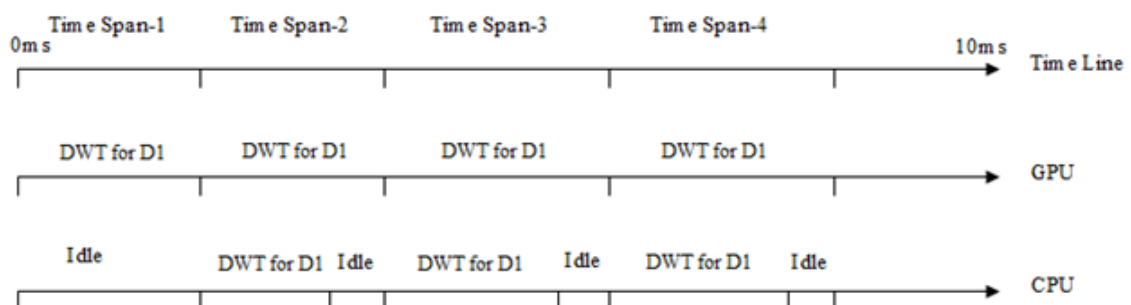


Figure 4.13: Time overlapping architecture of the implementation.





Figure 4.14: Color test images of size  $(256 \times 256)$ .

The MSE- human visual system (HVS) esteem for different test images is mentioned in Table 4.3. The tested GPGPU contains a large number of estimation threads executed in parallel. The most usually utilized GPGPU architecture is CUDA from Nvidia. The presented DWT-EDNN  $16 \times 16$  blocks fits better on a GPU. The performance assessment for the presented model against its sequential technique in terms of execution speed is specified in Table 4.4. It clearly portrays that the execution time of the presented strategy with GPU is taking much lesser time than its CPU model.

#### 4.3.1 Performance Analysis

In this section, the presented GPU model performance is compared with the RVSS (Mhala et al. 2017) and BPVSS (Hou et al. 2013a) models utilizing other statistical measures to further demonstrate its effectiveness.

##### Normalized Cross Correlation ( $NCC$ )

$NCC$  processes the resemblance among secret and reconstructed images. The highest esteem of  $NCC$  demonstrates the highest likeness amongst two images.  $NCC$  is calculated by using equation (4.15),

$$NCC = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [I(x, y) \times g'(x, y) / I^2(x, y)] \quad (4.15)$$

The presented technique provides  $NCC$  in the ranges of 0.9663 to 0.9987 and 0.639 to 0.8017 for noise-like shares and meaningful shares as evident in Table 4.5.

Table 4.3:  $MSE$  esteems for different test images.

Test images	Type	Presented scheme		RVSS		BPVSS	
		Noise like	Meaningful	Noise like	Meaningful	Noise like	Meaningful
Female	Color	0.0017	0.0132	0.031	0.032	0.3945	0.4706
Couple	Color	0.0019	0.0050	0.018	0.016	0.4511	0.5328
House	Color	0.0014	0.0370	0.054	0.093	0.4987	0.5642

Table 4.4: Comparison analysis in terms of execution speed.

Desktop platform	Presented scheme	Execution Time
		GPU time (DWT-EDNN)
	CPU time	7.3ms

Table 4.5:  $NCC$  esteems for test images.

Test images	Type	Presented scheme		RVSS		BPVSS	
		Noise like	Meaningful	Noise like	Meaningful	Noise like	Meaningful
Female	Color	0.9871	0.7832	0.777	0.718	0.4998	0.2489
Couple	Color	0.9663	0.8017	0.703	0.701	0.4879	0.2484
House	Color	0.9987	0.6390	0.798	0.641	0.5010	0.2479

Table 4.6:  $NAE$  esteems for the test images.

Test images	Type	Presented scheme		RVSS		BPVSS	
		Noise like	Meaningful	Noise like	Meaningful	Noise like	Meaningful
Female	Color	0.0987	0.597	0.650	0.616	0.5010	0.7502
Couple	Color	0.1879	0.501	0.245	0.263	0.4989	0.7492
House	Color	0.0787	0.361	0.372	0.431	0.5001	0.7498

### Normalized Absolute Error ( $NAE$ )

$NAE$  demonstrates the error among secret and reconstructed images. The lowest  $NAE$  infers the superior quality of the reconstructed image. The  $NAE$  is calculated using equation (4.16). Table 4.6 shows the  $NAE$  for the recovered images using existing schemes and the presented EDNN based secret image sharing technique. From the Table 4.6, the presented framework  $NAE$  esteems are superior to BPVSS and RVSS schemes.

$$NAE = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [I(x, y) - g'(x, y)] / I(x, y) \quad (4.16)$$

#### 4.3.2 Effectiveness of super-resolution

In this section, we demonstrate the degree to which the image quality is enhanced using super-resolution. Therefore, we compare the presented model's image quality using super-resolution with the image quality of the same model without super-resolution. We use the following statistical measures for this comparison.

#### Peak Signal to Noise Ratio ( $PSNR$ )

$PSNR$  is the proportion of the most extreme possible pixels to the noise in the input image as given in the equation (4.17).

$$PSNR = 10 \log_{10} \left[ \frac{MAX_i^2}{MSE} \right] \quad (4.17)$$

Where,  $MAX_i$  is the most extreme number of pixels, and  $MSE$  is the mean square error. The presented secret image sharing with super-resolution improves the quality of the secret image significantly as shown in Figure 4.15.

#### Mean Squared Error ( $MSE$ )

$MSE$  estimates the average squared difference between the assessed and the original pixel values. The  $MSE$  is given in equation (4.18).

$$MSE = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [g(x, y) - g'(x, y)]^2 \quad (4.18)$$

Here,  $g(x, y)$  represents the desired outcome, and  $g'(x, y)$  represents the current outcome,  $MN$  denotes the range of the chosen pixels. The performance analysis of the

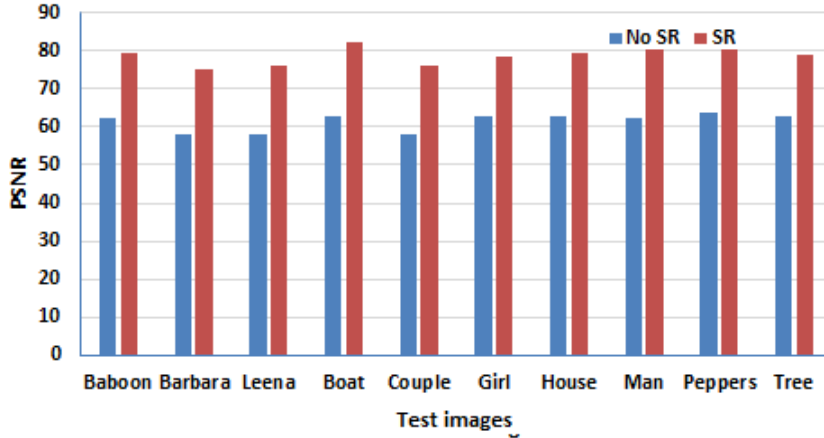


Figure 4.15: Accuracy of the presented model with and without super-resolution.

presented secret image sharing, with and without super-resolution is depicted in Figure 4.16. Figure 4.16 shows that super-resolution has resulted in a considerable reduction

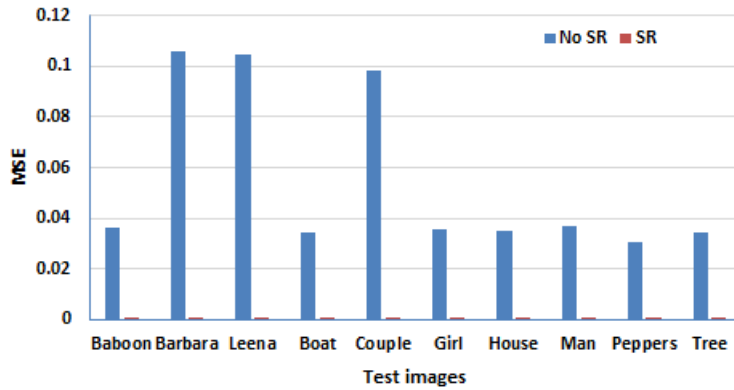


Figure 4.16: MSE of the presented model with and without super-resolution.

in MSE values of the recovered images.

### Signal to Noise Ratio (SNR)

SNR is the proportion of pixel signal value to the noise.

$$SNR = 10 \log_{10} \frac{P'}{N'} \quad (4.19)$$

Where,  $P'$  denotes the possible pixel value and  $N'$  denotes the corrupting noise. The performance investigation graph of the presented secret image sharing with and without super-resolution is depicted in Figure 4.17. Figure 4.17 shows that super-resolution of the presented secret image sharing resulted in better SNR against the model without super-resolution.

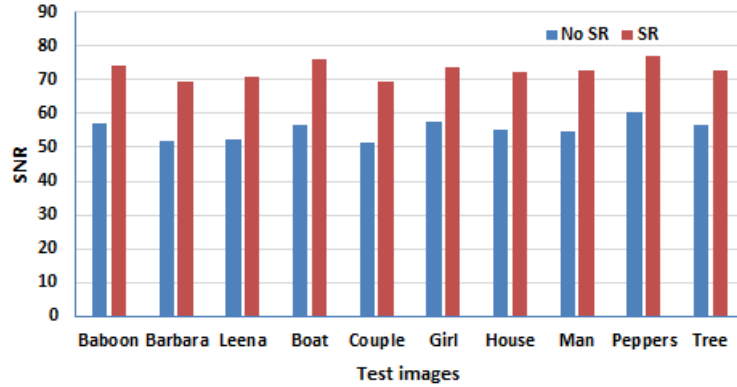


Figure 4.17:  $SNR$  of the presented model with and without super-resolution.

### 4.3.3 Multiple Scales based PSNR and Execution Time

The PSNR and execution time comparison results between presented and existing super-resolution convolution neural network (SRCNN) (Zeng et al. 2019), Efficient sub-pixel convolutional neural network (ESPCNN) (Zeng et al. 2019), BPVSS (Hou et al. 2013a) and RVSS (Mhala and Pais 2019a) techniques are depicted in Figure 4.18. We found that the network manages with any scale used during training. When train =  $(\times 2, 3, 4)$ , its  $PSNR$  for each scale is comparable to those achieved from the corresponding result of single-scale. The scale  $\times 2$   $PSNR$  and execution time measures are given in Figure 4.18. In Figure 4.18, the examination result portrays that our method

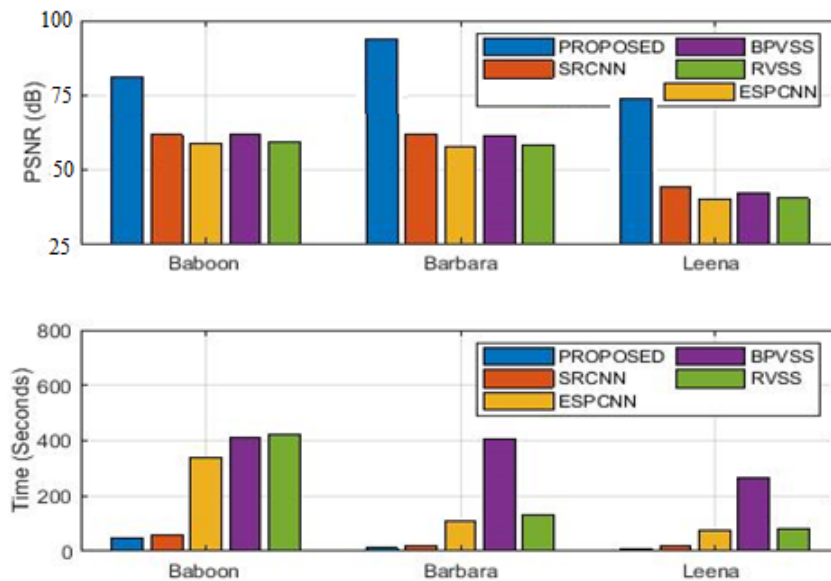


Figure 4.18: Comparison analysis of  $PSNR$  and time in Scale  $\times 2$ .

#### 4. VSS using Quantum logic and GPGPU based EDNN Super Resolution

outperforms the existing techniques. The comparison analysis of scale  $\times 3$  PSNR and execution time measures are given in Figure 4.19. In Figure 4.19, the examination

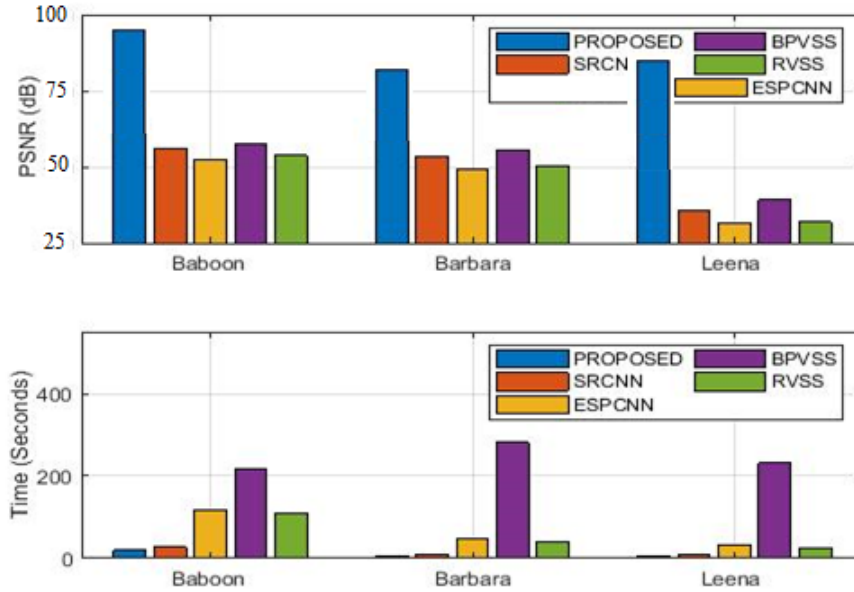


Figure 4.19: Comparison analysis of PSNR and time in Scale  $\times 3$ .

result portrays that our method outperforms the existing techniques. The comparison analysis of scale  $\times 4$  PSNR and execution time measures are given in Figure 4.20. In

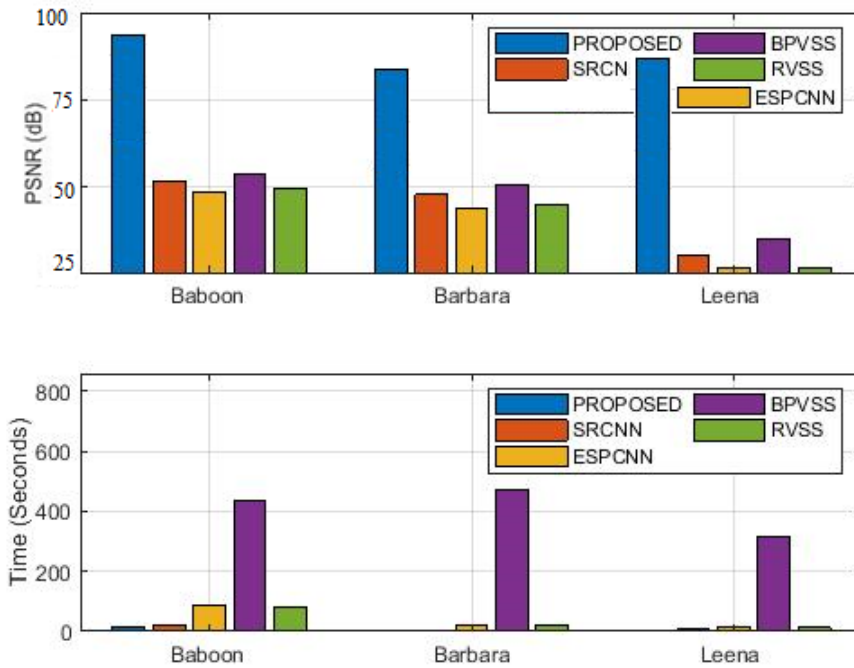


Figure 4.20: Comparison analysis of PSNR and time in Scale  $\times 4$ .

Figure 4.20, the examination result portrays that our method outperforms the existing

techniques.

#### 4.4 SUMMARY

We presented an effective secret image sharing with super-resolution utilizing quantum logic and enthalpy based adaptive deep neural network. The presented technique enhances the contrast of secret image sharing. To accomplish this objective, the presented system is designed with the GPGPU to enhance the resolution of the reconstructed images and to lessen the time intricacy in deep learning. The EDNN is adapted with the enthalpy based normalization to mitigate the over-fitting in layers before the max pooling layer in the deep neural network. The exploratory outcomes demonstrated that our secret image sharing with GPGPU outperforms the existing methodologies without utilizing a GPGPU system in terms of normalized cross correlation, normalized absolute error, peak signal to noise ratio, mean square error, PSNR, execution time and signal to noise ratio.

The CNNs learn to determine hierarchical descriptions of the visual data. This conducive condition of deep learning lets the automatic optimal low-level feature determination of data at different scales. Despite adjusting for minimal squared error, the convolutional network also proved its efficacy in yielding perceptual metrics.





## CHAPTER 5

### **GPGPU VSS BY CONTRAST-ADAPTIVE CNN SUPER-RESOLUTION**

A  $(k, n)$ -threshold Visual Cryptography (VC) scheme encrypts a covert image  $p$  into  $n$  share images in such a way that minimum  $k$  ( $k \leq n$ ) stacked shares expose the secret. Though the encryption and decryption are the two primary processes in any cryptography, their considerable distinction in VC is that encryption requires less computation and decryption needs no or little calculation. Stacking minimum  $k$  number of shares without calculation discloses the image  $p$ . In systems requiring simple logical operations for decryption, OR is preferred if it uses a 1 to represent the black pixel and the system based on light polarization utilizes XOR logic (Yan et al. 2020a). Because of the secret image distribution in the VC uses shares that are innocuous and irrelevant to the secret for the human visual system, another name Visual Secret Sharing (VSS) also refers to the VC.

The research in the area of VC is continually advancing primarily in optimizing three factors. They are the share size, the effectiveness of the disclosed image, and security level. The researchers invented an encryption approach that generates shares larger in proportion to the covert image (Naor and Shamir 1994). This pixel extension amounts to the share-alignment problem due to the contrast degradation (Liu et al. 2014).

The random-grid, probabilistic, and the block based models are the three broad categories of size invariant VSS (Li et al. 2020c; Wu and Yang 2020). Though these

models produce shadows of the original image size, they can not ideally disclose the covert image even using computing resources (Li et al. 2020b). Block-based progressive models are handy for small threshold value of  $k$  in a  $(k, n)$  scheme (Pandey et al. 2020). Strict the security imposed, less the contrast of the reconstructed image in the VC model (Yan et al. 2020a). One way to improve security is to embed the shadows in the meaningful cover image to fake invaders during transmission. Such models have to focus on a better trade-off among embedding volume, contrast, and security (Xiong et al. 2020). Embedding in the spatial domain with a least significant bit change approach can compromise security. Investigators used Discrete Wavelet Transform (DWT) with authentication to meet these objectives in Xiong et al. (2020). On the contrary, a model that minimizes the security requirement proved the better quality of the deciphered image along with a reduction in pixel extension (Liu et al. 2012).

Lossless and Lossy schemes are the two broad categories of VSS based on the nature of the revealed secret utilizing shadow images (Yan et al. 2020c). The lossless techniques result in complete non-distorted recovery, unlike lossy, but consume intensive computation. As a result, lossless approaches do not fit for poor-computational devices. The constraint on the fixed ordering of shares and no General Access Structure (GAS) during image reconstruction are the other shortcomings of these methods (Yan et al. 2020b). Threshold-based strategies, a class of lossy strategies, constrain the number of shadows stacked to reveal the secrecy. But they occupy additional space and bandwidth in memory and communication channel for a requirement of a codebook and the pixel extension problem (Al-Khalid et al. 2017). The pixel widening causes redundant computations in shadow creation. The research on combining advantages and advancements in all of these categories is a notable investigation in Sridhar and Sudha (2020).

Recently, an investigation on  $(n, n)$  access quantum sharing has been growing in the area of VSS (Bassirian et al. 2019). Quantum secret sharing method shares a confidential quantum state among  $n$  participants, allowing only  $k$  ( $k \subseteq n$ ) permitted participants can collaboratively recover that state whereas other subsets can not. In the quantum sharing in Bassirian et al. (2019), researchers investigated a fundamental  $(2, 3)$

---

quantum sharing using qubits and 7-qubit Calderbank-Shor-Steane (CSS) code. Then they extended it to produce general-access  $(n, n)$  scheme.

