

**AN INTELLIGENT SYSTEM FOR THE  
CLASSIFICATION AND SELECTION OF  
NOVEL AND EFFICIENT LOSSLESS IMAGE  
COMPRESSION ALGORITHMS**

**Mahmoud AL Qerom**

**Thesis Submitted in Partial Fulfilment of the Requirements of the Degree  
of Doctor of Philosophy**

**School of Computing, Science and Engineering  
University of Salford, Salford, UK**

**2020**

**Supervisor**

**Prof. Farid Meziane**

## Abstract

We are currently living in an era revolutionised by the development of smart phones and digital cameras. Most people are using phones and cameras in every aspect of their lives. With this development comes a high level of competition between the technology companies developing these devices, each one trying to enhance its products to meet the new market demands. One of the most sought-after criteria of any smart phone or digital camera is the camera's resolution. Digital imaging and its applications are growing rapidly; as a result of this growth, the image size is increasing, and alongside this increase comes the important challenge of saving these large-sized images and transferring them over networks. With the increase in image size, the interest in image compression is increasing as well, to improve the storage size and transfer time.

In this study, the researcher proposes two new lossless image compression algorithms. Both proposed algorithms focus on decreasing the image size by reducing the image bit-depth through using well defined methods of reducing the coloration between the image intensities.

The first proposed lossless image compression algorithm is called Column Subtraction Compression (CSC), which aims to decrease the image size without losing any of the image information by using a colour transformation method as a pre-processing phase, followed by the proposed Column Subtraction Compression function to decrease the image size. The proposed algorithm is specially designed for compressing natural images. The CSC algorithm was evaluated for colour images and compared against benchmark schemes obtained from (Khan *et al.*, 2017). It achieved the best compression size over the existing methods by enhancing the average storage saving of the BBWCA, JPEG 2000 LS, KMTF– BWCA, HEVC and basic BWCA algorithms by 2.5%, 15.6%, 41.6%, 7.8% and 45.07% respectively. The CSC algorithm simple implementation positively affects the execution time and makes it one of the fastest algorithms, since it needed less than 0.5 second for compressing and decompressing natural images obtained from (Khan *et al.*, 2017). The proposed algorithm needs only 19.36 seconds for compressing and decompressing all of the 10 images from the Kodak image set, while the BWCA, KMTF – BWCA and BBWCA need 398.5s, 429.24s and 475.38s respectively. Nevertheless, the CSC algorithm achieved less compression ratio, when compressing low resolution images since it was designed for compressing high resolution images. To solve this issue, the researcher proposed the Low-Resolution Column Subtraction

Compression algorithm (LRCSC) to enhance the CSC low compression ratio related to compressing low-resolution images.

The LRCSC algorithm starts by using the CSC algorithm as a pre-processing phase, followed by the Huffman algorithm and Run-Length Coding (RLE) to decrease the image size as a final compression phase. The LRCSC enhanced the average storage saving of the CSC algorithm for raster map images by achieving 13.68% better compression size. The LRCSC algorithm decreases the raster map image set size by saving 96% from the original image set size but did not reach the best results when compared with the PNG, GIF, BLiSE and BBWCA where the storage saving is 97.42%, 98.33%, 98.92% and 98.93% respectively. The LRCSC algorithm enhanced the compression execution time with acceptable compression ratio.

Both of the proposed algorithms are effective with any image types such as colour or greyscale images. The proposed algorithms save a lot of memory storage and dramatically decreased the execution time.

Finally, to take full benefits of the two newly developed algorithms, a new system is developed based on running both of the algorithm for the same input image and then suggest the appropriate algorithm to be used for the de-compression phase.

**Keywords:** Image Compression, Image Classification, Lossless Image Compression Techniques, Column Subtraction Compression (CSC), Low-Resolution Column Subtraction Compression (LRCSC), Deep Convolutional Neural Networks (DCNN), Artificial Intelligence (AI), Image Classification.

## **Acknowledgements**

My deep thanks and gratitude for my main supervisor Professor. Farid Meziane for all the guidance and encouragement he provided during the time of developing this work, and for his contribution in making this work successes. Your support and advice are invaluable. I also wish to express my deep gratitude to Professor. Mohammad Otair for his efforts in developing this work and for his helpful contributions and guidance during the last three years.

Furthermore, many thanks go to my family; especially to my sweet mother for all her supports and encouragement and for all her prayers- thank you mum. For my brothers and sisters my darling wife and my lovely daughters and dear sons; I wish to thank you all for your encouragements, your supports and for the lovely time we had together. Finally, many thanks to my friends for all of their supports.

## **Dedication**

This work is dedicated to my late Father; Mr. Mohammed Al-Qerom. The beautiful 24 years that we had together looks like 24 minutes. During this time, I remember one of your wishes of obtaining a Ph.D. degree for me; that's why I dedicated this work for your pure soul.

# Table of Contents

<b>ABSTRACT</b> .....	<b>1</b>
<b>CHAPTER ONE: INTRODUCTION AND MOTIVATION</b> .....	<b>16</b>
<b>OVERVIEW</b> .....	<b>16</b>
1.1 INTRODUCTION AND MOTIVATION.....	16
1.2 RESEARCH AIM AND OBJECTIVES .....	17
1.3 RESEARCH METHODOLOGY .....	17
1.3.1 Design Science Research in Information Systems.....	18
1.3.2 Philosophical Ground of DSR .....	19
1.3.3 DSR Process Model.....	20
1.3.4 DSR Knowledge Contribution Framework.....	22
1.4 RESEARCH CONTRIBUTIONS TO KNOWLEDGE .....	23
1.5 THESIS STRUCTURE .....	24
<b>CHAPTER TWO: RESEARCH BACKGROUND AND LITERATURE REVIEW</b> .....	<b>25</b>
<b>2 CHAPTER OVERVIEW</b> .....	<b>25</b>
2.1 LITERATURE REVIEW .....	25
2.1.1 Digital Images.....	26
2.1.2 The Most Popular Used Image Formats .....	27
2.1.3 Digital Image Processing .....	28
2.1.4 Operations Performed on Digital Images.....	29
2.1.4.1 Image Segmentation .....	29
2.1.4.2 Image Compression .....	30
2.1.5 General Image Compression Model .....	31
2.1.6 Description of the Processes Used in the Current Techniques.....	33
2.1.6.1 Correlation .....	33
2.1.6.2 Quantization .....	34
2.1.6.3 Entropy Coding.....	34
2.1.7 Measurement of Image Size and Quality.....	35
2.1.8 Classification of Compression Techniques.....	37
2.2 LOSSLESS COMPRESSION TECHNIQUES.....	38
2.2.1 Lossless Compression Techniques Phases.....	38
2.2.2 Lossless Compression Algorithms.....	39
2.2.3 Lossless Compression Related Work.....	40
2.2.3.1 Huffman Coding .....	40
2.2.3.2 Shannon's Coding.....	41
2.2.3.3 Bit Plane Slicing (BPS).....	41
2.2.3.4 Different Plus Coding Modulation Followed by Huffman .....	41
2.2.3.5 Improved Lempel-Ziv-Welch.....	41
2.2.3.6 Lempel-Ziv-Welch with Region of Interest.....	42
2.2.3.7 Run Length Coding .....	42
2.2.3.8 Arithmetic Coding .....	43
2.2.3.9 Median Edge Detection.....	44
2.2.3.10 Median Edge Detection and Different Plus Coding Modulation .....	44
2.2.3.11 Median Edge Detection and Activity Level Classification Model .....	44
2.2.3.12 BBWCA.....	44
2.3 LOSSY COMPRESSION TECHNIQUES.....	45
2.3.1 Lossy Compression Techniques Phases.....	45
2.3.2 Lossy Compression Algorithms.....	46
2.3.3 Lossy Compression Related Work.....	47
2.3.3.1 JPEG .....	47
2.3.3.2 JPEG 2000.....	48

2.3.3.3	Minimise-Matrix-Size algorithm.....	49
2.3.3.4	Discrete Cosine Transform Followed by Huffman.....	49
2.3.3.5	Different Plus Coding Modulation, Discrete Wavelet Transform Followed by Huffman.....	49
2.3.3.6	Radial Basis Function Neural Networks and Discrete Wavelet Transform.....	50
2.3.3.7	Discrete Wavelet Transform and Set Partitioning in Hierarchical Coding Techniques.....	50
2.3.3.8	Rounding the Intensity Followed by Dividing.....	50
2.3.3.9	Fractal Compression .....	51
2.3.3.10	EZW.....	51
2.4	SUMMARY AND COMPARISON OF FUNDAMENTAL ALGORITHMS.....	52
2.5	CHAPTER SUMMARY .....	54
<b>CHAPTER THREE: PROPOSED SOLUTIONS AND DEVELOPMENT TOOLS .....</b>		<b>55</b>
<b>3</b>	<b>CHAPTER OVERVIEW .....</b>	<b>55</b>
3.1	PROPOSED SOLUTIONS.....	55
3.1.1	Huffman Limitation and the Proposed Solution .....	56
3.1.2	LZW Limitation and the Proposed Solution .....	56
3.1.3	RLC Limitation and the Proposed Solution.....	56
3.2	SOFTWARE DEVELOPMENT PROCESS.....	57
3.2.1	The first Iteration Requirements .....	58
3.2.2	The Second Iteration Requirements.....	59
3.2.3	The third Iteration Requirements .....	60
3.3	ALGORITHM COMPLEXITY.....	61
3.4	MATLAB.....	62
3.5	USED DEVICES AND OPERATING SYSTEM.....	63
3.6	SET OF TESTED IMAGES .....	63
3.6.1	Image Set 1 .....	64
3.6.2	Image Set 2 .....	65
3.6.3	Image Set 3 .....	66
3.6.4	Image Set 4 .....	67
3.6.5	Image Set 5 .....	68
3.7	RATIONAL FOR THE SELECTION OF THE IMAGES SETS .....	68
3.8	CHARACTERISTICS OF THE SELECTED IMAGES.....	69
3.9	CHAPTER SUMMARY .....	69
<b>CHAPTER FOUR: THE PROPOSED LOSSLESS ALGORITHM FOR NATURAL IMAGES COMPRESSION .....</b>		<b>70</b>
<b>4</b>	<b>CHAPTER OVERVIEW .....</b>	<b>70</b>
4.1	INTRODUCING THE CSC COMPRESSION ALGORITHM .....	70
4.2	THE CSC FLOWCHART.....	73
4.3	DESCRIPTION OF THE CSC ALGORITHM.....	73
4.3.1	Colour Transformation: .....	74
4.3.2	Column Subtraction .....	77
4.4	CSC ALGORITHM TIME COMPLEXITY ANALYSES.....	80
4.4.1	CSC algorithm Time Complexity .....	80
4.4.2	Proof .....	82
4.5	VALIDATION AND TESTING .....	82
4.5.1	The CSC Algorithm Compression Size .....	83
4.5.2	The CSC Algorithm Image Quality.....	86
4.5.3	The CSC Algorithm Execution Time.....	87
4.6	EVALUATIONS, RESULTS AND OBSERVATIONS.....	90
4.6.1	Comparison Between the CSC Results and Huffman Algorithm Results.....	90
4.6.1.1	Comparison Between the CSC Algorithm and Huffman Algorithm in Terms of Image Size .....	90
4.6.1.2	Comparison Between the CSC Algorithm and Huffman in Terms of Image Quality.....	94
4.6.1.3	Comparison Between the CSC and Huffman in Terms of Execution Time .....	94
4.6.2	Comparison Between the CSC Compression Size and Other State of the Art Algorithm. ....	96

4.6.2.1	First Comparison for Natural Images Compression Size .....	97
4.6.2.2	Second Comparison for Natural Images Compression Size.....	98
4.6.2.3	Third Comparison for Raster Map Images Size .....	99
4.6.3	Comparison Between the CSC Execution Time and Other State of the Art Algorithms .....	101
4.7	CHAPTER SUMMARY .....	102
<b>CHAPTER FIVE: THE PROPOSED LOSSLESS ALGORITHM FOR RASTER MAP IMAGES COMPRESSION .....</b>		<b>104</b>
<b>5</b>	<b>CHAPTER OVERVIEW .....</b>	<b>104</b>
5.1	INTRODUCING THE LRCSC COMPRESSION ALGORITHM .....	104
5.2	THE LRCSC FLOWCHART .....	109
5.3	DESCRIPTION OF THE LRCSC ALGORITHM.....	110
5.3.1	CSC Algorithm: .....	110
5.3.2	Negative Value Removing .....	113
5.3.3	Huffman Algorithm .....	115
5.3.4	Negative Value Restoration .....	116
5.3.5	RLE.....	120
5.4	LRCSC TIME COMPLEXITY .....	121
5.4.1	LRCSC algorithm Time Complexity Analyses .....	121
5.4.2	Proof .....	124
5.5	VALIDATION AND TESTING .....	124
5.5.1	The LRCSC Algorithm Compression Size .....	125
5.5.2	The LRCSC Algorithm Image Quality. ....	128
5.5.3	The LRCSC Algorithm Execution Time. ....	130
5.6	EVALUATIONS, RESULTS AND OBSERVATIONS.....	133
5.6.1	Comparison Between the LRCSC Results and Huffman Algorithm Results.....	133
5.6.1.1	Comparison Between the LRCSC Algorithm and Huffman Algorithm in Terms of Image Size .....	133
5.6.1.2	Comparison Between the LRCSC Algorithm and Huffman in Terms of Image Quality .....	136
5.6.1.3	Comparison Between the LRCSC and Huffman in Terms of Execution Time .....	136
5.6.2	Comparison Between the LRCSC Compression Size and Other State of the Art Algorithm. ....	139
5.6.2.1	First Comparison for Natural Images Compression Size .....	139
5.6.2.2	Second Comparison for Natural Images Compression Size .....	140
5.6.2.3	Third Compression for Raster Map Images Size.....	142
5.6.3	Comparison Between the LRCSC Execution Time and Other State of the Art Algorithm. ....	144
5.7	CHAPTER SUMMARY .....	145
<b>CHAPTER SIX: AUTOMATED SYSTEM FOR IMAGE COMPRESSION .....</b>		<b>146</b>
<b>6</b>	<b>CHAPTER OVERVIEW .....</b>	<b>146</b>
6.1	INTRODUCTION .....	146
6.2	THE FULLY AUTOMATED SYSTEM COMPRESSION SIZE.....	147
6.3	THE FULLY AUTOMATED SYSTEM EXECUTION TIME.....	147
6.4	THE FULLY AUTOMATED SYSTEM IMAGE QUALITY .....	148
6.5	CHAPTER SUMMARY .....	150
<b>CHAPTER SEVEN: CONCLUSIONS AND FUTURE WORK .....</b>		<b>151</b>
<b>7</b>	<b>CHAPTER OVERVIEW .....</b>	<b>151</b>
7.1	RESEARCH SUMMARY .....	151
7.2	RESEARCH CONTRIBUTIONS AND REVIEW OF THE RESEARCH OBJECTIVES.....	152
7.2.1	Lossless and Lossy Image Compression State-of-the-Art .....	152
7.2.2	Addressing the Challenges of Lossless Image Compression Techniques.....	152
7.2.3	Developing a Lossless Image Compression Algorithm for Natural Images .....	152
7.2.4	Developing a Lossless Image Compression Algorithm for Synthetic Images .....	153
7.2.5	Develop a Fully Automated System for Choosing the Suitable Algorithm for Compressing the Images Regarding its Type. ....	153
7.3	RESEARCH LIMITATIONS .....	154



7.4	RECOMMENDATIONS FOR FUTURE WORK .....	154
7.4.1	The Proposed AI Algorithm (GF-FSAE) for Image Classification .....	155
7.4.2	Artificial Intelligence .....	156
7.4.3	The Chosen Artificial Intelligence Technique for Image Classification .....	157
<b>REFERENCES.....</b>		<b>158</b>
<b>APPENDIX A LRCSC COMPRESSION .....</b>		<b>167</b>
<b>APPENDIX B LRCSC DE-COMPRESSION .....</b>		<b>170</b>
<b>APPENDIX C HUFFMAN CODING FUNCTION .....</b>		<b>172</b>
<b>APPENDIX D RLE CODING FUNCTION .....</b>		<b>173</b>

## List of Figures

Figure 1. 1 - Levels of Research in Information Systems and DSR Role .....	18
Figure 1. 2 - DSR Process Model (Adapted from (Vijay, Bill and Stacie, 2015) .....	20
Figure 1. 3 - DSR Knowledge Contribution Framework. ....	23
Figure 2. 1 - Digital image (value and bit representing). Obtained From (Abdalla and Osman, 2016).26	
Figure 2. 2 - Colour Space Conversion from RGB to YCbCr. Adapted From (Kuppusamy and Mehala, 2013) .....	30
Figure 2. 3 - General Data Compression Scheme adapted by (Gupta, Bansal and Khanduja, 2017). ..	30
Figure 2. 4 - General Compression and Decompression Model .....	31
Figure 2. 5 - Lossless Compression Model .....	38
Figure 2. 6 - Lossless Compression Techniques and Algorithms Adaptive from (Zaineldin, Elhosseini and Ali, 2015) .....	39
Figure 2. 7 - Lossy Compression Model .....	45
Figure 2. 8 - Lossy Compression Techniques and Algorithms Adaptive from (Zaineldin, Elhosseini and Ali, 2015) .....	46
Figure 3. 1 - The Compression System Class Diagram .....	63
Figure 4. 1 - CSC Lossless Algorithm Flowchart.....	73
Figure 4. 2 - Transformation Example.....	76
Figure 4. 3 - Invers Transformation Example.....	77
Figure 4. 4 - CSC Example.....	79
Figure 4. 5 - CSC Decompression Example.....	80
Figure 4. 6 - The CSC Algorithm Storage saving .....	85
Figure 4. 7 - Compression and Decompression Time for the CSC Algorithm.....	90
Figure 4. 8 - The Average Compression Size for the CSC and Huffman algorithm .....	93
Figure 4. 9 - The Average Execution Time for Both Algorithms.....	96
Figure 4. 10 - Compression Ratio for the Five Algorithm.....	98
Figure 4. 11 - Total Compression Size for the Kodak Image Set .....	99
Figure 4. 12 - Average Compression Ratio for the Raster Map Image Set .....	101
Figure 4. 13 - Total Execution Time for the Kodak Image Set.....	102
Figure 5. 1 - LRCSC Lossless Algorithm Flowchart.....	109
Figure 5. 2 - Transformation Example.....	111
Figure 5. 3 - Invers Transformation Example.....	111
Figure 5. 4 - CSC Example.....	112
Figure 5. 5 - CSC Decompression Example.....	113
Figure 5. 6 - Negative Values Temporary Dictionary File.....	114
Figure 5. 7 - Positive Values Matrices .....	114
Figure 5. 8 - Huffman Dictionary Files.....	115
Figure 5. 9 - Huffman Results.....	116
Figure 5. 10 - Restoring the Negative Values.....	117
Figure 5. 11 - Negative Values Temporary Dictionary File.....	117
Figure 5. 12 - Positive Values matrices .....	118
Figure 5. 13 - Huffman Decompression Results.....	119
Figure 5. 14 - Huffman Positive Values matrices .....	119
Figure 5. 15 - RLE Results .....	120
Figure 5. 16 - The LRCSC Algorithm Storage Saving.....	127
Figure 5. 17 - The Average MSE Results for the Lossless LRCSC Algorithm .....	128

Figure 5. 18 - Compression and Decompression Time for the LRCSC Algorithm.....	133
Figure 5. 19 - The Average Compression size for the LRCSC and Huffman algorithm .....	136
Figure 5. 20 - The Average Execution Time for Both Algorithms .....	139
Figure 5. 21 - Compression Ratio for the Five Algorithm.....	140
Figure 5. 22 - Total Compression Size for the Kodak Image Set .....	142
Figure 5. 23 - LRCSC Average Compression Ratio for the Raster Map Image Set.....	143
Figure 5. 24 - Total Execution Time for the Kodak Image Set.....	145
Figure 7. 1 - The Fully Automated AI system Flowchart .....	155
Figure 7. 2 - Relationship Between ML and Deep Learning (DL) (Edureka, 2019).....	156

## List of Tables

Table 1. 1- Design Science Research Perspective. (Source: (Vijay, Bill and Stacie, 2015)).....	19
Table 1. 2 - DSR Process Steps and the Corresponding chapters in this research.....	22
Table 2. 1 - Summary of Lossless algorithms. ....	52
Table 3. 1 - The Algorithms Working Environments .....	63
Table 3. 2 - Image set 1 .....	64
Table 3. 3 - Image Set 2.....	65
Table 3. 4 - Image Set 3.....	66
Table 3. 5 - Image Set 4.....	67
Table 3. 6 - Image Set 5.....	68
Table 4. 1 - R,Dg,Db Transformation and Invers Transformation Equations .....	74
Table 4. 2 - The Modified Transformation and Invers Transformation Equations .....	75
Table 4. 3 - The Average Compression Ratio for the Original and Modified Transformation .....	75
Table 4. 4 - CSC Algorithm Complexity .....	81
Table 4. 5 - The Lossless CSC Algorithm Compression Size.....	83
Table 4. 6 - The Lossless CSC Algorithm Average Compression Size.....	85
Table 4. 7 - Sample Images from the Five Image Sets Before and After Compression .....	86
Table 4. 8 - The Lossless CSC Algorithm Compression Time in Seconds.....	88
Table 4. 9 - The Lossless CSC Algorithm Average Compression Time.....	89
Table 4. 10 - The Lossless CSC Compression Size and Huffman Compression Size .....	91
Table 4. 11 - The Average Lossless Approach Compression Size with Huffman Average Compression Size .....	93
Table 4. 12 - The Lossless CSC Algorithm Execution Time with Huffman Execution Time .....	94
Table 4. 13 - The Average Execution Time for Both Algorithms in Seconds.....	96
Table 4. 14 - The Proposed Algorithm Results Compared with Other Four Algorithm Results.....	97
Table 4. 15 - Comparison Between the CSC and various benchmark systems in Term of Compressed File Sizes of Kodak Colour Test Images (size in KBs) .....	98
Table 4. 16 - The CSC Results Compared with Other Four Algorithm Results .....	100
Table 4. 17 - System Requirements .....	101
Table 4. 18 - Execution Time in Seconds for the Kodak Image Set for Different Algorithms .....	101
Table 5.1 - LRCSC Algorithm Complexity.....	121
Table 5.2 - The Lossless LRCSC Algorithm Compression Size.....	125
Table 5.3 - The Lossless LRCSC Algorithm Average Compression Size.....	127
Table 5.4 - Image Quality Average Results for the Lossless LRCSC Algorithm.....	128
Table 5.5 - Sample Images for the Five Image Sets Before and After Compression by LRCSC .....	129
Table 5.6 - The LRCSC Algorithm Compression Time in Seconds.....	130
Table 5.7 - The LRCSC Algorithm Average Compression Time.....	132
Table 5.8 - The Lossless LRCSC Compression Size and Huffman Compression Size .....	133
Table 5.9 - The Average LRCSC Approach Compression Size with Huffman Average Compression Size .....	135
Table 5.10 - The Lossless LRCSC Algorithm Execution Time with Huffman Execution Time .....	136
Table 5.11 - The Average Execution Time for Both Algorithms in Seconds.....	138
Table 5.12 - The LRCSC Algorithm Results Compared with Other Four Algorithm Results .....	139
Table 5.13 - Comparison Between the LRCSC and various benchmark systems in Term of Compressed File Sizes of Kodak Colour Test Images (size in KBs) .....	141
Table 5.14 - The LRCSC Results Compared with Other Four Algorithm Results .....	142

Table 5.15 - Execution Time in Seconds for the Kodak Image Set for Different Algorithms .....	144
Table 6. 1 -Fully Automated System Compression Ratio .....	147
Table 6. 2 - The Fully Automated System Average Compression Time .....	148
Table 6. 3 - Sample Images from the Five Image Sets Before and After Compression .....	148

## List of Abbreviations

<b>2D</b>	Two Dimensional
<b>AC</b>	Arithmetic Coding
<b>AD</b>	Average Difference
<b>AI</b>	Artificial Intelligence
<b>AINN</b>	Artificial Intelligence Neural Network
<b>ALCM</b>	Activity Level Classification Model
<b>ALPC</b>	Adaptive Linear Prediction and Classification
<b>BMP</b>	Bitmap File Format
<b>BPS</b>	Bit Plane Slicing
<b>BWT</b>	Burrows-Wheeler Transform
<b>CALIC</b>	Context-Based, Adaptive, Lossless Image Codec
<b>CSC</b>	Column Subtraction Compression
<b>CNN</b>	Convolutional Neural Network
<b>CR</b>	Compression Ratio
<b>DCT</b>	Discrete Cosine Transform
<b>DCNN</b>	Deep Conventional Neural Networks
<b>DFT</b>	Discrete Fourier Transform
<b>DIP</b>	Digital Image processing
<b>DL</b>	Deep learning
<b>DPCM</b>	Differential Pulse Code Modulation
<b>DSR</b>	Design Science Research
<b>DSRIS</b>	Design Science Research for Information System
<b>DWT</b>	Discrete Wavelet Transform
<b>EBCOT</b>	Embedded Block Coding with Optimised Truncation
<b>EDT</b>	Enhanced Differential Pulse Code Modulation Transformation
<b>ERIFD</b>	Enhanced Rounding the Intensity Followed by Dividing
<b>EQ</b>	Equation
<b>EZW</b>	Embedded Zero Tree Wavelet
<b>F-DCT</b>	Forward-Discrete Cosine Transform
<b>FMS</b>	Fast-Match-Search algorithm

<b>FR</b>	Full Reference
<b>GAP</b>	Gradient Adaptive Predictor
<b>GCD</b>	Greatest Common Divisor
<b>GIF</b>	Graphics Interchange Format
<b>GPS</b>	Global Positioning System
<b>GRC</b>	Gaps Removal Compression
<b>HBC</b>	Hierarchic Pairwise Coding
<b>HC</b>	Huffman coding
<b>HEVC</b>	High Efficiency Video Coding Computing
<b>HH</b>	High-High
<b>HINT</b>	Hearing in Noise Test
<b>HL</b>	High-Low
<b>HRI</b>	High Resolution Image
<b>HVS</b>	Human Visual System
<b>IQA</b>	Image Quality Assessment
<b>ISO/CCITT</b>	International Organization for Standardisation / International Telegraph and Telephone Consultative Committee
<b>JFIF</b>	File Interchange Format
<b>JPEG</b>	Joint Photographic Expert Group
<b>K-NN</b>	k-Nearest Neighbours Algorithm
<b>LGRC</b>	Lossy Gaps Removal Compression
<b>LH</b>	Low-High
<b>LL</b>	Low-Low
<b>LPB</b>	Local-Adaptive Block-Based Prediction
<b>LPCDH</b>	Linear Predictive Coding (LPC) with Discrete Wavelet Transform (DWT) followed by Huffman
<b>LRCSC</b>	Low-Resolution Column Subtraction
<b>LRI</b>	Low Resolution Image
<b>LZW</b>	Lempel-Ziv-Welch
<b>LZ77</b>	Lempel-Ziv 77
<b>LZ78</b>	Lempel-Ziv 78
<b>MAE</b>	Mean Absolute Error

<b>MBIC</b>	Minimum Block Intensity Compression
<b>MCIC</b>	Minimum Column Intensity Compression
<b>MD</b>	Maximum Difference
<b>MED</b>	Median Edge Detection
<b>ML</b>	Machine learning
<b>MRIC</b>	Minimum Row Intensity Compression
<b>MSE</b>	Mean Squared Error
<b>NK</b>	Normalised Cross-Correlation
<b>PMSE</b>	Peak Mean Square Error
<b>PNG</b>	Portable Network Graphics
<b>PSNR</b>	Peak Signal to Noise Ratio
<b>RAW</b>	Raw image formats
<b>RBFNN</b>	Radial Basis Function Neural Networks
<b>RCT</b>	Reversible Colour Transform
<b>RGB</b>	Red, Green, Blue
<b>RIFD</b>	Rounding the Intensity Followed by Dividing
<b>RLE</b>	Run Length Encoding
<b>RLC</b>	Run Length Coding
<b>ROI</b>	Region of Interest
<b>SPIHT</b>	Set Partitioning in Hierarchical Trees
<b>SSIM</b>	Structural Similarity Index Metric
<b>TIF</b>	Tagged Image File Format
<b>TRLC</b>	Traditional Run Length Coding
<b>UIQI</b>	Universal Image Quality Index
<b>VQ</b>	Vector Quantization
<b>WSN</b>	Wireless Sensor Network
<b>YCbCr</b>	Luminance Green (Y), Chrominance Blue (Cb), Chrominance Red (Cr)



# CHAPTER ONE: INTRODUCTION AND MOTIVATION

---

## *Overview*

This chapter describes the research motivation and introduces the research topic in the field of image compression. Then the research aim and objective are explained. This is followed by the description of the adopted research methodology that is used to achieve the research aim and objectives and the contribution to knowledge. The chapter concludes by providing a short description of the remaining chapters constituting this thesis.

---

## **1.1 Introduction and Motivation**

At the beginning of the 1960s, the need for digital imaging development arose. From that date to the current time, there has been considerable growth in digital images and image applications. Indeed, images have become an important part of our daily lives and activities. Photos are now taken mainly by mobile phones for personal photos but also for newspapers, websites and other media (Lyon, 2006).

Nowadays, cameras and their applications have seen unprecedented development. Each smart phone has a good resolution digital camera, which provides more data about images such as location, date and time. With the development of the digital camera and the increase in the number of photos and their transmission through networks, comes the size issue of the images as some motion images have 16 bit-depth with an image size of 50.3 megabytes. Regarding the large image size, a massive amount of data is created and it is reported that approximately 2.5 quintillion bytes are created daily (Lu *et al.*, 2014). The estimated amount of the data created in the year 2020 is approximately 1.7 MB per second for every person (Anon, 2019). Many researches had been developed to enhance the methods of displaying images in applications; these researches gave computers and cell phones the capabilities of displaying high resolution complex graphical images such as those used in computer games and medical applications. Although the visual aesthetics of images in the applications is enhanced, these applications need a lot of disk storage.

With the result of producing such a high number of high-resolution images, came the challenge of saving and transmitting these images. The higher resolution images need more storage space and transmitting time; image compression is about developing algorithms to solve this problem. Given the enormous production rate of digital images, developing image compression algorithms has become a very important field for researchers to save storage space and channel bandwidth. Hence, the research motivation is to investigate the digital image processing challenges in the area of lossless image compression by developing new algorithms that aim to

enhance the current image compression rates and execution times to meet the demands of the information revolution and fast development of producing images with large sizes.

Lossless image compression aims to decrease the image size without effecting the image quality (Shukla, Alwani and Tiwari, 2010). The efforts developed by the researchers in the lossless domain aim to enhance the compression rate with zero distortion and an acceptable execution time.

## **1.2 Research Aim and Objectives**

Based on the motivation from the previous section, this research aims to create a fully automated lossless image compression system, that decreases the image size with high compression rate and zero percentage of distortion in a very fast time. To achieve this aim, the following objectives are identified.

- 1- Reviewing the literature critically on current image compression techniques.
- 2- Identify the weaknesses of the current techniques.
- 3- Design the new compression algorithms.
- 4- Implement the new algorithms.
- 5- Validate the algorithm results.
- 6- Evaluate the compression results by conducting an empirical comparison of the new algorithm with existing state of the art algorithms.

## **1.3 Research Methodology**

Vijay, Bill and Stacie, (2015) generally defined research as some activity that participate in understanding a particular phenomenon. The phenomenon is a set of interested actions for some entities found by the researchers, and the research methods are to produce knowledge by the researcher by using the convenient activities. (Hanid, 2014) defines research as an investigation methodology that test the resources to expand the knowledge scope within domain. (Alan and Samir, 2012) defined the difference between research and reasoning by providing the following definitions; reasoning can work in an abstract environment that is highly separated from the reality. However, research supports self-correcting by providing testing methods to obtain results for validation. Information systems employed a lot of methods for developing knowledge as it supports the phenomenon behaviours predictions (Vijay, Bill and Stacie, 2015).

### 1.3.1 Design Science Research in Information Systems

Design Science Research (DSR) is an Information Systems (IS) research methodology that provides problems solving methods and can handle the industry sectors challenges to contribute to the theories in the application domain (Hanid, 2014). Furthermore, DSR is a methodology for finding and exploring alternative solutions for problem solving for a specific domain.

(Hevner, Chatterjee and Juhani, 2010) emphasised that the DSR methodology become one of the best methodologies over the years. The DSR concentrates on organizations' problem solving by providing valid knowledge in a certain domain.

(Gregor and Hevner, 2013) stated that, the DSR is interested in the development of technical artefacts in the IS domain, such as information system evaluation, decision support systems and modelling tools. DSR become reliable information system research model, since it provides effective solutions for the research problems.

Figure 1.1 illustrate the IS three levels of research and the DSR role for each level as described by (Hevner, Chatterjee and Juhani, 2010).

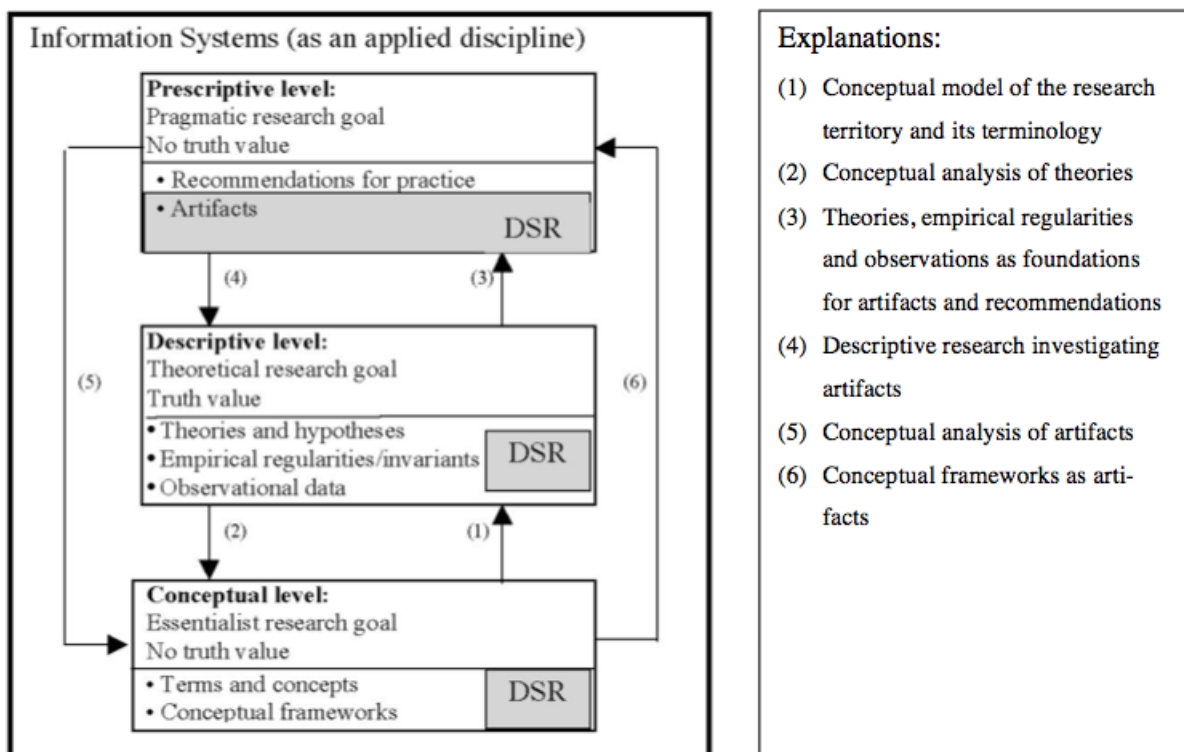


Figure 1. 1 - Levels of Research in Information Systems and DSR Role

(Adapted From: (Hevner, Chatterjee and Juhani, 2010)

The concepts of using the DSR with information systems (DSRIS) is continually evolving as acquiring knowledge process through design practices (Kuechler and Vaishnavi, 2012). DSRIS artefact and methodology is to decide what to build and the method of building it.

### 1.3.2 Philosophical Ground of DSR

(Ken Peffers *et al.*, 2007) stated that, early in the 1990s, IS researchers are starting to pay more attention to DSR. The DSRIS methodology is different from other methodologies in theory building and testing methods. (Ken Peffers *et al.*, 2007) described that the DSR is to develop solutions or models that helps humans.

(Nunamaker, Chen and Purdin, 1990) noted that some researchers tried to create an integrated model that combine the research process with the system development process to facilitate the system developments with the support of suitable experiments. Furthermore, the IS research can be enhanced by adopting the DSR methodology.

(Hanid, 2014) emphasise that the knowledge creation process begins with a substantive field of inquiry known as philosophy, where the inquiry, theories, facts and alternatives are gathered to help in the knowledge creation. (Scotland, 2012) noted that, several philosophical approaches to thinking exists and they are commonly categorized into positivism, interpretivism, phenomenology, realism, hermeneutics, critical theory and phenomenology.

The research strategy and research method selections rely on the philosophical stance. However, the previous categories did not cover the Design Science Research (DSR) that is interested in problem-solving (Hanid, 2014). Table 1.1 lists a comparison of DSR with some other well-known research perspectives.

Table 1. 1- Design Science Research Perspective. (Source: (Vijay, Bill and Stacie, 2015))

Basic Belief	Research Perspectives		
	Positivist	Interpretive	Design
Ontology	A single knowable and probabilistic reality	The construction of multiple realities in a social manner.	Multiple realities with contextually situated alternative world-states. Socio-technologically enabled.
Epistemology	Objective; dispassionate. Detached observer of truth	Subjective, i.e. values and knowledge emerge from the researcher-participant interaction.	Knowing through making objectively constrained construction within a context. Iterative circumscription reveals meaning.
Methodology	Observational in nature with quantitative and statistical measures	Participation; qualitative. Hermeneutical, dialectical.	Developmental in nature with an impact measurement of artefact on compound systems
Axiology	Truth: universal and beautiful; prediction	Understanding: situated and description	Control; creation; progress (i.e. improvement); understanding

Observing Table 1.1, the DSR involves a multi-dimensional perspective from ontological and epistemological beliefs. The methodological belief is suitable for IS domain, since its nature is developmental. Furthermore, DSR controlled the environment and provide improvements to the research process as the Axiological belief. This is why the researcher adapted the DSR in this research as it supports the research objectives and meet the research requirements.

### 1.3.3 DSR Process Model

The DSR model describes the research environment and variables according to the variables practiced. This research adapted the DSR process model from (Vijay, Bill and Stacie, 2015).

Figure 1.2 illustrate the adapted DSR process model in this research.

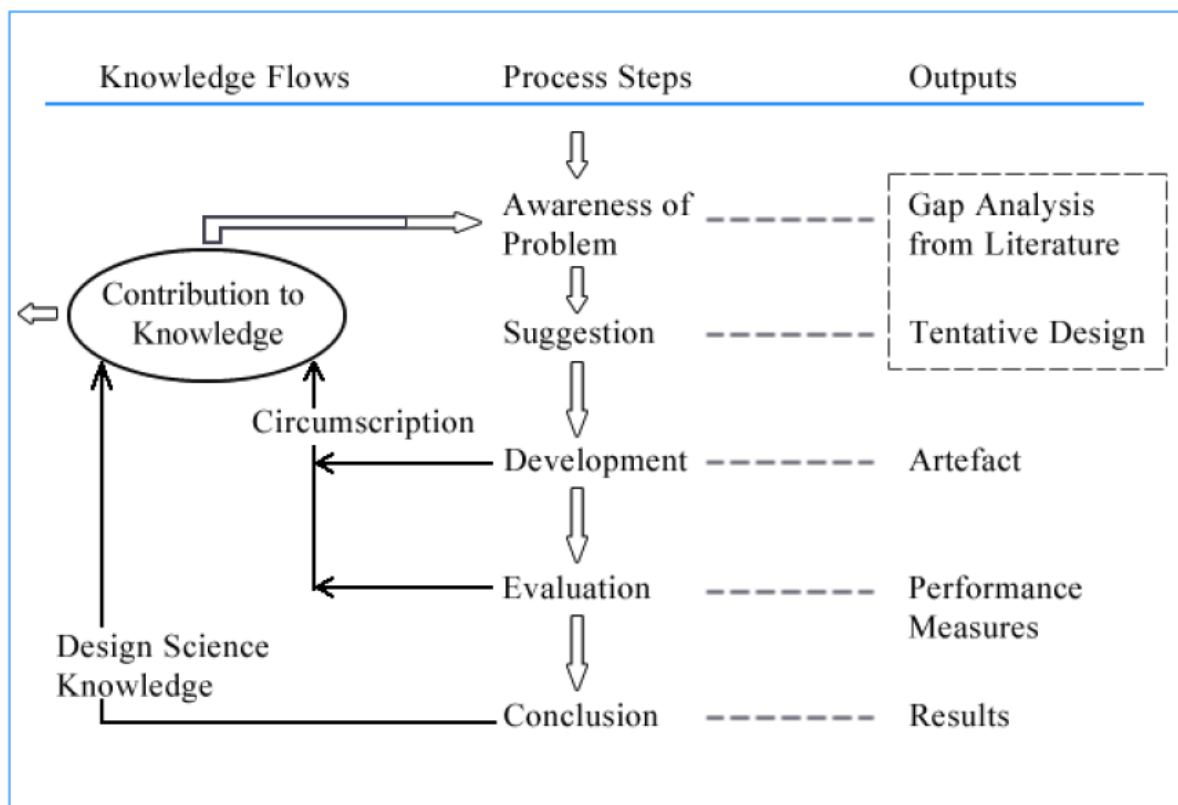


Figure 1. 2 - DSR Process Model (Adapted from (Vijay, Bill and Stacie, 2015))

#### Awareness of the Problem

Awareness of the problem is the first step in the DSR process model. It aims to gain the knowledge of the research domain to cover and understand the research area and identifies the research problem that needed to be solved (Alan and Samir, 2012). Critical review of the existing literature in the research domain is conducted to identify the problem. Once the critical review is completed, the gaps from the current literature were analysed to provide

comprehensive awareness of the current problem. Sections 2.1 to 2.3 in chapter two describe this in detail.

**Suggestion:** The second DSR process model step is solutions suggestion. This step aims to propose solutions to the problem from the previous step. This step suggested a solution for the lossless image compression problems such as low compression rate and slow execution time. Sections 3.1 in chapter three describe the solutions suggestion.

**Development:** The third step of the DSR process model is development. This step considers the development of a solution for the problems from the first step by using the suggestions from the second step to meet the research objectives. (Gregor and Hevner, 2013) emphasised that, the development is to create artefact such as constructs, models or methods. The artefact should meet the research contribution. Sections 4.1 to 4.3 in chapter four describe the development phase for the first algorithm and sections 5.1 to 5.3 in chapter five describe the development phase for the second algorithm.

**Evaluation:** The fourth step of the DSR process model is evaluation. (Gregor and Hevner, 2013) emphasised that evaluation should be rigorous and appropriately implemented to prove that the developed artefact meets the research aim and objectives. It should clearly describe the outputs as a solution for the research problem based on the requirements. The DSR evaluation could use one or more of the following evaluation methods:

- Observational.
- Analytical.
- Experimental.
- Testing.
- Descriptive.

Section 4.5 describe the evaluation process for the first algorithm and Section 5.5 describe the evaluation process for second algorithm.

**Conclusion:** The final DSR process model step is the conclusion. This step is to describe how the research achieves its aim and objectives from the research outputs, by identifying the research problem in the domain, the design rigour, artefact and novelty (Offermann *et al.*, 2009). Furthermore, it provides recommendations for future research (Vijay, Bill and Stacie, 2015). Chapter 7 describe the conclusion phase for this research. Table 1.2 lists the DSR model steps for this research and each process steps corresponding chapters.

Table 1. 2 - DSR Process Steps and the Corresponding chapters in this research

DSR Process Model Steps	Corresponding Chapter(s)	Outputs
Awareness of Problem	Sections 2.1 to 2.3	Gain the knowledge of the research domain and identifies the research problem.
Suggestion	Sections 3.1	Propose solution for lossless image compression methods to enhance the compression ratio and execution time.
	Sections 3.2 to 3.4	Requirements specification for the research area
Development	Chapter 4	The proposed lossless image compression algorithm for natural images (CSC)
	Chapter 5	The proposed lossless image compression algorithm for synthetic images (LRCSC)
	Chapter 6	Automated System for image compression
Evaluation	Sections 4.5	Experimental Evaluation for the CSC algorithm to evaluate the algorithm performance.
	Sections 5.5	Experimental Evaluation for the LRCSC algorithm to evaluate the algorithm performance.
Conclusion	Chapter 7	Research Summaries
		Research Contributions
		Recommendations for future work in the domain
Communication	Chapters 1 to 7	Research Thesis

### 1.3.4 DSR Knowledge Contribution Framework

The DSR outputs contribute to the knowledge in specific domains. (Gregor and Hevner, 2013) classified the knowledge contribution into four types.

1. Invention: provides knowledge contribution by inventing or providing new solutions to a new problem area in a specific domain.
2. Improvement: develop new knowledge or solutions that contributes a well-known problem area.
3. Adaptation: adapting knowledge solution for current problem from different area or domain.
4. Routine Design: the use of existing knowledge or solutions for a well-known problem area.

(Vijay, Bill and Stacie, 2015) mentioned that the DSR may use more than one type of knowledge contribution. Figure 1.3 displays the DSR knowledge contribution framework.

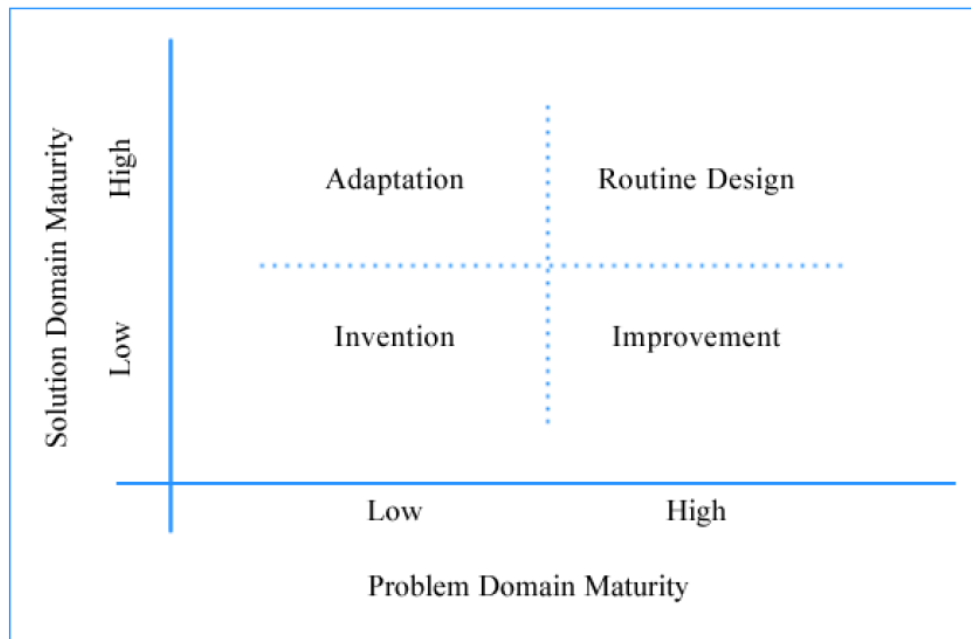


Figure 1. 3 - DSR Knowledge Contribution Framework.

Adapted from: (Gregor and Hevner, 2013).

This research aims to develop a new knowledge and solutions for known challenges within the lossless image compression domain to enhance the current algorithms performance. Therefore, this research is related to “Improvement” type on the DSR Knowledge Contribution Framework.

#### 1.4 Research Contributions to Knowledge

The research contributions to knowledge are:

1. A comprehensive literature review on lossless and lossy image compression technologies.
2. Addressing the challenges of lossless image compression techniques.
3. Developing a lossless image compression algorithm for natural images.
4. Developing a lossless image compression algorithm for synthetic images.
5. Adapting an artificial intelligence system for image classification to develop a fully automated system for choosing the suitable algorithm for compressing the images regarding the image type.
6. Evaluating the compression algorithms.



## **1.5 Thesis Structure**

The thesis is composed of seven chapters:

### **Chapter 1: Introduction**

This chapter describes the research motivation and the research domain in the area of image compression. Then the research aim and objective were explained. It also includes the adopted research methodology to achieve the research contribution of knowledge, and finally, the description of the thesis structure is described.

### **Chapter 2: Literature Review**

This chapter covers the literature review for the digital image processing in the domain of image compression; with a view of a critical analyses of the currently used algorithms regarding the image compression domain. Furthermore, the advantages and limitations of the current algorithms are described to identify the challenges related to the research problem to provide directions for the development chapters.

### **Chapter 3: Solutions and Development Tools**

This chapter covers the research requirements and the environmental parameters that surround the research, such as the development tools, the used programming language and the tested image sets.

### **Chapter 4: Column Subtraction Compression Algorithm**

This chapter describes in detail the proposed lossless Column Subtraction Compression algorithm (CSC). It starts with a detailed explanation for all the procedures used in the proposed technique. This was then followed by validating the algorithm by testing its results and verifying it by comparing it with the state-of-the-art algorithms.

### **Chapter 5: Low-Resolution Column Subtraction Compression Algorithm**

This chapter describes in detail the proposed lossless Low-Resolution Column Subtraction Compression algorithm (LRCSC). It starts with a detailed explanation of all the procedures used in the proposed technique, followed by the validation of the algorithm by testing its results, and its verification by a comparison between the lossless approach results with state-of-the-art results.

### **Chapter 6: Automated System for Image Compression**

This chapter describes in detail the concepts of artificial intelligence, machine learning, artificial neural networks and deep learning, followed by the fully automated artificial intelligence system for choosing the suitable compression algorithm.

### **Chapter 7: Conclusion and Recommendations**

This chapter summarizes the research results and provides recommendations for future work.

### **2 Chapter Overview**

This chapter covers the literature review for digital image processing in the domain of image compression; with a view of critically analysing the current state of the art used algorithms. Furthermore, the advantages and limitations of the current algorithms are discussed to identify the challenges related to the research problem and provide directions for the development chapters.

---

#### **2.1 Literature Review**

Given the enormous production rate of digital images, image compression has become a very important field for researchers. Image compression aims to decrease the image size without losing its main information content, or it may have some acceptable loss provided that the human visual system cannot notice it.

Image compression techniques are either lossless or lossy. The lossless compression techniques aim to produce an image that is similar to the original one after the decompression phase. In contrast, the lossy compression techniques miss some data from the source image during the compression process; it aims to decrease the image size by enhancing the compression rate, although the decompressed image quality will be less than the original image. In practice, the lossy compression techniques is more used than the lossless compression techniques (Otair and Shehadeh, 2016).

Most compression techniques take advantage of the fact that neighbouring pixels are strongly related to each other and their values are very close or similar. In addition, each pixel should have a high correlation with the surrounding pixels, which leads to a high data redundancy called spectral correlation. Furthermore, an image can be compressed by removing this redundancy and the compression algorithm will be more effective if we remove the data redundancy before starting compression (Husseen, Mahmud and Mohammed, 2017).