Halftoning VSS schemes use an image conversion method called halftoning to hide secrets. The converted halftone image simulates continual-tone image using two tones while preserving continual originality (Wu et al. 2020). The human visual structure is a low-pass filter, and hence the average local intensity is more crucial than individual pixel values (Liu et al. 2020). A novel error diffusion with varying threshold (EDVT) method in Srividhya et al. (2019) coalesces two conventional halftoning approaches, namely ordered dither and error diffusion to achieve high-resolution colour halftone images, at the cost of complex computation.

Another approach called the super-resolution (SR) reconstructs a high-contrast image using one or more low contrast images and a priori knowledge. Moustafa et al. (2016) used advanced general-purpose graphics processing unit (GPGPU or only GPU) to compensate for the time complexity overhead associated with the SR for hyperspectral images. Also, the graphic computation using GPUs has attracted the researchers as those devices empowered the light-weight computation devices recently (Moustafa et al. 2016). Utilizing GPU for super-resolution balances the performance concerning computational complexity and image contrast (Jung et al. 2018). SR methods involve demanding redundant computations (Yamanaka et al. 2017), and hence the recent area of research focused on accelerating them (Georgis et al. 2019). The investigation in Qiu et al. (2020) used SR reconstruction for medical image contrast enhancement. The method utilized deep learning cascaded small convolution neural networks to attain a reconstruction quality with speed. To alleviate the problem of more memory and mathematical demand of deep learning-based SR utilizing a single low-resolution image, the investigators Yang et al. (2020) presented a fast and lightweight cascaded convolution layers.

Addressing the cheating problem is one more way of achieving more security that detects cheating by some fraudsters submitting fake shadows in the image reconstruction stage (Liu et al. 2018). The model called Randomized Visual Secret Sharing (RVSS) used a frequency domain transformation of shares to embed information which

made the quality of the reconstructed image with a contrast range 70 – 90% for the noise-like and 70 – 80% for the meaningful shares (Mhala et al. 2017). The researchers compared the effectiveness of the RVSS model with a novel Progressive Visual Secret Sharing (PVSS) called Block based PVSS (BPVSS) model (Hou et al. 2013a). A super-resolution based PVSS model (hereafter, referred to as SR-PVSS) investigated in Mhala and Pais (2019a) utilized a super-resolution method to improve the effectiveness of the reconstructed image. This model achieved a contrast range of 70 – 80% for meaningful and 99% for noise-like shares. For efficiency, visual multiple secret sharing (VMSS) allowed encryption of any number of secrets simultaneously, but with a bottleneck in managing shares. The meaningful VSS strategies overcome shadow management’s problem but with a constraint of a fixed number of shadows (Huang and Juan 2020).

Despite these innovations in the field of VC, there has been little research on efficient and effective VC models using advanced techniques and technologies such as deep learning super-resolution (SR) models and General Purpose Graphics Processing Unit (GPGPU or only GPU). The GPGPUs continue to empower the deep learning models through their throughput-based efficiency (Mittal and Vaishay 2019). The deep learning SR based VC models fueled by GPGPUs positively contribute efficiency, effectiveness, and computing resource utilization.

The schemes discussed above do not address the following issues:

- Computationally intensive VSS schemes do not fit for real-time applications.
- The VSS schemes demand super-resolution approaches to enhance the contrast of the reconstructed image, but at the computation cost.
- Existing SR techniques are standalone models either reduce mathematical demand or use GPU for efficiency.

We presented an efficient and effective VC utilizing contrast-adaptive convolutional neural network (CCNNs / CConvNet) and GPGPU. The performance of the GPGPU-based model is evaluated in two ways. Firstly, by comparing it with the SR-PVSS and

RVSS models. Then by contrasting with its CPU model.

This work stands unique in the following aspects:

- The model exploits the power-efficient and throughput-optimized GPGPU for the computationally intensive data-parallel tasks in the VSS pipeline.
- GPGPU constant memory for storing read-only data for massive thread broadcasting in all the activities contributes to further performance gain.
- The contribution this work is twofold. Firstly we improved the resulting image quality by utilizing a lightweight ConvNet-based super-resolution. The use of the GPGPU reduced the ConvNet training time.
- The speedup achieved saves power enabling this VSS scheme suitable for mobile and real-time security applications, including online banking, resolution variant cryptography, digital watermarking.

The rest of this chapter is structured as follows: In section 5.1, a concise discourse about our methodology is provided. Experimental results are given in Section 5.2 and we summarize our work in Section 5.3.

## 5.1 GPGPU VSS

Figure 5.1 depicts the two stages of the presented model. Firstly, the share generation stage generates information-embedded shadows equal to the number of participants. The steps involved in share generation are as follows: creating halftone image (Section 5.1.1), creating shadows (Section 5.1.2), and transform halftone image to wavelet domain using Discrete Wavelet Transformation (DWT) (Section 5.1.3) to embed information into shares. Then the image reconstruction stage is the inverse of the share generation. This stage involves steps such as extraction of embedded information, reconstruction of the image by logical XOR (Section 5.1.4), and contrast enhancement of the reconstructed image using CCNN (Section 5.1.5).

Initially, we convert the colour image to the halftone image. The halftone image is then encrypted based on the halftone image's pixel value and the random choice of a

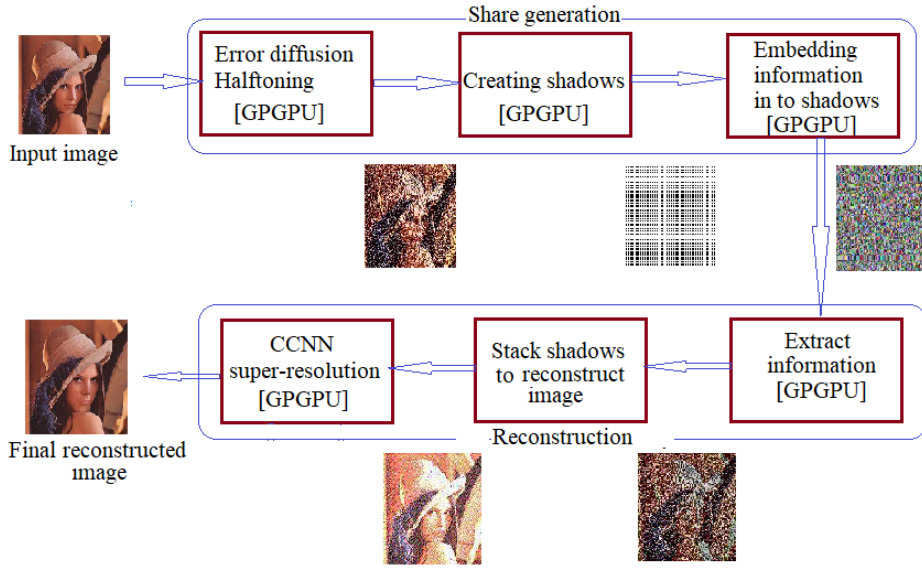


Figure 5.1: Block diagram of the presented methodology.

row of the assigned basis matrix to create shadows. A wavelet transformation of the halftone image produces a detailed component of the image embedded in the shadows. Individual secured shadows do not decipher any input image during their transmission. However, when all shares are in coalition by the authorized users, the secret input gets revealed. The super-resolution with the trained ConvNet improves the contrast of the resultant image.

The assignment of the data-parallel activities by the CPU pertaining to the share generation activities to the GPGPU is given in Algorithm 5.1. This is the host module which invokes the functions to be accomplished from the GPGPU. Inputs to Algorithm 5.1 are the secret image and the basis matrices. However, this module acts as a staging area to copy required data between CPU and GPGPU memory locations. This host module also copies error diffusion parameters  $r[4]$  as well Daubechies wavelet parameters  $h[4]$  and  $g[4]$  to the GPGPU constant memory as these values are only read by all the threads. The *configuration parameters* are used to define the thread organization. The different data-parallel activities used in Algorithm 5.1 are halftoning using GPGPU error diffusion, creating shadows using GPGPU, information embedding using Discrete Wavelet Transformation (DWT), Secret image reconstruction, and contrast-adaptive convolution neural network (CCNN). These steps are explained below.

**Algorithm 5.1: Share\_generation( $I, Q_p$ ) //CPU function.**


---

```

// n- number of participants, W- width, H- height,
I- input image, r[4]- error diffusion paramters,
h[4], g[4] -Daubechies wavelet parameters,  $I_{Tg}$ -
halftone image in GPGPU memory,  $S_{ig}$ -shadow images
in GPGPU memory,  $I_{Wg}[W \times H]$ - wavelet image in GPGPU
memory,  $E_{ig}$ - embedded shadow images in GPGPU
memory,  $I_W$ - wavelet image in CPU memory,  $I_T$ -
output halftone image in CPU memory, and  $E_i$ -
output embedded shadow images in CPU memory.
Input: 1.  $I[W \times H]$ .
          2.  $\{Q_p[2 \times n] | (0 \leq p \leq n)\}$ .
Output:  $I_T[W \times H]$ ,  $\{S_i[W \times H], E_i[W \times H] | (1 \leq i \leq n)\}$ .
1  $r_c[4] = r[4]$ ,  $Q_{pc}[2 \times n] = \{Q_p[2 \times n] | (0 \leq p \leq n)\}$ ,  $h_c[4] = h[4]$ ,  $g_c[4] = g[4]$ .
   // allocate GPGPU memory to store output from GPGPU.
2 Allocate  $\{I_{gs}[W \times H] | s = R, G, B\}$ ,  $\{I_{Tgs}[W \times H] | s = R, G, B\}$ ,
    $S_{ig}[W \times H]$ ,  $I_{Tg}[W \times H]$ ,  $I_{Wg}[W \times H]$ ,  $E_{ig}[W \times H]$  in GPGPU global
   memory.
   // split  $I[W \times H]$  in to RGB components and store in
   GPGPU global memory.
3  $I_{gs}[W \times H] = split\_rgb(I[W \times H])$ 
4 Compute GPGPU kernel configuration parameters and launch
   kernel_halfone( $I_{gs}, I_{Tgs}, configuration\ parameters$ ).
   // merge  $I_{Tgs}[W \times H]$  and store in CPU memory.
5  $I_T = merge\_rgb(I_{Tgs}[W \times H])$ ,  $I_{Tg} = I_{Wg} = I_T$ ,  $N = W \times H$ 
   // Launch the kernel to generate shadows.
6 kernel_shadows( $I_{Tg}, S_{ig}, configuration\ parameters$ ).
   // store  $S_{ig}[W \times H]$  in CPU memory.
7  $\{S_i\} = S_{ig}[W \times H]$ 
8  $N = W \times H$ 
   // Launch the kernel to generate wavelet.
9 while  $N \geq 4$  do
10 | kernel_dwt( $I_{Tg}, I_{Wg}, N, configuration\ parameters$ )
11 |  $I_{Tg} = I_{Wg}$ 
12 |  $N = N/2$ 
13 end
   // store  $I_{Wg}[W \times H]$  in CPU memory.
14  $\{I_W\} = I_{Wg}[W \times H]$ .
15 Generate key $[W/2, H/2]$ .
   // share embedding.
16 kernel_embed( $I_{Wg}, S_{ig}, key, N, E_{ig}, configuration\ parameters$ )
   // store  $E_{ig}$  in CPU memory.
17  $\{E_i\} = E_{ig}$ 
18 Free GPGPU memory.
19 Return.

```

---

### 5.1.1 Halftoning using GPGPU error diffusion

Let  $I_T = i_{T_{pq}}$  be the halftone equivalent of the colour image  $I = i_{pq}$  for  $p = 0, 1, 2, \dots, W$  and  $q = 0, 1, 2, \dots, H$ , where  $W$  is the width and  $H$  is the height of the  $I_T$  and  $I$ . Then equation 5.1 gives the each pixel value  $i_{T_{pq}}$ , where where  $\theta$  represents the binarization threshold value.

$$i_{T_{pq}} = \begin{cases} 0, & \text{if } i_{pq} > \theta \\ 1, & \text{otherwise} \end{cases} \quad (5.1)$$

Equation 5.2 defines the error generated by equation 5.1.

$$d_{pq} = i_{pq} - i_{T_{pq}} \quad (5.2)$$

The error  $d_{pq}$  is diffused to the neighbouring pixels as specified in equation 5.3

$$i'_{p+m \ q+n} = i_{p+m \ q+n} + r_{mn}d_{pq} \quad (5.3)$$

where the pixel  $i_{p+m \ q+n}$  at  $(p + m \ q + n)$  updated to  $i'_{p+m \ q+n}$  by adding  $d_{pq}$  weighted by Floyd-Steinberg coefficients  $r_{mn}$  (Floyd 1976).

Algorithm 5.2 shows the steps involved in GPGPU based data-parallel halftoning. The use of constant memory to store read-only Floyd-Steinberg coefficients  $r_{mn}$  offers its aggressive broadcasting to the massively parallel threads. The efficiency by utilizing constant memory is considerable for the larger image.

### 5.1.2 Creating shadows using GPGPU

The presented model uses  $(n + 1)$  basis matrices to create size-invariant  $n$  shadows where  $n$  is the number of participants. Algorithm 5.3 generates the two sets of basis matrices  $S_1 = \{Q_0\}$  and  $S_2 = \{Q_p | (1 \leq p \leq n)\}$  to share the white and the black pixels of the confidential image in the  $(n + 1)$  shadows. Table 5.1 is an example of 5 basis matrices created by Algorithm 5.3 for  $n = 4$  shadows each corresponding to a participant.

Algorithm 5.4 shows the tasks performed by the GPGPU in generating shadows using basis matrices. The basis matrices stored in the GPGPU-constant memory offer aggressive caching of read-only data contained in them for the parallel threads. Another advantage of utilizing constant memory is that it reduces GPGPU global memory traffic and hence improves bandwidth. As the number participants increases with increase in



**Algorithm 5.2: *kernel\_halfone*( $I_{gs}, I_{Tgs}$ ) //GPGPU kernel.**


---

```

//  $r_c[4]$  available in GPGPU constant memory for
// aggressive caching to read-only operations by
// threads
// assume  $e[W \times H] = 0$ .
Input: 1.  $\{I_{gs}[W \times H] \mid s = R, G, B\}$ .
          2.  $\{I_{Tgs}[W \times H] \mid s = R, G, B\}$ .
Output: Updated  $\{I_{Tgs}[W \times H] \mid s = R, G, B\}$  in GPGPU global memory.
// set global memory index.
1  $t = blockIdx.x \times blockDim.x + threadIdx.x$ 
// only  $H$  threads active.
2 if ( $t < H$ ) then
    // each thread processes data in the respective
    // row successively and performs error-diffusion
    // halftoning.
3   for  $j = 0 : (W - 1)$  do
4     if ( $I_{gs}[t \times W + j] + e[t \times W + j] < 128$ ) then
5        $I_{Tgs}[t \times W + j] = 0$  else
6          $I_{Tgs}[t \times W + j] = 255$ 
7       end
8     end
9      $d = I_{gs}[t \times W + j] + e[t \times W + j] - I_{Tgs}[t \times W + j]$ 
10    if ( $j + 1 < W$ ) then
11       $e[t \times W + j + 1] = e[t \times W + j + 1] + d \times r[0]$ 
12    end
13    if ( $t \times W + 1 < H$ ) then
14       $e[t \times W + j + 1] = e[t \times W + j + 1] + d \times r[1]$ 
15    end
16    if ( $t \times W + 1 < H$  and  $j - 1 \geq 0$ ) then
17       $e[t \times W + j] = e[t \times W + j] + d \times r[2]$ 
18    end
19    if ( $t \times W + 1 < H$  and  $j + 1 < W$ ) then
20       $e[t \times W + j + 2] = e[t \times W + j + 2] + d \times r[3]$ 
21    end
22  end
23 end

```

---

---

**Algorithm 5.3:** Generate basis matrices. //CPU function

---

```

// n- number of participants,  $\{Q_p[2 \times n] | (0 \leq p \leq n) \}$ -
// set of basis matrices.
Input:  $n$ .
Output:  $\{Q_p[2 \times n] | (0 \leq p \leq n) \}$ .
// Set  $Q_0$ 
1 for  $j = 0 : (n - 1)$  do
2    $Q_0[0, j] = 0$ 
3    $Q_0[1, j] = \overline{Q_0[0, j]}$ 
4 end
// Set  $Q_p$  ( $1 \leq p \leq n$ )
5 for  $p = 1:n$  do
6   for  $j = 0 : (n - 1)$  do
7     if  $j = p$  then
8        $Q_p[0, j] = 1$  else
9        $Q_p[0, j] = 0$ 
10      end
11     end
12      $Q_p[1, j] = \overline{Q_p[0, j]}$ 
13   end
14 end

```

---

Table 5.1: Generated basis matrices

$$\begin{aligned}
 Q_0 &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} & Q_1 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \\
 Q_2 &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} & Q_3 &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \\
 Q_4 &= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}
 \end{aligned}$$


---

size of the image, this technique will lead to significant improvement in speedup.

### 5.1.3 Information embedding using Discrete Wavelet Transform (DWT)

DWT dyadically decomposes a discrete spatial signal  $p$ ,  $\{p(i, j) \in P(w, h) | (0 \leq i \leq w - 1), (0 \leq j \leq h - 1)\}$  of an image  $P$  where  $w$  is the width, and  $h$  is the height of  $P$  to four discrete frequency bands  $f_\Phi$  and  $\{f_\varphi^s | s = \{H, V, D\}\}$  as shown in the equation

**Algorithm 5.4:** *kernel.shadows*( $I_{Tg}$ ,  $S_{ig}$ ) //GPGPU kernel.

---

```

//  $Q_{pc}$  available in GPGPU constant memory for
// aggressive caching to read-only operations by
// threads.
//  $n$ - number of participants.
Input: 1.  $I_{Tg}[W \times H]$ .
         2.  $S_{ig}[W \times H]$ .
Output: Updated  $S_{ig}[W \times H]$  in GPGPU global memory.
// set global memory index.
1  $t = blockIdx.x \times blockDim.x + threadIdx.x$ 
// set index to basis matrices.
2 if ( $threadIdx.x < n$ ) then
3 |    $t_p = threadIdx.x$ 
4 end
// randomly generate  $r$ .
5  $r = 0$  or 1
// set shadow pixles in parallel for ( $1 \leq (i \& p) \leq n$ ).
6 if  $r = 0$  then
7 |   if  $I_{Tg}[t] = White$  then
8 | |    $S_{ig}[t] = Q_{0c}[t_p]$  else
9 | | |    $S_{ig}[t] = Q_{pc}[t_p]$ 
10 | |   end
11 |   end
12 |   else
13 | |   if  $I_{Tg}[t] = White$  then
14 | | |    $S_{ig}[t] = Q_{0c}[n + t_p]$  else
15 | | | |    $S_{ig}[t] = Q_{pc}[n + t_p]$ 
16 | | |   end
17 | |   end
18 |   end
19 end

```

---

5.4 and equation 5.5. In equation 5.4 and equation 5.5,  $d$  is the present input scale,  $d_+$  is the scale in the next iteration,  $k$  is the index along  $i$ , and  $l$  is the index along  $j$  with a down sampling scale 2 ( $\downarrow 2$ ).

$$f_{\Phi(d_+,k,l)} = \frac{1}{\sqrt{wh}} \sum_{i=0}^w \sum_{j=0}^h p(i, j) \Phi_{(d,k,l)}(i, j) \quad (5.4)$$

In equation 5.4,  $f_{\Phi}$  is the LL component and  $\Phi$  is the filter. In equation 5.5,  $f_{\varphi}^s$  contains HL, LH, and HH components corresponding to  $s = \{H, V, D\}$  where  $H$  is horizontal,  $V$  is the vertical, and  $D$  is the diagonal component of the signal with the corresponding filters  $\varphi^s$ .

$$f_{\varphi(d_+,k,l)}^s = \frac{1}{\sqrt{wh}} \sum_{i=0}^w \sum_{j=0}^h p(i, j) \varphi_{(d,k,l)}^s(i, j) \quad (5.5)$$

As the low pass filtered component  $LL$  contains maximum signal information, embedding that component of the halftone image in the shares retains the coarser signal detail. Before doing so, a reversible secret key is added to the shares. The embedding of information in the our model utilizes DWT centered reversible information concealing technique. Consequently, the outcome of the inverse DWT is indistinguishable from the input of the DWT (Bao and Zhang 2020).