Any image may include different information types, such as redundant information, which helps in producing the original image without losing any of its information. Another type of information is irrelevant information; this type saves enormous details, where the human visual system cannot observe (Hussain and Al-Fayadh, 2018).

### 2.1.1 Digital Images

Several tools can be used to convert images in their original form (E.g. paper form) to a digital image (computerised image), such as scanners and photography devices. Such tools have contributed significantly in digital imaging development. A digital image is represented in the form of a two-dimensional matrix (Shinde and Dani, 2011).

Each of the matrix values is referred to as a pixel or image element. An image could be represented with a small number of pixels or with a large number of pixels (up to millions) depending on its resolution (Abdalla and Osman, 2016). Intensity is used to represent the pixel's value, whereas the number of bits needed to represent each intensity is called bit-depth. An example of a representation of a digital image is shown in Figure 2.1 where the  $f$  function is used to locate the intensity address  $f(\text{row}, \text{column})$  and 181 represent the value for the acquired address which is the intensity of the pixel (in the Figure we also show the binary representation of 181).

	1	2	3	4	5	6	7	8	9	10	11	12
1	178	180	179	183	180	181	184	180	179	182	181	182
2	179	183	181	182	182	184	185	184	182	184	182	184
3	181	181	181	184	183	184	182	186	184	184	181	181
4	179	181	179	181	181	181	182	181	182	183	180	181
5	181	178	178	181	178	181	180	183	181	182	180	181
6	181	178	180	174	179	180	174	179	184	178	178	178
7	179	183	181	183	181	$f(4,5) = 181$			181	182	181	181
8	180	182	178	182	181	$181_{10} = (10110101)_2$			181	182	180	182
9	182	185	184	184	180	183	183	184	184	185	183	182
10	181	180	180	182	181	183	181	183	184	184	184	184
11	182	180	177	182	180	183	179	181	181	183	180	181
12	178	179	179	181	183	178	183	181	182	180	183	181

Figure 2. 1 - Digital image (value and bit representing). Obtained From (Abdalla and Osman, 2016).

With the fast-growing use of digital images, image processing has become a significant aspect in various fields. The image will be the input for processing, while the processed output could be an image or information related to the input image.

#### Digital Images Types:

(Otair and Shehadeh, 2016), (Kuppusamy and Mehala, 2013), (John and Joe, 2005) and (Padmavathi and Thangadurai, 2016) described the digital image types as:

- 1- Binary Image: represents the images in black and white only – black has the value zero while white has the value one.

- 2- Greyscale Image: In this type of image, the image colour is black and white with all the colour gradations between them. The colour values are assigned values from 0 to 255, where each pixel is represented by 8 bits, and the internal value interval is from 0 to  $2^{N-1}$  (where  $N$  is the bit-depth). For example, if the bit-depth is 8, then the intensity values range should be between 0 and 255. These values are represented using binary code; For example, 135 is represented by 10000111 (Starosolski, 2007).
- 3- Colour Image: For images represented by colours, each colour can be represented by a combination of the main three colours – Red, Green and Blue (RGB). Each pixel represents any colour by three parts (RGB), each part consists of 8 bits with the total of 24 bits. It is worth noting that, coloured image compression is derived from the greyscale compression algorithm (Starosolski, 2007).

By combining all the RGB values, we can represent any colour. For example, the minimum value of the three colours should be 24 bits of zeros, which represents the black colour, while if we want to represent the white colour, we should set the value of the RGB to the maximum value (255). All of the other colours can have values between 0 and 255 (John and Joe, 2005).

### **2.1.2 The Most Popular Used Image Formats**

Images can use a huge amount of memory. Image size is different according to the information they store and the format type they use; some image formats use more size than others due to the application with which they are used (Chawla, Beri, and Mudgil, 2014).

- 1- Joint Photographic Expert Group (JPEG, JPEG or JPG) is a lossy compression technique used with images that store a huge number of colours such as 24 bits (16 million colour) photographic images. JPEG was created to compress continuous-tone images (colour or greyscale) of normal daily-used real-world images, natural images, animations, documents or videos. Nonetheless, the image quality will never be as before, since some of the data will be lost during the compression process. This loss of data (distortion) may not be noticed by the human visual system. JPEG/JFIF format is used by most of the digital camera. One of the JPEG files' limitations is generational degradation; the algorithm suffers when we repeatedly edit and save an image. JPEG also provides lossless image storage, but the lossless version is not widely supported.

Note that JPG files and GPEG files are pronounced “jay-peg”, and some GPEG files use the JPG extension because they both have the exact same format, even though the extension name is different. JPE and JPEG File Interchange Format (JFIF) files are also extensions for GPEG, but they are not very commonly used (Lifewire, 2018).

- 2- JPEG 2000 was created in the beginning of the year 2000 by the Joint Photographic Expert Group. This standard supports both lossless and lossy techniques. It improves the image quality and compression ratios. One of its limitations is that it needs more computational power, which makes the JPEG format more common; it has been used in professional movie editing and distribution.
- 3- Tagged Image File Format (TIF) is mostly used with lossless image compression techniques, while some applications use it with lossy compression techniques. The image with this format is normally represented with 8 bits or 16 bits for each colour (R, G, B) or for 24 bits or 48 bits as well. The image compression algorithm should store in this format the needed image compression details. TIF is not recommended to be used with web images, as they produce a large image file size.
- 4- Graphics Interchange Format (GIF) is a lossless image compression format, used only with black and white text or with greyscale images, which have fewer than 256 colours (8 bits); consequently, it will not work with coloured images since most coloured images represent the image with 24 bits. To solve this problem, we need to convert the colour image into an 8 bits image, but the compression will be lossy in this case.
- 5- Portable Network Graphics (PNG) is a lossless image compression format. It has a 10 – 30% compression rate enhancement compared to the GIF format; also, it can handle more colour. It supports partial transparency, which may be used in fades and antialiasing for text.
- 6- Bitmap (BMP) file format is used in Microsoft Windows' operating system graphics files. Normally, BMP files are of large size before compression. The main advantages of the BMP file format are the wide use and simplicity.
- 7- The raw image formats family (RAW) generally uses lossless or nearly lossless compression techniques that are available as options on some digital cameras. It has a better compression rate than the TIF format since it produces smaller images from the cameras. The drawback is that there are many manufacturers, and every manufacturer has its own RAW format and application to view the image.

### **2.1.3 Digital Image Processing**

Digital Image Processing (DIP) is described as the process of executing a computer algorithm to perform image processing on a computerised image. DIP allows us to use several different image compression algorithms to compress images; it also helps in avoiding many difficulties, such as signal distortion (Madhu and Dalal, 2017). Due to the new image size, faster

transmitting and better representation are needed. DIP became a part of many fields such as communications systems, Global Positioning System (GPS), Radiology Information Systems (RIS), medical science, telemedicine networks, picture archiving, social media and image compression (Kuppusamy and Mehala, 2013); (Li-Hui and Chen, 2017).

The development of digital image applications is growing continuously; with this growth comes the real difficulties of transferring and saving the remarkable volume of images and their related information (Abdalla and Osman, 2016);(Agustsson *et al.*, 2019); (Li-Hui and Chen, 2017); (Poobal and Ravindran, 2014); (Talukder and Harada, 2010); (Vidal and Amigo, 2012)

## **2.1.4 Operations Performed on Digital Images**

### **2.1.4.1 Image Segmentation**

Image segmentation aims to group the digital image pixels using the benefits of the neighbouring pixel, as they have a strong relationship with each other (their values are very close to each other). Image segmentation is commonly used to calculate image features, which helps to get information about numerous image parameters (Hazarika, Nath, and Bhuyan, 2015). Every pixel should have a high correlation with the surrounding pixels, which leads to higher data redundancy. One of the major objectives of image compression is to decrease or remove the data redundancy using the correlated pixels.

(Wei, 2008) and (Kuppusamy and Mehala, 2013) described the method of reducing the correlation between pixels. Because of the high correlation in the neighbouring region or pixel, an image can be compressed (each pixel is similar or close to its adjacent pixel). When we decrease or remove each pixel's correlations, we can decrease the image size by using statistical characteristic and any compression techniques, such as variable length coding. The process of compression starts with the conversion of colour space. We use the transform matrix to convert the three dimensions' colour matrix of the image from red, green and blue (RGB) to YCbCr, pixel by pixel, as shown in Figure 2.2, where Y is commonly called the luminance and Cb, Cr are commonly called the chrominance (blue difference and red difference). Luminance receives the brightness of the light, which is proportional to the total energy in the visible band, while chrominance describes the perceived colour tone of light, which depends on the wavelength composition of the light. Chrominance is in turn characterised by two attributes: hue, which specifies the colour tone (which depends on the peak wavelength of the light); and saturation, which describes how pure the colour is (which depends on the spread or bandwidth of the light spectrum). As displayed in Figure 2.2, the Y,Cb,Cr matrices maintain the same matrices

dimensions from the R,G,B matrices ( $m \times n$ ), the only difference is the values of the Y,Cb,Cr matrices that are smaller than the R,G,B matrices.



Figure 2. 2 - Colour Space Conversion from RGB to YCbCr. Adapted From (Kuppusamy and Mehala, 2013)

Quantization is a structure that is concerned with converting a continuous set of data into a finite set of discrete data. The image should be the input to the quantizer while the output should know the number of levels. The best quantizer should represent the original image (signal) with the minimum loss of data (Ballé et al., 2017).

The objective of quantization is to produce a higher compression rate by reducing the image accuracy. Each image compression standard, such as JPEG and JPEG 2000, has its own quantization methods (Paul and Kumar, 2003); (Solomon and Breckon, 2011).

#### 2.1.4.2 Image Compression

Compression is the process that is concerned with decreasing the number of bits required to represent an image, audio or video files. Suppose that the image is using 8 bits to represent each pixel; if we decrease the bit-depth to 6 bits to represent the intensity for each pixel, then we should have a new image with less size. Figure 2.3 represents the structure of a general data compression, which uses an encoder for compression and decoder for decompression (Gupta, Bansal and Khanduja, 2017); (Li, Drew and Liu, 2014).

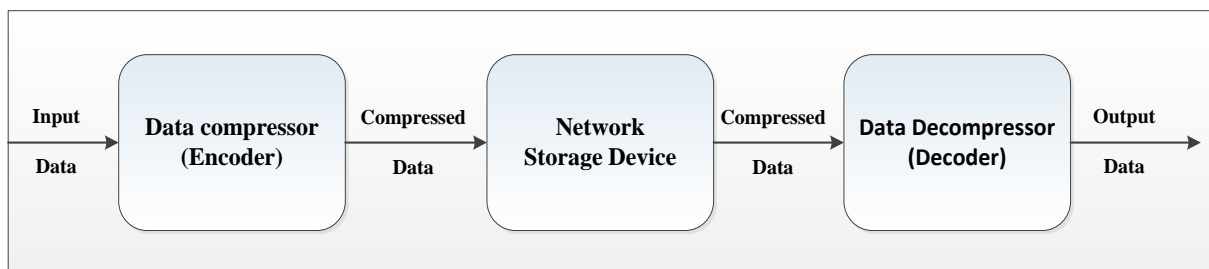


Figure 2. 3 - General Data Compression Scheme adapted by (Gupta, Bansal and Khanduja, 2017).

Image compression is the process that is concerned with decreasing the number of bits required to represent an image. It could use a lossless or lossy algorithm. A lossless algorithm is responsible for decreasing the image size without losing any of its information; thus, it has a low compression rate compared to a lossy algorithm. A lossy algorithm decreases the image

size by eliminating redundant and irrelevant data and is considered to have a high compression rate, but it still loses some of the original image information(Otair and Shehadeh, 2016).

Compression is achieved by the removal of one or more of the three basic data redundancies (Mehanna, 2013);(Vijayvargiya, Silakari and Pandey, 2013). There are three types of redundancies which are:

1. Coding redundancy, which is present when less than optimal (i.e. the smallest length) code words are used, uses variable length code words to match the statistics of the source image. Some of the algorithms that use this type of redundancy are the Huffman coding algorithm (Huffman, 1951) and Arithmetic Coding algorithm (Shaw, Rahman and Routray, 2018).
2. Inter-pixel redundancy (spatial redundancy) results from the correlations between the pixels of an image; Run length coding (RLE) is one of the algorithms that use this type of redundancy (Al-Wahaib and Wong, 2010).
3. Psycho visual redundancy, which is due to data that is ignored by the human visual system (i.e. visually non-essential information), is considered for use with a lossy compression techniques, since some information is lost from the source image (Mehanna, 2013) (Ernawan, Abu and Suryana, 2013).

### 2.1.5 General Image Compression Model

Figure 2.4 describes in detail the general image compression and decompression model. The compression model aims to produce a compressed image, which comprises of two phases: pre-processing and encoding. In contrast, the decompression model seeks to reconstruct the compressed image by using the decoding phase and post-processing phase. To compress any image, the digital image will be loaded into the compression algorithm.

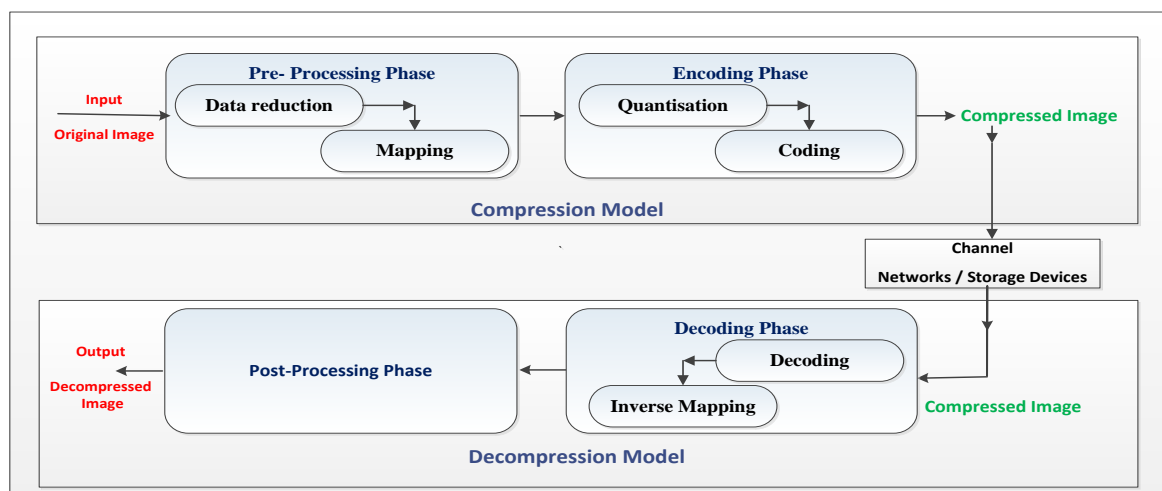


Figure 2. 4 - General Compression and Decompression Model



The general compression model starts with the pre-processing phase, which is responsible for data reduction (elimination of the inter-pixel redundancy) (Sood, Bhathal, and Singh, 2018). It aims to remove any irrelevant data (irrelevant data usually comes from applications) to prepare the image for the next phase. After removing irrelevant data, the image will be loaded into the mapping process to rearrange its data using a mathematical operation, which will make the coming encoding phase much easier and more effective. This mathematical operation is responsible mainly for removing the spatial redundancy, since neighbouring pixel values are similar or very close to each other. The image is now ready to be compressed in the encoding process.

As it can be seen in Figure 2.4, the second step in the compression phase is encoding. The input for this phase will be an image that is smaller than the original one. A quantizer is responsible for removing or decreasing psycho-visual redundancy (Sood, Bhathal and Singh, 2018). The coding process is a reversible process, and no data should be lost during this process. The coder delivers a 1:1 mapping where each input symbol should be coded to a unique output in the decoding phase. The code may have two types: equal length code and unequal length code. In the equal length code, the code-words have the same size, whereas in the unequal length code, the code-words can have variable lengths (Sood, Bhathal and Singh, 2018).

It is important to note the following facts:

- 1- Some compression algorithms use all the previous processes while other algorithms may use some of them only.
- 2- Some information might be lost in the quantization process, since quantization is not a reversible process. Therefore, the decompression model does not have a quantization phase. One exception for this role is the Discrete Wavelet Transform (DWT) algorithm, since it has its own quantizer for compressing the image. The quantizer in the DWT is one of two types, being reversible or non-reversible (Mehanna, 2013).
- 3- Non-reversible quantizer could be used with lossy compression techniques only, due to the unrecoverable loss of information in this process.

The decompression model is responsible for reconstructing the compressed image in two phases. The first phase is the decoding phase, which starts by loading the compressed image into the decoder, where the codes will be mapped to the original quantized values using the reversal process. Then, these values should be loaded to the inverse mapping process. The second phase in the decompression model should be the post-processing, which places the last detail that could enhance the view of the final image.

### 2.1.6 Description of the Processes Used in the Current Techniques

As it has been explained earlier, there are three main processes in the compression model. Some algorithms use all three while others may use some of them; however, some algorithms use more than one as a hybrid technique. Choosing the algorithm depends on the compression technique type (lossless or lossy) and the applications' needs. The following sections discuss the main three compression processes (Wei, 2008).

#### 2.1.6.1 Correlation

Correlation is also known as Transforming or Mapping. Images can be compressed by taking advantage of the high correlation between neighbouring pixels. In another word, each pixel value is similar or very close to the value of its adjacent pixels. When this correlation is decreased, then any entropy coding algorithm can be used to decrease the image size. In fact, decreasing the pixel correlation is the most important part in compressing the image.

Correlation usually starts with a prediction and is followed by an entropy coding algorithm to enhance the compression rate. As such, prediction is used to reduce colour components' redundancy (pixel intensity redundancy). Prediction methods are either low-complexity or high complexity approaches; the main factors that may affect image prediction accuracy are edges and noise (Novikov, Egorov, and Gilmutdinov, 2016).

There are various relevant processing techniques that are used to remove this correlation. The most popular methods are:

- 1- **Predictive Coding:** such as Differential Pulse Code Modulation (DPCM), Median Edge Detection (MED), Gradient Adaptive Predictor (GAP), Activity Level Classification Model (ALCM) and Adaptive Linear Prediction and Classification (ALPC), which are all algorithms that are used with lossless compression techniques, where the compressed image should be exactly like the source image. Decorrelating the image data is to remove the inter-pixel redundancy (mutual redundancy), which will lead to a much more efficient and better compression rate. Furthermore, predictive coding is to remove the interpixel redundancies by using the neighbouring pixel value to predict the current pixel value and generate a new value; the new value should be encoded using a variable length algorithm. Subtracting the predicted value from the original will produce the error signal value; this value will be rounded to its nearest integer and then encoded by using a variable length algorithm. Using the rounding function may cause some distortion since this function is not reversible, which makes the predictive coding related to the lossy techniques; but predictive coding is considered to be a lossless compression technique, since there is no need for a

quantizer. After removing the inter-pixel redundancies, the image will be compressed using a variable length algorithm (Hussain and Al-Fayadh, 2018).

- 2- **Orthogonal Transform:** Discrete Cosine Transform (DCT) is the most popular transform method in lossy compression techniques. It is used with the JPEG image compression standard, where the original image loses some of its information during the compression process. The transformation algorithm is to transform the pixels from the image domain into another domain to produce a set of coefficients (Hussain and Al-Fayadh, 2018).
- 3- **Subband Coding:** Discrete Wavelet Transform (DWT) is the most common subband method, which divides the image spectrum into high-pass and low-pass types. It is used in the JPEG 2000 standard as a two-dimension matrix.

#### **2.1.6.2 Quantization**

Quantization is a lossy compression process that aims to increase the compression rate by decreasing the image precision. For example, if the original image represents each intensity bit-depth by 8 bits, and the bit-depth for the same image can be decreased to 5 bits, then we can say that we reduced the image size and prepare the image to be compressed even more in the entropy coding phase as a post processing phase. The limitation of this process is the loss of data or, as it known, distortion. This distortion is not recoverable since the quantization process is not reversible. JPEG and JPEG 2000 standards use different types of quantization. Two of the most popular quantizers algorithms are the scaler quantizer and vector quantizer (Ballé, Laparra and Simoncelli, 2017).

#### **2.1.6.3 Entropy Coding**

Entropy Coding is a lossless compression method that aims to reduce the coding redundancy to reach less average length for the image, by implementing any of the entropy coding algorithms such as Huffman coding. Entropy Coding maps the input data into bit sequences to produce shorter output (Kodituwakku and Amarasinghe, 2010); (Manjinder and Gaganpreet, 2013). Entropy Coding depends on presenting the highest probability appearance symbols in the original data stream (source image) with short code-words in the compressed bit stream (compressed image), while the lowest probability appearance symbols should have the longest code-words. Some lossy compression algorithms use Entropy Coding due to its efficacy in decreasing the number of bits generated by the quantizer's output (Zaineldin, Elhosseini, and Ali, 2015).

### 2.1.7 Measurement of Image Size and Quality

After compressing any image we should have mathematical procedures to calculate the percentage of compression, image quality efficiency, and algorithm performance, which will lead us to a better understanding of the strongest and weakness points for each algorithms (Hussain and Al-Fayadh, 2018).

- Measuring the compression proportion, known as the compression rate, is important to find out the image compression percentage, which can be calculated by dividing the compressed image size by the original image size, as in Eq. 2.1. This percentage is high in lossless compression techniques, while it is low in lossy compression techniques (John and Joe, 2005); (Wang and Li, 2011).

$$\text{compression rate} = \frac{\text{Compressed Image Size}}{\text{Original Image Size}} \quad 2.1$$

The best compression percentage is when the compression rate equation result is far away from one and near to zero; therefore, the compression algorithm should provide a compressed image size smaller than the original image.

When we get the compression rate results, we can calculate the algorithm storage saving percentage easily by using Eq. 2.2.

$$\text{Storage Saving} = 1 - \text{compression rate} \quad 2.2$$

Image compression ratio (Cr): is the ratio between the original image size and the compressed image size. Thus, when decreasing the image size from 100 MB to 20 MB then we can represent the compression ratio of  $100/20=5$  or 5:1 (read five to one) Eq. 2.3. is for calculating the algorithm Cr.

$$\text{Cr} = \frac{\text{Original Image Size}}{\text{Compressed Image Size}} \quad 2.3$$

- The image compression algorithm performance can be calculated by measuring the compression speed and the decompression speed. Compression speed is the number of bits that the compression algorithm can process per second while the decompression speed is the number of bits that the decompression algorithm can process per second.
- Since the Cr equation is concerned only with image size, it cannot give us any hint about the image quality. Therefore, we need to use different measurement techniques that can

measure the similarity and differences between both the source image and the compressed image. There are two methods for measuring image quality: subjective assessment and objective assessment (Gogoi and Ahmed, 2016); (Varnan *et al.*, 2011).

1. Subjective evaluations can be done by selecting several observers who have been tested for their visual abilities. The observers are then shown many test scenes to score their quality. Image fidelity is to represent the similarity between the reference image and the compressed image and how close they are. It is the only “correct” method of quantifying visual image quality.

For the Human Visual System (HVS), the basic image quality measures are:

- i. Universal Image Quality Index (UIQI)
- ii. Structural Similarity Index Metric (SSIM).

Nevertheless, subjective evaluation has some limitations, such as:

- i. It usually needs a lot of time for measuring images.
  - ii. It is expensive.
  - iii. It cannot be used with real-time systems.
2. Objective evaluation uses well-defined mathematical techniques for quality assessment. It focuses on analysing the image and describing its quality by making a report, without human involvement. During the past few years, objective methods have been used more than the subjective methods because of their benefits, such as fast execution and no human interaction.

A distortion measure aims to specify the degree of similarity between an original image and the compressed image by using a mathematical procedure. We can classify image quality metrics based on the availability of the source for original image (zero-distortion) compared with the compressed image (distort-image). The existing methodologies are:

- 1- Full reference (FR): original image should be available.
- 2- No-reference: where the original image is not available.
- 3- Reduced reference: where the reference image is not available 100%.

Simple statistics error metrics is a method of full reference objective quality (distortion assessment), which includes:

- a) **Mean Squared Error (MSE)**: the simplest and one of the most widely used methods, with full-reference image quality measurement. Eq. 2.4 describes the MSE equation and its parameter (Gogoi and Ahmed, 2016).

$$\text{MSE} = \frac{\sum_r^j [x1(r, j) - x2(r, j)]^2}{r \times j} \quad 2.4$$

Where  $r$  and  $j$  are the elements coordinates, and  $x1$  represents the compressed image pixels and  $x2$  represents the source image pixels.

- b) **Peak Signal to Noise Ratio (PSNR)**: PSNR, as represented in Eq. 2.5, is also one of the most widely used methods, with full-reference image quality measurement. It is mathematically useful in the framework of optimisation (Gogoi and Ahmed, 2016).

$$\text{PSNR} = \frac{10 \times \log_{10}(\text{Intensity})^2}{\text{MSE}} \quad 2.5$$

Limitation of MSE and PSNR: they both have computational complexities while assessing image similarity through distortion types. They cannot model the human visual system.

- c) **Average Difference (AD)**: Calculates the average difference between the original image and the compressed one.
- d) **Maximum Difference (MD)**: Calculates the difference between the original image and the compressed image, by finding the maximum of the error signal.
- e) **Mean Absolute Error (MAE)**: Eq. 2.6 describes the MAE by calculating the average of the absolute difference between the reference image and the compressed image (Gogoi and Ahmed, 2016); (Varnan *et al.*, 2011).

$$\text{MAE} = \frac{\sum_r^j |x1(r, j) - x2(r, j)|}{r \times j} \quad 2.6$$

Where  $r$  and  $j$  are the elements' coordinates, and  $x1$  represents the compressed image pixels and  $x2$  represents the source image pixels.

### 2.1.8 Classification of Compression Techniques

Image compression is mainly classified into two techniques, based on the capability of reconstructing the image after decompressing (Hussain and Al-Fayadh, 2018); (Kavitha and Anandhi, 2015); (Singh, Potnis and Kumar, 2016). These are namely: Lossless compression techniques and Lossy Compression Techniques. These techniques will be described in the following subsections.

## 2.2 Lossless Compression Techniques

Lossless compression results in no data being lost during the compression and decompression phases, the decompressed image should be a bit-for-bit ideal match with the source image. The lossless compression techniques have a low compression rate (Sharma, 2010).

### 2.2.1 Lossless Compression Techniques Phases

Figure 2.5 describes the lossless compression model phases:

1. Decorrelation (Transforming or Mapping): phase one is to remove the inter-pixel redundancy by using any of several decorrelation techniques, such as run-length coding, predictive techniques, or transform techniques (Kuppusamy and Mehala, 2013); (Yang and Bourbakis, 2005).
2. Entropy Coding (phase two): used to reduce a coding redundancy, by using any of the lossless algorithms such as LZW, Arithmetic Coding or Huffman coding (Kodituwakku and Amarasinghe, 2010).
3. Decompressing the image starts at phase three by using the reversible entropy coding algorithm.
4. Phase number four is for reconstructing the image by using the reversible transformation method to reconstruct the decompressed image exactly as it was before compression.

Lossless compression techniques are used in many applications, where any loss of the original image data leads to an improper diagnosis. Some of these applications are the medical application (Lucas *et al.*, 2017), digital radiography, camera systems (Sengupta and Roy, 2018) and remote sensing applications such as monitoring forest fires (Rusyn *et al.*, 2016). The next section is to describe the most popular used lossless algorithms.

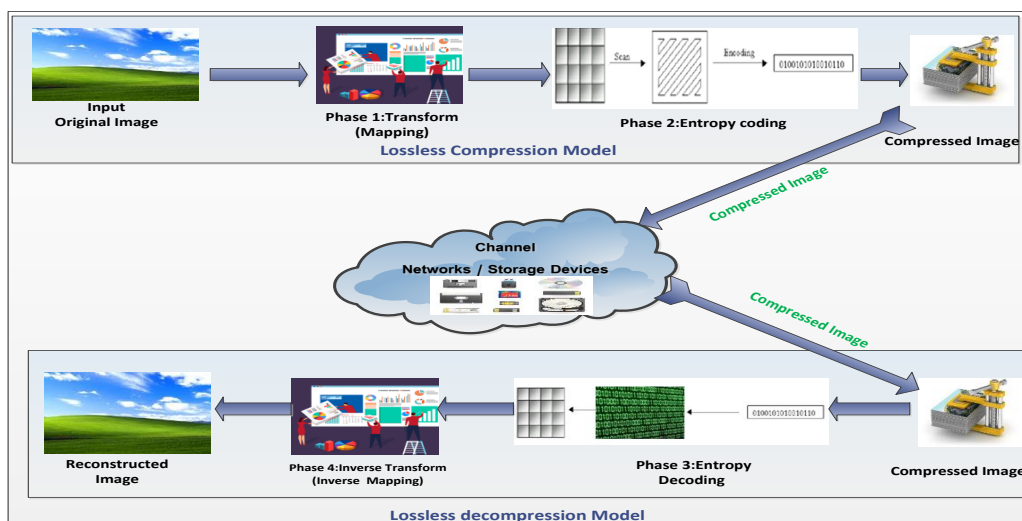


Figure 2. 5 - Lossless Compression Model

## 2.2.2 Lossless Compression Algorithms

Figure 2.6 illustrates the hierarchical diagram that represents the lossless compression techniques phases and each phase-related algorithm (Zaineldin, Elhosseini and Ali, 2015).

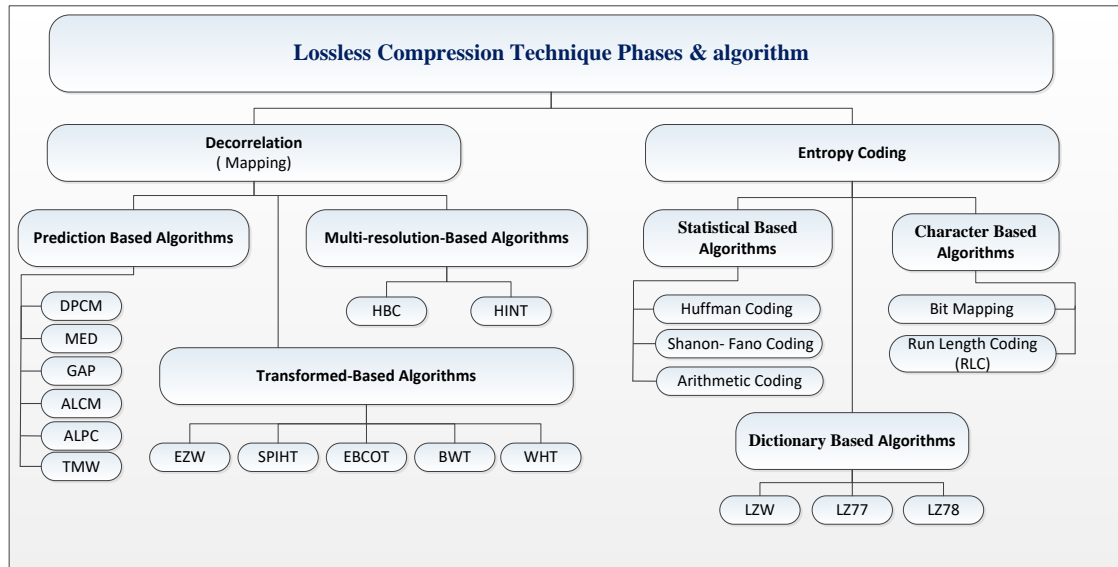


Figure 2. 6 - Lossless Compression Techniques and Algorithms Adaptive from (Zaineldin, Elhosseini and Ali, 2015)

1. Decorrelation techniques use the following methods to remove the inter-pixel redundancy:
  - a) Prediction-based methods use the following image compression algorithms (Shukla, Alwani, and Tiwari, 2010).
    - i. Differential Pulse Code Modulation (DPCM) (Oliveira *et al.*, 2013).
    - ii. Median Edge Detection (MED) (Dadgostar and Afsari, 2016).
    - iii. Gradient Adaptive Predictor (GAP) (Novikov, Egorov, and Gilmudinov, 2016).
    - iv. Activity Level Classification Model (ALCM) (Karimi *et al.*, 2015).
    - v. TMW (Shukla, Alwani and Tiwari, 2010); (Meyer and Tischer, 1997).
    - vi. Adaptive Linear Prediction and Classification (ALPC) (Motta, Storer, and Carpentieri, 2000).
  - b) Multi-resolution-based method used the following image compression algorithms (Carreto-Castro *et al.*, 1993).
    - i. Hierarchic Pairwise Coding (HBC) (Begum and Aygun, 2012).
    - ii. Hearing in Noise Test (HINT) (Carreto-Castro *et al.*, 1993).
  - c) Transform-based method use the following image compression algorithms (Shukla, Alwani, and Tiwari, 2010).
    - i. Embedded Zero Tree Wavelet (EZW) (Singh, Potnis, and Kumar, 2016).
    - ii. Set Partitioning in Hierarchical Trees (SPIHT) (Paul *et al.*, 2015).



- iii. Embedded Block Coding with Optimised Truncation (EBCOT) (Wiseman, 2015).
  - iv. Burrows-Wheeler Transform (BWT) (Shukla, Alwani, and Tiwari, 2010).
  - v. WHT (Tajne and Kulkarni, 2015).
2. Entropy Coding techniques uses the following methods to remove coding redundancy (Raghavendra, Sivasubramanian, and Kumaravel, 2018):
- a) Statistical-based such as:
    - i. Huffman coding (Praisline Jasmi, Perumal, and Pallikonda Rajasekaran, 2015).
    - ii. Shannon-Fano Coding.
    - iii. Arithmetic Coding (AC) (Shaw, Rahman, and Routray, 2018)
  - b) Character-based such as:
    - i. Bit Mapping (Raghavendra, Sivasubramanian, and Kumaravel, 2018).
    - ii. Run Length Coding (RLC) (Al-Wahaib and Wong, 2010).
  - c) Dictionary-based such as:
    - i. Lempel-Ziv-Welch (LZW) (Gagie, Gawrychowski, and Puglisi, 2015).
    - ii. LZ77 (Gagie, Gawrychowski, and Puglisi, 2015).
    - iii. LZ78 (Gagie, Gawrychowski, and Puglisi, 2015).

### **2.2.3 Lossless Compression Related Work**

The relevant literature has been reviewed. All of the conducted related work aims to improve the compression rate and preserves the image quality with zero distortion. The main parameters that may affect any compression algorithm are the compression rate, computational complexity (explained in ch-3) and image quality (Mehanna, 2013). The researcher reviewed the last few years' related work, to understand the weaknesses and strengths of the most popularly used techniques. Understanding the weaknesses can help in avoiding them in the proposed strategy to reach the research objectives. The following section describes the current state-of-the-art in lossless compression algorithms.

#### **2.2.3.1 Huffman Coding**

David A. Huffman presented his coding algorithm in 1952, which depends on the statistical model of data, which starts with measuring the frequency of occurrence for each symbol (giving for each intensity its weight), and then gives prefix codes to those symbols according to their probabilities (creates a frequency Table of the symbols). Shorter codes will be assigned to the more frequently occurring symbols while larger codes will be assigned to the less frequently occurring ones. After that, the Huffman tree is formed to extract the Huffman codes for each symbol. Huffman coding removes the redundancy from the two-dimensional image

and converts it into a one-dimension row of bits by assigning a binary code for all the image intensity. This code is located in the Huffman Table (Huffman, 1951). Huffman algorithm has a high-quality performance; therefore, it is one of the most popular lossless compression algorithms. One limitation of Huffman is that when having more values in the dictionary Table, this will increase the size of Huffman tree, which affects the code-word size to be larger (Vidhya *et al.*, 2016) (Kumar *et al.*, 2015) (Gupta, Bansal and Khanduja, 2017).

#### **2.2.3.2 Shannon's Coding**

Shannon's algorithm is similar to the Huffman coding algorithm; the code-word creation is the only difference, as they use different tree structures in creating code-words. Shannon algorithm has a high-quality performance and low compression rate (Zhang *et al.*, 2015).

#### **2.2.3.3 Bit Plane Slicing (BPS)**

The Bit Plane Slicing (BPS) is a greyscale image compression technique, which splits the source image into 8-bit planes for the 256-level images. Coloured images should have 24-bit planes, since it has three channels (RGB) and each channel is represented with 8-bit planes. After splitting the image into its bit planes, the eight binary images can be compressed by using any entropy coding algorithm such as Run Length Coding (RLC), Huffman or Lempel-Ziv-Welch (LZW) (Maheshwari *et al.*, 2019).

#### **2.2.3.4 Different Plus Coding Modulation Followed by Huffman**

(Tomar and Jain, 2015) produced a new lossless compression algorithm based on the differential pulse code modulation (DPCM) followed by the Huffman algorithm, to enhance the image compression rate. They developed a new transformation algorithm to improve the (DPCM) and named it an Enhanced Differential Pulse Code Modulation Transformation (EDT). The technique has two phases. The first phase is the transformation phase, which uses the EDT algorithm to remove the inter-pixel redundancy, whereas the second phase is the entropy encoding by using the Huffman algorithm to reduce the coding redundancy. The new technique enhanced the lossless compression rate and complexity, but this enhancement is still far away from a lossy techniques compression rate. The DPCM is efficient for lossless compression and near-lossless medical image compression. One of the DPCM limitation is the large error signal when there is a sharp change in the input image edges (Hussain and Al-Fayadh, 2018) (Sanchez, 2015) (Oliveira *et al.*, 2013).

#### **2.2.3.5 Improved Lempel-Ziv-Welch**

The Lempel-Ziv-Welch (LZW) compression algorithm starts with creating a dictionary for a single symbol based on their probability of appearance, then represents each symbol with a

prefix code, starting from the first one to the last. When scanning each symbol, if the symbol is not the last, it will be added to the dictionary until we reach the last symbol in the algorithm output (Yan-li *et al.*, 2010). LZW is a lossless data compression algorithm. It is simple to use but has some limitations, though, such as:

- 1- The LZW algorithm has a problem with the big space redundancy, due to the dictionary file having all the single characters from the source image, but only some of them have been used in the coding process.
- 2- The LZW algorithm has a problem with the small and large dictionary. If the dictionary has a small number of symbols, we will not have a good compression rate; furthermore, when the dictionary has large number of symbols, it will overflow, due to its limited storage.

(Yan-li *et al.*, 2010) proposed a new approach to enhance the lossless LZW algorithm for Wireless Sensor Network (WSN); by understanding the LZW limitation, they avoid the previous problems in the new improved LZW algorithm. The proposed algorithm aims to decrease the dictionary length, which will enhance the compression rate.

The new approach provides the following improvements for the LZW:

1. The dictionary should have no single symbol at the beginning, as it will reduce the dictionary size.
2. The dictionary capacity is stored using two-bytes since the nodes have limited memory space.
3. Reducing the dictionary size by adopting different methods for decreasing the data range; this is positively reflected in high efficiency and memory size.

The proposed algorithms improve the compression rate dramatically by reducing the dictionary size.

#### **2.2.3.6 Lempel-Ziv-Welch with Region of Interest**

The Region of Interest (ROI) image compression algorithm was developed to compress the large medical images. The algorithm starts by locating the region of interest and use the LZW for each region separately. The LZW is a dictionary-based algorithm, which scans the image to find each symbol's probability of appearance and creates the symbol dictionary, and then replaces the symbol with single codes (Kaur and Kaur, 2017). The compression rate is enhanced when using the ROI with LZ77, LZ78, LZW (Singh and Pandey, 2016).

#### **2.2.3.7 Run Length Coding**

The Run Length Coding (RLC), also known as Run Length Encoding (RLE), one of the simplest lossless compression algorithms. It scans the image to find the runs (pixels with the

same value); the runs should be encoded by their probabilities and values (value; probability). This value with its probability is called a unit, and each unit should be coded by using the Huffman algorithm. The best (RLE) results come with the images that have large areas of contiguous colour such as monochrome images (where the value is repeated often). On the other hand, RLE is not an efficient compression algorithm with colour images, since they have many values, with minimum probability of repetition (Husseen, Mahmud and Mohammed, 2017). (Al-Wahaib and Wong, 2010) proposed an algorithm to solve the duplication problem from the traditional RLC algorithm (duplication comes from the escape character). Where the proposed algorithm uses the traditional RLC to decode the three consecutive values (Run) and uses the Traditional Run Length Coding1 algorithm (TRLC1) and the Traditional Run Length Coding2 algorithm (TRLC2) to convert the consecutive two pixels of the same value into a single code-word. The new compression algorithm has a better compression rate than the traditional RLC. The proposed algorithm solves the problem of the large file size of the encoded images and it is considered to be one of the simplest algorithms. RLC is highly efficient with the images that have long runs of pixels with the same value and has low efficiency with images that have high spatial activity, because of the high variation in pixel intensity values. (Szoke, Lungeanu and Holban, 2015) (Al-Laham and Emary, 2007) (Carreto-Castro *et al.*, 1993).

#### **2.2.3.8 Arithmetic Coding**

(Masmoudi, Puech and Masmoudi, 2015) developed a new lossless image compression algorithm by using finite mixture models and adaptive Arithmetic Coding. The algorithm separated the image into blocks and encoded each block separately by using Arithmetic Coding. Arithmetic Coding provides the probability distribution for the image intensity to be compressed. Each block should have its own probability distribution; the statistic for each block will be estimated by a finite mixture model of non-parametric distributions by exploiting the high correlation between neighbouring blocks. The algorithm enhanced lossless compression algorithm efficiency and gave an effective result of 9.7% more than the JPEG-LS, when switching between pixel and prediction error domains (Shaw, Rahman and Routray, 2018). An increase of the image number of blocks leads to a decrease in the compression efficiency; this happens since the total overhead generated by block-based histogram-packing methods depends on the number of blocks used in image and the mapping Table size (Jallouli *et al.*, 2017).

### **2.2.3.9 Median Edge Detection**

The Median Edge Detection (MED) is a filtering technique, which is responsible for removing the image signal (noise). The process of decreasing or removing the noise is a pre-processing phase that should enhance the result of the next image compression phase. The MED predictor has been used in the JPEG-LS format. The MED algorithm predicts the current pixel value by examining three neighbours' pixel values: the north (N), the west (W) and the north-west (NW). If there is a high correlation in the horizontal direction, then the predictive value will be calculated by using the following equation:  $N+W-NW$  (Sanchez, 2015); (Shukla, Alwani and Tiwari, 2010); (Novikov, Egorov and Gilmutdinov, 2016).

### **2.2.3.10 Median Edge Detection and Different Plus Coding Modulation**

The lossless algorithm was developed based on an intra-prediction method and the sample-based angular prediction (SAP) with median and edge prediction (ME): SAP-ME. The main objective to proposing the algorithm is to reduce the average bitrate for the current lossless coding intra-prediction coding. It combines the DPCM, median prediction and edge prediction. The proposed algorithm decreases the average bitrate by 16.13% when compared with the High Efficiency Video Coding Computing (HEVC) intra-prediction coding. The algorithm predicts each pixel value separately by using the neighbouring pixels' values without increasing the computational complexity (Sanchez, 2015).

### **2.2.3.11 Median Edge Detection and Activity Level Classification Model**

A new pixel value prediction algorithm for lossless image compression was developed based on a low-complexity predictors method and neighbouring blocks data analysis followed by an entropy coding. The new algorithm is called local-adaptive block-based prediction (LPB), in which the researchers combined between the MED, ALCM and GAP algorithms and used the best advantage of each. The new algorithm aims to improve the prediction performance, which will enhance the compression rate. The result of the algorithm shows that the LPB has the best result in a comparison with the most popular low-complexity prediction methods (Shukla, Alwani and Tiwari, 2010); (Novikov, Egorov and Gilmutdinov, 2016).

### **2.2.3.12 BBWCA**

The Bi-level Burrows-Wheeler Compression (BBWCA) algorithm is developed based on block sorting structure and it does not compress the image by itself, but works as a pre-processing phase to prepare the image to be compressed by the next phase as in Kernel Move-To-Front (KMTF). (Khan *et al.*, 2017) modified the KMTF algorithm and developed the BBWCA by using the Reversible Colour Transform (RCT) to transfer the image from the RGB colour space

to the UVU colour space with less coloration between the image pixels. The BBWCA achieve high compression ratio with zero distortion (Burrows and Wheeler, 1994).

## 2.3 Lossy Compression Techniques

Lossy compression indicates that some information will be missed during the image decompression phase. It depends on the fact that digital images save a lot of information, more than a human can understand, so it evacuates some of the less important information from the original image. Lossy compression techniques have a high compression rate but also they cannot reconstruct the image exactly as it was, due to the loss of the less important information (Sharma, 2010).

### 2.3.1 Lossy Compression Techniques Phases

Lossy compression techniques mostly have three compression phases, as displayed in Figure 2.7. Phase one is responsible for eliminating the inter-pixel redundancy, in phase two, the quantizer is used to eliminate another kind of redundancy, called psycho-visuals, and create quantized bits as an output, which compress the image a second time. Finally, in phase three, the techniques compresses the image for the third time by encoding the quantized bits (Hussain and Al-Fayadh, 2018); (Sood, Bhathal and Singh, 2018).

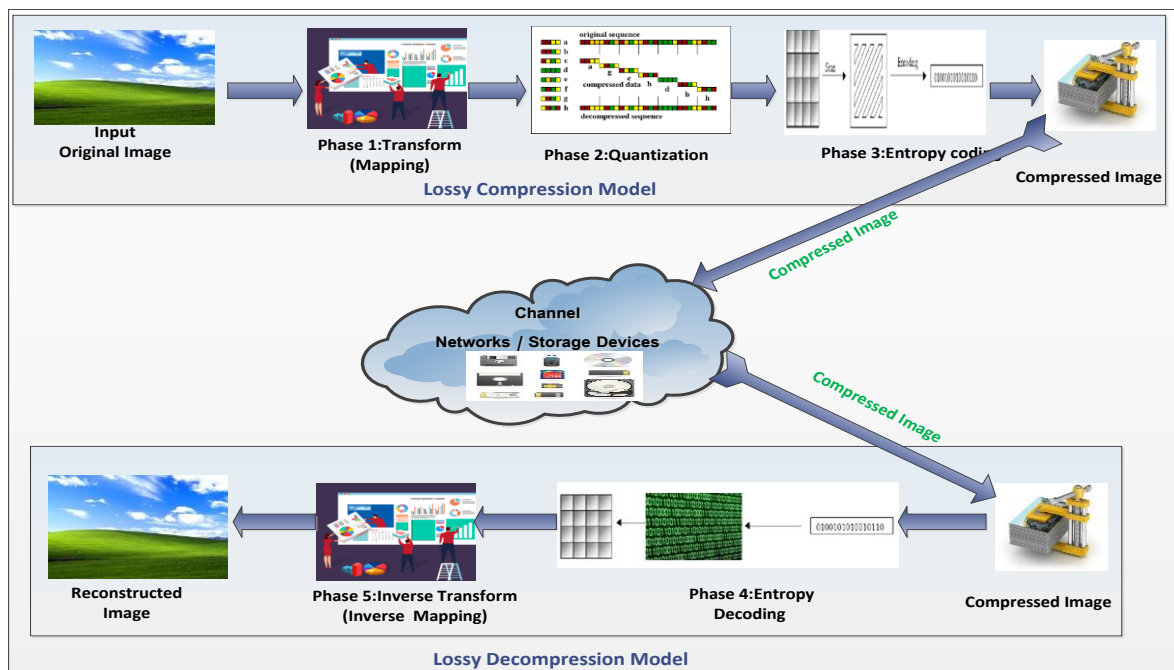


Figure 2. 7 - Lossy Compression Model

To decompress the compressed image, we should inverse the compression phases, starting with the entropy decoding as in phase four. Finally, the inverse transform is used in phase five to reconstruct the image. Lossy compression is used in many applications such as transferring

images through the internet (Abubaker, Eshtay and AkhoZahia, 2016) and the construction of image vegetation (Hussain and Al-Fayadh, 2018). The next section is to describe the most popular used lossy algorithms.

### 2.3.2 Lossy Compression Algorithms

Figure 2.8 shows the hierarchical diagram that represent the lossy compression techniques phases and each phase related algorithm (Zaineldin, Elhosseini and Ali, 2015).

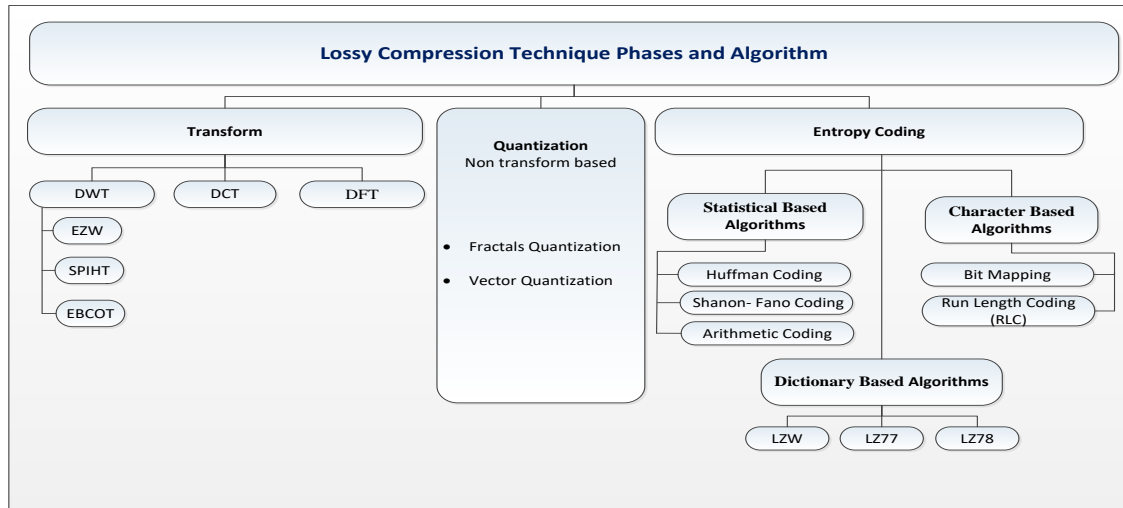


Figure 2. 8 - Lossy Compression Techniques and Algorithms Adaptive from (Zaineldin, Elhosseini and Ali, 2015)

1. Transformation algorithms such as:
  - i. Discrete Cosine Transform (DCT) (Masmoudi, Puech, and Masmoudi, 2015).
  - ii. Discrete Wavelet Transform (DWT) (Mofreh, Barakat, and Refaat, 2016).
  - iii. Embedded Zero Tree (EZW) (Kumar, Kumar, and Singh, 2016).
  - iv. Set Partitioning in Hierarchical Coding Techniques (SPIHT) (Paul *et al.*, 2015).
  - v. Embedded Block Coding with Optimal Truncation (EBCOT) (Wiseman, 2015).
  - vi. Discrete Fourier Transform (DFT) (Ouyang, Coatrieux, and Shu, 2015).
2. Quantization algorithms such as:
  - i. Vector Quantization (VQ) (Karri and Jena, 2016).
  - ii. Fractals Quantization (Ibrahim, Youssef, and Elkaffas, 2015).
3. Entropy Coding techniques uses the following methods to remove coding redundancy (Raghavendra, Sivasubramanian, and Kumaravel, 2018):
  - a) Statistical-based such as:
    - i. Huffman coding (Praisline Jasmi, Perumal, and Pallikonda Rajasekaran, 2015).
    - ii. Shannon-Fano Coding (Zhang *et al.*, 2015).
    - iii. Arithmetic Coding (AC) (Shaw, Rahman, and Routray, 2018)

- b) Character-based such as:
  - i. Bit Mapping (Raghavendra, Sivasubramanian, and Kumaravel, 2018).
  - ii. Run Length Coding (RLC) (Al-Wahaib and Wong, 2010).
- c) Dictionary-based such as:
  - i. Lempel-Ziv-Welch (LZW) (Gagie, Gawrychowski, and Puglisi, 2015).
  - ii. LZ77 (Gagie, Gawrychowski, and Puglisi, 2015).
  - iii. LZ78 (Gagie, Gawrychowski, and Puglisi, 2015).