Algorithm 5.5 shows the tasks performed by the GPGPU for DWT transformation. The image transformation into the wavelet domain uses constant-memory to store Daubechies wavelet coefficients for memory optimization and performance gain. Algorithm 5.6 shows the embedding of  $LL$  halftone image information into the shares along with a reversible secret key. For the analysis, we combine the execution times by the GPGPU to perform wavelet transformation and embedding.

#### 5.1.4 Secret image reconstruction

Finally, the system extracts the embedded information, and a logical XOR of the shares reconstructs the low-resolution secret image. The super-resolution model based on a Contrast-adaptive Convolution Neural Network (CCNN) produces an effective high-resolution secret image.

**Algorithm 5.5:** *kernel\_dwt*( $I_{Tg}$ ,  $I_{Wg}$ ,  $N$ ) //GPGPU kernel.

---

```

//  $h_c[4]$  and  $g_c[4]$  available in GPGPU constant memory for
// aggressive caching to read-only operations by
// threads.
// assume  $blockDim.x = W$ .
Input: 1.  $I_{Tg}[W \times H]$ .
         2.  $I_{Wg}[W \times H]$ .
         3.  $N$ .
Output: Updated  $I_{Wg}(W \times H)$  in GPGPU global memory.
// set global memory index.
1  $t = blockDim.x \times blockDim.x + threadIdx.x$ 
2  $h = N/2$ 
3 if ( $2 \times t < (N - 3)$ ) then
4    $I_{Wg}[t] = I_{Tg}[2 \times t] \times h_c[0] + I_{Tg}[2 \times t + 1] \times h_c[1] + I_{Tg}[2 \times t + 2] \times$ 
    $h_c[2] + I_{Tg}[2 \times t + 3] \times h_c[3]$ 
5    $I_{Wg}[t + h] = I_{Tg}[2 \times t] \times g_c[0] + I_{Tg}[2 \times t + 1] \times g_c[1] + I_{Tg}[2 \times t + 2] \times$ 
    $g_c[2] + I_{Tg}[2 \times t + 3] \times g_c[3]$ 
6 end
7 if ( $2 \times t = (N - 2)$ ) then
8    $I_{Wg}[t] = I_{Tg}[t - 2] \times h_c[0] + I_{Tg}[t - 1] \times h_c[1] + I_{Tg}[0] \times h_c[2] + I_{Tg}[1] \times h_c[3]$ 
9    $I_{Wg}[t + h] = I_{Tg}[t - 2] \times g_c[0] + I_{Tg}[t - 1] \times g_c[1] + I_{Tg}[0] \times g_c[2] + I_{Tg}[1] \times g_c[3]$ 
10 end

```

---

**Algorithm 5.6:** *kernel\_embed*( $I_{Wg}$ ,  $S_{ig}$ ,  $key$ ,  $N$ ,  $E_{ig}$ ) //GPGPU kernel.

---

```

Input: 1.  $I_{Wg}[W, H]$ .
         2.  $\{S_{ig}[W \times H] \mid (1 \leq i \leq n)\}$ .
         3.  $key[W/2, H/2]$ .
         4.  $N$ .
Output: Updated  $\{E_{ig}[W \times H] \mid (1 \leq i \leq n)\}$  in GPGPU global memory.
// set global memory index.
1  $c = blockDim.x \times blockDim.x + threadIdx.x$ 
2  $r = blockDim.y \times blockDim.y + threadIdx.y$ 
// only  $N/4$  threads active.
3 if ( $c < N/2 \ \&\& \ r < N/2$ ) then
   // perform embedding.
4    $E_{ig}[c, r] = S_{ig}[c, r] + key[c, r] \oplus I_{Wg}[c, r]$ 
5 end

```

---

### 5.1.5 Contrast-adaptive Convolution Neural Network (CCNN)

Figure 5.2 shows the layered architecture of the contrast-adaptive ConvNet (CCNN) used in the presented system. The presented CCNN comprises of convolution, contrast-

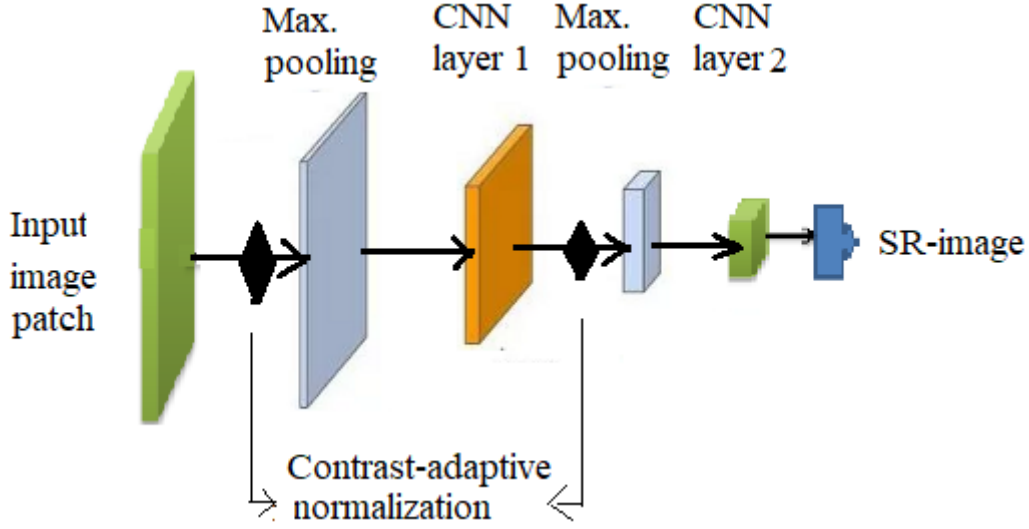


Figure 5.2: Architecture of the presented CCNN.

adaptive normalization, max-pool and a fully connected layer. The purpose of CCNN is to provide the GPGPU based super-resolution. The use of GPGPU improves the efficiency of deep learning in CNN.

The convolutional layers consist of successive layers of a rectangular grid of the input image processed with their weighted filters. Let  $y^{\ell-1}$  be a convolution layer processed with a filter bank  $\hat{f}$  of size  $(f \times f)$  producing an output of size  $(P - f + 1) \times (P - f + 1)$  where  $P \times P$  is the size of the succeeding neuron layer  $x^\ell$ . Equation 5.6 and Equation 5.7 signifies the linear and the non-linear input to  $x^\ell$ .

$$x_{pq}^\ell = \sum_{i=0}^{f-1} \sum_{j=0}^{f-1} \hat{f}_{ij} y_{(p+i)(q+j)}^{\ell-1} \quad (5.6)$$

$$y_{pq}^\ell = \sigma(x_{pq}^\ell) \quad (5.7)$$

For the error function  $E$ , the error for the previous layer is  $\left(\frac{\partial E}{\partial y_{pq}^\ell}\right)$ . Equation 5.8 signifies the gradient part. Delta is computed using equation 5.9.

$$\frac{\partial E}{\partial \hat{f}_{ij}} = \sum_{p=0}^{N-f} \sum_{q=0}^{N-f} \frac{\partial E}{\partial x_{pq}^\ell} \frac{\partial x_{pq}^\ell}{\partial \hat{f}_{ij}} = \sum_{p=0}^{N-f} \sum_{q=0}^{N-f} \frac{\partial E}{\partial x_{pq}^\ell} y_{(p+i)(q+j)}^{\ell-1} \quad (5.8)$$

$$\frac{\partial E}{\partial x_{pq}^\ell} = \frac{\partial E}{\partial y_{pq}^\ell} \frac{\partial y_{pq}^\ell}{\partial x_{pq}^\ell} = \frac{\partial E}{\partial y_{pq}^\ell} \frac{\partial}{\partial x_{pq}^\ell} (\sigma(x_{pq}^\ell)) = \frac{\partial E}{\partial y_{pq}^\ell} \sigma'(x_{pq}^\ell) \quad (5.9)$$

The weights of the previous layer obtained through back propagation are shown in equation 5.10.

$$\frac{\partial E}{\partial y_{pq}^{\ell-1}} \left\{ \begin{array}{l} = \sum_{i=0}^{f-1} \sum_{j=0}^{f-1} \frac{\partial E}{\partial x_{(p-i)(q-j)}^\ell} \frac{\partial x_{(p-i)(q-j)}^\ell}{\partial y_{pq}^{\ell-1}} \\ = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \frac{\partial E}{\partial x_{(i-a)(q-b)}^\ell} \widehat{f}_{ab} \end{array} \right. \quad (5.10)$$

In the CCNN model, the contrast-adaptive normalization between successive layers in CCNN mitigates the over-fittings errors. Equation 5.15 shows the contrast-adaptive normalization esteem computation where  $Y' = (y_1, y_2, , y_3, \dots, y_l)$  with  $l$  number of layers and  $W'$  is the number of resultant windows. Then the  $Z$ -score normalization in Equation 5.12 deals with any outliers where  $\mu$  and  $\sigma$  representing mean and the standard deviation.

$$\widehat{C}_x = W' + Y' \quad (5.11)$$

$$Z_{score} = \frac{\widehat{C}_x - \mu}{\sigma} \quad (5.12)$$

The max-pooling layer subsamples using a maximum in the block to reduce computational overhead. The errors backpropagated to this layer are sparse. The softmax function (Equation 5.13) to each input element  $x_i$  at the fully connected layer produces the best super-resolution image equivalent of the reconstructed image.

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \quad (5.13)$$

By deploying images among the CPU and the GPGPU during super-resolution, the presented model proved efficient.

## 5.2 EXPERIMENTAL RESULTS

The presented GPGPU model is designed and analyzed with the CUDA, OpenCV, and MATLAB running on the PARAM Shavak Super Computer with a CPU: Intel® Xeon®-E5-2670 with two cores each containing 24 cores, 8TB RAM, 2.3 GHz., and single NVIDIA® Tesla K40 GPU: having 2880 cores, 12 GB GDDR5, 745 MHz. Figure 5.3 shows the sample Baboon (Figure 5.3 (a) - color), Barbara (Figure 5.3 (b) - grayscale) and Lena (Figure 5.3 (c) - color) input images, each of size  $256 \times 256$ . Figure 5.4



Figure 5.3: Test input images (a) Baboon-color image, (b) Barbara-grayscale image, and (c) Lena-color image.

shows the halftone images. Figure 5.5 (a), Figure 5.5 (b), and Figure 5.5 (c) shows the

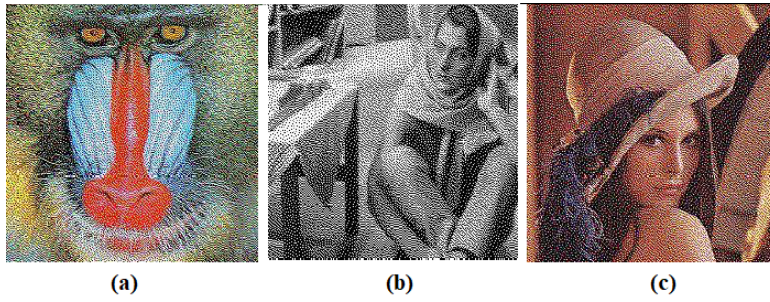


Figure 5.4: Halftone images (a) Baboon-color image, (b) Barbara-grayscale image, and (c) Lena-color image.

generated corresponding shadows. Figure 5.6 denotes the embedded shadows. Figure 5.7 denotes the extracted low-resolution shadows. Figure 5.8 shows the reconstructed low-resolution images. Finally, Figure 5.9 shows the resultant high-resolution images. The presented architecture contains the following three steps.

1. Generate the image patches and apply Contrast-Adaptive Normalisation (CAN):

Let  $Max$  and  $Min$  be the maximum and the minimum intensity across all the channels. For the given input training image  $I$ , the entropy  $E$  is computed as in equation 5.14 that is added as shown in equation 5.15.

$$E = Max + (Min \times 1.5) \quad (5.14)$$

$$I = I + E \quad (5.15)$$

Using the presented CAN for training images helps the model to learn and generate a better contrast-enhanced image for a given input image.



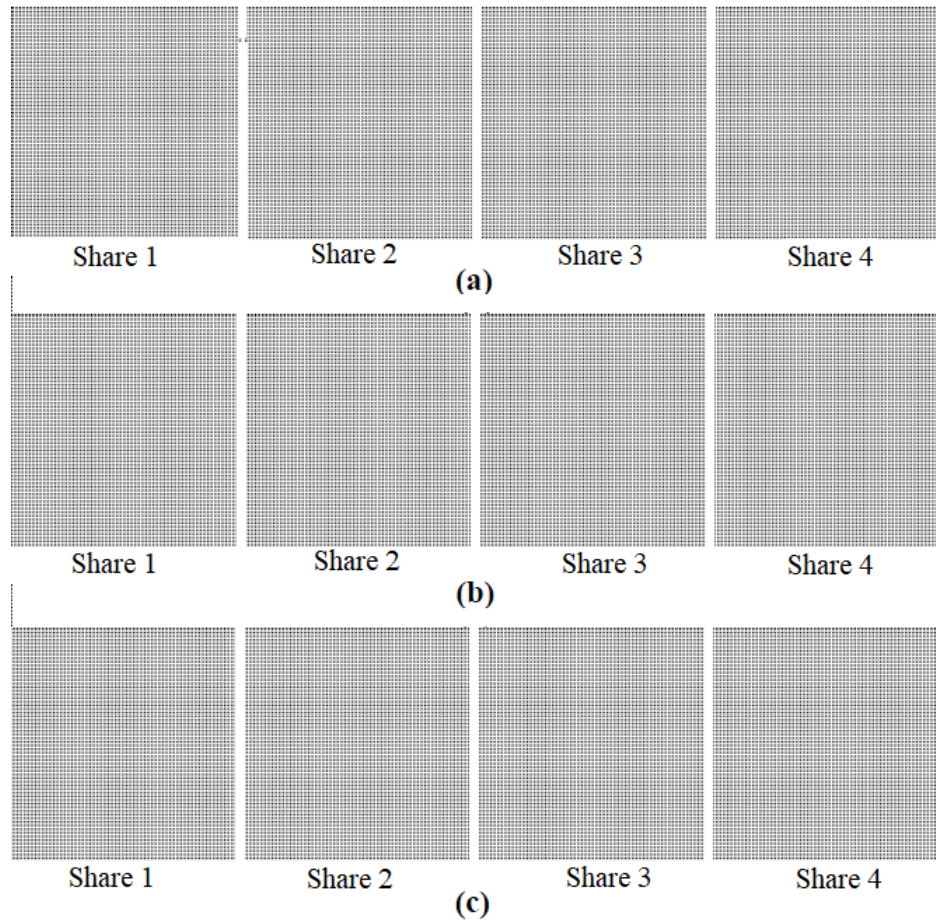


Figure 5.5: Shadow (Share) images of halftone images (a) Baboon-color image, (b) Barbara-grayscale image, and (c) Lena-color image.

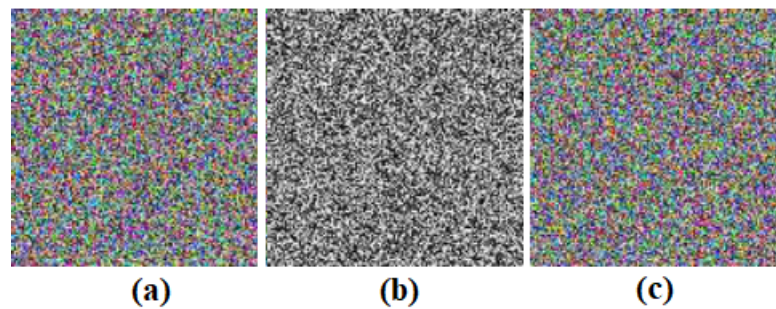


Figure 5.6: Embedded images (a) Baboon-color image, (b) Barbara-grayscale image, and (c) Lena-color image.

2. Learn the contrast enhancing features of the image using deep CNN model: We followed the generalized deep CNN model with two CNN blocks, each followed by a max-pooling layer with Sigmoid activation function. We split the images into  $4 \times 4$  patches after the reshaping of CAN filtered images. It results in  $4 \times 4 \times 12288$

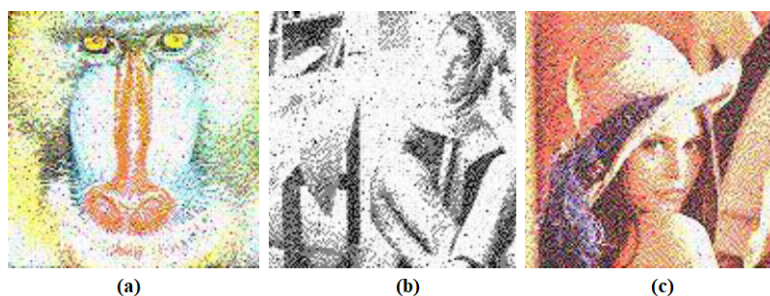


Figure 5.7: Extracted images (a) Baboon-color image, (b) Barbara-grayscale image, and (c) Lena-color image.

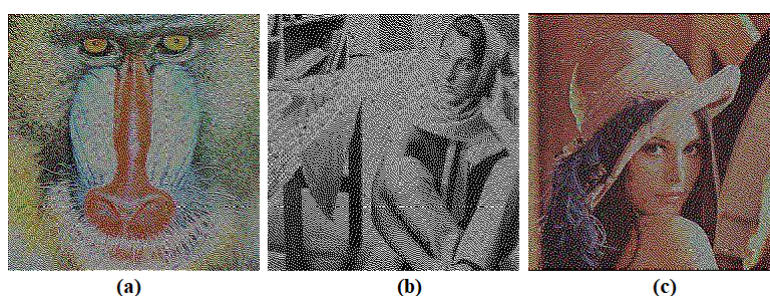


Figure 5.8: Reconstructed images after extracting data (a) Baboon-color image, (b) Barbara-grayscale image, and (c) Lena-color image.

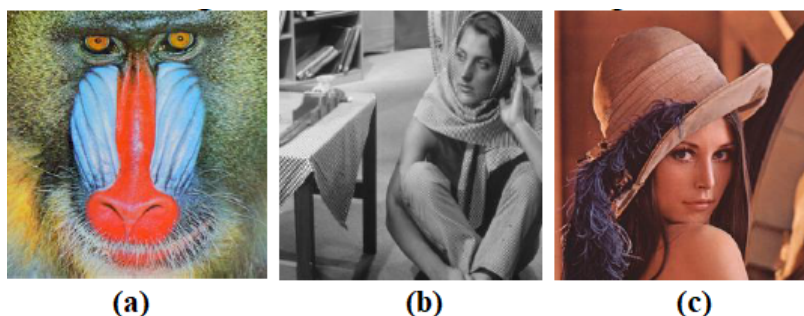


Figure 5.9: Final reconstructed images after super resolution (a) Baboon-color image, (b) Barbara-grayscale image, and (c) Lena-color image.

image patches. Then we apply CNN followed by the max-pooling layer. We used  $5 \times 5$  convolution for each of three image patches resulting in one patch with three output channels, and then down-sampling is carried out using max-pooling with  $2 \times 2$  kernel size. Binary cross-entropy loss is used to find the loss between predicted and actual contrast-enhanced images. We have trained the the model for ten epochs. Table 5.2 shows the CNN summary utilized.

3. Generate the contrast enhanced image: Finally, we concatenate all end layer fea-

Table 5.2: *CNN* summary utilized by the proposed system.

Layer (type)	Output shape	# Parameters
CNN layer 1 (Conv2D)	(None, 5, 5, 3) * 4096	319488
CNN layer 2 (Conv2D)	(None, 5, 5, 3) * 4096	319488
Total parameters: 638976		
Trainable parameters: 638976		
Non-trainable parameters: 0		

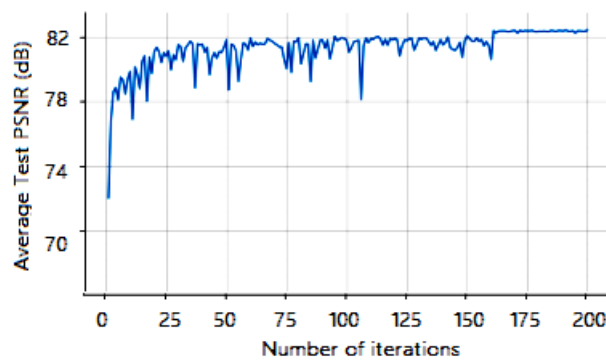


Figure 5.10: PSNR convergence curve of CCNN for noise-like shares.

ture maps into a vector for generating the output  $256 \times 256$  contrast-enhanced image.