Lossy compression techniques can be a transform- or non-transform-based techniques:

1. Transform-based techniques are used to reduce the correlation in a vector (the original image) by transforming it into a new, less correlated image, and then compressing it in the quantization phase; some of the algorithms used in this technique are DCT and DWT.
2. Non-transform-based techniques depend on the vector quantizer as the quantization process. Quantizers have two types: scalar and vector quantizers (Zaineldin, Elhosseini, and Ali, 2015).

### **2.3.3 Lossy Compression Related Work**

The relevant literature has been reviewed. All the related work from the literature aims to improve the compression rate and provide less distortion for the compressed image quality. To understand the weaknesses and strengths of the most popularly used techniques, the researcher analyses the related work, to avoid the limitation from the current algorithms. The following sections describes the current state of the art lossy compression algorithms.

#### **2.3.3.1 JPEG**

In recent years, the use of greyscale image and colour image is increasing, and an international compression standard is needed to represent both types of images (Mehanna, 2013). Accordingly, to this need, the International Organization for Standardisation (ISO) together with the International Telegraph and Telephone Consultative Committee (CCITT) created the JPEG standard for images and continuous-tone still images (video), such as MPEG and H.261 (Jeon, Park, and Jeong, 1998) (Wallace, 1992). The JPEG compression algorithm used the most efficient and most common transform techniques, DCT (known as a lossy compression techniques), to improve the compression process (Lakhani, 2004). This was followed by entropy coding (a lossless compression techniques). The JPEG algorithm starts by converting the RGB image into Y,Cb,Cr colour space by separating the luminance and the chrominance. Then, the algorithm divides the image into 8\*8 blocks of pixels and transforms each block separately using forward DCT (F-DCT) to create the DCT coefficients (a number that



represents the contribution of each block to the image). This coefficient will be normalised by a specific quantization Table to get normalised coefficients (the process of removing the high-frequency cosine waves that should have the smallest coefficient value, since they do not really contribute very much to the image), and then an entropy coding algorithm, such as the Huffman algorithm, will be used to reduce redundancy. As a result, the JPEG algorithm enhances the compression rate. DCT-based algorithms, such as JPEG standard, cannot satisfy the image coding required today because of its distortion. Although it provides good image quality, there is still some distortion in the images, and several applications will not be satisfied using it, such as medical applications. A higher JPEG compression rate leads to more distortion, due to the artefacts that may result from the block based DCT. JPEG 2000 standard was developed to solve the JPEG problem. JPEG 2000 is a Wavelet-based coding that enhances the image quality at low bit rates (Wei, 2008).

### **2.3.3.2 JPEG 2000**

The JPEG 2000 standard was developed to meet the new needs of image compression, such as enhancing the compression rate and image quality more than the JPEG standard. The JPEG 2000 could be lossless or lossy techniques regarding the use of the JPEG 2000 quantizer (Reversible Quantizer or Irreversible Quantizer). In addition, the JPEG 2000 uses its transformation based on the Region-of-Interest Coding (ROI), and it has a powerful mechanism for Error Resilience (Mehanna, 2013); (Wei, 2008).

The first process in JPEG 2000 is the Reversible Colour Transform (RCT), which is a modified YUV colour transform that will not lead to any quantization error. The second process is the use of subband coding application. Discrete Wavelet Transform (DWT) is used in the JPEG 2000 to reduce the image size by reducing the undesirable noise. The third process is the quantization, which is conducted to decrease the number of bits that are required to represent the transform coefficients. The final phase is the Entropy Coding, which is done by using the embedded Block Coding with Optimised Truncation (EBCOT). EBCOT has two phases: the first one uses the context formation and Arithmetic Coding by dividing the DWT coefficients into code blocks, and each block should be encoded separately into block-based bit-stream; the second entropy phase is to minimise the embedded bit-stream. In comparison, the JPEG 2000 enhances the image quality and the image compression rate more than the JPEG standard by using the advantages of DWT, such as better energy compression property (Manjinder and Gaganpreet, 2013) (Sudhakar, Karthiga and Jayaraman, 2005).

### **2.3.3.3 Minimise-Matrix-Size algorithm**

Developing a new compression algorithm that enhanced the image quality more than the most popular compression techniques such as JPEG and JPEG 2000 is increased. Regarding this need, the Minimise-Matrix-Size algorithm was developed. This algorithm has four steps: first, transforming the image by using a two-level DCT and a two-level DWT to create two matrices, one for DC and the other one for AC (low- and high-frequency matrix); second, applying the DCT again, but for the DC matrix only to create non-zero array and zero array; third, applying the Minimise-Matrix-Size algorithm to the AC-matrix and to the high frequencies from the second step; finally, the entropy coding, which is conducted by using AC algorithm on the data resulting from the third step. The high frequencies are calculated by using a Fast-Match-Search (FMS) algorithm through calculating the probabilities of the compressed data by using the Table of probabilities. The Minimise-Matrix-Size algorithm enhanced the image quality more than JPEG and JPEG 2000, as it removes the block artefacts from the 8x8 DCT matrix, and also by using the levels of DWT (single level or two level) instead of multi-level DWT, which reduced the blurring that typically exists in JPEG 2000. The main limitation of this approach is the large number of steps and the execution time is more than JPEG 2000 standard (Siddeq and Rodrigues, 2015).

### **2.3.3.4 Discrete Cosine Transform Followed by Huffman**

A new hybrid compression algorithm is developed to enhance the image compression rate by reducing the image redundancy. The new algorithm is a combination of the Discrete Cosine Transform (DCT) and the Huffman coding algorithm. The algorithm starts by removing the inter-pixel redundancy using the DCT algorithm as a transforming phase. Followed by the Huffman algorithm to encode the image as a second phase. The algorithm enhanced the quality of the compressed image with high PSNR value, and a higher compression rate was achieved (Shukla and Gupta, 2015).

### **2.3.3.5 Different Plus Coding Modulation, Discrete Wavelet Transform Followed by Huffman**

A new medical image compression algorithm was developed by (Abo-Zahhad *et al.*, 2015) to enhance the compression rate. The algorithm called (DPCM-DWT-Huffman) and involved three phases. The first phase is the pre-processing phase, which uses Differential Pulse Code Modulator (DPCM) as transformation algorithm; the second phase receives the outcomes from the first phase and uses them as input to the wavelet transformed (DWT) to reduce the redundancy and spatial reputation; the third phase uses the Huffman algorithm to encode the coefficients that resulted from the second phase. The algorithm performance has been measured

for the three phases: the new algorithm enhanced the compression rate (Abo-Zahhad *et al.*, 2015). (Mofreh, Barakat and Refaat, 2016) Emphasise that the enhancement in the compression rate using the three phases algorithm (DPCM-DWT-Huffman) is done, but the image compression algorithm is still suffering from the low compression rate. Furthermore, the compression rate can be even better if we used some other compression techniques, such as LPCDH, which presents the Linear Predictive Coding (LPC) with Discrete Wavelet Transform (DWT) followed by Huffman (LPCDH).

#### **2.3.3.6 Radial Basis Function Neural Networks and Discrete Wavelet Transform**

Developing a new image compression algorithm that can be employed in both lossless and lossy compression techniques is required to meet the organizational demands. A new algorithm was developed in this regard by (Wozniak *et al.*, 2015) and called the (DWT-RBFNN). The algorithm was developed based on Radial Basis Function Neural Networks (RBFNN) and has two processes: the first process uses the Radial Basis Function Neural Networks (RBFNN) to decrease the wavelet coefficients' numbers while maintaining the main features; the second process implements the DWT algorithm on the outcomes from the first process. The new algorithm enhanced the performance of time consuming and compression rate (Wozniak *et al.*, 2015).

#### **2.3.3.7 Discrete Wavelet Transform and Set Partitioning in Hierarchical Coding Techniques**

(Vijayaran and Sakila, 2016) studied the current lossless and lossy compression techniques and compared them according to their compression rate and image quality. Then, they proposed a new hybrid image compression technique, which combined the DWT with the Set Partitioning in Hierarchical Trees (SPIHT). The result of the hybrid algorithm produces better accuracy than DWT and SPIHT.

#### **2.3.3.8 Rounding the Intensity Followed by Dividing**

Lossy Image Compression by Rounding the Intensity Followed by Dividing (RIFD) algorithm is developed to reduce the image redundancy as much as possible; also, it is to reduce the image bit-depth to 5 bits instead of 8 bits. More decreasing of the bit-depth should provide a higher compression rate. For example, when reducing the bit-depth from 8 bits to 5 bits then the image is compressed. The algorithm starts by rounding the intensities to the nearest ten, then the image should be divided by ten to reduce the bit-depth from 8 bits to 5; the resulting matrix should be encoded by the Huffman algorithm. The algorithm is suitable for greyscale image and coloured image; since colour images are represented by RGB, then the algorithm should be applied three times (one time for each colour). If the bit-depth in the original image is 16 bits, then the round

function should be rounded to the nearest thousand instead of ten. The algorithm enhances the compression rate, but it still has some distortion regarding the use of the rounding function; this distortion is not visible by the human visual system (Otair and Shehadeh, 2016).

#### **2.3.3.9 Fractal Compression**

Fractal compression is a lossy compression method, used fractals instead of pixels. Fractal compression is suitable for natural images. It takes advantage from the similarity between image fractions, since a part of the image can be identical to other parts. (Ibrahim, Youssef and Elkaffas, 2015) introduced a new fractal compression algorithm, by using quantized quad trees and entropy coding. The algorithm divided the quantized image into various blocks, via using the threshold value and the most important features presented in the image. Then, they used an entropy coding algorithm to improve the compression rate. After testing the new algorithm and comparing the results with the fractal image compression and iterations technique, there was a good improvement in compression rate and image quality (Ibrahim, Youssef and Elkaffas, 2015).

#### **2.3.3.10 EZW**

The EZW coding is an effective image-coding algorithm based on a wavelet transform and works with lossless and lossy compression techniques. It depends on the fact that bits in the bits array are generated based on their importance. EZW encoder depends on progressive encoding to compress an image into a bit stream with increasing accuracy (Senturk and Kara, 2016).

The main important observations of EZW encoder are:

- Low pass spectrum (low scale means high resolution – highly correlated) usually comes with natural images. When an image is wavelet transformed, the subband energy will be decreased as the scale decreases, so the higher subband wavelet coefficients will be smaller than those of the lower subband.
- Smaller wavelet coefficients are less important than the larger.

EZW uses a sequential approximation quantization process to present multi-representations of the transformed coefficients. EZW coded the transformed coefficients in decreasing order in several scans, where each scan has two phases: significant map encoding and refinement pass.

Some of the EZW limitation are:

- Requires complicated bit allocation procedures (Kale and Deshmukh, 2010).
- Pixel values of detailed images are mostly composed of zeros or close to zero, therefore, quantizing them to zero leads to visually visible blurring. (Sudhakar, Karthiga and Jayaraman, 2005).

## 2.4 Summary and Comparison of Fundamental Algorithms

This section summarises the most important lossless algorithms by focusing on their advantages and disadvantages, as listed in Tables 2.1.

Table 2. 1 - Summary of Lossless algorithms.

Algorithms	Short Description	Advantages	Disadvantages
Huffman algorithm (Huffman, 1951)	Huffman coding rely on the statistical model of data, which starts by measuring the frequency of occurrence for each symbol (giving for each intensity its weight), and then gives prefix codes to those symbols according to their probabilities (creates a frequency table of the symbols). Shorter codes will be assigned to the more frequently occurring symbols while larger codes will be assigned to the less frequently occurring symbol. After that, the Huffman tree is formed to extract the Huffman codes for each symbol. Huffman coding removes the redundancy from the two-dimensional image and converts it into a one-dimension row of bits by assigning a binary code for all the image intensity. This code is located in the Huffman table.	Huffman algorithm stands as one of the best compression algorithms for JPEG and many other standards, due to its high compression rate.  Huffman algorithm has a high-quality performance; therefore, it is one of the most popular lossless compression algorithms.	We cannot use the Huffman algorithm unless we know the probability distribution for each symbol.  Likewise, we cannot use the Huffman algorithm when changing the source statistics.  The number of code-words should be equal to the number of unique values in the table of probability. More values in this table will increase the size of Huffman tree, which affects the code-word size to be larger.
RLC algorithm Al-Wahaib and Wong, (2010);	RLC, known as RLE, scans the source image (row by row from top left, downwards) to find any repeating values (redundancies) and classifies this symbol to runs for the consecutive sequences of the data (three as minimum) or non-runs for the other data. Quite often, most of the images have rows of the same colour. Then, RLE saves each runs value and its length in a file. The	Simple to use.  Highly efficient with the images that have long runs of pixels with the same value (such as black and white images) For example, the runs (77778889999) will be	Low efficiency with images that have high spatial activity, because of the high variation in pixel intensity values. For example, the non-repeated sequence (non-runs) such as (6 7 8 9) will be decoded as (1 6 1 7 1 8 1 9), which increases the

	<p>compression algorithm uses only the runs to compress the source file.</p> <p>RLC is used to compress the images or as a pre-processor for other compression algorithms.</p> <p>An RLC-encoded file will be decoded again, by using any entropy coding algorithms to create variable length code-words.</p>	<p>decoded as (4 7 3 8 4 9), the file size is reduced from (11) to (6).</p> <p>High image quality (Lossless image compression technique)</p>	<p>cardinality (file size) instead of decreasing it.</p>
<p>LZW algorithm</p> <p>Nelson, M. R. (1989)</p>	<p>LZW is a dictionary-based algorithm derived from the LZ78 algorithm. It scans the image to find each symbol's probability of appearance and creates the symbol dictionary. It then replaces each symbol from the dictionary with a single code.</p>	<p>High image quality (Lossless image compression technique).</p> <p>High performance algorithm in terms of execution speed.</p> <p>LZW is an adaptive algorithm; because of the dynamic creation of the dictionary in both processes (compression and decompression), there is no need to transmit the dictionary with the compressed message.</p>	<p>LZW has a problem with the big space redundancy, due to the dictionary file having all the single characters from the source image, but only some of them having been used in the coding process.</p> <p>LZW also has a problem with the small and large dictionary, where if the dictionary has a small number of symbols it will not have a good compression rate, and when the dictionary has a large number of symbols it will be overflowing, due to the dictionary limit storage.</p> <p>For the low repetition rate, the dictionary size will be increased and its contribution to the compression rate improvement will be less.</p>

## **2.5 Chapter Summary**

This chapter presented a comprehensive review of the literature for the lossless and lossy compression techniques by focusing on the lossless techniques as it is the pivotal contribution of this research. Both techniques were critically reviewed to understand their impact on the current challenges related to the image compression domain by reviewing each technique development model; characteristics; advantages and limitations. The next chapter provide a gap analyses for the current most popular used algorithm's by describing their limitation and the proposed solution for each of the limitations, followed by the development process.

## **CHAPTER THREE: PROPOSED SOLUTIONS AND DEVELOPMENT TOOLS**

---

### **3 Chapter Overview**

This chapter covers the proposed solutions and the research requirements that are needed to develop the proposed solutions, such as the development tools, the used programming language and the tested image sets.

---

#### **3.1 Proposed Solutions**

From the previously mentioned literature review from chapter two, it can be understood that the ideal image compression algorithm should provide a high image quality performance and good compression rate. To achieve this objective, the researcher reviewed and analysed the literature to understand the parameters that can affect the compression rate and the image quality. Lossless compression techniques have good quality performance and low compression rate, whereas lossy compression techniques have low quality performance and higher compression rate.

In lossless compression techniques, the low compression rate is mostly related to the number of unique values needed to represent the image. If the number of unique values is large, this will affect the probability Table in the entropy algorithm to have more data and the dictionary file size should be larger as well. In fact, entropy coding is a lossless compression technique that has a great influence on both lossless and lossy compression techniques. It has three approaches: dictionary based, statistical based, and character based. All these approaches share some steps in their coding process. They scan the image to calculate each value probability of appearance and store these values and their probability in a file (a dictionary or Table); this file will be used to give codes for each unique symbol by using their probabilities. Each symbol should represent the decoded image as a string of code-words or as a one-dimension row of bits instead of the two-dimensional matrix. Huffman coding, Arithmetic Coding, run-length encoding, LZW and Golomb-Rice coding are different algorithms that were used in entropy coding; they all suffer from a low compression rate when compressing high resolution images. To solve the problem of the low compression rate in the entropy coding approaches, we need to understand how the algorithm works and what the limitations of the most popular entropy coding algorithms are. The reviewed literature helped the researcher to determine some of these limitations.



### **3.1.1 Huffman Limitation and the Proposed Solution**

In Huffman coding, the table of probability will be created to use the symbol probabilities to create the code-words for each unique symbol in the dictionary table. Shorter codes will be assigned to the most frequently occurring symbols, whereas larger codes will be assigned to the least frequently occurring symbols. The number of code-words should be equal to the number of unique values in the table of probability. More values in this table will increase the size of the Huffman tree, which enlarges the code-word size (Vidhya *et al.*, 2016). Storage in a single word is complex because of the least symbol probability represented by a large length of code (Gupta, Bansal and Khanduja, 2017); (Hazarika, Nath and Bhuyan, 2015); (Muntean, Căbulea and Vălean, 2014).

Reducing the image bit-depth before compressing the image using any entropy algorithms should enhance the compression rate, the proposed algorithm uses a pre-processing transformation phase to decrease the image bit-depth before compressing, followed by the CSC compression phase to decrease the image size and to represent the compressed image with less number of unique values.

### **3.1.2 LZW Limitation and the Proposed Solution**

The Lempel-Ziv-Welch (LZW) algorithm is a dictionary-based approach. It scans the image to find each symbol's probability of appearance and creates the symbol dictionary, and then it replaces the symbol with single codes (Kaur and Kaur, 2017). LZW has a problem with the big space redundancy; this is due to the dictionary file having all the single characters from the source image, only some of which were used in the coding process. In addition, it has a problem with the small and large dictionary where if the dictionary has a small number of symbols, we will not have a good compression rate, and when the dictionary has a large number of symbols, it will be overflowing, due to its limited storage capacity (Li *et al.*, 2018); (Yan-li *et al.*, 2010). Reducing the dictionary size by adopting different methods for decreasing the data range will increase the probability value; this is positively reflected in the compression rate. The proposed algorithms use well defined functions to represent the image with a smaller number of unique values.

### **3.1.3 RLC Limitation and the Proposed Solution**

The Run Length Coding (RLC), also known as Run Length Encoding (RLE), is one of the simplest lossless compression algorithms. It scans the image (matrix) to find the runs (pixels with the same value); the runs should be encoded by their probabilities and values (value;

probability). This value with its probability is called a unit. The RLC best results come with the images that have large areas of contiguous colour such as monochrome images (where the value is repeated often). On the other hand, RLC is not an efficient compression algorithm with colour images (since they have many values, with minimum probability of repetition) (Husseen, Mahmud, and Mohammed, 2017).

Reducing the colour image bit-depth will decrease the data range and increase the probability value; this positively influences the compression rate. The proposed algorithm uses a pre-processing transformation phase to decrease the image bit-depth before compressing to decrease the data range; followed by using the CSC function for decreasing the number of unique values.

This research aims for developing new lossless compression algorithms to enhance the current algorithms compression rate and execution time by using a pre-processing transformation phase for mapping the input image into new colour space to prepare the image to be compressed by using the proposed CSC function; the CSC compression phase is to decrease the image size by reducing the image bit-depth and to represent the compressed image with less number of unique values.

### **3.2 Software Development Process**

The proposed compression system model is developed based on the incremental prototype methodology, where the system is divided into prototypes that represent the main system functions, then the prototypes were developed and tested individually to provide feedback for enhancing each of the current prototype by reworking it until it achieved the function aim. Eventually, the different prototypes are merged into a single system to deliver a compressed image.

(Hoffer, J. A., 2012) describe the incremental prototyping process phases as:

1. Requirements gathering and analysis.

This phase identifies the system requirements to define the system expectations through understanding the gap in the current techniques.

2. Quick design.

A simple design of the system is created to help in developing the prototype by providing a brief idea of the compression system. This phase is responsible for identifying the main system functions (Prototypes).

3. Build a Prototype.

A small model of the required system is developed based on the quick design. The model should combine all the prototypes to deliver a compressed image and should be ready for evaluation.

4. Initial user evaluation.

The first prototype is ready for evaluation to find out its strength and weakness. This phase should provide many feedback points to the developers. The evaluation phase is to evaluate the output image size, quality, execution time and the algorithm complexity.

5. Refining prototype.

This phase uses the feedback from the previous phase to refine the current prototype. Once the Initial user evaluation phase provides zero suggestion, then this phase is over, and a final system is developed based on the approved final prototype.

6. Implement Product and maintenance.

The final system is developed based on the final prototype. The system deliverables (compressed images) are evaluated in chapter four and five.

### **3.2.1 The first Iteration Requirements**

The first compression model was designed, developed and tested in this iteration, to analyse its strengths and weaknesses to provide feedback for the next iteration. The system development phases are described as:

1. Requirements gathering and analysis phase.

The system should compress the input image with high compression rate and fast execution time with zero percentage of distortion.

2. Quick design phase.

A simple design of the system is created to help in developing the prototype by providing a brief idea of the compression system. This phase is responsible for identifying the main system functions (Prototypes) as follows.

The proposed CSC compression system is divided into two prototypes:

- Read the input image and determine its information.
- Column Subtraction Compression.

Each of the functions (prototypes) should be programmed individually as a class and should be called in the final system by using an object-oriented structure.

3. Build a Prototype.

A small model of the required system is developed based on the quick design. The model should combine the two prototypes to deliver a compressed image and should be ready for evaluation.

4. Initial user evaluation.

The evaluation phase is to evaluate the output image size, quality, execution time and the algorithm complexity. The first model achieved high compression speed performance with zero distortion and less compression rate than expected.

5. Refining the prototype.

Having the feedback from the previous phase, we need to enhance the compression rate in the next iteration without effecting the compression speed and the image quality.

### 3.2.2 The Second Iteration Requirements

The second compression model was designed, developed and tested in this iteration to enhance the system compression rate and to analyse its strength and weaknesses points to provide feedback for the next iteration. The system development phases are described as:

1. Requirements gathering and analysis phase.

The second system should enhance the compression rates of the first model by compressing the input image with a higher compression rate and fast execution time with zero percentage of distortion.

2. Quick design phase.

An updated version of the first design of the system is created to enhance the compression rate. This phase is responsible for identifying the second system main functions (Prototypes) as follows.

The second compression system is divided into three prototypes:

- Read the input image and determine its information.
- Transformation
- Column Subtraction Compression.

Each of the functions (prototypes) should be programmed individually as a class and should be called in the final system by using an object-oriented structure.

3. Build a Prototype.

A small model of the required system is developed based on the quick design. The model should combine the three prototypes to deliver a compressed image and should be ready for evaluation.

4. Initial user evaluation.

The evaluation phase is to evaluate the output image size, quality, execution time and the algorithm complexity. The second model achieved high compression speed performance with zero distortion and enhances the compression rate. The only limitation of this system is when compressing low-resolution images; the system achieved less compression rates.

5. Refining prototype.

Having the feedback from the previous phase, we need to enhance the compression rate of the low-resolution images in the next iteration without effecting the compression speed and the image quality.

6. The final compression system for the High-Resolution images is developed based on the final prototype. The system deliverables (compressed images) are evaluated in chapter four.

### 3.2.3 The third Iteration Requirements

The third compression model is designed, developed and tested in this iteration, to enhance the compression rate for the low-resolution images. The system development phases are described as:

1. Requirements gathering and analysis phase.

The third system is to enhance the compression rates of the second model for the low-resolution images by compressing the input image with a higher compression rate and fast execution time with zero percentage of distortion.

2. Quick design phase.

An updated version of the second design of the system is created to enhance the compression rate. This phase is responsible for identifying the third system main functions (Prototypes) as follows.

The third compression system is divided into five prototypes:

- Read the input image and determine its information.
- Transformation
- Column Subtraction Compression.
- Huffman Algorithm.
- RLE Algorithm.

Each of the functions (prototypes) should be programmed individually as a class and should be called in the final system by using an object-oriented structure.

3. Build a Prototype.

A small model of the required system is developed based on the quick design. The model should combine the five prototypes to deliver a compressed image and should be ready for evaluation.

4. Initial user evaluation.

The evaluation phase is to evaluate the output image size, quality, execution time and the algorithm complexity. The second model achieved high compression speed performance with zero distortion and enhances the compression rate for the low-resolution images. The only limitation of this system is when compressing a high-resolution image; the system achieved less compression rates with high-resolution images.

5. Refining prototype.

Having the feedback from the previous phase, we need to develop a fully automated system to compress the image according its type (Low-Resolution Image or High-Resolution Image). The new system is to enhance the compression rate for the input image.

6. Implement Product and Maintain.

The final compression system for the Low-Resolution images is developed based on the final prototype. The system deliverables (compressed images) are evaluated in chapter five.