The model is trained using 200 images and validated on 100 images available in the image repository BSDS500 of 30 human subjects (Martin et al. 2001). A total of 5,37,600 images used for training. Table 5.2 shows that we used two convolution layers, which uses a total 638976 learning parameters to train the system. We evaluated the test performance using the image database available in SIP (2020). Figure 5.10 shows the convergence curve of the CCNN model for the noise-like shares. The receiver operating characteristic (ROC) curve in Figure 5.11 plots the True Positive Rate (TPR) versus FPR False Positive Rate (FPR).

We compare the model’s qualitative estimates with the SR-PVSS and RVSS models in the following subsections using four-colour and four-grayscale test images shown in Figure 5.12 and Figure 5.13. In two ways, we objectively evaluate the performance of the GPGPU-based system, including the parameters in Athar and Wang (2019). Firstly,



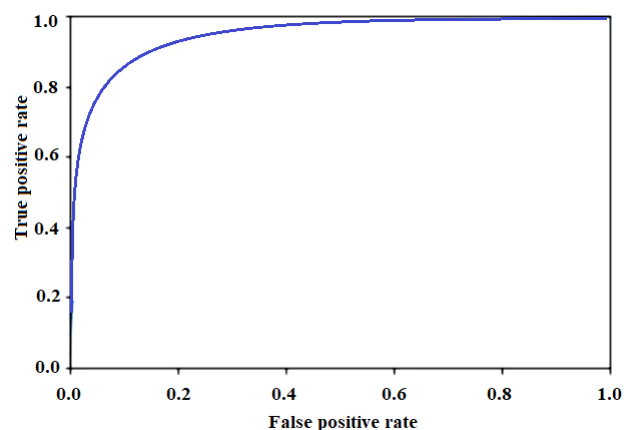


Figure 5.11: The receiver operating characteristic (ROC) curve.

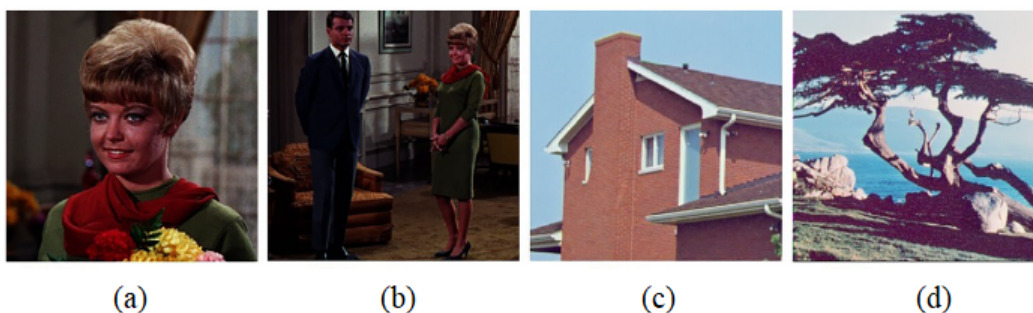


Figure 5.12: Test colour images (a) Female ( $256 \times 256$ ) (b) Couple ( $256 \times 256$ ) (c) House ( $256 \times 256$ ), and (d) Tree ( $256 \times 256$ ).

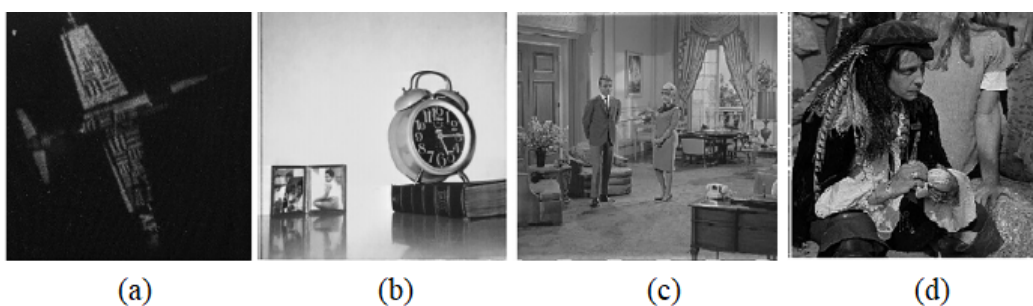


Figure 5.13: Test grayscale images (a) Airplane ( $256 \times 256$ ) (b) Clock ( $256 \times 256$ ) (c) Couple ( $512 \times 512$ ), and (d) Man ( $1024 \times 1024$ ).

we examine its effectiveness with the SR-PVSS and RVSS reference models using specific statistical measures in sections 5.2.1, 5.2.2, 5.2.3, and 5.2.4. After that, we compare the efficiency of the GPGPU model with its CPU model in section 5.2.6.

### 5.2.1 Mean Squared Error ( $MSE$ )

We have already discussed  $MSE$  using equation 4.18 in chapter 4. Lower the  $MSE$ ,

Table 5.3: *MSE* comparison.

Test images	SR-PVSS		RVSS		Presented scheme	
	Noise-like	Meaningful	Noise-like	Meaningful	Noise-like	Meaningful
Fig. 5.12 (a)	0.0018	0.014	0.031	0.032	0.00041	0.00914
Fig. 5.12 (b)	0.0020	0.007	0.018	0.016	0.00109	0.00325
Fig. 5.12 (c)	0.0017	0.049	0.054	0.093	0.00089	0.02927
Fig. 5.12 (d)	0.0031	0.060	0.050	0.107	0.00091	0.03211
Fig. 5.13 (a)	0.0050	0.061	0.048	0.071	0.00051	0.02952
Fig. 5.13 (b)	0.0050	0.024	0.051	0.026	0.00039	0.01813
Fig. 5.13 (c)	0.0030	0.120	0.023	0.110	0.00032	0.07529
Fig. 5.13 (d)	0.0030	0.034	0.017	0.035	0.00036	0.01942

the better is the effectiveness meaning that the reference and the target images are similar. From Table 5.3, it is seen that the presented plan gives the values much close to zero for the reconstructed secret image. We contrast the GPGPU model with the SR-PVSS and RVSS models concerning Mean Square Error (*MSE*) in Table 5.3. From Table 5.3, it is evident that the GPGPU model has a range of *MSE* estimates (0.00032 – 0.00109) and (0.00325 – 0.03211) for the noise-like and meaningful shares. These ranges are significantly lesser than the corresponding values (0.0017 – 0.0050) and (0.007 – 0.120) in SR-PVSS as well as (0.017 – 0.054) and (0.016 – 0.11) in RVSS.

### 5.2.2 Peak Signal to Noise Ratio (*PSNR*)

*PSNR* is explained in chapter 4 using equation 4.17. A positively-oriented *PSNR* score offers a more effective visual quality of the image. Table 5.4 shows the performance analysis of the GPGPU model concerning *PSNR* over the SR-PVSS and RVSS models. From Table 5.4, it is observed that the presented plan gives larger values of *PSNR* relative to SR-PVSS model. Table 5.4 contains the *PSNR* range (77.256 – 83.079) and (59.363 – 73.011) for the noise-like and meaningful shares in the presented model. These ranges are considerably larger than the corresponding values, (70.817 – 75.460) and (58.623 – 69.627) in SR-PVSS as well as (60.786 – 65.692) and (57.717 – 65.958) in RVSS..

### 5.2.3 Normalized Cross Correlation (*NCC*)

We have explained *NCC* using equation 4.15 in chapter 4. Table 5.5 gives the comparison of *NCC* values of the GPGPU model over the SR-PVSS and RVSS models.

Table 5.4: PSNR comparison.

Test images	SR-PVSS		RVSS		Presented scheme	
	Noise-like	Meaningful	Noise-like	Meaningful	Noise-like	Meaningful
Fig. 5.12 (a)	75.460	66.731	63.185	63.020	82.002	68.521
Fig. 5.12 (b)	73.731	69.627	65.692	65.958	77.256	73.011
Fig. 5.12 (c)	74.945	61.213	60.786	58.450	78.630	63.466
Fig. 5.12 (d)	72.869	60.366	61.15	57.828	78.540	63.064
Fig. 5.13 (a)	70.915	58.623	61.339	59.623	81.055	63.429
Fig. 5.13 (b)	70.817	65.072	61.088	64.062	82.220	65.546
Fig. 5.13 (c)	72.801	58.890	64.44	57.717	83.079	59.363
Fig. 5.13 (d)	73.033	64.820	65.755	59.694	82.567	65.248

In Table 5.5, the GPGPU model has a maximum 99.7% correlation for the noise-like shares. The  $NCC$  values for the meaningful shares of the GPGPU model lie in the range (0.784 – 0.871) compared to the corresponding ranges (0.582 – 0.773) of SR-PVSS and (0.526 – 0.807) of RVSS models.

Table 5.5: NCC comparison.

Test images	SR-PVSS		RVSS		Presented scheme	
	Noise-like	Meaningful	Noise-like	Meaningful	Noise-like	Meaningful
Fig. 5.12 (a)	0.978	0.754	0.777	0.718	0.987	0.813
Fig. 5.12 (b)	0.949	0.773	0.703	0.701	0.973	0.798
Fig. 5.12 (c)	0.995	0.582	0.798	0.641	0.994	0.871
Fig. 5.12 (d)	0.993	0.682	0.853	0.671	0.995	0.897
Fig. 5.13 (a)	0.993	0.589	0.640	0.526	0.991	0.784
Fig. 5.13 (b)	0.999	0.587	0.815	0.574	0.993	0.829
Fig. 5.13 (c)	0.995	0.695	0.792	0.730	0.989	0.836
Fig. 5.13 (d)	0.997	0.598	0.906	0.807	0.997	0.801

#### 5.2.4 Normalized Absolute Error (NAE)

We have discussed  $NAE$  in chapter 4 using equation 4.16. Table 5.6 lists the  $NAE$  values of the GPGPU model, along with the SR-PVSS and RVSS models. The  $NAE$  values of the GPGPU model are significantly lesser compared to the SR-PVSS model for both noise-like and meaningful shares. For the noise-like shadows,  $NAE$  range is (0.051 – 0.396) in the SR-PVSS model and (0.2445 – 0.650) in the RVSS model whereas it is (0.043 – 0.097) in the GPGPU model.  $NAE$  range is (0.224-0.392) in the SR-PVSS model and (0.155 – 0.494) in the RVSS model whereas it is (0.104 – 0.211) in the GPGPU model for the meaningful shares.

Table 5.6: *NAE* comparison.

Test images	SR-PVSS		RVSS		Presented scheme	
	Noise-like	Meaningful	Noise-like	Meaningful	Noise-like	Meaningful
Fig. 5.12 (a)	0.128	0.263	0.650	0.616	0.092	0.104
Fig. 5.12 (b)	0.234	0.341	0.245	0.263	0.097	0.120
Fig. 5.12 (c)	0.051	0.224	0.372	0.431	0.062	0.203
Fig. 5.12 (d)	0.078	0.286	0.364	0.494	0.043	0.198
Fig. 5.13 (a)	0.396	0.258	0.252	0.399	0.078	0.176
Fig. 5.13 (b)	0.068	0.359	0.259	0.155	0.051	0.154
Fig. 5.13 (c)	0.082	0.289	0.255	0.124	0.065	0.211
Fig. 5.13 (d)	0.112	0.392	0.262	0.380	0.094	0.201

### 5.2.5 Structural Similarity Index (*SSIM*)

The Structural Similarity Index ( $-1 \leq SSIM \leq 1$ ) is a spatial domain metric that compares luminance, contrast, and structure of two images of a single capture (Athar and Wang 2019). The  $SSIM = 1$  for the two identical images and approaches minimum for the distinct images. Equation (5.16) gives the  $SSIM$  for the comparison of a

Table 5.7: *SSIM* comparison.

Test images	SR-PVSS		RVSS		Presented scheme	
	Noise-like	Meaningful	Noise-like	Meaningful	Noise-like	Meaningful
Fig. 5.12 (a)	0.800	0.309	0.625	0.221	0.974	0.728
Fig. 5.12 (b)	0.713	0.496	0.710	0.410	0.893	0.804
Fig. 5.12 (c)	0.668	0.087	0.337	0.052	0.998	0.716
Fig. 5.12 (d)	0.777	0.146	0.404	0.088	0.965	0.847
Fig. 5.13 (a)	0.826	0.662	0.826	0.654	0.899	0.793
Fig. 5.13 (b)	0.464	0.093	0.245	0.094	0.901	0.834
Fig. 5.13 (c)	0.763	0.161	0.448	0.166	0.890	0.876
Fig. 5.13 (d)	0.946	0.536	0.722	0.528	0.997	0.901

distorted image,  $I'$  with a reference plain image,  $I$ .

$$SSIM(I, I') = [l(I, I')]^\alpha \cdot [c(I, I')]^\beta \cdot [s(I, I')]^\gamma \quad (5.16)$$

Where,

$$l(I, I') = \frac{2\mu_I\mu_{I'} + C1}{\mu_I^2 + \mu_{I'}^2 + C2}$$

$$c(I, I') = \frac{2\sigma_I\sigma_{I'} + C2}{\sigma_I^2 + \sigma_{I'}^2 + C2}$$

$$s(I, I') = \frac{\sigma_{II'} + C3}{\sigma_I\sigma_{I'} + C3}$$

Equation (5.17) is the simplified form of equation (5.16), assuming  $\alpha = \beta = \gamma = 1$  and  $C3 = C2/2$ .

$$SSIM(I, I') = \frac{(2\mu_I\mu_{I'} + C1)(2\sigma_{II'} + C2)}{(\mu_I^2 + \mu_{I'}^2 + C1)(\sigma_I^2 + \sigma_{I'}^2 + C2)} \quad (5.17)$$

Table 5.8: Comparison of execution times (in seconds) and speedups.

Sl.No.	Share generation activities	Execution time in seconds		Speedup
		CPU model	Presented GPGPU model	
1.	Halftoning (section 5.1.1)	0.893001	0.002984	299
2.	Creating shadows (section 5.1.2)	0.327010	0.003811	86
3.	DWT and embedding (section 5.1.3)	0.011613	0.000818	15
4.	CCNN (section 5.1.5)	0.013871	0.007561	2

where  $\sigma_I, \sigma_{I'}$  are the local standard deviations of  $I$  and  $I'$ ,  $\mu_I, \mu_{I'}$  are the means of  $I$  and  $I'$ , and  $\sigma_{II'}$  is the cross-covariance of  $I$  and  $I'$ .

Table 5.7 lists the *SSIM* values of the two compared images  $I$  and  $I'$  for the GPGPU model against the SR-PVSS and RVSS models. The presented model achieves considerably higher *SSIM* of (89% – 99.8%) for the noise-like shares in contrast with the (46.4% – 94.6%) of the SR-PVSS model. Similarly, for the meaningful shadows, it is (71.6% – 90%) when compared to the range (8.7% – 66.2%) of the SR-PVSS model.

### 5.2.6 Speedup

The GPGPU contains a large number of threads executed in parallel. The most usually utilized GPGPU architecture is Compute Unified Device Architecture (CUDA) from NVIDIA. The presented CCNN model fits better with  $16 \times 16$  blocks on the GPGPU. Table 5.8 specifies the performance assessment of GPGPU relative to its sequential model. It is evident from Table 5.8 that the execution time of the GPGPU technique takes much less time resulting in a speedup of  $400\times$  during share generation for the four shadows. The CUDA is a scalable architecture model. The increase in the speedup is significant as the number of shadows increase. But rationale here is that the speedup is the ratio of execution times of the sequential model over the GPGPU model. So, the overall speedup of the model is around  $800\times$  for four shares when compared to the sequential method.

## 5.3 SUMMARY

We presented an efficient secret sharing model with a contrast-adaptive convolution neural network for super-resolution. The GPGPU utilization in the presented model of-



fers efficiency, and the GPGPU based deep-learning augments the effectiveness of the reconstructed image. The extensive use of constant-memory further supplements efficiency. The contrast-adaptive normalization mitigates the overfitting problem in the layers. We use two sequential models for objective quality assessment of the reconstructed image. The speedup of the GPGPU model is assessed with its equivalent sequential model. The experimental results proved that the GPGPU model outperforms the sequential reference models concerning image quality with a time efficiency of  $800\times$  over its sequential model.

In the presented model, we applied a convolutional neural network to super-resolution for the image quality enhancement after its recovery in a VSS scheme. We achieved good performance but with a long training penalty involving the repetitive back-propagation learning algorithm over the training data. The use of GPGPU drastically reduced the training time with the superior quality of the image intact.

In the presented model, we exploited the data-parallel tasks in all stages of the VSS pipeline. We applied a convolutional neural network to super-resolution for the image quality enhancement after its recovery in a VSS scheme. We achieved good performance but with a long training penalty involving the repetitive back-propagation learning algorithm over the training data. The use of GPGPU drastically reduced the training time with the superior quality of the image intact.



## CHAPTER 6

### **MEDICAL IMAGE SECRET SHARING USING SUPER-RESOLUTION FOR CAD SYSTEMS**

Medical imaging systems produce images conveying information about the structure and functioning of the human organs. These medical images pose several challenges to automate their analysis owing to their low quality. The trade-offs in the design and manufacturing of such medical equipment lead to compromise the quality degradation of the images they produce, with an absence of useful information (Qiu et al. 2021). In the thirst to improve the medical images' effectiveness, medical-practitioners started accepting the concept of super-resolution (SR) assisting medical diagnosis (Deeba et al. 2020). The biomedical advancements necessitated the communication of medical images over the public channels. Therefore, many recent investigations focusing on the privacy and security of medical images (Pandey et al. 2020).

The visual cryptography (VC) paradigm has emerged to provide safe communication of visual information like images in the public channels. VC embodies perfect safety and recovery of the secret image (Punithavathi and Geetha 2017). Traditional VC schemes use computations not only during encryption but also during decryption (Sharma et al. 2018). Visual secret sharing (VSS) is a VC category that allows sharing secret information among shares equal to the number of parties. These shares are random matrices (Sharma et al. 2018). A coalition of a subset of these shares helps the human visual system (HVS) perceive the secret. Parties with their individual shares or with their shares less than the subset's size fail to learn the secret information. The

earlier state-of-the-art VSS approaches limited to process binary secret images are insufficient for many medical image applications like telemedicine and forensic images where security during communication is paramount (Shivani 2018). Most of the recent literature preferred using machine learning (ML) models to analyze histopathological images (Rachapudi and Devi 2020). The most challenging part in applying ML-based computer-aided diagnosis (CAD) is the feature extraction (Mangal et al. 2020).

The computer-aided detection and diagnosis system brings objectivity to the analysis of medical images. Also, the CAD system offers non-tenderness, non-invasive, speedy evaluation, and accuracy. In Arora et al. (2020), researchers innovated an early detection of skin cancer using bag-of-features (BOF) and support vector machines (SVM). The Scale Invariant Feature Transform (SIFT) feature extractor in conjunction with random forest multiclass classifier has proved the maximum 69.03% accuracy with 95% area under the curve (AUC) (Bansal et al. 2021) compared to k-NN, Nave Bayes, and Decision Tree for the Caltech-101 public dataset.

In the proposed Improved Randomized VSS (IRVSS) with super-resolution (SR), Mhala and Pais (2019b) demonstrated that the BOF with the Discrete Cosine Transformation (DCT) outperformed the BOF with SIFT features. Recently many researchers utilized discrete wavelet transformation (DWT) to extract the features needed for categorization (Noyum et al. 2021; Rajashekhar et al. 2021). DWT method reduces the storage requirement (Rajashekhar et al. 2021). In this chapter, we present a novel medical image VSS approach harnessing state-of-the-art GPU for VSS and CNN-based SR to improve efficiency and effectiveness. The rest of the chapter is structured as follows: Related work in this area is given in Section 6.1. In section 6.2, a concise discourse about the presented methodology is provided. Experimental results are given in Section 6.3 and we conclude this chapter in Section 6.4.

## **6.1 VSS SCHEMES APPLIED TO MEDICAL IMAGES**

Improving the contrast, minimizing the share sizes, or reducing the computations continued to be the significant research challenges in the field of VSS (Sarosh et al. 2021). A novel flexible random grid-based system invented by Huang and Juan (2020) tried

to negotiate the number of shares and the recovered image quality. Though not reported for the medial images, this VSS model reduced the time complexity to  $O(Nhw + hw \log(hw))$  where  $N$  is the number of shares,  $h$  is the height, and  $w$  is the width of the image.