### 3.3 Algorithm complexity

This work is about developing algorithms. The proposed algorithms time complexity is measured by using the big O notation to describe the algorithm efficiency in term of execution time. The big O notation focusses on the algorithm efficiency and how it varies according to the size of the input image. (Bsoul, 2011) emphasise that, “the Big-O notation is used to define an upper bound on the worst-case scenario for a given algorithm.

Let  $f(n)$  be a function that approximate the worst-case running time of an algorithm of input size  $n$ . Let  $g(n)$  be a function mapping nonnegative integer to real numbers. We say that  $f(n)$  is  $O(g(n))$ , if there exist a constant  $C > 0$  and an integer constant  $n_0 > 0$  such that  $f(n) \leq Cg(n)$  for sufficiently large  $n \geq n_0$ . This means that  $f$  is asymptotically upper bounded by  $g$ . The definition is often referred to as the “big-oh” notation. In Chapter 4 and 5, an approximation of  $f(n)$  is analysed. and the corresponding  $g(n)$  is proved using Eq. 3.1”.

Let  $f$  and  $g$  be two functions that

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} \quad \text{Eq.3.1}$$

exists and is equal to some number  $c > 0$ . Then  $f(n) = O(g(n))$

The big O notation is used to calculate the algorithms complexities in the evaluation chapters.

The researcher testbed is composed of four parameters which are:

- 1- The algorithm time complexity by calculating the big O notation.
- 2- The compressed image size by calculating the compression ratio.
- 3- The image quality by calculating the (MSE and PSNR).
- 4- The algorithm execution time by using the Tic-Toc Matlab function.

Chapters four and five discuss the evaluation phase in detail.

### 3.4 Matlab

MATLAB is one of the fourth-generation programming languages. It provides an easy platform for numerical analysis and a very suitable language for matrices operations. It provides a graphical user interface and data visualisation. The MATLAB libraries have many pre-defined functions that support most of the image processing researchers; therefore, it is very popular among image processing. MATLAB has many different versions; in this research we will use MATLAB R2016b version. Some of the MATLAB features include:

1. MATLAB supports matrices mathematical operation and provides analysis features that facilitate the programmers' work.
2. MATLAB has a very well-documented toolbox that helps any programmer to develop their applications.
3. With MATLAB, programmers can draw Figures to analyse data without the need to export the data to other software.

Figure 3. 1 describe the compression system class diagram, by describing the main building blocks of the object-oriented methods. The first class (Read Input Image Class) is responsible for reading the image to specify its details, and the second class (Transformation Class) is to map the image into new colour space to prepare it for the next phase. The (CSC Compression Class) is to compress the image by using the proposed CSC function and followed by the Huffman class to decrease the values of the resulted image from the CSC phase. The (RLE Class) is used for decreasing the image size as a final phase.

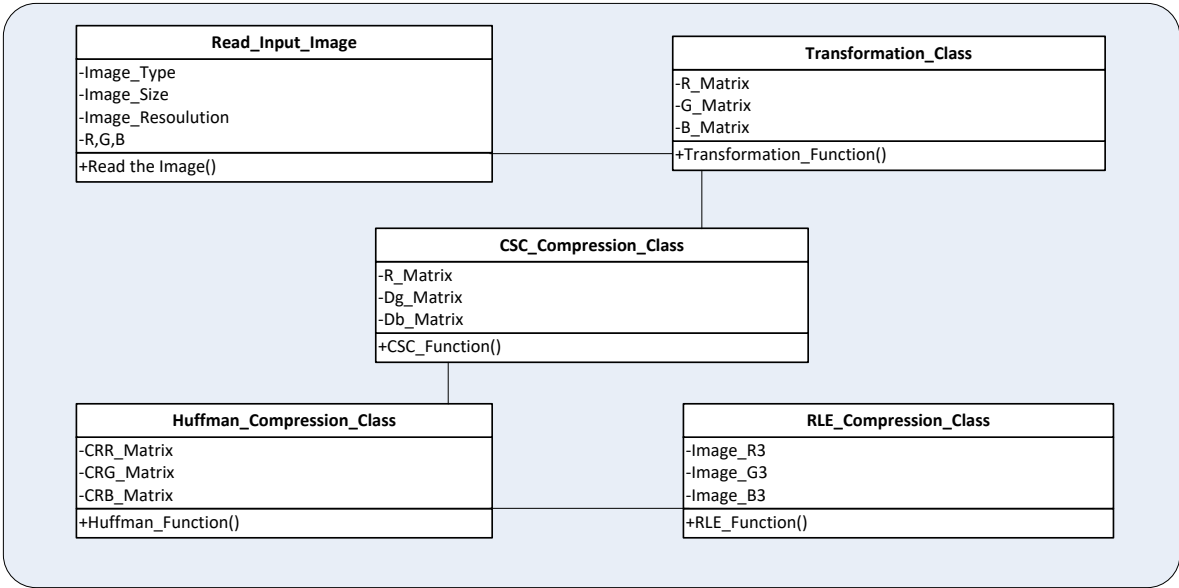


Figure 3. 1 - The Compression System Class Diagram

### 3.5 Used Devices and Operating System

The researcher started programming the algorithm by using MATLAB programming language on Microsoft Windows 10 operating system, and a laptop. Table 3.1 lists the hardware specifications.

Table 3. 1 - The Algorithms Working Environments

Used Device	Dell Laptop
<b>Processor</b>	Intel (R) Core (TM) i7- 7500 CPU @ 2.7 GHz 2.9 GHz
<b>Random Access Memory</b>	8.00 GB
<b>Display Adapter</b>	Intel (R) HD Graphics 620
<b>Operating System</b>	windows 10 64-bit
<b>Programming Language</b>	MATLAB R2016b

### 3.6 Set of Tested Images

To cover all the aspects that may affect the compression performance and to ensure the accuracy and reliability of the results, the researcher used an inclusive set of test images from different image types and different image resolutions. This test image samples should represent the most popularly used formats, such as JPEG, BMP, GIF, PNG, and the most popularly used images in the literature and image processing research community. To reach the best image results, the researcher tested the algorithm on five images sets with a total of 52 images.



### 3.6.1 Image Set 1

Set one, includes nine of the most popularly used greyscale images, including Baboon, Barbara, Boat, Camera Man, House, Lena, as shown in Table 3.2.

Table 3. 2 - Image set 1

Image set 1		
		
Baboon (GIF)	Barbara (PNG)	Boats (PNG)
		
Boats (BMP)	Camera Man (BMP)	Camera Man (GIF)
		
House (PNG)	Lena (PNG)	Lena (JPEG)

### 3.6.2 Image Set 2

Set two, includes nine of the most popularly used coloured images, such as Airplane, Baboon, Barbara, Boats, Gold Hill, Lena, Pepper, as shown in Table 3.3.





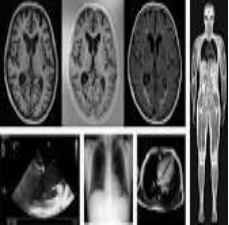
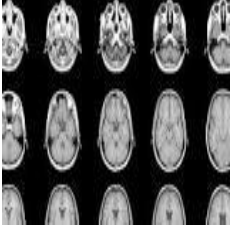

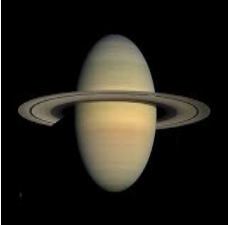






Table 3. 3 - Image Set 2

Image set 2		
		
Airplane (BMP)	Baboon (BMP)	Barbara (BMP)
		
Boats (BMP)	Gold Hill (BMP)	Lena (BMP)
		
Lena (PNG)	Lena (JPEG)	Pepper (BMP)

### 3.6.3 Image Set 3

Set three, includes 14 images from the Stanford image set, as shown in Table 3.4. This image set has more than 9000 images, so we choose different images from this image set with different resolutions, to test their results on the proposed algorithms.

Table 3. 4 - Image Set 3






Image set 3			
			
Boats (JPEG)	Butterfly (JPEG)	Earth (JPEG)	Lake (JPEG)
			
Medic (JPEG)	Medic1 (JPEG)	Mountain (JPEG)	Saturn (JPEG)
			
Swarm (JPEG)	Waterfall (JPEG)	Grandfather (JPEG)	Car (JPEG)
			
Eagle (JPEG)	Shape (JPEG)		



### 3.6.4 Image Set 4

Set four, includes 10 natural images obtained from the Kodak image set <http://www.cs.albany.edu/~xypan/research/snr/Kodak.html>, as shown in Table 3.5. All of the images in this image sets have the same resolution 768 X 512- and 24 bit-depth and PNG data type.

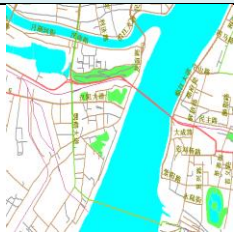









Table 3. 5 - Image Set 4

<b>Image set 4</b>		
		
Knob & Bolt (PNG)	Houses (PNG)	Landscape (PNG)
		
Light House (PNG)	Barn (PNG)	Parrots (PNG)
		
Flowers & Sill (PNG)	Six-Shooter (PNG)	Motocross (PNG)
		
Zentime (PNG)		

### 3.6.5 Image Set 5

Set five, includes 10 raster map images obtained from <https://sites.google.com/site/qinzoucn/documents/> image set, as shown in Table 3.6. All of the images in this image sets have the same resolution 600 X 480 and 24 bit-depth and BMP data type.

Table 3. 6 - Image Set 5

Image set 5			
			
Map 1 (BMP)	Map 2 (BMP)	Map 3 (BMP)	Map 4 (BMP)
			
Map 5 (BMP)	Map 6 (BMP)	Map 7 (BMP)	Map 8 (BMP)
			
Map 9 (BMP)	Map 10 (BMP)		

### 3.7 Rational for the Selection of the Images Sets

All the images had been chosen carefully by the researcher to cover all the aspects that may affect the compression performance. The first and second image sets represent the most used images by the scientific community (Grayscale and Colour). (Al-azawi *et al.*, 2011); (Kale and Deshmukh, 2010) and (Shukla and Gupta, 2015) used some of these images in their research. The third image set represent small size images obtained from the Stanford image set; this image set is used to find out the algorithm performance on the small size images. The fourth and fifth image sets are obtained from lossless benchmark compression schemes for kodak

colour images and raster map and they were used for evaluation. (Khan et al., 2017); (Khan et al., 2016); (Mao et al., 2015) used this image sets in their research.

### **3.8 Characteristics of the Selected Images**

- All the images from image set 1 are greyscale images, while the images in the other image sets have coloured images only.
- The image size is different from one image to another, so we discuss the image size in the discussion section in Chapters 4 and 5.
- A scope of various image scenes, such as medical image, natural image, architectural images were used as displayed in image set three.
- Images that have smoothly graduating signals were used as displayed in image set four.
- Highly coloured images were used, such as images from image set four.
- Low coloured content images were used, such as images from image set five.
- Old and new images with different resolution were used, the minimum image resolution used is (128x85) and the maximum used image resolution is (768x512).
- The used image formats are (GIF, PNG, BMP, JPG).

### **3.9 Chapter Summary**

This chapter presented the gap analyses for the current most popular used algorithm's in the domain of lossless image compression by describing their limitation and the proposed solution for each of the limitations, followed by the software development process and the research requirements that are needed to develop the proposed solutions, such as the used programming language and the tested image sets.

The next chapter describes the developments of the proposed image compression algorithms.

## **CHAPTER FOUR: THE PROPOSED LOSSLESS ALGORITHM FOR NATURAL IMAGES COMPRESSION**

---

### **4 Chapter Overview**

This chapter describes in detail the proposed lossless Column Subtraction Compression algorithm (CSC). It starts with a detailed explanation of all the procedures used in the proposed technique, followed by the validation of the algorithm. Finally, the algorithm was compared to the state-of-the-art algorithms.

---

#### **4.1 Introducing the CSC Compression Algorithm**

Lossless compression techniques have been used in many applications, where any loss of the original image data leads to an improper diagnosis. Some of these applications are medical application, Global Positioning System (GPS), digital radiography, camera systems (Sengupta and Roy, 2018) and remote sensing applications such as monitoring forest fires (Rusyn et al., 2016).

The lossless CSC algorithm is designed to work with any application and supports all image formats whether the input image is of a high resolution or low-resolution. The CSC algorithm is suitable for natural image compression and can be used as stand-alone algorithm or as a pre-processing phase for any lossless or lossy techniques. The CSC algorithm includes two procedures as follows.

Let  $NI$  be a coloured image referring to a Natural Image represented with three colour matrices (R, G and B). The three matrices have the same resolution of  $(m \times n)$  where  $m$  and  $n$  represents the matrix dimensions (Row ( $m$ ), Columns ( $n$ )).

1. **Procedure transformation** is to map the input natural image  $NI(R,G,B)$  from the  $RGB$  colour space into  $RDgDb$  colour space. The transformation procedure output is an image with less coloration matrices, it maps the pixel values into new space that includes smaller values  $TI(R,Dg,Db)$ . The transformed matrices have the same dimensions of the input matrices ( $m \times n$ ).  $Dg$  refers to the transformed green matrix,  $Db$  refers to the transformed blue matrix and  $TI$  referred to the Transformed Image.
2. **Procedure compression** is to decrease the size of the transformed image  $TI(R,Dg,Db)$  by using the column subtraction compression where  $CRR$  refers to the compressed red channel,  $CRG$  refers to the compressed green channel and  $CRB$  refers to the compressed blue channel. The compression procedure output is the compressed image  $CI(CRR, CRG, CRB)$ . Algorithm 1 illustrates the procedures of natural image compression.

## **Algorithm1: CSC For Natural Images Compression**

### **1: Procedure Transformation**

- 2: input NI: natural image NI(R,G,B)
- 3: m = total rows number; n = total columns number;
- 4: //Dg, Db are (m X n) matrices of natural number;
- 5: // Dg matrix is resulted by subtracting the G matrix from the R matrix using the following formula
$$Dg(\text{row}, \text{column}) = R(\text{row}, \text{column}) - G(\text{row}, \text{column})$$
- 6: // Db matrix is resulted by subtracting the G matrix from the B matrix using the following formula
$$Db(\text{row}, \text{column}) = B(\text{row}, \text{column}) - G(\text{row}, \text{column})$$
- 7: output: transformed image TI(R,Dg,Db)

### **8: End Procedure Transformation**

### **1: Procedure Compression**

- 2: input TI: transformed image TI(R,Dg,Db)
- 3: //CRR, CRG, CRB; are matrices of natural number
- 4: // CRR matrix is resulted by the following nested for loop
- 5: from the first column to the last column - 1
- 6: for all rows in the column do
- 7: CRR (row, column) = R(row, column) - R(row, column+1);
- 8: end
- 9: end
- 10: // CRG matrix is resulted by the following nested for loop
- 11: from the first column to the last column - 1
- 12: for all rows in the column do
- 13: CRG (row, column) = Dg(row, column) - Dg(row, column+1);
- 14: end
- 15: end
- 16: // CRB matrix is resulted by the following nested for loop
- 17: from the first column to the last column - 1
- 18: for all rows in the column do
- 19: CRB (row, column) = Db(row, column) - Db(row, column +1);
- 20: end
- 21: end
- 22: output: compressed image CI(CRR, CRG, CRB)

### **23: End Procedure Compression**



The Natural Images Decompression algorithm used two procedures for reconstructing the compressed image as follows.

## **Algorithm2: CSC For Natural Images Decompression**

### **1: Procedure Decompression**

```

2:  input compressed image CI(CRR, CRG, CRB)
3:    m = total rows number; n = total columns number;
4: //  R,Dg,Db are (m X n) matrices of natural number;
5: // R matrix is resulted by the following nested for loop
6:   from the last column to the first column + 1
7:     for all rows in the column do
8:       R (row, column-1) = CRR(row, column) + CRR(row, column-1);
9:     end
10:  end
11: // Dg matrix is resulted by the following nested for loop
12:  from the last column to the first column + 1
13:   for all rows in the column do
14:     Dg (row, column-1) = CRG(row, column) + CRG(row, column-1);
15:   end
16:  end
17: // Db matrix is resulted by the following nested for loop
18:  from the last column to the first column + 1
19:   for all rows in the column do
20:     Db (row, column-1) = CRB(row, column) + CRB(row, column-1);
21:   end
22:  end
23: output: transformed image TI(R,Dg,Db)

```

### **24: End Procedure Decompression**

### **1: Procedure Revers Transformation**

```

2:  input R,Dg,Db; matrices of natural number
3: //  R,G,B are (m X n) matrices of natural number
4: //  G matrix is resulted by subtracting the Dg matrix from the R matrix
5:   G(row, column) = R(row, column) – DG(row, column)
6: //  B matrix is resulted by adding the G matrix values to the Db matrix values
7:   B(row, column) = G(row, column) + DB(row, column)

```

### **8: End Procedure Revers Transformation**

By implementing the CSC algorithm, we expect to decrease the image size and maintain the image quality as it was before compression in a very fast time. Figure 4.1 shows the CSC flowchart.

## 4.2 The CSC Flowchart

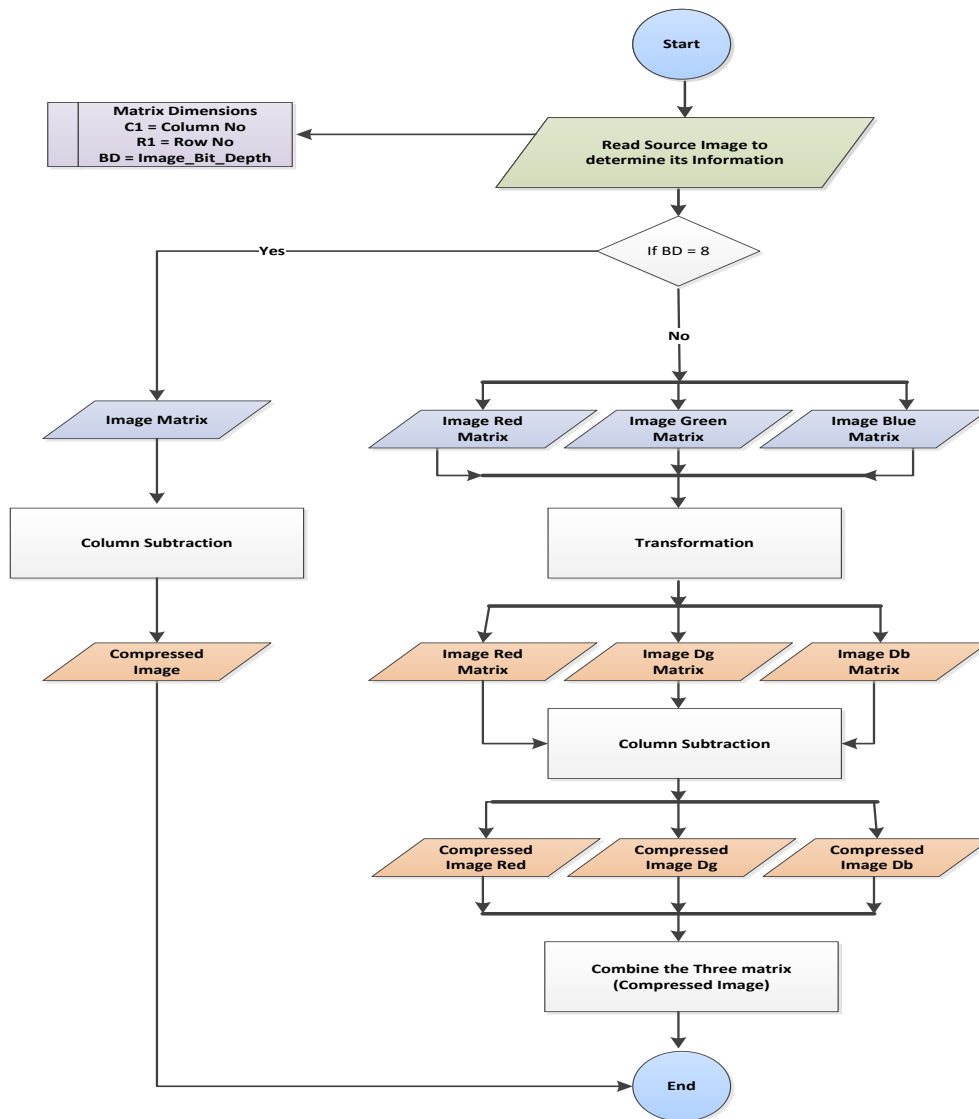


Figure 4. 1 - CSC Lossless Algorithm Flowchart

## 4.3 Description of the CSC Algorithm

The algorithm starts with loading the source image to identify the matrix dimension and the input image bit-depth, followed by specifying the suitable compression steps that meet the input image; if the image bit-depth is equal to eight then the image cannot be transformed, and the image will be sent to the subtraction function phase directly, and if the image bit-depth is

24 then the image will be loaded to the transformation phase as a pre-processing to the column subtraction function.

### 4.3.1 Colour Transformation:

It is a fact that the main three colours in the RGB colour space have high coloration in natural images. That indicates that, two or more of the three main components have the same information for the same pixel's addresses, e.g., if a particular area of an image is bright in the red channel then mostly it should be bright in the green and blue channels as well. The aim of colour transformation is for decreasing the coloration between the image components (R,G,B) to produce a new colour space with less correlation between the three components.

(Starosolski, 2014) described the most popular transformation methods such as R,Dg,Db and YCoCg. In this research, the researcher implements and tests many transformation methods to choose the best method that achieves the best compression rate with the proposed algorithm.

After implementing and testing many transformation methods; the researcher modified the R,Dg,Db transformation method and used it as a pre-processing phase for the CSC algorithm since it achieved the best compression rate. Table 4.1. lists the original R,Dg,Db transformation formula. Where, the image colour space is transformed from RGB to R,Dg,Db colour space by applying Eq 4.1 for the red colour space, Eq 4.2 for the green colour space and Eq 4.3 for the blue colour space, where **R** represents the luminance and **Dg**, **Db** are used to represent the chrominance. Eq 4.4, Eq 4.5, Eq 4.6 are used for the invers transformation.

Table 4. 1 - R,Dg,Db Transformation and Invers Transformation Equations

Colour Space Transformation Equation		Inverse Colour Space Transformation Equation	
$R = R$	<a href="#">Eq 4.1</a>	$R = R$	<a href="#">Eq 4.4</a>
$Dg = R - G$	<a href="#">Eq 4.2</a>	$G = R - Dg$	<a href="#">Eq 4.5</a>
$Db = G - B$	<a href="#">Eq 4.3</a>	$B = G - Db$	<a href="#">Eq 4.6</a>

The researcher modified the original R,Dg,Db transformation by using Eq. 4.9 for the Db colour space instead of using Eq. 4.3. The modified transformation is used as pre-processing phase for the CSC compression algorithm. Table 4.2 lists the modified transformation formula where, the image colour space is transformed from RGB to R,Dg,Db colour space by applying Eq 4.7 for the red colour space, Eq 4.8 for the green colour space and the modified Eq 4.9 for the blue colour space, where **R** represents the luminance and **Dg**, **Db** are used to represent the chrominance.

Table 4. 2 - The Modified Transformation and Invers Transformation Equations

Colour Space Transformation Equation		Inverse Colour Space Transformation Equation	
$R = R$	<a href="#">Eq 4.7</a>	$R = R$	<a href="#">Eq 4.10</a>
$Dg = R - G$	<a href="#">Eq 4.8</a>	$G = R - Dg$	<a href="#">Eq 4.11</a>
$Db = B - G$	<a href="#">Eq 4.9</a>	$B = G + Db$	<a href="#">Eq 4.12</a>

The inverse colour transform equations are used for restoring the original colour space values, where Eq. 4.10 is used for restoring the red channel value and Eq. 4.11 and Eq. 4.12 for restoring the green and blue channel values respectively.

As displayed in Table 4.3, the modified colour transformation enhances the average compression ratio for the CSC algorithm by having 0.147 better compression ratio than the original transformation.

Table 4. 3 - The Average Compression Ratio for the Original and Modified Transformation

Image Set	Average Compression Ratio (Original Transformation)	Average Compression Ratio (Modified Transformation)
Image Set 1	2.319	2.346
Image Set 2	2.384	2.571
Image Set 3	2.254	2.324
Image Set 4	3.094	3.541
Image Set 5	28.240	28.243
<b>Average</b>	<b>7.6582</b>	<b>7.805</b>

Figure 4.2 describe the modified transformation for the three colours space by using a sample example of 8x8 block for the three matrices.

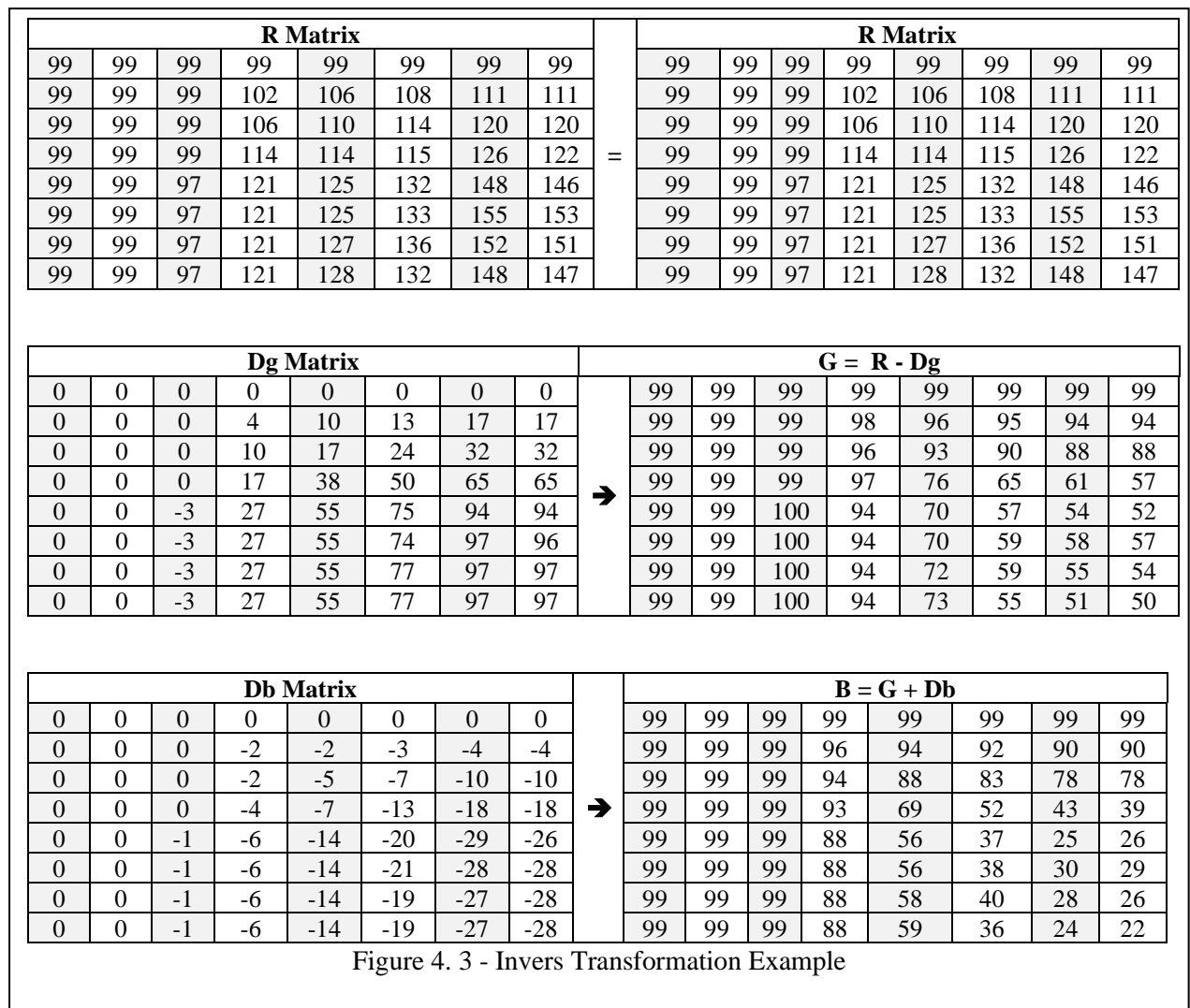
Sample 8x8 for the Red Matrix (R)								=	R = Red Matrix R							
99	99	99	99	99	99	99	99		99	99	99	99	99	99		
99	99	99	102	106	108	111	111		99	99	99	102	106	108	111	111
99	99	99	106	110	114	120	120		99	99	99	106	110	114	120	120
99	99	99	114	114	115	126	122		99	99	99	114	114	115	126	122
99	99	97	121	125	132	148	146		99	99	97	121	125	132	148	146
99	99	97	121	125	133	155	153		99	99	97	121	125	133	155	153
99	99	97	121	127	136	152	151		99	99	97	121	127	136	152	151
99	99	97	121	128	132	148	147		99	99	97	121	128	132	148	147

Sample 8x8 for the Green Matrix (G)								→	Dg = R - G							
99	99	99	99	99	99	99	99		0	0	0	0	0	0	0	0
99	99	99	98	96	95	94	94		0	0	0	4	10	13	17	17
99	99	99	96	93	90	88	88		0	0	0	10	17	24	32	32
99	99	99	97	76	65	61	57		0	0	0	17	38	50	65	65
99	99	100	94	70	57	54	52		0	0	-3	27	55	75	94	94
99	99	100	94	70	59	58	57		0	0	-3	27	55	74	97	96
99	99	100	94	72	59	55	54		0	0	-3	27	55	77	97	97
99	99	100	94	73	55	51	50		0	0	-3	27	55	77	97	97

Sample 8x8 for the Blue Matrix (B)								→	Db = B - G							
99	99	99	99	99	99	99	99		0	0	0	0	0	0	0	0
99	99	99	96	94	92	90	90		0	0	0	-2	-2	-3	-4	-4
99	99	99	94	88	83	78	78		0	0	0	-2	-5	-7	-10	-10
99	99	99	93	69	52	43	39		0	0	0	-4	-7	-13	-18	-18
99	99	99	88	56	37	25	26		0	0	-1	-6	-14	-20	-29	-26
99	99	99	88	56	38	30	29		0	0	-1	-6	-14	-21	-28	-28
99	99	99	88	58	40	28	26		0	0	-1	-6	-14	-19	-27	-28
99	99	99	88	59	36	24	22		0	0	-1	-6	-14	-19	-27	-28

Figure 4. 2 - Transformation Example

Figure 4.3 describe the invers transformation for the three colours space by using the sample example of 8x8 block for the three transformed matrices TI(R,Dg,Db).



### 4.3.2 Column Subtraction

Images can be compressed by taking advantage of the high correlation between neighbouring pixels. In another word, each pixel value is similar or very close to the value of its adjacent pixels (Novikov, Egorov, and Gilmutdinov, 2016). To accomplish a high compression ratio from this fact, a new method called Column Subtraction Compression (CSC) is developed to decrease the image intensities by subtracting each column from the nearest column and save the resulting value in the first column starting from the first column from the left side of each matrix. The resulting three matrices from the previous transformation phase TI(R, Dg and Db) should be loaded to the CSC function to decrease the image size as a second phase by using

Eq. 4.13 for the red matrix (**R**), Eq. 4.14 for the green matrix (**Dg**) and Eq. 4.15 for the blue matrix (**Db**) respectively.

$$CRR(i, j) = R(i, j) - R(i, j + 1) \quad \forall i, j; 1 \leq i \leq m \text{ and } 1 \leq j \leq n-1 \quad 4.13$$

$$CRG(i, j) = Dg(i, j) - Dg(i, j + 1) \quad \forall i, j; 1 \leq i \leq m \text{ and } 1 \leq j \leq n-1 \quad 4.14$$

$$CRB(i, j) = Db(i, j) - Db(i, j + 1) \quad \forall i, j; 1 \leq i \leq m \text{ and } 1 \leq j \leq n-1 \quad 4.15$$

The three matrices (**CRR**, **CRG** and **CRB**) have the same resolution of (m×n) where **m** and **n** represents the matrix dimensions (Row (m), Columns (n)) and (i,j) refers to the elements coordinates, and **CRR** represents the compressed matrix for the red channel, **CRG** represents the compressed matrix for the green channel and **CRB** represents the compressed matrix for the blue channel. Figure 4.4 describes the CSC results for the three transformed matrices for the sample example.

For reconstructing the compressed image, the three compressed matrices **CRR**, **CRG** and **CRB** should be loaded to the decompression algorithm which uses the reversible CSC equations for each of the three matrices separately, by applying Eq. 4.16 for the **CRR** matrix, Eq. 4.17 for the **CRG** matrix and Eq. 4.18 for the **CRB** matrix.

$$R(i, j - 1) = CRR(i, j) + CRR(i, j - 1) \quad \forall i, j; 1 \leq i \leq m \text{ and } 2 \leq j \leq n \quad 4.16$$

$$Dg(i, j - 1) = CRG(i, j) + CRG(i, j - 1) \quad \forall i, j; 1 \leq i \leq m \text{ and } 2 \leq j \leq n \quad 4.17$$

$$Db(i, j - 1) = CRB(i, j) + CRB(i, j - 1) \quad \forall i, j; 1 \leq i \leq m \text{ and } 2 \leq j \leq n \quad 4.18$$

Figure 4.4 describe the CSC compression phase for the three transformed matrices. The resulted three matrices should be combined to produce the compressed image CI(**CRR**,**CRG**,**CRB**).

Red Matrix R								→	CRR							
99	99	99	99	99	99	99	99	0	0	0	0	0	0	0	0	99
99	99	99	102	106	108	111	111	0	0	-3	-4	-2	-3	0	111	
99	99	99	106	110	114	120	120	0	0	-7	-4	-4	-6	0	120	
99	99	99	114	114	115	126	122	0	0	-15	0	-1	-11	4	122	
99	99	97	121	125	132	148	146	0	2	-24	-4	-7	-16	2	146	
99	99	97	121	125	133	155	153	0	2	-24	-4	-8	-22	2	153	
99	99	97	121	127	136	152	151	0	2	-24	-6	-9	-16	1	151	
99	99	97	121	128	132	148	147	0	2	-24	-7	-4	-16	1	147	

Dg Matrix								→	CRG							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	4	10	13	17	17	0	0	-4	-6	-3	-4	0	17	
0	0	0	10	17	24	32	32	0	0	-10	-7	-7	-8	0	32	
0	0	0	17	38	50	65	65	0	0	-17	-21	-12	-15	0	65	
0	0	-3	27	55	75	94	94	0	3	-30	-28	-20	-19	0	94	
0	0	-3	27	55	74	97	96	0	3	-30	-28	-19	-23	1	96	
0	0	-3	27	55	77	97	97	0	3	-30	-28	-22	-20	0	97	
0	0	-3	27	55	77	97	97	0	3	-30	-28	-22	-20	0	97	

Db Matrix								→	CRB							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	-2	-2	-3	-4	-4	0	0	2	0	1	1	0	-4	
0	0	0	-2	-5	-7	-10	-10	0	0	2	3	2	3	0	-10	
0	0	0	-4	-7	-13	-18	-18	0	0	4	3	6	5	0	-18	
0	0	-1	-6	-14	-20	-29	-26	0	1	5	8	6	9	-3	-26	
0	0	-1	-6	-14	-21	-28	-28	0	1	5	8	7	7	0	-28	
0	0	-1	-6	-14	-19	-27	-28	0	1	5	8	5	8	1	-28	
0	0	-1	-6	-14	-19	-27	-28	0	1	5	8	5	8	1	-28	

Figure 4. 4 - CSC Example

Figure 4.5 describe the CSC decompression phase for the three compressed matrices CI(CRR,CRG,CRB). The resulted three matrices should be combined to produce the decompressed image TI(R,Dg,Db).



<b>CRR Matrix</b>								→	<b>R</b>							
0	0	0	0	0	0	0	99		99	99	99	99	99	99	99	99
0	0	-3	-4	-2	-3	0	111		99	99	99	102	106	108	111	111
0	0	-7	-4	-4	-6	0	120		99	99	99	106	110	114	120	120
0	0	-15	0	-1	-11	4	122		99	99	99	114	114	115	126	122
0	2	-24	-4	-7	-16	2	146		99	99	97	121	125	132	148	146
0	2	-24	-4	-8	-22	2	153		99	99	97	121	125	133	155	153
0	2	-24	-6	-9	-16	1	151		99	99	97	121	127	136	152	151
0	2	-24	-7	-4	-16	1	147		99	99	97	121	128	132	148	147
<b>CRG Matrix</b>								→	<b>Dg Matrix</b>							
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	-4	-6	-3	-4	0	17		0	0	-4	-6	-3	-4	0	17
0	0	-10	-7	-7	-8	0	32		0	0	-10	-7	-7	-8	0	32
0	0	-17	-21	-12	-15	0	65		0	0	-17	-21	-12	-15	0	65
0	3	-30	-28	-20	-19	0	94		0	3	-30	-28	-20	-19	0	94
0	3	-30	-28	-19	-23	1	96		0	3	-30	-28	-19	-23	1	96
0	3	-30	-28	-22	-20	0	97		0	3	-30	-28	-22	-20	0	97
0	3	-30	-28	-22	-20	0	97		0	3	-30	-28	-22	-20	0	97
<b>CRB Matrix</b>								→	<b>Db Matrix</b>							
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	2	0	1	1	0	-4		0	0	2	0	1	1	0	-4
0	0	2	3	2	3	0	-10		0	0	2	3	2	3	0	-10
0	0	4	3	6	5	0	-18		0	0	4	3	6	5	0	-18
0	1	5	8	6	9	-3	-26		0	1	5	8	6	9	-3	-26
0	1	5	8	7	7	0	-28		0	1	5	8	7	7	0	-28
0	1	5	8	5	8	1	-28		0	1	5	8	5	8	1	-28
0	1	5	8	5	8	1	-28		0	1	5	8	5	8	1	-28

Figure 4. 5 - CSC Decompression Example

#### 4.4 CSC Algorithm Time Complexity Analyses

A simplified time-complexity analysis for the CSC algorithm is performed based on the Big-O notation, which defines the worst-case scenario. The CSC algorithm starts by loading the input (R,G,B) image to identifies the matrix size  $n$ . Then the transformation is used to map the (R,G,B) image into new colour space (R,Dg,Db) by using the transformation equations, followed by the proposed CSC compression as a final phase.

##### 4.4.1 CSC algorithm Time Complexity

The CSC algorithm is divided into two main components (Transformation and Compression). Table 4.4 lists the time complexity for each of the CSC algorithm components by using the O notation for describing each components computation complexity.

Table 4. 4 - CSC Algorithm Complexity

Line	Description	O
1	<b>Procedure Transformation</b>	-
2	input NI: natural image NI(R,G,B)	$O(n^2)$
3	m = total rows number; n = total columns number;	$O(1)$
5	Dg(row, column) = R(row, column) – G(row, column)	$O(n^2)$
6	Db(row, column) = B(row, column) - G(row, column)	$O(n^2)$
1	<b>Procedure Compression</b>	-
2	input TI: transformed image TI(R,Dg,Db)	$O(n^2)$
7	CRR (row, column) = R(row, column) - R(row, column+1);	$O(n^2)$
13	CRG (row, column) = Dg(row, column) - Dg(row, column+1);	$O(n^2)$
19	CRB (row, column) = Db(row, column) - Db(row, column + 1);	$O(n^2)$
1	<b>Procedure De-Compression</b>	-
2	input compressed image CI(CRR, CRG, CRB)	$O(n^2)$
3	m = total rows number; n = total columns number;	$O(1)$
8	R (row, column-1) = CRR(row, column) + CRR(row, column-1)	$O(n^2)$
14	Dg (row, column-1) = CRG(row, column) + CRG(row, column-1)	$O(n^2)$
20	Db (row, column-1) = CRB(row, column) + CRB(row, column-1)	$O(n^2)$
1	<b>Procedure Revers Transformation</b>	-
2	input R,Dg,Db; matrices of natural number	$O(n^2)$
5	G(row, column) = R(row, column) – DG(row, column)	$O(n^2)$
7	B(row, column) = G(row, column) + DB(row, column)	$O(n^2)$

The growth rate function in terms of time for the CSC components is analysed as:

- 1- The growth rate function of the transformation component is

$$f(n) = 3O(n^2) + O(1) \text{ Therefore, the overall rate of growth for this component is}$$

$$f(n) = O(n^2) \text{ after removing the constants.}$$

- 2- The growth rate function of the CSC component is

The second phase of the algorithm is to compress the R,Dg,Db image by using the CSC function. This phase is to apply the CSC function for each of the three matrices individually. For each colour space we used nested two loops. The outer loop runs  $n$  times and the inner loop runs  $n$  times for each iteration of the outer loop; this indicate

that, this function will be running for  $n^2$  total times, thus the function is running  $O(n^2)$  time for each colour and the complexity of this phase is

$f(n) = 4 O(n^2)$  Therefore, the overall rate of growth for this component is

$f(n) = O(n^2)$  after removing the constants.

The decompression phase is to reconstruct the image by using two reversible procedures.

3- The growth rate of the CSC decompression component is

$f(n) = 4On^2 + O(1)$  Therefore, the overall rate of growth for this component is

$f(n) = O(n^2)$  after removing the constants.

4- The growth rate of the inverse transformation component is

$f(n) = 3 O(n^2)$  Therefore, the overall rate of growth for this component is

$f(n) = O(n^2)$  after removing the constants.

After approximating the computation complexity for the individual components of the system, the overall complexity is calculated by summing up the overhead for the individual parts.

$f(n) = [O(n^2)] + [O(n^2)] + [O(n^2)] + [O(n^2)]$ .

$f(n) = 4 O(n^2)$ .

Therefore, the overall complexity of the algorithm is:

$f(n) = O(n^2)$  after removing the constants.

#### 4.4.2 Proof

Assume that  $g(n) = n$ , the time complexity of  $f(n)$  is  $O(n^2)$ . To prove that  $f(n)$  in Equation 3.1 is  $O(g(n))$ , we will apply the limit to find a constant  $c > 0$ .

We have  $f(n) = O(n^2)$  and  $g(n) = n$ . That is

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)}$$

$$\lim_{n \rightarrow \infty} \frac{n^2}{n} = n$$

As the proof shows, there is a constant  $c > 0$  that satisfy the limit in the proof Theorem. Since  $n_0$  must be positive integer, we can say the  $f(n)$  in Equation 3.1 is  $O(n)$ , for  $n \geq n_0$ .

## 4.5 Validation and Testing

After the implementation of the algorithm, the researcher needs to estimate the compression performance by comparing the new algorithm results with other state-of-the-art results. The main parameters that may affect any compression algorithm are compression size, image

quality and execution time. Using our testbed, we are going to evaluate the CSC algorithm based on:

- 1- The compressed image size by calculating the compression ratio.
- 2- The image quality by calculating the (MSE and PSNR).
- 3- The algorithm execution time by using the Tic-Toc matlab function.

#### 4.5.1 The CSC Algorithm Compression Size

Measuring the compression ratio (Cr) is important in order to find out the algorithm storage saving, which can be measured by dividing the input image size by the output compressed image size (John and Joe, 2005); (Wang and Li, 2011). The best compression ratio is when the Cr results have larger values; therefore, the compression algorithm should provide a compressed image size smaller than the original image. Table 4.5 lists the compression size results for the CSC algorithm by listing the new image size, compression ratio, compression rate and storage saving percentage for the three image sets results.

Table 4. 5 - The Lossless CSC Algorithm Compression Size

Image Set 1							
Image	Format	Resolution	Original Size in KBs	New Size in KBs	Compression Ratio (CR)	Compression Rate	Storage Saving %
Baboon	GIF	512*512	225	94	2.394	0.418	58.345
Barbara	PNG	512*512	230	121	1.901	0.526	47.238
Boats	PNG	512*512	239	113	2.115	0.473	52.728
Boats	BMP	720*576	368	148	2.486	0.402	59.751
Camera Man	BMP	256*256	60	26	2.308	0.433	56.423
Camera Man	GIF	256*256	56	23	2.435	0.411	58.366
House	PNG	256*256	59	21	2.81	0.356	64.995
Lena	PNG	256*256	59	27	2.185	0.458	54.390
Lena	JPG	512*512	226	91	2.484	0.403	59.510
Image Set 2							
Image	Format	Resolution	Original Size in KBs	New Size in KBs	Compression Ratio (CR)	Compression Rate	Storage Saving %
Lena	PNG	330*330	290	118	2.458	0.408	59.205
Lena	BMP	220*220	129	46	2.804	0.355	64.452
Lena	JPG	225*225	135	48	2.813	0.355	64.540
Airplane	BMP	512*512	751	249	3.016	0.331	66.876
Baboon	BMP	500*480	599	359	1.669	0.600	40.020
Barbara	BMP	720*576	1064	397	2.680	0.373	62.690
Boats	BMP	787*576	1148	373	3.078	0.324	67.556
Goldhill	BMP	720*576	1031	378	2.728	0.367	63.307
Pepper	BMP	512*512	651	346	1.882	0.531	46.899
Image Set 3							
Image	Format	Resolution	Original Size in KBs	New Size in KBs	Compression Ratio (CR)	Compression Rate	Storage Saving %
Medic	JPG	168*90	32	12	2.667	0.376	62.441

Medic1	JPG	160*90	24	11	2.182	0.444	55.650
Butterfly	JPG	128*85	20	13	1.538	0.627	37.300
Mountain	JPG	128*96	28	11	2.545	0.399	60.129
swarm	JPG	128*85	25	11	2.273	0.436	56.360
Lake_jpg	JPG	128*85	25	12	2.083	0.468	53.159
Saturn_jpg	JPG	128*100	19	9	2.111	0.497	50.282
Earth_jpg	JPG	225*225	78	43	1.814	0.551	44.921
boat_jpg	JPG	128*85	25	10	2.5	0.404	59.609
Waterfall	JPG	128*96	26	13	2	0.506	49.426
Eagle	JPG	128*96	17	8	2.125	0.478	52.246
Grand_Sone	JPG	128*96	28	12	2.333	0.425	57.475
Car	JPG	128*85	28	10	2.8	0.353	64.744
Shape	JPG	128*95	31	15	2.067	0.471	52.929
Image Set 4							
Image	Format	Resolution	Original Size in KBs	New Size in KBs	Compression Ratio (CR)	Compression Rate	Storage Saving %
Knob & Bolt	PNG	768*512	1155	320	3.609	0.277	72.31
Houses	PNG	768*512	1149	395	2.909	0.344	65.64
Landscape	PNG	768*512	1153	396	2.912	0.343	65.66
Light House	PNG	768*512	1150	311	3.698	0.271	72.91
Barn	PNG	768*512	1149	355	3.237	0.309	69.08
Parrots	PNG	768*512	1153	297	3.882	0.258	74.20
Flowers & Sill	PNG	768*512	1154	282	4.092	0.245	75.53
Six-Shooter	PNG	768*512	1152	270	4.267	0.235	76.53
Motocross	PNG	768*512	1150	376	3.059	0.327	67.34
Zentime	PNG	768*512	1155	307	3.762	0.266	73.42
Image Set 5							
Image	Format	Resolution	Original Size in KBs	New Size in KBs	Compression Ratio (CR)	Compression Rate	Storage Saving %
Map 1	BMP	600*480	761	149	5.107	0.195	80.46
Map 2	BMP	600*480	821	136	6.037	0.166	83.38
Map 3	BMP	600*480	801	141	5.681	0.176	82.39
Map 4	BMP	600*480	838	120	6.983	0.143	85.68
Map 5	BMP	600*480	800	151	5.298	0.188	81.15
Map 6	BMP	600*480	826	147	5.619	0.178	82.20
Map 7	BMP	600*480	838	133	6.301	0.158	84.17
Map 8	BMP	600*480	814	154	5.286	0.189	81.11
Map 9	BMP	600*480	769	158	4.867	0.205	79.46
Map 10	BMP	600*480	768	149	5.154	0.194	80.58

One of the main purposes of any compression algorithm is to decrease the image size. As listed in Table 4.6, the CSC algorithm dramatically decreases the image size for the five image sets. In image sets 1 and 2, the image size is decreased by having an average compression ratio of 2.35 for the first image set and 2.57 for the second image set, which indicates that the algorithm saved 56.9% from the original size of the first image set and 59.5% from the original size of the second image set. By observing the third image set results, the algorithm achieved the

lowest compression ratio of 2.22, since this image set is already compressed by the JPEG algorithm; however, the algorithm decreases the JPEG images from the third image set as well, by saving 54.1% from the original image's size. The Cr resulted from the fourth image set is 3.54 and saved 71.3% from the original image size. The best Cr came with compressing the fifth image set by having 5.63 Cr and saves 82.1 % from the original image set size.

Table 4. 6 - The Lossless CSC Algorithm Average Compression Size

Image Sets	Average Compression Ratio CR	Average Compression Rate	Average Storage Saving %
Image Set 1	2.346	0.431	56.86
Image Set 2	2.571	0.405	59.51
Image Set 3	2.224	0.460	54.05
Image Set 4	3.543	0.288	71.26
Image Set 5	5.634	0.179	82.06
<b>Average</b>	<b>3.263</b>	<b>0.352</b>	<b>64.74</b>

Figure 4.6 describes the compression storage saving percentages for the proposed lossless algorithm CSC. By observing the results for each of the image set, we can say that the CSC algorithm decreases the image size by different percentages for different images types and different images resolution.

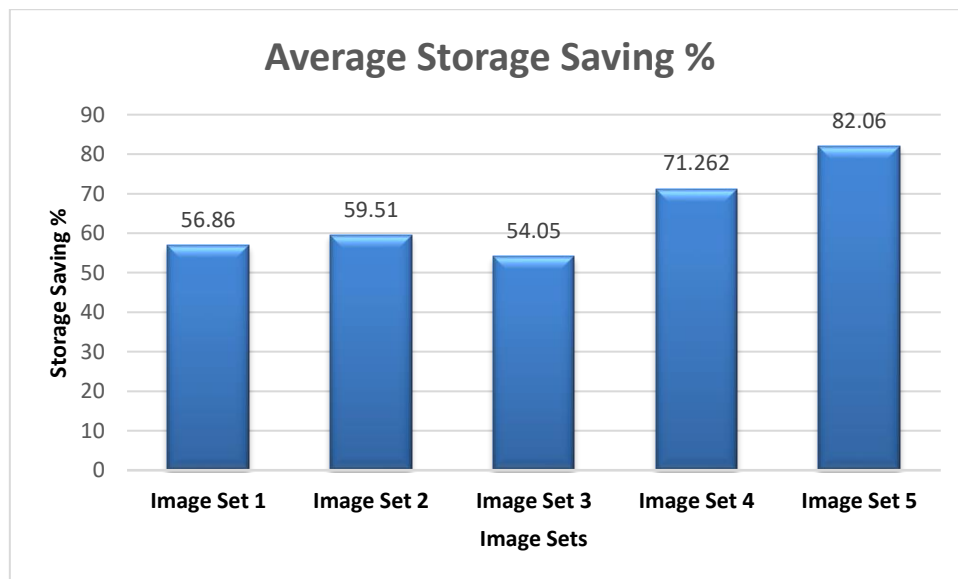


Figure 4. 6 - The CSC Algorithm Storage saving

#### 4.5.2 The CSC Algorithm Image Quality.

For distortion assessment, we used the following two metrics:

- i. Mean Squared Error (MSE)
- ii. Peak Signal to Noise Ratio (PSNR)

When the MSE is closer to zero we have better image quality, while in the PSNR, larger values suggest better image quality. Since the CSC algorithm is a lossless technique, this indicates that the image quality should not be compromised, and the original image is a 100% perfect match with the decompressed image. The proposed algorithm MSE results is zero and the PSNR is infinite for all the images in the five image sets, the PSNR is infinite, because we calculate its value by dividing the distortion value on the MSE; in this case, the MSE is zero so the PSNR should be infinite. After averaging the distortion value for the five image sets, we prove that the proposed image compression algorithm is a lossless algorithm. All the images from the five image sets have been restored exactly as they were before compression. The five image sets have been perfectly restored after decompression with zero distortion, since the MSE value is zero for all the test images. Table 4.7 display two image samples for each image set.