There is a shortage of literature concerning "Medical Image Secret Sharing (MISS)" that are the VSS schemes exclusively designed and applied to secure medical images. Recently Zhang et al. (2020), in their review on medical image confidentiality (or security) technology tried to map related works of literature into a coherent taxonomy. They collected the latest five year's 123 pieces of literature from renowned academic repositories and classified them into five categories. The first classification, namely "medical image security algorithms", is a collection of 77 (about 63%) works of literature in which only 4 (only 3%) papers are under its sub-category MISS. They further classified MISS into 1) Shamir's MISS schemes (Marwan et al. 2019; Sah et al. 2018) and 2) Visual Cryptography based MISS schemes (Bakshi and Patel 2019; Kanso and Ghebleh 2018). Therefore, we discuss other recent state-of-the-art VC schemes applied to medical images with their time complexity.

Analysis of medical images considerably assists medical practitioners in saving the life of the patients by diagnosis and prognosis. A category of image encryption called the chaotic mapped encryption of medical images suffers from less security, inconsiderable keyspace, and more computational time. Recently, Selvi et al. (2021) coupled the couple-map-lattice with the salp-swarm procedure for better efficiency, security, and reliability of the medical images through the communication channel. Telemedicine offers a remote diagnosis of diseases and more frequent communication of data over the insecure channel. The vulnerabilities in the confidentiality, integrity, and authentication of medical images hindered mobile clinical applications' growth. The forbidden computation time of chaotic schemes made them do not fit for real-time applications. Therefore, Banday and Pandit (2021) presented chaotic cryptography with an encryption efficiency of 0.83 seconds for the benchmark images of  $512 \times 512$  medical modality. The foremost downside of such schemes is the computation complexity increases drastically with the increase in the image size.

To secure extensive medical image communication in telemedicine, Fares et al. (2021) proposed two watermarking techniques. The first method combined DCT (Discrete Cosine Transformation) and Schur decomposition with integration in the image's medium frequencies. Another one is a coalition of DWT and Schur decomposition which improved imperceptibility and robustness. Another VSS category called the Reversible Data Hiding (RDH) techniques applicable to medical records ensure image fidelity without distortion of the original secret image. RDH approaches are computationally expensive (Abdel-Nabi and Al-Haj 2021; Anushiadevi et al. 2021) proposed an RDH concerning the security infracts, especially for the offshored medical images in the cloud infrastructure using the difference in the most-significant-bit of the pixel values. This approach's overall execution time complexity was  $O(h)$  where  $h$  is the height of the image. Mhala and Pais (2019b) proposed an improved Randomized VSS (IRVSS) model applied to the medical images achieving a better contrast over the RVSS they proposed earlier (Mhala et al. 2017).

Reconceiving sequential algorithms with parallel computations using GPGPU has been the recent research trend having significance on par with the novel parallel algorithms (Wang et al. 2018). In the recent four years, only a few VSS review papers published in reputable journals. Findings in the survey papers Punithavathi and Geetha (2017), Sharma et al. (2018), Chanu and Neelima (2019) and the very recent paper Ibrahim et al. (2021) have not listed GPGPU-based VSS schemes. Those findings also pave the way for understanding the rationale, challenges and opportunities to exploit GPGPU in the VSS domain a priori.

The state-of-the-art VSS schemes deficit in the following aspects:

- Conventional sequential VSS models are inherently computationally extensive.
- The VSS models' execution time, which leverages super-resolution (SR) to improve the recovered image quality, is prohibitive, making such models do not fit for time-critical applications.
- The application of existing VSS models to medical images demands more performance gains in efficiency and effectiveness.

We presented a novel high-performance Medical Image Secret Sharing (**MISS**) for the histopathological medical images. The contribution of this work is as follows.

- Use of GPU for the tasks involved in the VSS technique and extensive use of GPU's constant memory for further performance gain.
- Leverage GPU power to improve the recovered image using super-resolution in less time.
- Presented a fused feature extraction with a random forest multi-class classifier to evaluate the CAD system model.

Table 6.1 explains important symbols used in the algorithms shown in the following section.

Table 6.1: Description of symbols

Symbol	Description
$p$	Number of parties
$M$	Width of the image
$N$	Height of the image
$P$	Input secret image in host memory
$R, G, \text{ and } B$	Red, Green, and Blue channels of the colour image
$P_{gs}$	Input secret image in GPU global memory where $s = R, G, \text{ and } B$
$G_j$	Basis matrices in host memory
$G_{jc}$	Basis matrices in GPU constant memory
$q$	Error diffusion constants in host memory
$q_c$	Error diffusion constants in GPU constant memory
$d$ and $e$	Daubechies DWT constants
$d_c$ and $e_c$	Daubechies DWT constants in GPU constant memory
$H$	Halftone image in host memory
$H_g$	Halftone image in GPU global memory with $R, G, \text{ and } B$ channels combined
$H_{gs}$	Halftone image in GPU global memory where $s = R, G, \text{ and } B$
$S_i$	Shadow images in host memory where $(1 \leq i \leq p)$ }
$S_{ig}$	Shadow images in GPU memory where $(1 \leq i \leq p)$ }
$E_i$	Embedded shadow images in host memory where $(1 \leq i \leq p)$ }
$E_{ig}$	Shadow images with embedded data in GPU memory where $(1 \leq i \leq p)$ }
$W$	DWT image in host memory
$W_g$	DWT image in GPU global memory

## 6.2 MISS METHODOLOGY

Figure 6.1 shows the presented model's architecture containing two parts, namely shadow generation and image reconstruction. Shadow generation requires the creation of the halftone image, produce shadows and embed data in the shadows. Algorithm 6.1 gives the sequence of these activities during shadow generation. This is the host module which invokes the functions to be accomplished from the GPGPU. Inputs to Algorithm 5.1 are the secret image  $P$  and the basis matrices  $G_j$ . This module copies required data between CPU and GPGPU memory locations. This host module also copies error diffusion parameters  $q[4]$  as well Daubechies DWT parameters  $d[4]$  and  $e[4]$  to the GPGPU constant memory as these values are only read by all the threads. The *configuration parameters* are used to define the thread organization. Reconstruction is the process of recovering the original secret image by reversing these operations. In these activities, data-parallel tasks are exploited utilizing GPU and its constant memory. The super-resolution using CNN with GPU results in the model's high performance regarding the model's speed and the quality of the recovered image. The following sections discuss each of these tasks in detail. .

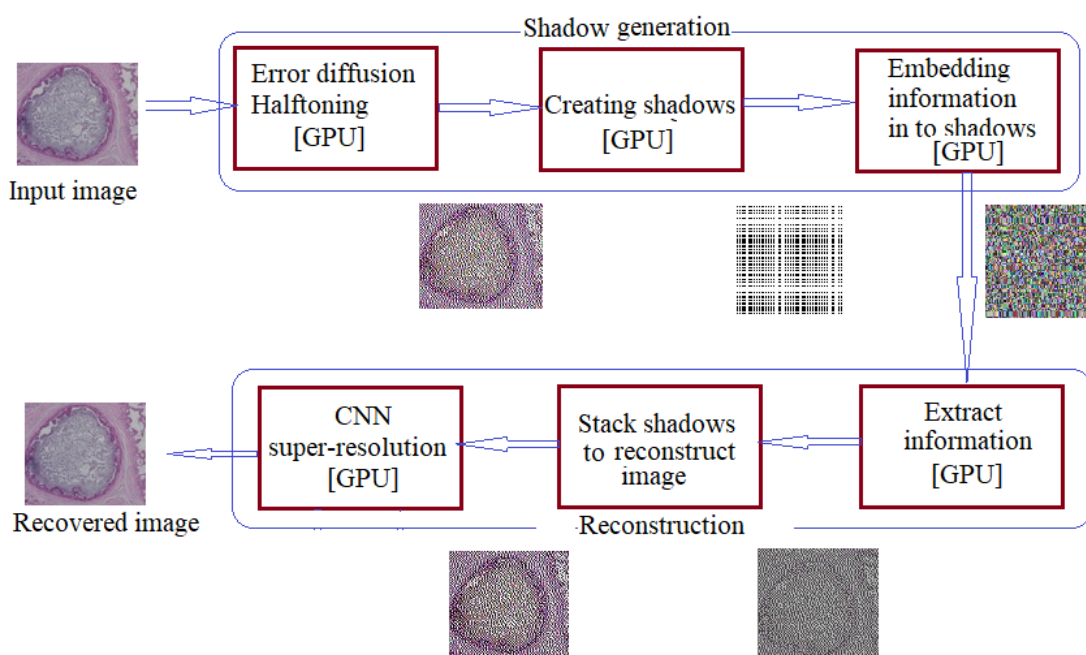


Figure 6.1: Block diagram of the MISS methodology.



**Algorithm 6.1: Shadow\_creation( $P, G_j$ ) //CPU function.**


---

```

// p- number of parties, M- width, N- height, P- secret image, Gj- basis
// matrices, q[4]- error diffusion constants, d[4] and e[4] -Daubechies DWT
// constants.
// Hg- halftone image in GPU memory, Sig-shadow images in GPU memory, Wg[M × N]-
// DWT image in GPU memory and Eig- shadow images with embedded data in GPU
// memory.
// W- DWT image in CPU memory.
// H- output halftone image in host memory, Ei- output embedded shadow images in
// host memory.
Input: 1. P[M × N].
          2. {Gj[2 × p] | (0 ≤ j ≤ p) }.
Output: H[W × H], {Si[M × N], Ei[M × N] | (1 ≤ i ≤ p) }.
// use GPU constant memory to aggressively broadcast read-only data.
1 qc[4] = q[4], Gjc[2 × p] = {Gj[2 × p] | (0 ≤ j ≤ p) }, dc[4] = d[4], ec[4] = e[4].
// allocate GPU memory.
2 Allocate {Pgs[M × N] | s = R, G, B}, {Hgs[M × N] | s = R, G, B}, Sig[M × N], Hg[M × N], Wg[M × N],
   Eig[M × N] in GPU memory.
// separate I[W × H] in to RGB values stored in GPU memory.
3 Pgs[M × N] = split_rgb(P[M × N])
// Configure threads in the kernel configuration.
4 Compute GPU kernel configuration parameters.
// Invoke kernel to create halftone.
5 kernel_error_diffusion(Pgs, Hgs, configuration parameters).
// merge Hgs[M × N] and store in host memory.
6 H = merge_rgb(Hgs[M × N])
// keep color halftone in GPU memory.
7 Hg = Wg = H
// Invoke kernel to create shadows.
8 kernel_create_shadows(Hg, Sig, configuration parameters).
// store Sig[M × N] in CPU memory.
9 {Si} = Sig[M × N]
10 size = M × N
// Invoke kernel to create DWT shadows.
11 while size ≥ 4 do
12     kernel_dwt(Hg, Wg, size, configuration parameters)
13     Hg = Wg
14     size = size/2
15 end
// store Wg[M × N] in host memory.
16 {W} = Wg[M × N].
17 Generate key[M/2, N/2].
// Embed data.
18 kernel_embed_data(Wg, Sig, key, size, Eig, configuration parameters)
// store Eig in host memory.
19 {Ei} = Eig
20 Release GPU memory.
21 Return.

```

---

**6.2.1 Error diffusion halftoning using GPU**

Many error-diffusion approaches are presented in the literature, varying in pixel rastering way and the weights used. We use the Floyd-Steinberg error diffusion (Floyd 1976) approach to obtain the input image's halftone equivalent, as shown in Figure 6.2. This dithering scans the image row-wise and column-wise to quantize each pixel. The resulting error is spread to the adjacent pixels keeping the already quantized pixels unaltered, ensuring average quantization error near zero. Algorithm 6.2 describes the general error diffusion technique.

Figure 6.3 shows the breast cancer biopsy sample images (256 × 256) (a) normal, (b) carcinoma in situ, and (c) invasive. Figure 6.4 shows the corresponding halftone

**Algorithm 6.2:** Error\_diffusion

```

1 foreach  $row \in image$  do
2   foreach  $pixel, px \in row$  do
3     Calculate nearest colour  $\phi$  to  $px$ .
4      $error, e = px - \phi$ .
5     Diffuse  $e$  to adjacent pixels using Floyd-Steinberg scheme (Floyd
6       1976).
7   end
8 end

```

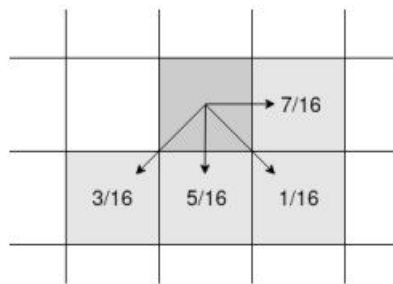


Figure 6.2: Floyd-Steinberg error diffusion.

images. Algorithm 6.3 describes the GPU kernel for the data-parallel error diffusion

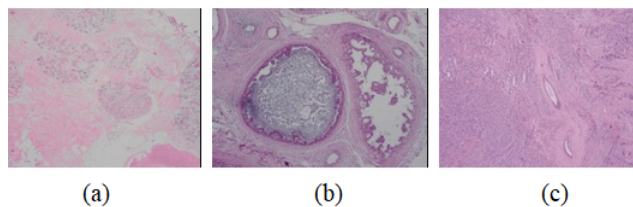


Figure 6.3: Breast cancer biopsy sample images (a) normal, (b) carcinoma in situ, and (c) invasive.

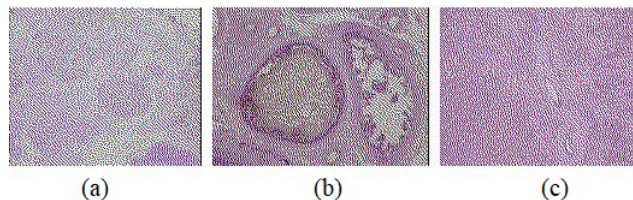


Figure 6.4: Breast cancer biopsy halftone sample images (a) normal, (b) carcinoma in situ, and (c) invasive.

halftoning. We use constant read-only memory to aggressively broadcast the Floyd-Steinberg constants to the threads to offer improved speedup.

### 6.2.2 Generating shadow images using GPU

For the  $p$  parties, the presented model requires  $(p + 1)$  basis matrices to generate  $p$  shadow images of size equal to the input image size. Table 6.2 shows five basis matrices used to generate  $p = 4$  shadow images.

The presented approach uses Algorithm 6.4 to create shadow images using GPU. The

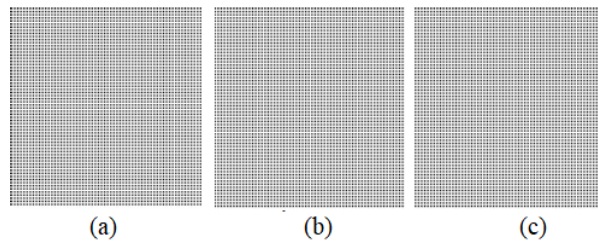


Figure 6.5: Share image of breast cancer biopsy sample images (a) normal, (b) carcinoma in situ, and (c) invasive.

basis matrices stored in the GPU-constant memory facilitate aggressive broadcasting of those data and reduce the GPU memory traffic, improving bandwidth. As the number of parties and the image size increases, this broadcasting yields considerable speedup. The randomness in row selection of the basis matrices strengthens the dynamic and secured sharing of visual information. Meanwhile, it may lead to blocking artefacts during secret recovery. This impact does not influence the reconstruction results as we use super-resolution to obtain a high-quality image. Figures 6.5 (a), 6.5 (b), and 6.5 (c) shows a constructed corresponding shadow.

### 6.2.3 Information embedding using Discrete Wavelet Transform (DWT)

The use of wavelet transform using filters in the image processing domain has been increasing in recent years (Hemdan 2021). The image gets partitioned down into specific frequencies depending on the filter. The  $cA$  (all detail),  $cH$  (horizontal detail),  $cV$  (vertical detail), and  $cD$  (diagonal detail) are the four primary decomposed components of the Discrete Wavelet Transform (DWT) image. This dyadic decomposition, if repeated, results in multi-scale wavelet segments, as shown in Figures 6.6 and 6.7. Figure 6.6 denotes a 2-level DWT decomposition considering the image in the matrix form, and

---

**Algorithm 6.3:** *kernel\_error\_diffusion*( $P_{gs}, H_{gs}$ ) //GPU kernel.

---

```

//  $q_c[4]$ - Error diffusion constants already in GPU
// constant memory for aggressive broadcasting of
// read-only data
// assume  $err[M \times N] = 0$ .
Input: 1.  $\{P_{gs}[M \times N] \mid s = R, G, B\}$ .
         2.  $\{H_{gs}[M \times N] \mid s = R, G, B\}$ .
Output: Updated  $\{H_{gs}[M \times N] \mid s = R, G, B\}$  in GPU global memory.
// Fetch GPU memory index.
1  $t = blockIdx.x \times blockDim.x + threadIdx.x$ 
// Active  $N$  threads.
2 if ( $t < N$ ) then
    // Process data row-wise per thread and perform
    // error-diffusion halftoning.
3 for  $j = 0 : (M - 1)$  do
4     if ( $P_{gs}[t \times M + j] + err[t \times M + j] < 128$ ) then
5          $H_{gs}[t \times M + j] = 0$  else
6              $H_{gs}[t \times M + j] = 255$ 
7         end
8     end
9      $d = P_{gs}[t \times M + j] + err[t \times M + j] - H_{gs}[t \times M + j]$ 
10    if ( $j + 1 < M$ ) then
11         $err[t \times M + j + 1] = err[t \times M + j + 1] + d \times q_c[0]$ 
12    end
13    if ( $t \times M + 1 < N$ ) then
14         $err[t \times M + j + 1] = err[t \times M + j + 1] + d \times q_c[1]$ 
15    end
16    if ( $t \times M + 1 < N$  and  $j - 1 \geq 0$ ) then
17         $err[t \times M + j] = err[t \times M + j] + d \times q_c[2]$ 
18    end
19    if ( $t \times M + 1 < N$  and  $j + 1 < M$ ) then
20         $err[t \times M + j + 2] = err[t \times M + j + 2] + d \times q_c[3]$ 
21    end
22 end
23 end

```

---

---

**Algorithm 6.4:** *kernel\_create\_shadows*( $H_g, S_{ig}$ ) //GPU kernel.

---

```

//  $G_{jc}$ - basis matrices already in GPU constant memory
// for aggressive broadcasting of read-only data.
//  $p$ - number of parties.
Input: 1.  $H_g[M \times N]$ .
          2.  $S_{ig}[M \times N]$ .
Output:  $S_{ig}[M \times N]$  in GPU memory.
// Fetch GPU memory index.
1  $t = \text{blockIdx}.x \times \text{blockDim}.x + \text{threadIdx}.x$ 
// Point to basis matrices.
2 if ( $\text{threadIdx}.x < p$ ) then
3   |  $t_j = \text{threadIdx}.x$ 
4 end
// Random  $r$ .
5  $r = 0$  or 1
// Obtain shadow pixels parallel for ( $1 \leq (i \ \& \ j) \leq p$ ).
6 if  $r = 0$  then
7   | if  $H_g[t] = \text{White}$  then
8     | |  $S_{ig}[t] = G_{0c}[t_j]$  else
9     | | |  $S_{ig}[t] = G_{jc}[t_j]$ 
10    | | end
11   | end
12   | else
13     | if  $H_g[t] = \text{White}$  then
14     | | |  $S_{ig}[t] = G_{0c}[n + t_j]$  else
15     | | | |  $S_{ig}[t] = G_{jc}[n + t_j]$ 
16     | | | end
17     | | end
18   | end
19 end

```

---

Table 6.2: Five basis matrices to create four shadow images.

$$G_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad G_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

$$G_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} \quad G_3 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

$$G_4 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

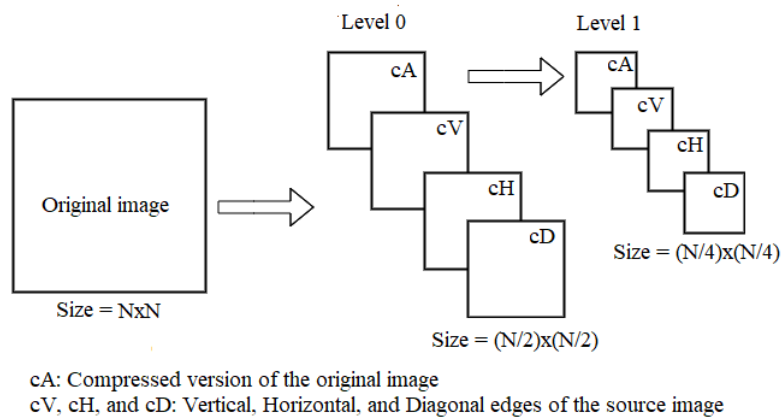


Figure 6.6: DWT transformation (2 level).

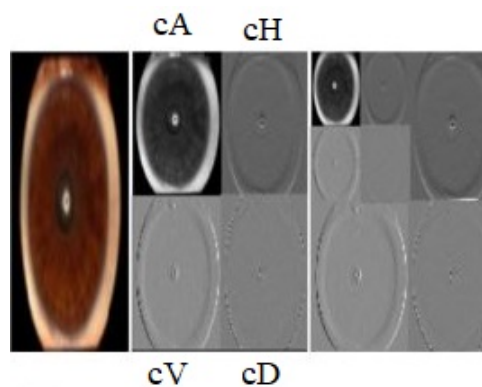


Figure 6.7: DWT transformation (2 level) (Dhage et al. 2015).

Figure 6.7 augments the same information by visualizing with an image. As  $cA$  is the low pass filtered component containing rich input information, shares embedded with it with a reversible key retain the input's coarser detail. Consequently, the inverse DWT outcome is indistinguishable from the input of the DWT (Bao and Zhang 2020).

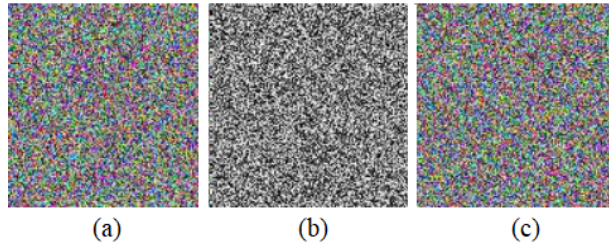


Figure 6.8: Embedded image of breast cancer biopsy sample images (a) normal, (b) carcinoma in situ, and (c) invasive.

Algorithm 6.5 contains the steps in DWT using GPU. Daubechies wavelet constants stored in GPU constant memory to improve the performance. Embedding  $cA$  component using a reversible key is shown in Algorithm 6.6. Figure 6.8 denotes the data embedded shadow images. The length of this key is  $(r \times c \times d)$ , where  $r$ ,  $c$ , and  $d$  indicates rows, column, and depth of the input image. The key is a unique combination of values to facilitate the reverse recovery of the embedded share. The key generated ensures that it should not reveal information by maintaining the maximum pixel value using a threshold mechanism. For example, the key values range from 0 to 255 for the 8-bit input image.

#### 6.2.4 Reconstruction of the secret image

Finally, we de-embed the information and recover the low-resolution secret image with a logical XOR operation of the shares. Figure 6.9 denotes the extracted low-resolution shadows. Figure 6.10 shows the recovered low-resolution images. The super-resolution on a Convolution Neural Network (CNN) constructs the high-resolution secret image. Finally, Figure 6.11 shows the recovered high-resolution images.

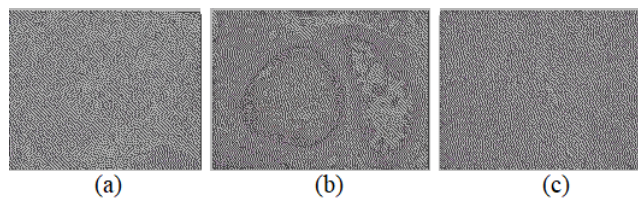


Figure 6.9: Extracted image of breast cancer biopsy sample images (a) normal, (b) carcinoma in situ, and (c) invasive.

---

**Algorithm 6.5:** *kernel\_dwt*( $H_g, W_g, size$ ) //GPU kernel.

---

```

//  $d_c[4]$  and  $e_c[4]$ – Daubechies DWT constants already in
GPU constant memory for aggressive broadcasting
of read-only data.
// Let  $blockDim.x = M$ .
Input: 1.  $H_g[M \times N]$ .
          2.  $W_g[M \times N]$ .
          3.  $size$ .

Output:  $W_g(M \times N)$  in GPU memory.
// Fetch GPU memory index.
1  $t = blockDim.x \times blockDim.x + threadIdx.x$ 
2  $s = size/2$ 
3 if ( $2 \times t < (size - 3)$ ) then
4    $W_g[t] = H_g[2 \times t] \times d_c[0] + H_g[2 \times t + 1] \times d_c[1] + H_g[2 \times t + 2] \times$ 
    $d_c[2] + H_g[2 \times t + 3] \times d_c[3]$ 
5    $W_g[t + s] = H_g[2 \times t] \times e_c[0] + H_g[2 \times t + 1] \times d_c[1] + H_g[2 \times t + 2] \times$ 
    $e_c[2] + H_g[2 \times t + 3] \times e_c[3]$ 
6 end
7 if ( $2 \times t = (size - 2)$ ) then
8    $W_g[t] = H_g[t - 2] \times d_c[0] + H_g[t - 1] \times d_c[1] + H_g[0] \times d_c[2] + H_g[1] \times d_c[3]$ 
9    $W_g[t + s] = H_g[t - 2] \times e_c[0] + H_g[t - 1] \times e_c[1] + H_g[0] \times e_c[2] + H_g[1] \times e_c[3]$ 
10 end

```

---



---

**Algorithm 6.6:** *kernel\_embed\_data*( $W_g, S_{ig}, key, size, E_{ig}$ ) //GPU kernel.

---

```

Input: 1.  $W_g[M \times N]$ .
          2.  $\{S_{ig}[M \times N] \mid (1 \leq i \leq p)\}$ .
          3.  $key[M/2, N/2]$ .
          4.  $size$ .

Output:  $\{E_{ig}[M \times N] \mid (1 \leq i \leq p)\}$  in GPU memory.
// Fetch GPU memory index.
1  $col = blockDim.x \times blockDim.x + threadIdx.x$ 
2  $row = blockDim.y \times blockDim.y + threadIdx.y$ 
// only  $size/4$  threads active.
3 if ( $col < size/2 \ \&\& \ row < size/2$ ) then
4   // Embedding.
    $E_{ig}[col, row] = S_{ig}[col, row] + key[col, row] \oplus W_g[col, row]$ 
5 end

```

---



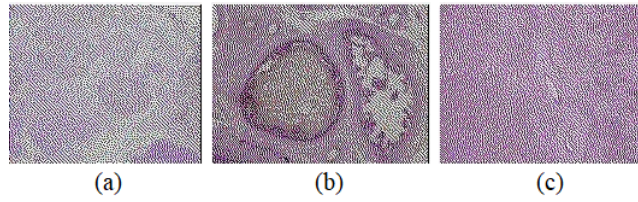


Figure 6.10: Recovered image of breast cancer biopsy sample images (a) normal, (b) carcinoma in situ, and (c) invasive.

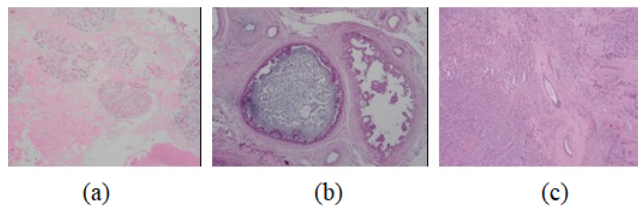


Figure 6.11: MISS system: Final SR recovered image of breast cancer biopsy sample images (a) normal, (b) carcinoma in situ, and (c) invasive.

### 6.2.5 Convolution Neural Network (CNN)

The CNN architecture shown in figure 6.12 provides SR using GPGPU. The CNN

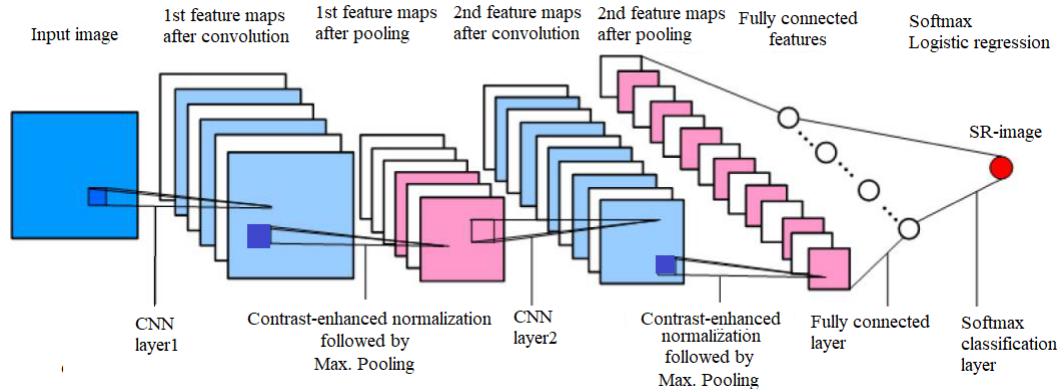


Figure 6.12: Layered CNN architecture of the proposed SR model.

consists of a convolution layer, contrast-enhanced normalization layer, max-pool layer, and a fully connected layer. The GPGPU augments the deep learning time of the CNN. Table 6.3 displays the CNN layer summary of the presented system. The following paragraphs briefly explain the layers of CNN.

Figure 6.13 depicts the convolution computation on the input values with a learnable kernel of filter to extract a feature. Precisely, a matrix is an input to this layer convolved using, say,  $m$  learnable kernels to produce  $m$  feature maps. A feature map is essentially

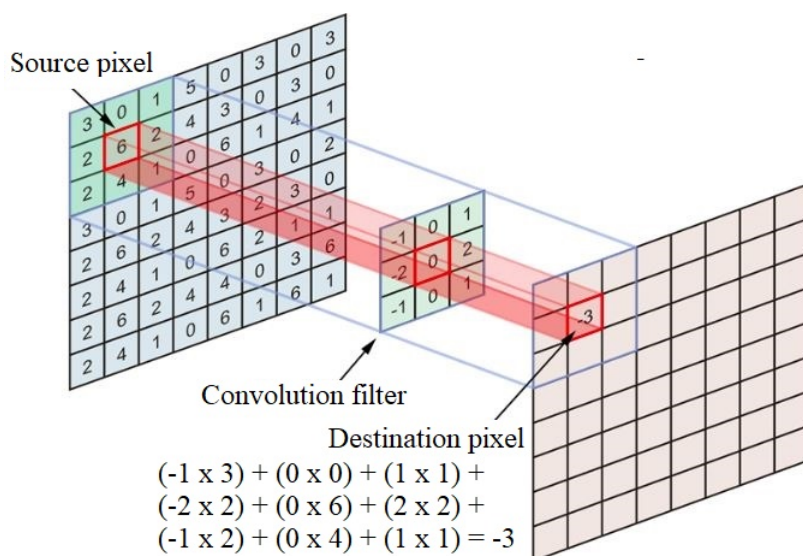


Figure 6.13: Convolution computation with a filter (con 2021).

Table 6.3: CNN layer summary of the proposed SR model.

CNN- (type)	Output configuration	Number of parameters
Layer 1- (2D)	$(None, 5, 5, 3) \times 4096$	319488
Layer 2- (2D)	$(None, 5, 5, 3) \times 4096$	319488
Total number of parameters = 638976		
Trainable number of parameters = 638976		
Non-trainable number of parameters = 0		

a dot product of the kernel values, and the input values added a bias with a predefined stride.

Generally, the convolution layer merged with the activation layer. This activation layer results in a nonlinear output. We used rectified linear unit (ReLU) activation function shown in equation 6.1 for the input value  $x$ .

$$f(x) = \max\{0, x\} \quad (6.1)$$

Because ReLU is linear for positive input values and nonlinear for negative values, it preserves many properties easy to optimize with gradient-based methods and generalize well.

The normalization between convolution layers minimizes the over-fitting errors. Pixel centering uses a Z-score normalization that subtracts the mean ( $\mu$ ) to center the pixel values and then divides it by the standard deviation ( $\sigma$ ) of the feature or pixel. This normalization guarantees each feature to contain a similar span so that the gradients are

under control during backpropagation. This normalization also mitigates outlier and over-fitting issues.

$$Z_{score} = \frac{data - \mu}{\sigma} \quad (6.2)$$

The pooling layer aims to control overfitting by reducing the dimensionality, size of the feature map, number of parameters, training time, and network calculation while keeping the vital information. The widespread max-pooling takes the maximum value in each window.

Each input element  $x_i$  of the fully connected layer with the softmax function (Equation 6.3) generates high-contrast SR image.

$$\text{Softmax } \sigma(\vec{x}_i) = \frac{e^{x_i}}{\sum_{j=1}^k e^{x_j}} \quad (6.3)$$

We use GPGPU during SR so that the images get deployed between CPU and GPGPU to minimize the computational overhead. Two CNN layers are used in the proposed model with 638976 learning coefficients for training as shown in Table 6.3.

### 6.3 EXPERIMENTAL RESULTS

We developed the presented parallel system using CUDA, OpenCV, and MATLAB on the PARAM Shavak Super Computer (CPU: Dual core Intel<sup>®</sup>Xeon<sup>®</sup>-E5-2670 inside 24 cores each, 8TB RAM, 2.3 GHz., and a NVIDIA<sup>®</sup>Tesla K40 GPGPU: 2880 cores, 12 GB GDDR5, 745 MHz). In this section, we analyze the performance of the presented model using breast cancer data available at can (2015). Three classes of this dataset's 361 images are normal (119 images), carcinoma in situ (102 images), and invasive carcinoma (140 images). Initially, we compare the objective parameters (Athar and Wang 2019) of the presented system with the state-of-the-art two models in Mhala et al. (2017) and Mhala and Pais (2019b) in section 6.3.1. Section 6.3.2 proves that existing CAD system can use our model for diagnosis. In section 6.3.3, we highlight the speed of the presented GPU model over its CPU model.

#### 6.3.1 Objective performance analysis of MISS

We have explained *PSNR*, *NCC*, and *NAE* in chapter 4 using equations 4.17, 4.15, and 4.16. *SSIM* is discussed in chapter 5 using equation 5.16. We used the these pa-

rameters to demonstrate that the presented model outperforms the two state-of-the-art models RVSS and IRVSS. Table 6.4 evinces that the presented model outperforms the

Table 6.4: Objective analysis of MISS with RVSS and IRVSS.

Objective analysis (average values)	VSS models		
	RVSS	IRVSS	MISS
<i>PSNR</i>	60.745	75.5495	82.71
<i>NCC</i>	0.2024	0.8416	0.993
<i>NAE</i>	0.3019	0.0510	0.031
<i>SSIM</i>	0.0601	0.6988	0.982

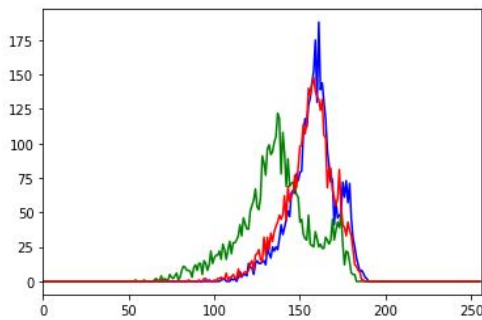


Figure 6.14: RGB histogram of the input secret image (rescaled to the size  $64 \times 64$ ) analysis.

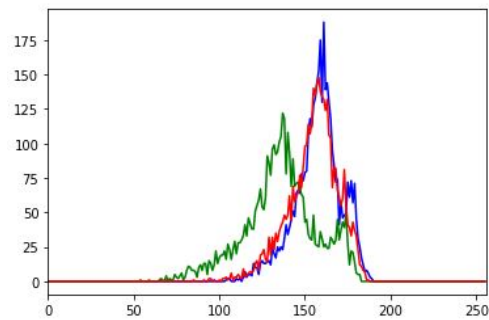


Figure 6.15: RGB histogram of the recovered secret image (rescaled to the size  $64 \times 64$ .) analysis

two benchmark models in *PSNR* (82.71), *NCC* (0.993), *NAE* (0.031), and *SSIM* (98.2%) parameters. RGB histogram analysis in Figures 6.14 and 6.15 elicits that the RGB histogram of the rescaled recovered image is similar to the original image, demonstrating the superior quality of the proposed model.

### 6.3.2 Random forest classifier for Computer Aided Diagnosis (CAD)

We evaluate the presented system for CAD using a random forest classifier as follows.

1. **Fused (SIFT+DWT+DCT) feature extraction:** Lowe (2004) proposed the SIFT detector and descriptor that extracts the object features considering its various transformations and illuminations apriori. SIFT outputs a feature vector of size (keypoints  $\times$  128) for 128 keypoint descriptors. Using the input image, SIFT constructs a multi-resolution pyramid. First, the SIFT algorithm finds local extrema

Table 6.5: Accuracy, sensitivity, and specificity of MISS for the CAD system.

CAD parameters $\Rightarrow$	Accuracy			Sensitivity			Specificity		
	300	500	1000	300	500	1000	300	500	1000
Cluster size $\Rightarrow$									
VSS Models $\Downarrow$									
BOF-DCT OI	0.99061	0.99530	0.99531	0.98148	1	0.98148	1	0.99375	1
RVSS-DCT	0.27230	0.28638	0.24882	1	1	1	0.03125	0.05000	0
IRVSS-DCT	0.99530	0.99530	0.99531	0.98148	1	0.98148	0.99371	0.99375	1
MISS	0.9971	0.9954	0.99613	0.99827	1	1	1	0.99463	1

Table 6.6: Precision, recall, and F-measure of MISS for the CAD system.

CAD parameters $\Rightarrow$	Precision			Recall			F-measure		
	300	500	1000	300	500	1000	300	500	1000
Cluster size $\Rightarrow$									
VSS Models $\Downarrow$									
BOF-DCT OI	1	0.98148	1	0.98148	1	0.98148	0.99065	0.99065	0.99065
RVSS-DCT	0.25480	0.25853	0.24882	1	1	1	0.40612	0.41084	0.39849
IRVSS-DCT	0.98148	0.98148	1	0.98148	1	0.98148	0.98148	0.99065	0.990655
MISS	0.99451	0.99831	1	0.99827	1	1	0.99639	0.99915	1

Table 6.7: MISS performance against benchmark deep learning networks.

Reference	Method used	Performance metrics
(Sun et al. 2017)	CNN and semi-supervised learning	Accuracy = 82.43% AUC = 88.18%
(Antropova et al. 2017)	VGG19 system with fusion classifier for MRI images (Pre-trained)	AUC = 89%
(Samala et al. 2018)	Multi-stage transfer learning	Accuracy = 91%
(Chougrad et al. 2018)	VGG16, ResNet50 Inception v3 (Pre-trained)	Accuracy = 98.23%
(Vo et al. 2019)	Inception network, Deep-CNN, gradient boosting tree classifier	Accuracy = 99.5%
(Yurttakal et al. 2020)	Deep-CNN	Accuracy = 98.33%
(Li et al. 2020a)	Xception network (Pre-trained)	Accuracy = 93.42%
(Li et al. 2020d)	VGG16 (Pre-trained)	Accuracy = 95.1%
MISS	Contrast-enhanced CNN with fused feature extractor and random forest classifier	Accuracy = 99.71%

in the space range by applying the Difference-of-Gaussian (DoG). The keypoints are then the preferred extrema among the local extrema. Second, exacted locations of the keypoints are identified with a threshold value. In the third stage, keypoints unvarying to image rotation are described with the orientation. Finally, 128 keypoint descriptor set is calculated to obtain 16 blocks (each  $4 \times 4$  size), each creating eight bin histograms of orientation. As discussed in section 6.2.3, the DWT decomposes data into different frequency components. Four coefficients computed by DWT are called  $cA$  (all detail),  $cH$  (horizontal detail),  $cV$  (vertical detail), and  $cD$  (diagonal detail). We use only  $cA$  to derive a reduced group of features. DWT works in the spatial domain. We use Discrete Cosine Transformation (DCT) to augment with better discerning features, transforming the input into its frequency domain. Discrete Cosine Transformation (DCT) transforms the  $8 \times 8$  blocks of the image into the frequency domain (using equation 6.4).

$$F_{u,v} = \frac{C(u) \times C(v)}{4} \sum_{s=0}^7 \sum_{t=0}^7 B_l^i(s, t) \times \bar{f}(s, t, u, v) \quad (6.4)$$

where  $0 \leq u, v \leq 7$  and

$$\bar{f}(s, t, u, v) = \cos\left(\frac{(2s+1)u\pi}{16}\right) \cos\left(\frac{(2t+1)v\pi}{16}\right),$$

$$C(e) = \begin{cases} \frac{1}{\sqrt{2}} & e = 0 \\ 1 & e > 0 \end{cases}.$$

2. **Codebook creation:** Then K-means clustering algorithm is used to generate a codebook utilizing features from the image set, which reduces the dimensionality of the extracted features (Mhala and Pais 2019b). Extracted features are grouped into 300, 500, and 1000 sizes.
3. **Bag-of-features (BOF) representation:** The histogram representation for the image consists of bins equal to the number of clusters. The more clusters, the more accurate is the classification process.
4. **Random forest classification:** Finally, we use a random forest multi-class classifier to classify the test image samples into three categories: Normal, carcinoma in situ, and invasive carcinoma.

We used cluster sizes of 300, 500, and 1000 to obtain the presented model's CAD performance measures. Table 6.5 lists the accuracy, sensitivity, and specificity values of the presented model compared with the RVSS and IRVSS standard models with DCT features. BOF-DCT OI is the bag-of-features with DCT values on the original image. Table 6.6 compares the precision, recall, and F-measure values of these models. It is evident from Tables 6.5 and 6.6 that the analysis of the recovered images of the MISS demonstrated promising results compared to the reference models proving the suitability of those images for CAD systems for automated medical diagnosis. Table 6.7 compares the performance of the MISS with the recent benchmark deep learning networks designed for the medical images. From Table 6.7, it is evident that our model outperforms the other state-of-art deep learning models designed for the histopathological medical images with an achieved 99.71% accuracy. We attribute outperformed accuracy of this model to the synergistic features by combining SIFT, DWT, and DCT features.

Table 6.8: Speedups of the MISS over the sequential model.

Sl.No.	Tasks	Execution time(seconds)		Speedup
		Sequential model	MISS GPU model	
1.	Error diffusion halftoning	0.89300	0.00298	300
2.	Generating shadow images	0.32701	0.00381	86
3.	Wavelet data embedding	0.01161	0.00081	14
4.	CNN	0.01387	0.00756	2

Table 6.9: MISS Speedup against benchmark GPU-based models.

Reference	Application	NVIDIA GPGPU	Speedup
(Smith et al. 2012)	Compressed sensing (split Bregman regularization)	Tesla C2050	10×
(Nam et al. 2013)	Compressed sensing (3D Radial Cardiac MRI)	GeForce GTX 480	54×
(Sabbagh et al. 2016)	Compressed sensing (L1-ESPIRiT algorithm)	Tesla K20m	15×
(Inam et al. 2017)	Parallel imaging (GRAPPA operator gridding)	GeForce GTX 780	30×
(Chang et al. 2017)	Compressed sensing (ADMM algorithm)	GeForce GTX 650	30×
MISS	VSS with super-resolution (encryption and decryption)	Tesla K40	800×

### 6.3.3 Speedup

MISS uses NVIDIA Compute Unified Device Architecture (CUDA), an evolving data-parallel paradigm. Table 6.8 lists the achieved speedup of around 800× in the MISS model over its sequential model for four shadow images. We compare the MISS model's speedup against the state-of-the-art GPGPU-based medical image reconstruction models designed for different applications in Table 6.9. The 800× speedup of our model is more efficient than those models.

## 6.4 SUMMARY

MISS is a typical VSS arena intended to address the challenges like execution time and quality of the recovered image in automating the image diagnosis. We have proposed a high-performance MISS using GPU. The super-resolution with a CNN model utilizing GPU reconstructs the high-resolution medical image. Also, we presented a



fused feature extraction-based random forest multi-class classifier to evaluate the proposed model's performance for the CAD system. Analysis of objective and CAD parameters demonstrate that the proposed model outperforms the reference models RVSS and IRVSS. Empirical 99.71% accuracy of the MISS is superior compared to the presented well-performed deep learning network-based medical models. We also achieved a speedup of about  $800\times$ , enabling the model fit for time-critical applications.

We applied our VSS scheme with super-resolution to the medical images. The medical images are of inherent poor quality. Therefore VSS approaches to secure those images result in further quality degradation of the recovered image. But with the GPGPU super-resolution, our model has addressed this quality issue effectively and efficiently. Therefore this model can also be used in health-critical systems.



## CHAPTER 7

### CONCLUSIONS AND FUTURE SCOPE

VSS is a field of cryptography embodying encryption and decryption of images. Traditional VSS schemes, in general, consume more execution time and produce low-quality images. GPGPU-based VSS schemes address these two shortcomings by taking advantage of the high-performance computing power of a GPGPU and deep-learning super-resolution models. We conclude our work on GPGPU-based VSS models as follows.

- The experimental results of the presented random grid-based VSS approach proved its efficiency over the traditional random-grid method with a considerably increased speedup as the secret image size increases. This scheme outperforms  $3651\times$  and  $1720\times$  in generating halftone and share images for a colour image size  $1024 \times 1024$ . For this image, the total performance gain is  $2688\times$  more than the conventional method.

GRVSS achieved a speedup range of  $1.6\times$  to  $1.8\times$  for four shares and  $2.63\times$  to  $3\times$  for eight shares for different image sizes while retaining the RVSS recovered image contrasts of 70% to 80% for noise-like shares and 70% to 90% for meaningful shares.

- To further enhance the visual quality of the retrieved image, we presented an effective secret image sharing with super-resolution that leverage the deep learning super-resolution technique. This approach outperformed the benchmark models in many-objective parameters with the recovered image contrasts of 96.6% to

99.8% for noise-like shares and 64% to 80 % for meaningful shares and a speedup of  $1.92\times$  over the CPU super-resolution.

- Also, we presented an effective VSS model with super-resolution utilizing a Convolution Neural Network (CNN) intending to increase both the contrast of the recovered image and the speedup. The objective quality assessment proved that the proposed model produces a high-quality reconstructed image with the recovered image contrasts of 97.3% to 99.7% and 78.4% to 89.7 % having the Structural Similarity Index (SSIM) of 89% to 99.8% and 71.6% to 90% for the noise-like shares and the meaningful shares respectively. This technique achieved an average speedup of  $800\times$  in comparison with the sequential model.
- The presented scheme is applied to the medical images to design a medical image secret sharing (MISS) suitable for the Computer-Aided Diagnostic(CAD) tools. The result analysis confirmed the high-performance of MISS with a 99.3% contrast of the reconstructed image and  $800\times$  speedup over the sequential MISS. We used cluster sizes of 300, 500, and 1000 to obtain the MISS model's CAD performance measures accuracy, sensitivity, specificity, precision, and F-measure using a proposed fused feature extractor and a random forest classifier. The achieved precision of the MISS of 99.45%, 99.83%, and 100% for cluster sizes 300, 500, and 1000 respectively fits the CAD systems.
- GRVSS utilizes DCT data-embedding, the technique used in RVSS, to improve the contrast of the recovered image. The rest of the presented models utilized reversible DWT data-embedding with a deep neural network super-resolution to enhance the reconstructed image contrast. We empirically demonstrated that the DWT embedding with neural super-resolution is more effective in improving the recovered image quality. We also infer that exploiting GPGPU computational power in the presented VSS models elevated them more suitable for real-life applications.

### **Future Scope**

Although our GPGPU-based VSS models outperformed the state-of-the-art VSS schemes

---

in improving the speed and visual quality, there is scope for future refinement and investigation in the VSS domain. We enumerate few future works as follows.

- The parallel paradigms like Compute Unified Device Architecture (CUDA) have been evolving with matured optimization catalysts. Applying them to the proposed schemes and analyzing the performance implications is both an opportunity and a challenge.
- It is also possible to extend this work to the existing traditional VSS schemes by considering memory efficiency possibilities.
- Many VSS schemes in the literature are amenable to both task parallelism and data parallelism. Retreating them with due consideration on efficiency, resource utilization, and CPU-GPGPU growth contributes to novel approaches in VSS.



## BIBLIOGRAPHY

- (1999(accessed June 10, 2020)). *SIFI Image Database*. [sipi.usc.edu/database/database.php?volume=misc?](http://sipi.usc.edu/database/database.php?volume=misc?)
- (2012 (accessed June 10, 2020)). *Index of /ychou/BPVSS*. [mail.im.tku.edu.tw/~ychou/BPVSS/](http://mail.im.tku.edu.tw/~ychou/BPVSS/).
- (2015). <ftp://ftp.cs.technion.ac.il/pub/projects/medical-image/breastcancerdata/>.
- (2018(accessed March 12, 2021)). *An intuitive guide to Convolutional Neural Networks*. <https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/>.
- (2020 (accessed June 10, 2020)). *CUDA Zone — NVIDIA Developer*. <https://developer.nvidia.com/cuda-zone>.
- Abdel-Nabi, H. and Al-Haj, A. (2021). “Reversible data hiding in adjacent zeros.” *Multimedia Systems*, 1–17.
- Al-Khalid, R. I., Al-Dallah, R. A., Al-Anani, A. M., Barham, R. M., Hajir, S. I. et al. (2017). “A secure visual cryptography scheme using private key with invariant share sizes.” *Journal of Software Engineering and Applications*, 10(01), 1.
- Antropova, N., Huynh, B. Q. and Giger, M. L. (2017). “A deep feature fusion methodology for breast cancer diagnosis demonstrated on three imaging modality datasets.” *Medical physics*, 44(10), 5162–5171.
- Anushiadevi, R., Praveenkumar, P., Rayappan, J. B. B. and Amirtharajan, R. (2021).

- “Uncover the cover to recover the hidden secret-a separable reversible data hiding framework.” *Multimedia Tools and Applications*, 1–20.
- Arora, G., Dubey, A. K., Jaffery, Z. A. and Rocha, A. (2020). “Bag of feature and support vector machine based early diagnosis of skin cancer.” *Neural Computing and Applications*, 1–8.
- Ateniese, G., Blundo, C., De Santis, A. and Stinson, D. R. (1996). “Visual cryptography for general access structures.” *Information and computation*, 129(2), 86–106.
- Ateniese, G., Blundo, C., De Santis, A. and Stinson, D. R. (2001). “Extended capabilities for visual cryptography.” *Theoretical Computer Science*, 250(1-2), 143–161.
- Athar, S. and Wang, Z. (2019). “A comprehensive performance evaluation of image quality assessment algorithms.” *Ieee Access*, 7, 140030–140070.
- Bakshi, A. and Patel, A. K. (2019). “Secure telemedicine using roni halftoned visual cryptography without pixel expansion.” *Journal of Information Security and Applications*, 46, 281–295.
- Banday, S. A. and Pandit, M. K. (2021). “Texture maps and chaotic maps framework for secure medical image transmission.” *Multimedia Tools and Applications*, 1–17.
- Bansal, M., Kumar, M. and Kumar, M. (2021). “2d object recognition: a comparative analysis of sift, surf and orb feature descriptors.” *Multimedia Tools and Applications*, 1–19.
- Bao, C. and Zhang, S. (2020). “Algorithm-based fault tolerance for discrete wavelet transform implemented on gpust1.” *Journal of Systems Architecture*, 101823.
- Bassirian, R., Boreiri, S. and Karimipour, V. (2019). “Computing on quantum shared secrets for general quantum access structures.” *Quantum Information Processing*, 18(4), 109.
- Bharanivendhan, N. and Amitha, T. (2014). “Visual cryptography schemes for secret image sharing using gas algorithm.” *International Journal of Computer Applications*, 92(8).



- Blundo, C., De Santis, A. and Naor, M. (2000). “Visual cryptography for grey level images.” *Information Processing Letters*, 75(6), 255–259.
- Bovik, A. C. (2009). *The essential guide to image processing*, Academic Press.
- Brown, L. G. (1992). “A survey of image registration techniques.” *ACM computing surveys (CSUR)*, 24(4), 325–376.
- Cao, Y., He, Z., Ye, Z., Li, X., Cao, Y. and Yang, J. (2019). “Fast and accurate single image super-resolution via an energy-aware improved deep residual network.” *Signal Processing*, 162, 115–125.
- Chang, C.-C., Lin, C.-C., Tseng, C.-S. and Tai, W.-L. (2007). “Reversible hiding in dct-based compressed images.” *Information Sciences*, 177(13), 2768–2786.
- Chang, C.-H., Yu, X. and Ji, J. X. (2017). “Compressed sensing mri reconstruction from 3d multichannel data using gpus.” *Magnetic resonance in medicine*, 78(6), 2265–2274.
- Chanu, O. B. and Neelima, A. (2019). “A survey paper on secret image sharing schemes.” *International Journal of Multimedia Information Retrieval*, 8(4), 195–215.
- Chen, T.-H. and Tsao, K.-H. (2011a). “Threshold visual secret sharing by random grids.” *Journal of Systems and Software*, 84(7), 1197–1208.
- Chen, T.-H. and Tsao, K.-H. (2011b). “User-friendly random-grid-based visual secret sharing.” *IEEE Transactions on Circuits and Systems for Video Technology*, 21(11), 1693–1703.
- Chougrad, H., Zouaki, H. and Alheyane, O. (2018). “Deep convolutional neural networks for breast cancer screening.” *Computer methods and programs in biomedicine*, 157, 19–30.
- Cristani, M., Cheng, D. S., Murino, V. and Pannullo, D. (2004). “Distilling information with super-resolution for video surveillance.” In *Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, 2–11.

- DArco, P. and De Prisco, R. (2016). “Visual cryptography.” In *International Conference for Information Technology and Communications*, Springer, 20–39.
- De Prisco, R. and De Santis, A. (2014). “On the relation of random grid and deterministic visual cryptography.” *IEEE transactions on information forensics and security*, 9(4), 653–665.
- Deeba, F., Kun, S., Dharejo, F. A. and Zhou, Y. (2020). “Wavelet-based enhanced medical image super resolution.” *IEEE Access*, 8, 37035–37044.
- Deka, B., Datta, S., Mullah, H. U. and Hazarika, S. (2020). “Diffusion-weighted and spectroscopic mri super-resolution using sparse representations.” *Biomedical Signal Processing and Control*, 60, 101941.
- Dhage, S. S., Hegde, S. S., Manikantan, K. and Ramachandran, S. (2015). “Dwt-based feature extraction and radon transform based contrast enhancement for improved iris recognition.” *Procedia Computer Science*, 45, 256–265.
- Elad, M. and Feuer, A. (1997). “Restoration of a single superresolution image from several blurred, noisy, and undersampled measured images.” *IEEE transactions on image processing*, 6(12), 1646–1658.
- Elad, M. and Hel-Or, Y. (2001). “A fast super-resolution reconstruction algorithm for pure translational motion and common space-invariant blur.” *IEEE Transactions on image Processing*, 10(8), 1187–1193.
- Fares, K., Khaldi, A., Redouane, K. and Salah, E. (2021). “Dct & dwt based watermarking scheme for medical information security.” *Biomedical Signal Processing and Control*, 66, 102403.
- Farsiu, S., Robinson, D., Elad, M. and Milanfar, P. (2004a). “Advances and challenges in super-resolution.” *International Journal of Imaging Systems and Technology*, 14(2), 47–57.
- Farsiu, S., Robinson, M. D., Elad, M. and Milanfar, P. (2004b). “Fast and robust multi-frame super resolution.” *IEEE transactions on image processing*, 13(10), 1327–1344.

- Floyd, R. W. (1976). "An adaptive algorithm for spatial gray-scale." In *Proc. Soc. Inf. Disp.*, volume 17, 75–77.
- Freeman, W. T., Jones, T. R. and Pasztor, E. C. (2002). "Example-based super-resolution." *IEEE Computer graphics and Applications*, 22(2), 56–65.
- Georgis, G., Lentaris, G. and Reisis, D. (2019). "Acceleration techniques and evaluation on multi-core cpu, gpu and fpga for image processing and super-resolution." *Journal of Real-Time Image Processing*, 16(4), 1207–1234.
- Gujjunoori, S. and Amberker, B. (2013a). "Busyembed: an hvs based reversible data embedding scheme for video using dct." .
- Gujjunoori, S. and Amberker, B. (2013b). "Dct based reversible data embedding for mpeg-4 video using hvs characteristics." *Journal of information security and applications*, 18(4), 157–166.
- Hardie, R. (2007). "A fast image super-resolution algorithm using an adaptive wiener filter." *IEEE Transactions on Image Processing*, 16(12), 2953–2964.
- Hardie, R. C., Barnard, K. J. and Armstrong, E. E. (1997). "Joint map registration and high-resolution image estimation using a sequence of undersampled images." *IEEE transactions on Image Processing*, 6(12), 1621–1633.
- He, Z., Tang, S., Yang, J., Cao, Y., Yang, M. Y. and Cao, Y. (2018). "Cascaded deep networks with multiple receptive fields for infrared image super-resolution." *IEEE transactions on circuits and systems for video technology*, 29(8), 2310–2322.
- Hemdan, E. E.-D. (2021). "An efficient and robust watermarking approach based on single value decompression, multi-level dwt, and wavelet fusion with scrambled medical images." *Multimedia Tools and Applications*, 80(2), 1749–1777.
- Holla, M. R. and Pais, A. R. (2021a). "An effective gpgpu visual secret sharing by contrast-adaptive convnet super-resolution." *Wireless Personal Communications*, 1–25.

- Holla, M. R. and Pais, A. R. (2021b). “An effective secret image sharing using quantum logic and gpgpu based ednn super-resolution.” *Multimedia Tools and Applications*, 80(6), 9255–9280.
- Holla, M. R. and Pais, A. R. (2021c). “Random grid-based visual cryptography for grayscale and colour images on a many-core system.” In *Computational Vision and Bio-Inspired Computing*, Springer, 287–302.
- Holla, R., Mhala, N. C. and Pais, A. R. (2020). “Gpgpu-based randomized visual secret sharing (grvss) for grayscale and colour images.” *International Journal of Computers and Applications*, 1–9.
- Hou, Y.-C. (2003). “Visual cryptography for color images.” *Pattern recognition*, 36(7), 1619–1629.
- Hou, Y.-C., Lin, C. and Chang, C.-Y. (2001). “Visual cryptography for color images without pixel expansion.” *Journal of Technology*, 16(4), 595–603.
- Hou, Y.-C. and Quan, Z.-Y. (2011). “Progressive visual cryptography with unexpanded shares.” *IEEE transactions on circuits and systems for video technology*, 21(11), 1760–1764.
- Hou, Y.-C., Quan, Z.-Y. and Tsai, C.-F. (2018). “Progressive visual cryptography with friendly and size invariant shares.” *Int. Arab J. Inf. Technol.*, 15(2), 321–330.
- Hou, Y.-C., Quan, Z.-Y., Tsai, C.-F. and Tseng, A.-Y. (2013a). “Block-based progressive visual secret sharing.” *Information Sciences*, 233, 290–304.
- Hou, Y.-C., Wei, S.-C. and Lin, C.-Y. (2013b). “Random-grid-based visual cryptography schemes.” *IEEE Transactions on Circuits and Systems for Video Technology*, 24(5), 733–744.
- Huang, B.-Y. and Juan, J. S.-T. (2020). “Flexible meaningful visual multi-secret sharing scheme by random grids.” *Multimedia Tools and Applications*, 1–25.
- Ibrahim, D. R., Teh, J. S. and Abdullah, R. (2021). “An overview of visual cryptography techniques.” *Multimedia Tools and Applications*, 1–26.

- Inam, O., Qureshi, M., Malik, S. A. and Omer, H. (2017). “Gpu-accelerated self-calibrating grappa operator gridding for rapid reconstruction of non-cartesian mri data.” *Applied Magnetic Resonance*, 48(10), 1055–1074.
- Jesalkumari, J. and Sedamkar, R. (2013). “Modified visual cryptography scheme for colored secret image sharing.” *Int. Journal of Computer Applications Technology and Research*, 2(3), 350–356.
- Joshi, M. V., Chaudhuri, S. and Panuganti, R. (2005). “A learning-based method for image super-resolution from zoomed observations.” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(3), 527–537.
- Jung, C., Ke, P., Sun, Z. and Gu, A. (2018). “A fast deconvolution-based approach for single-image super-resolution with gpu acceleration.” *Journal of Real-Time Image Processing*, 14(2), 501–512.
- Kabirirad, S. and Eslami, Z. (2019). “Improvement of (n, n)-multi-secret image sharing schemes based on boolean operations.” *Journal of Information Security and Applications*, 47, 16–27.
- Kafri, O. and Keren, E. (1987). “Encryption of pictures and shapes by random grids.” *Optics letters*, 12(6), 377–379.
- Kanakkath, P., Madathil, S. and Krishnan, R. (2019). “Deterministic extended visual cryptographic schemes for general access structures with or-and and xor-and operations.” *Multimedia Tools and Applications*, 78(2), 1315–1344.
- Kanso, A. and Ghebleh, M. (2018). “An efficient lossless secret sharing scheme for medical images.” *Journal of Visual Communication and Image Representation*, 56, 245–255.
- Kennedy, J. A., Israel, O., Frenkel, A., Bar-Shalom, R. and Azhari, H. (2006). “Super-resolution in pet imaging.” *IEEE transactions on medical imaging*, 25(2), 137–147.

- Lai, W.-S., Huang, J.-B., Ahuja, N. and Yang, M.-H. (2018). “Fast and accurate image super-resolution with deep laplacian pyramid networks.” *IEEE transactions on pattern analysis and machine intelligence*, 41(11), 2599–2613.
- Lee, K.-H. and Chiu, P.-L. (2011). “An extended visual cryptography algorithm for general access structures.” *IEEE transactions on information forensics and security*, 7(1), 219–229.
- Li, F., Jia, X. and Fraser, D. (2008). “Universal hmt based super resolution for remote sensing images.” In *2008 15th IEEE International Conference on Image Processing*, IEEE, 333–336.
- Li, L., Pan, X., Yang, H., Liu, Z., He, Y., Li, Z., Fan, Y., Cao, Z. and Zhang, L. (2020a). “Multi-task deep learning for fine-grained classification and grading in breast cancer histopathological images.” *Multimedia Tools and Applications*, 79(21), 14509–14528.
- Li, P., Ma, J., Yin, L. and Ma, Q. (2020b). “A construction method of (2, 3) visual cryptography scheme.” *IEEE Access*, 8, 32840–32849.
- Li, P., Yin, L. and Ma, J. (2020c). “Visual cryptography scheme with essential participants.” *Mathematics*, 8(5), 838.
- Li, X., Qin, G., He, Q., Sun, L., Zeng, H., He, Z., Chen, W., Zhen, X. and Zhou, L. (2020d). “Digital breast tomosynthesis versus digital mammography: integration of image modalities enhances deep learning-based breast mass classification.” *European radiology*, 30(2), 778–788.
- Lin, F., Fookes, C., Chandran, V. and Sridharan, S. (2005). “Investigation into optical flow super-resolution for surveillance applications.” In *WDIC 2005: APRS Workshop on Digital Image Computing: Workshop Proceedings*, University of QLD, 73–78.
- Liu, F., Wu, C., Qian, L. et al. (2012). “Improving the visual quality of size invariant visual cryptography scheme.” *Journal of Visual Communication and Image Representation*, 23(2), 331–342.

- Liu, F., Yan, W. Q., Li, P. and Wu, C. (2014). “Essvcs: an enriched secret sharing visual cryptography.” In *Transactions on Data Hiding and Multimedia Security IX*, Springer, 1–24.
- Liu, W., Yin, X., Lu, W., Zhang, J., Zeng, J., Shi, S. and Mao, M. (2020). “Secure halftone image steganography with minimizing the distortion on pair swapping.” *Signal Processing*, 167, 107287.
- Liu, Y.-X., Sun, Q.-D. and Yang, C.-N. (2018). “(k, n) secret image sharing scheme capable of cheating detection.” *EURASIP Journal on Wireless Communications and Networking*, 2018(1), 72.
- Lou, D.-C., Chen, H.-H., Wu, H.-C. and Tsai, C.-S. (2011). “A novel authenticatable color visual secret sharing scheme using non-expanded meaningful shares.” *Displays*, 32(3), 118–134.
- Lowe, D. G. (2004). “Distinctive image features from scale-invariant keypoints.” *International journal of computer vision*, 60(2), 91–110.
- MacPherson, L. (2002). “Grey level visual cryptography for general access structures.” Master’s thesis, University of Waterloo.
- Maintz, J. A. and Viergever, M. A. (1998). “A survey of medical image registration.” *Medical image analysis*, 2(1), 1–36.
- Malczewski, K. and Stasinski, R. (2008). “Toeplitz-based iterative image fusion scheme for mri.” In *2008 15th IEEE International Conference on Image Processing*, IEEE, 341–344.
- Mangal, S., Chaurasia, A. and Khajanchi, A. (2020). “Convolution neural networks for diagnosing colon and lung cancer histopathological images.” *arXiv preprint arXiv:2009.03878*.
- Martin, D., Fowlkes, C., Tal, D. and Malik, J. (2001). “A database of human segmented natural images and its application to evaluating segmentation algorithms and measur-

- ing ecological statistics.” In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, IEEE, 416–423.
- Marwan, M., AlShahwan, F., Sifou, F., Kartit, A. and Ouahmane, H. (2019). “Improving the security of cloud-based medical image storage.” *Engineering Letters*, 27(1).
- Mayya, V. and Nayak, A. (2017). “Parallel implementation of rotation visual cryptography on gpu using cuda.” In *Proceedings of the International Conference on Advances in Image Processing*, ACM, 62–65.
- Mhala, N. C., Jamal, R. and Pais, A. R. (2017). “Randomised visual secret sharing scheme for grey-scale and colour images.” *IET Image Processing*, 12(3), 422–431.
- Mhala, N. C. and Pais, A. R. (2019a). “Contrast enhancement of progressive visual secret sharing (pvss) scheme for gray-scale and color images using super-resolution.” *Signal Processing*, 162, 253–267.
- Mhala, N. C. and Pais, A. R. (2019b). “An improved and secure visual secret sharing (vss) scheme for medical images.” In *2019 11th International Conference on Communication Systems & Networks (COMSNETS)*, IEEE, 823–828.
- Mittal, S. and Vaishay, S. (2019). “A survey of techniques for optimizing deep learning on gpus.” *Journal of Systems Architecture*, 99, 101635.
- Mittal, S. and Vetter, J. S. (2015). “A survey of cpu-gpu heterogeneous computing techniques.” *ACM Computing Surveys (CSUR)*, 47(4), 69.
- Monoth, T. (2019). “Contrast-enhanced recursive visual cryptography scheme based on additional basis matrices.” In *Smart Intelligent Computing and Applications*, Springer, 179–187.
- Moustafa, M., Ebeid, H. M., Helmy, A., Nazmy, T. M. and Tolba, M. F. (2016). “Rapid real-time generation of super-resolution hyperspectral images through compressive sensing and gpu.” *International Journal of Remote Sensing*, 37(18), 4201–4224.



- Nam, S., Akçakaya, M., Basha, T., Stehning, C., Manning, W. J., Tarokh, V. and Nezafat, R. (2013). “Compressed sensing reconstruction for whole-heart imaging with 3d radial trajectories: a graphics processing unit implementation.” *Magnetic resonance in medicine*, 69(1), 91–102.
- Naor, M. and Shamir, A. (1994). “Visual cryptography.” In *Workshop on the Theory and Application of Cryptographic Techniques*, Springer, 1–12.
- Naphade, M. R. A. and Khobaragade, D. R. N. (2016). “Multiple share images using random grids and xor-based visual cryptography.” .
- Noyum, V. D., Mofenjou, Y. P., Feudjio, C., Göktug, A. and Fokoué, E. (2021). “Boosting the predictive accuracy of singer identification using discrete wavelet transform for feature extraction.” *arXiv preprint arXiv:2102.00550*.
- Padiya, I., Manure, V. and Vidhate, A. (2015). “Visual secret sharing scheme using encrypting multiple images.” *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 4(1).
- Pandey, D., Rawat, U., Rathore, N. K., Pandey, K. and Shukla, P. K. (2020). “Distributed biomedical scheme for controlled recovery of medical encrypted images.” *IRBM*.
- Park, S. C., Park, M. K. and Kang, M. G. (2003). “Super-resolution image reconstruction: a technical overview.” *IEEE signal processing magazine*, 20(3), 21–36.
- Peled, S. and Yeshurun, Y. (2001). “Superresolution in mri: application to human white matter fiber tract visualization by diffusion tensor imaging.” *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 45(1), 29–35.
- Pham, T. Q., Van Vliet, L. J. and Schutte, K. (2006). “Robust fusion of irregularly sampled data using adaptive normalized convolution.” *EURASIP Journal on Advances in Signal Processing*, 2006(1), 083268.

- Protter, M. and Elad, M. (2009). “Super resolution with probabilistic motion estimation.” *IEEE Transactions on Image Processing*, 18(8), 1899–1904.
- Punithavathi, P. and Geetha, S. (2017). “Visual cryptography: A brief survey.” *Information Security Journal: A Global Perspective*, 26(6), 305–317.
- Qin, J., Huang, Y. and Wen, W. (2020). “Multi-scale feature fusion residual network for single image super-resolution.” *Neurocomputing*, 379, 334–342.
- Qiu, D., Zhang, S., Liu, Y., Zhu, J. and Zheng, L. (2020). “Super-resolution reconstruction of knee magnetic resonance imaging based on deep learning.” *Computer methods and programs in biomedicine*, 187, 105059.
- Qiu, D., Zheng, L., Zhu, J. and Huang, D. (2021). “Multiple improved residual networks for medical image super-resolution.” *Future Generation Computer Systems*, 116, 200–208.
- Rachapudi, V. and Devi, G. L. (2020). “Improved convolutional neural network based histopathological image classification.” *Evolutionary Intelligence*, 1–7.
- Rajashekhar, U., Neelappa, D. and Rajesh, L. (2021). “Electroencephalogram (eeg) signal classification for brain–computer interface using discrete wavelet transform (dwt).” *International Journal of Intelligent Unmanned Systems*.
- Robinson, D. and Milanfar, P. (2006). “Statistical performance analysis of super-resolution.” *IEEE Transactions on Image Processing*, 15(6), 1413–1428.
- Sabbagh, M., Uecker, M., Powell, A. J., Leeser, M. and Moghari, M. H. (2016). “Cardiac mri compressed sensing image reconstruction with a graphics processing unit.” In *2016 10th International Symposium on Medical Information and Communication Technology (ISMICT)*, IEEE, 1–5.
- Sah, H. R., Gunasekaran, G. and Parthiban, L. (2018). “A novel privacy preserving visual cryptography based scheme for telemedicine applications..” *Biomedical Research (0970-938X)*.

- Samala, R. K., Chan, H.-P., Hadjiiski, L., Helvie, M. A., Richter, C. D. and Cha, K. H. (2018). “Breast cancer diagnosis in digital breast tomosynthesis: effects of training sample size on multi-stage transfer learning using deep neural nets.” *IEEE transactions on medical imaging*, 38(3), 686–696.
- Sarosh, P., Parah, S. A. and Bhat, G. (2021). “Utilization of secret sharing technology for secure communication: a state-of-the-art review.” *Multimedia Tools and Applications*, 80(1), 517–541.
- Selvi, C. T., Amudha, J. and Sudhakar, R. (2021). “A modified salp swarm algorithm (ssa) combined with a chaotic coupled map lattices (cml) approach for the secured encryption and compression of medical images during data transmission.” *Biomedical Signal Processing and Control*, 66, 102465.
- Sharma, R. G., Dimri, P. and Garg, H. (2018). “Visual cryptographic techniques for secret image sharing: a review.” *Information Security Journal: A Global Perspective*, 27(5-6), 241–259.
- Shivani, S. (2018). “Vmvc: Verifiable multi-tone visual cryptography.” *Multimedia Tools and Applications*, 77(5), 5169–5188.
- Shyu, S. J. (2009). “Image encryption by multiple random grids.” *Pattern Recognition*, 42(7), 1582–1596.
- Smith, D. S., Gore, J. C., Yankeelov, T. E. and Welch, E. B. (2012). “Real-time compressive sensing mri reconstruction using gpu computing and split bregman methods.” *International journal of biomedical imaging*, 2012.
- Sridhar, S. and Sudha, G. F. (2020). “Quality improved (k, n) priority based progressive visual secret sharing.” *Multimedia Tools and Applications*, 1–28.
- Srividhya, S., Jayasree, J. and Sudha, G. F. (2019). “Error diffusion with varying threshold halftoning for enhancing contrast of color images.” In *Innovations in Computer Science and Engineering*, Springer, 289–298.

- Srujana, O. S., Mhala, N. C. and Pais, A. R. “Secure transmission of hyperspectral images.” In *2020 Third ISEA Conference on Security and Privacy (ISEA-ISAP)*, IEEE, 94–99.
- Sun, W., Tseng, T.-L. B., Zhang, J. and Qian, W. (2017). “Enhancing deep convolutional neural network scheme for breast cancer diagnosis with unlabeled data.” *Computerized Medical Imaging and Graphics*, 57, 4–9.
- Tsai, R. (1984). “Multiframe image restoration and registration.” *Advance Computer Visual and Image Processing*, 1, 317–339.
- Ulutas, M. (2010). “Meaningful share generation for increased number of secrets in visual secret-sharing scheme.” *Mathematical Problems in Engineering*, 2010.
- Vo, D. M., Nguyen, N.-Q. and Lee, S.-W. (2019). “Classification of breast cancer histology images using incremental boosting convolution networks.” *Information Sciences*, 482, 123–138.
- Wang, H., Peng, H., Chang, Y. and Liang, D. (2018). “A survey of gpu-based acceleration techniques in mri reconstructions.” *Quantitative imaging in medicine and surgery*, 8(2), 196.
- Wu, K., Inoue, K. and Hara, K. (2020). “Neugebauer models for color error diffusion halftoning.” *Journal of Imaging*, 6(4), 23.
- Wu, X. and Yang, C.-N. (2020). “Probabilistic color visual cryptography schemes for black and white secret images.” *Journal of Visual Communication and Image Representation*, 102793.
- Xiong, L., Zhong, X. and Yang, C.-N. (2020). “Dwt-sisa: a secure and effective discrete wavelet transform-based secret image sharing with authentication.” *Signal Processing*, 107571.
- Xiong, Z., Orchard, M. T. and Ramchandran, K. (1999). “Inverse halftoning using wavelets.” *IEEE transactions on image processing*, 8(10), 1479–1483.

- Yamaguchi, Y. (2014). “Extended visual cryptography scheme for multiple-secrets continuous-tone images.” In *Transactions on Data Hiding and Multimedia Security IX*, Springer, 25–41.
- Yamanaka, J., Kuwashima, S. and Kurita, T. (2017). “Fast and accurate image super resolution by deep cnn with skip connection and network in network.” In *International Conference on Neural Information Processing*, Springer, 217–225.
- Yan, B., Wang, Y.-F., Song, L.-Y. and Yang, H.-M. (2016). “Size-invariant extended visual cryptography with embedded watermark based on error diffusion.” *Multimedia Tools and Applications*, 75(18), 11157–11180.
- Yan, B., Xiang, Y. and Hua, G. (2020a). “Improving visual quality for probabilistic and random grid schemes.” In *Improving Image Quality in Visual Cryptography*, Springer, 75–95.
- Yan, X., Liu, X. and Yang, C.-N. (2018). “An enhanced threshold visual secret sharing based on random grids.” *Journal of real-time image processing*, 14(1), 61–73.
- Yan, X. and Lu, Y. (2019). “Generalized general access structure in secret image sharing.” *Journal of Visual Communication and Image Representation*, 58, 89–101.
- Yan, X., Lu, Y. and Liu, L. (2020b). “A common general access structure construction approach in secret image sharing.” *International Journal of Digital Crime and Forensics (IJDCF)*, 12(3), 96–110.
- Yan, X., Lu, Y., Liu, L., Wan, S., Ding, W. and Liu, H. (2020c). “Exploiting the homomorphic property of visual cryptography.” In *Cryptography: Breakthroughs in Research and Practice*, IGI Global, 416–427.
- Yang, A., Yang, B., Ji, Z., Pang, Y. and Shao, L. (2020). “Lightweight group convolutional network for single image super-resolution.” *Information Sciences*, 516, 220–233.

- Yang, J., Wright, J., Huang, T. and Ma, Y. (2008). "Image super-resolution as sparse representation of raw image patches." In *2008 IEEE conference on computer vision and pattern recognition*, IEEE, 1–8.
- Yang, Y., Xiang, P., Mantor, M. and Zhou, H. (2012). "Cpu-assisted gpgpu on fused cpu-gpu architectures." In *IEEE International Symposium on High-Performance Comp Architecture*, IEEE, 1–12.
- Yuan, Y., Yang, X., Wu, W., Li, H., Liu, Y. and Liu, K. (2019). "A fast single-image super-resolution method implemented with cuda." *Journal of Real-Time Image Processing*, 16(1), 81–97.
- Yurttakal, A. H., Hasan, E., Türkan, İ. and Seyhan, K. (2020). "Detection of breast cancer via deep convolution neural networks using mri images." *Multimedia Tools and Applications*, 79(21-22), 15555–15573.
- Zeller, C. (2011). "Cuda c/c++ basics." *NVIDIA Coporation*.
- Zeng, K., Ding, S. and Jia, W. (2019). "Single image super-resolution using a polymorphic parallel cnn." *Applied Intelligence*, 49(1), 292–300.
- Zhang, B., Rahmatullah, B., Wang, S. L., Zaidan, A., Zaidan, B. and Liu, P. (2020). "A review of research on medical image confidentiality related technology coherent taxonomy, motivations, open challenges and recommendations." *Multimedia Tools and Applications*, 1–40.
- Zhou, Z., Arce, G. R. and Di Crescenzo, G. (2006). "Halftone visual cryptography." *IEEE transactions on image processing*, 15(8), 2441–2453.
- Zitova, B. and Flusser, J. (2003). "Image registration methods: a survey." *Image and vision computing*, 21(11), 977–1000.

## RESEARCH OUTCOMES

1. Holla, Raviraja and Mhala, Nikhil C and Pais, Alwyn R (2020). GPGPU-based randomized visual secret sharing (GRVSS) for grayscale and colour images. *International Journal of Computers and Applications (Taylor & Francis)*, 1-9. [DOI: <https://doi.org/10.1080/1206212X.2020.1830246>]
2. Holla, Raviraja and Pais, Alwyn R (2020). An effective secret image sharing using quantum logic and GPGPU based EDNN super-resolution. *Multimedia Tools and Applications (Springer)*, 80(6), 9255 - 9280. [DOI: <https://doi.org/10.1007/s11042-020-10065-7>]
3. Holla, Raviraja and Pais, Alwyn R and Suma, D (2021). An Accelerator-based Logistic Map Image Cryptosystems for Grayscale Images. *Journal of Cyber Security and Mobility (River Publishers)*, 10(3), 487-510. [DOI: <https://doi.org/10.13052/jcsm2245-1439.1031>]
4. Holla, Raviraja and Pais, Alwyn R (2022). An effective GPGPU visual secret sharing by contrast-adaptive ConvNet super-resolution. *Wireless Personal Communications (Springer)*, 123(3), 2367-2391. [DOI: <https://doi.org/10.1007/s11277-021-09245-x>]
5. Holla, Raviraja and Pais, Alwyn R (2022). High-performance medical image secret sharing using super-resolution for CAD systems. *Applied Intelligence. (Springer)*, 1-17. [DOI: <https://doi.org/10.1007/s10489-021-03095-7>]
6. Holla, Raviraja and Pais, Alwyn R (2021). Random Grid-based Visual Cryptography for Grayscale and Colour Images on a Many-core System. *In Compu-*

## *BIBLIOGRAPHY*

---

*tational Vision and Bio-Inspired Computing. (Springer), 1318, 287-302. [DOI:  
[https://doi.org/10.1007/978-981-33-6862-0\\_25](https://doi.org/10.1007/978-981-33-6862-0_25)]*



## BIO-DATA

Name: Raviraja Holla M  
Date of Birth: 16/03/1971  
Gender: Male  
Marital Status: Married  
Father's Name: Venkataramana Holla M  
Mother's Name: Susheela A  
Email Id: hollamr.v@gmail.com  
Present Address: "Hemadri", Door No. 16-181R, 2nd Cross,  
ALN Layout, Manipal, Udupi District and  
Taluk, Karnataka - 576104  
Educational Qualifications: B.E (CSE) - R.V. Engineering College, Ben-  
galuru,  
M.Tech (Information Technology) - KSOU,  
Mysore  
Areas of Interest: Information Security, High Performance  
Computing, Semantic Web.  
Mobile No.: (+91) 9480570768