Table 4. 7 - Sample Images from the Five Image Sets Before and After Compression



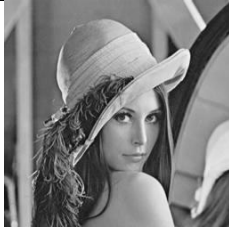
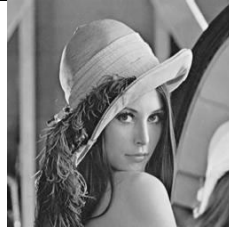




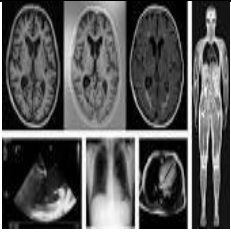
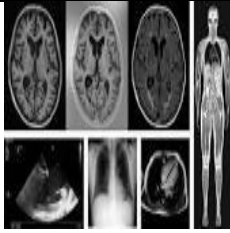








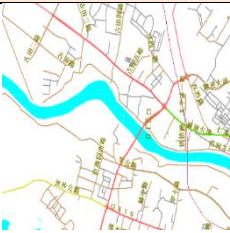
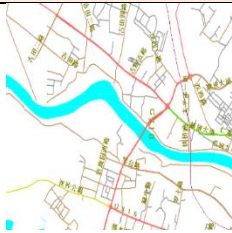
Image Set 1			
Original Image	Compressed Image	Original Image	De-Compressed Image
Camera Man (GIF)		Lena (PNG)	
			
Image Set 2			
Original Image	Compressed Image	Original Image	De-Compressed Image
Lena (PNG)		Goldhill (BMP)	
			



Image Set 3			
Original Image	Compressed Image	Original Image	De-Compressed Image
Medic (JPEG)		Boat (JPEG)	
			
Image Set 4			
Original Image	Compressed Image	Original Image	De-Compressed Image
Parrots (PING)		Motocross (PING)	
			
Image Set 5			
Original Image	Compressed Image	Original Image	De-Compressed Image
Map 1 (BMP)		Map 2 (BMP)	
			

#### 4.5.3 The CSC Algorithm Execution Time.

Image compression algorithm performance can be calculated by measuring the compression speed and the decompression speed in seconds. Compression speed is the time needed for compressing the image while the decompression speed is the time needed for decompressing the image. Table 4.8 displays the needed execution time for each image from the five sets in seconds.



Table 4. 8 - The Lossless CSC Algorithm Compression Time in Seconds

Image Set 1			
Image	Compression Time	Decompression Time	Total Time
Baboon	0.303	0.003	0.306
Barbara	0.313	0.007	0.32
Boats	0.254	0.11	0.364
Boats	0.32	0.012	0.332
Camera Man	0.28	0.003	0.283
Camera Man	0.25	0.003	0.253
House	0.225	0.002	0.227
Lena	0.257	0.002	0.259
Lena	0.519	0.007	0.526
Image Set 2			
Image	Compression Time	Decompression Time	Total Time
Lena	0.22	0.01	0.23
Lena	0.12	0.009	0.129
Lena	0.15	0.009	0.159
Airplane	0.46	0.02	0.48
Baboon	0.5	0.02	0.52
Barbara	0.64	0.03	0.67
Boats	0.55	0.03	0.58
Goldhill	0.54	0.03	0.57
Pepper	0.56	0.02	0.58
Image Set 3			
Image	Compression Time	Decompression Time	Total Time
Medic	0.06	0.0003	0.0603
Medic1	0.08	0.0005	0.0805
Butterfly	0.08	0.0005	0.0805
Mountain	0.04	0.005	0.045
swarm	0.07	0.0004	0.0704
Lake_jpg	0.04	0.0004	0.0404
Saturn_jpg	0.04	0.0006	0.0406
Earth_jpg	0.04	0.002	0.042
boat_jpg	0.04	0.0006	0.0406
Waterfall	0.04	0.0006	0.0406
Eagle	0.04	0.0007	0.0407
Grand_Sone	0.04	0.0005	0.0405
Car	0.04	0.006	0.046
Shape	0.04	0.0004	0.0404
Image Set 4			
Image	Compression Time	Decompression Time	Total Time
Knob & Bolt	0.35	0.01	0.36
Houses	0.34	0.01	0.35

Landscape	0.38	0.01	0.39
Light House	0.35	0.01	0.36
Barn	0.41	0.01	0.42
Parrots	0.34	0.01	0.35
Flowers & Sill	0.3	0.01	0.31
Six-Shooter	0.36	0.01	0.37
Motocross	0.42	0.01	0.43
Zentime	0.34	0.01	0.35
<b>Image Set 5</b>			
<b>Image</b>	<b>Compression Time</b>	<b>Decompression Time</b>	<b>Total Time</b>
Map 1	0.12	0.01	0.13
Map 2	0.2	0.02	0.22
Map 3	0.13	0.018	0.148
Map 4	0.1	0.018	0.118
Map 5	0.14	0.01	0.15
Map 6	0.09	0.01	0.1
Map 7	0.1	0.01	0.11
Map 8	0.15	0.01	0.16
Map 9	0.12	0.01	0.13
Map 10	0.14	0.01	0.15

The CSC algorithm need 2.87 seconds to compress and decompress images for the first image set, while the second image set needs 3.91 seconds for both compression and decompression. The third image set has the best computation time with 0.7 seconds, because all of the images in the third image sets are JPEG images (JPEG format represent images with small intensities values). The fourth image set execution time is 3.69 seconds and the fifth image sets needed 1.41 second.

One of the main terms for measuring the compression algorithm performance is the computation time. Table 4.9 lists the average execution time resulted by calculating the average compression time and the average decompression time for the five image sets.

Table 4. 9 - The Lossless CSC Algorithm Average Compression Time

Image Sets	Average (Compression Time)	Average (De-Compression Time)	Average (Total Time)
Image Set 1	0.302	0.017	0.319
Image Set 2	0.416	0.019	0.435
Image Set 3	0.049	0.001	0.051
Image Set 4	0.359	0.01	0.369
Image Set 5	0.129	0.126	0.255
<b>Average</b>	<b>0.251</b>	<b>0.034</b>	<b>0.285</b>

Figure 4.7 represents the execution time for each image set. For a better understanding of the results, we displayed the algorithm results in bar-charts, where each column represents the value for the average needed time for each image sets.

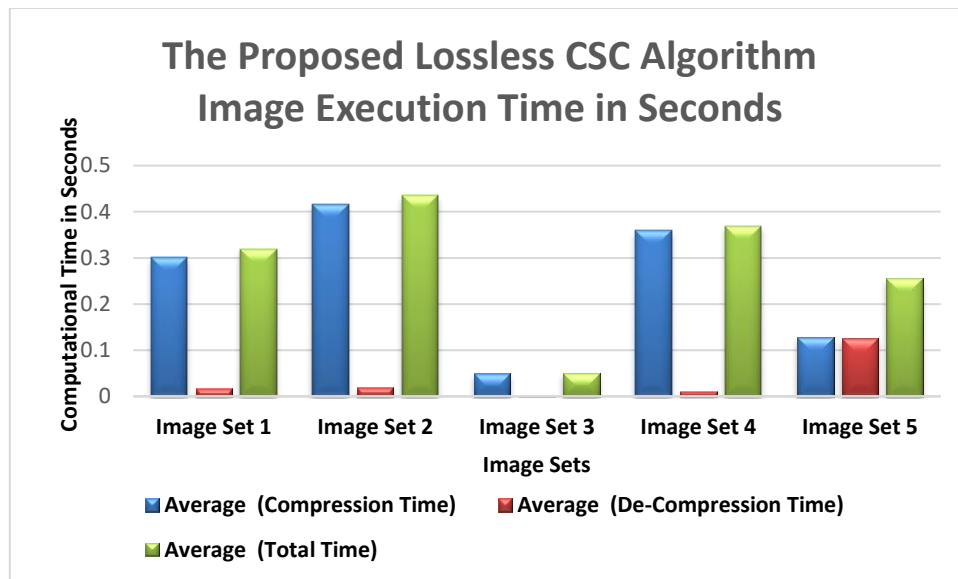


Figure 4. 7 - Compression and Decompression Time for the CSC Algorithm

## 4.6 Evaluations, Results and Observations

To describe the algorithm contribution, we investigated the results regarding the compression size, image quality and execution time. To reach the best conclusion from the results investigation, we need to compare our proposed lossless CSC algorithm results with the most common state of the art lossless algorithms and describe the analytical results to reach the best conclusion.

### 4.6.1 Comparison Between the CSC Results and Huffman Algorithm Results.

To compare the proposed lossless algorithm results with Huffman algorithm results, we need to compare both algorithm results regarding the main three features (image size, image quality and execution time).

#### 4.6.1.1 Comparison Between the CSC Algorithm and Huffman Algorithm in Terms of Image Size

By measuring the compression ratio for both algorithms, we can easily compare the results between both algorithms, to determine which one has the better compression ratio. Therefore, Table 4.10 lists the compression ratio for CSC algorithm and the Huffman algorithm.

Table 4. 10 - The Lossless CSC Compression Size and Huffman Compression Size

The Proposed Lossless CSC Algorithm Results				Huffman Results	
Image Set 1				Image Set 1	
Image	Original Size in KBs	New Size in KBS	Compression Ratio Cr	New Size in KBS	Compression Rate (Cr)
Baboon	225	94	2.394	104	2.163
Barbara	230	121	1.901	204	1.127
Boats	239	113	2.115	174	1.374
Boats	368	148	2.486	276	1.333
Camera Man	60	26	2.308	40	1.500
Camera Man	56	23	2.435	43	1.302
House	59	21	2.810	36	1.639
Lena	59	27	2.185	49	1.204
Lena	226	91	2.484	194	1.165
The Proposed Lossless CSC Algorithm Results				Huffman Results	
Image Set 2				Image Set 2	
Image	Original Size in KBs	New Size in KBS	Compression Ratio Cr	New Size in KBS	Compression Rate (Cr)
Lena	290	118	2.458	234	1.239
Lena	129	46	2.804	104	1.240
Lena	135	48	2.813	110	1.227
Airplane	751	249	3.016	459	1.636
Baboon	599	359	1.669	426	1.406
Barbara	1064	397	2.680	955	1.114
Boats	1148	373	3.078	898	1.278
Goldhill	1031	378	2.728	933	1.105
Pepper	651	346	1.882	526	1.238
The Proposed Lossless CSC Algorithm Results				Huffman Results	
Image Set 3				Image Set 3	
Image	Original Size in KBs	New Size in KBS	Compression Ratio Cr	New Size in KBS	Compression Rate (Cr)
Medic	32	12	2.667	32	1.000
Medic1	24	11	2.182	22	1.091
Butterfly	20	13	1.538	19	1.053
Mountain	28	11	2.545	26	1.077
swarm	25	11	2.273	24	1.042
Lake_jpg	25	12	2.083	24	1.042
Saturn_jpg	19	9	2.111	17	1.118
Earth_jpg	78	43	1.814	75	1.040
boat_jpg	25	10	2.500	24	1.042
Waterfall	26	13	2.000	25	1.040
Eagle	17	8	2.125	14	1.214
Grand_Sone	28	12	2.333	27	1.037
Car	28	10	2.800	21	1.333
Shape	31	15	2.067	25	1.240
The Proposed Lossless CSC Algorithm Results				Huffman Results	
Image Set 4				Image Set 4	

Image	Original Size in KBs	New Size in KBS	Compression Ratio Cr	New Size in KBS	Compression Rate (Cr)
Knob & Bolt	1155	320	3.609	544	2.123
Houses	1149	395	2.909	878	1.309
Landscape	1153	396	2.912	844	1.366
Light House	1150	311	3.698	754	1.525
Barn	1149	355	3.237	809	1.420
Parrots	1153	297	3.882	829	1.391
Flowers & Sill	1154	282	4.092	773	1.493
Six-Shooter	1152	270	4.267	632	1.823
Motocross	1150	376	3.059	851	1.351
Zentime	1155	307	3.762	749	1.542
<b>The Proposed Lossless CSC Algorithm Results</b>				<b>Huffman Results</b>	
<b>Image Set 5</b>				<b>Image Set 5</b>	
Image	Original Size in KBs	New Size in KBS	Compression Ratio Cr	New Size in KBS	Compression Rate (Cr)
Map 1	761	149	5.107	119	6.395
Map 2	821	136	6.037	115	7.139
Map 3	801	141	5.681	118	6.788
Map 4	838	120	6.983	110	7.618
Map 5	800	151	5.298	123	6.504
Map 6	826	147	5.619	118	7.000
Map 7	838	133	6.301	112	7.482
Map 8	814	154	5.286	119	6.840
Map 9	769	158	4.867	121	6.355
Map 10	768	149	5.154	117	6.564

As listed in Table 4.11, the proposed CSC algorithm decreases the image size more than the Huffman algorithm for all of the image sets except image set five. In the first, second, third and fourth image sets, the CSC algorithm results decrease the storage saving percentage more than the Huffman algorithm by saving 29.8%, 38.85%, 45.55% and 37.79% respectively. By observing the results for the fifth image sets, we conclude that the Huffman algorithm saved 3.35% more than the proposed algorithm, since the raster map image has low resolution and less unique values to represent the image.

By averaging the five-test image storage saving results for both algorithms, the CSC algorithm saved 29.72% more than the Huffman algorithm.

Table 4. 11 - The Average Lossless Approach Compression Size with Huffman Average Compression Size

The CSC Algorithm Average Results			Huffman Average Results	
Image Sets	Average Compression Ratio Cr	Average Storage Saving %	Average Compression Ratio Cr	Average Storage Saving %
Image Set 1	2.35	56.9	1.42	27.10
Image Set 2	2.569	59.48	1.276	20.63
Image Set 3	2.224	53.85	1.10	8.3
Image Set 4	3.543	71.26	1.534	33.47
Image Set 5	5.635	82.05	6.868	85.4
<b>Average</b>	<b>3.1688</b>	<b>63.828</b>	<b>2.4328</b>	<b>34.364</b>

Figure 4.8 describes the average storage saving to the proposed CSC algorithm and Huffman algorithm for each of the five image sets. For better results understanding, we displayed the algorithm results in bar-charts, where each column represents the image set and the value for the bits-saving percentage. The proposed lossless algorithm has a better compression ratio.

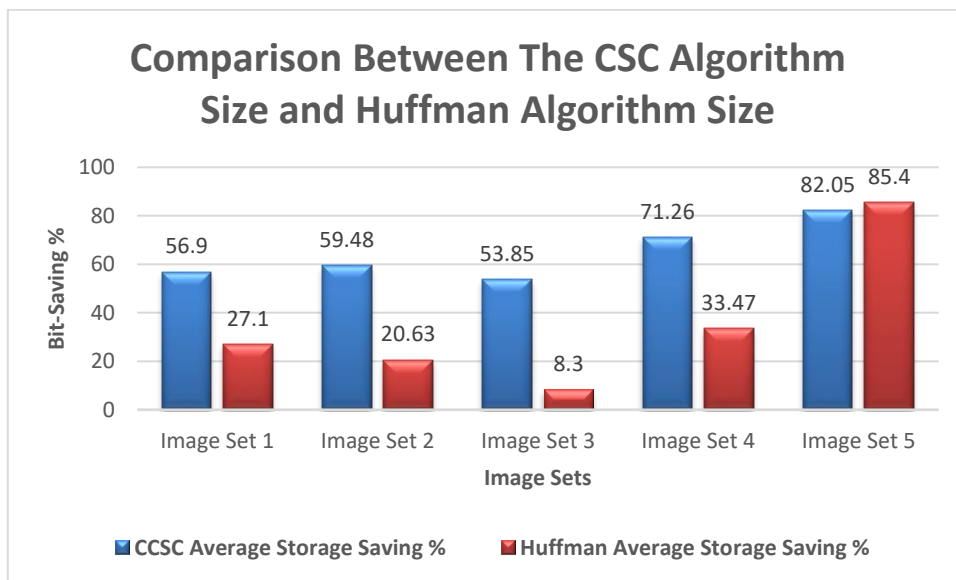


Figure 4. 8 - The Average Compression Size for the CSC and Huffman algorithm

#### 4.6.1.2 Comparison Between the CSC Algorithm and Huffman in Terms of Image Quality

Since both the proposed CSC algorithm and Huffman algorithm are lossless techniques, the two algorithm results should have zero distortion after decompressing the tested image.

#### 4.6.1.3 Comparison Between the CSC and Huffman in Terms of Execution Time

Table 4.12 lists the execution time (compression and decompression time) for the proposed algorithm and the execution time for the Huffman algorithm, for all the five image sets.

Table 4. 12 - The Lossless CSC Algorithm Execution Time with Huffman Execution Time

CSC Algorithm Execution Time				Huffman Algorithm Execution Time
<b>Image Set 1</b>				
Image	Compression Time	Decompression Time	Total Time	Huffman Total Time
1	0.303	0.003	0.306	0.4
2	0.313	0.007	0.32	0.8
3	0.254	0.11	0.364	0.06
4	0.32	0.012	0.332	0.4
5	0.28	0.003	0.283	0.4
6	0.25	0.003	0.253	0.4
7	0.225	0.002	0.227	0.4
8	0.257	0.002	0.259	0.6
9	0.519	0.007	0.526	0.6
CSC Algorithm Execution Time				Huffman Algorithm Execution Time
<b>Image Set 2</b>				
Image	Compression Time	Decompression Time	Total Time	Huffman Total Time
1	0.22	0.01	0.23	0.2
2	0.12	0.009	0.129	0.3
3	0.15	0.009	0.159	0.06
4	0.46	0.02	0.48	0.8
5	0.5	0.02	0.52	0.55
6	0.64	0.03	0.67	1.6
7	0.55	0.03	0.58	1.6
8	0.54	0.03	0.57	1.8
9	0.56	0.02	0.58	1.8
CSC Algorithm Execution Time				Huffman Algorithm Execution Time
<b>Image Set 3</b>				
Image	Compression Time	Decompression Time	Total Time	Huffman Total Time
1	0.06	0.0003	0.0603	0.12
2	0.08	0.0005	0.0805	0.12
3	0.08	0.0005	0.0805	0.1
4	0.04	0.005	0.045	0.08

5	0.07	0.0004	0.0704	0.08
6	0.04	0.0004	0.0404	0.06
7	0.04	0.0006	0.0406	0.06
8	0.04	0.002	0.042	0.34
9	0.04	0.0006	0.0406	0.06
10	0.04	0.0006	0.0406	0.06
11	0.04	0.0007	0.0407	0.06
12	0.04	0.0005	0.0405	0.02
13	0.04	0.006	0.046	0.06
14	0.04	0.0004	0.0404	0.15
<b>CSC Algorithm Execution Time</b>				<b>Huffman Algorithm Execution Time</b>
<b>Image Set 4</b>				
<b>Image</b>	<b>Compression Time</b>	<b>Decompression Time</b>	<b>Total Time</b>	<b>Huffman Total Time</b>
1	0.35	0.01	0.36	2.34
2	0.34	0.01	0.35	2.3
3	0.38	0.01	0.39	2.2
4	0.35	0.01	0.36	2.2
5	0.41	0.01	0.42	2.2
6	0.34	0.01	0.35	2.2
7	0.3	0.01	0.31	2.1
8	0.36	0.01	0.37	2.1
9	0.42	0.01	0.43	2.2
10	0.34	0.01	0.35	2.1
<b>CSC Algorithm Execution Time</b>				<b>Huffman Algorithm Execution Time</b>
<b>Image Set 5</b>				
<b>Image</b>	<b>Compression Time</b>	<b>Decompression Time</b>	<b>Total Time</b>	<b>Huffman Total Time</b>
1	0.12	0.01	0.13	0.14
2	0.2	0.02	0.22	0.16
3	0.13	0.018	0.148	0.16
4	0.1	0.018	0.118	0.15
5	0.14	0.01	0.15	0.2
6	0.09	0.01	0.1	0.15
7	0.1	0.01	0.11	0.16
8	0.15	0.01	0.16	0.16
9	0.12	0.01	0.13	0.16
10	0.14	0.01	0.15	0.15

Table 4.13 lists the average total time for compression and decompression for both algorithms for each test image.



Table 4. 13 - The Average Execution Time for Both Algorithms in Seconds

The Proposed Lossless CSC Algorithm Average Execution Time				Huffman Average Execution Time
Image Sets	Average Compression Time	Average De-Compression Time	Total Execution Time	Total Execution Time
Image Set 1	0.30	0.02	0.32	0.45
Image Set 2	0.42	0.02	0.44	0.97
Image Set 3	0.05	0.001	0.05	0.10
Image Set 4	0.36	0.01	0.37	2.19
Image Set 5	0.13	0.01	0.14	0.16
<b>Average</b>	<b>0.25</b>	<b>0.01</b>	<b>0.26</b>	<b>0.77</b>

Figure 4.9 shows the total execution time needed for compression and decompression for each of the image sets for the two algorithms. The Figure shows that the CSC algorithm has better execution time for all image sets. Due to the CSC algorithm’s simplicity of implementation and execution speed.

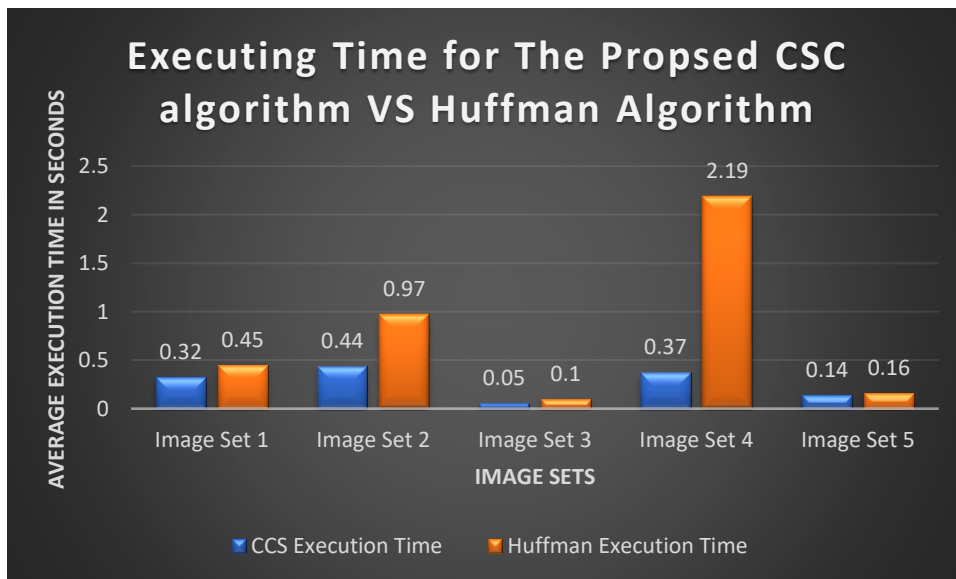


Figure 4. 9 - The Average Execution Time for Both Algorithms

#### 4.6.2 Comparison Between the CSC Compression Size and Other State of the Art Algorithm.

The CSC algorithm was tested as lossless compression technique and compared against other benchmark scheme for natural images compression obtained from (Khan *et al.*, 2017), the

comparison with the most recent state of the art algorithms is needed for a better evaluation for the proposed algorithm results.

#### 4.6.2.1 First Comparison for Natural Images Compression Size

Table 4.14 shows the compression sizes in KBs and the compression ratios Cr, for the basic BWCA, KMTF based BWCA, JPEG 2000 LS, RCT-BWCA algorithm obtained from (Khan *et al.*, 2017) and the proposed CSC algorithm results.

Table 4. 14 - The Proposed Algorithm Results Compared with Other Four Algorithm Results

Image	BWCA		KMTF - BWCA		JPEG-2000 LS		RCT - BWCA		CSC	
	Size	Cr	Size	Cr	Size	Cr	Size	Cr	Size	Cr
<b>Knob &amp; Bolt</b>	765	1.510	750	1.540	487	2.370	381	3.020	320	3.609
<b>Houses</b>	1008	1.140	981	1.170	578	1.990	463	2.490	395	2.909
<b>Landscape</b>	1020	1.130	965	1.190	612	1.880	352	3.270	396	2.912
<b>Light House</b>	827	1.390	783	1.470	509	2.260	480	2.400	311	3.698
<b>Barn</b>	891	1.290	839	1.370	525	2.190	358	3.220	355	3.237
<b>Parrots</b>	791	1.460	739	1.560	447	2.580	346	3.330	297	3.882
<b>Flowers &amp; Sill</b>	780	1.480	743	1.550	457	2.520	350	3.290	282	4.092
<b>Six-Shooter</b>	591	1.950	560	2.060	433	2.660	332	3.470	270	4.267
<b>Motocross</b>	991	1.160	947	1.220	574	2.010	244	4.720	376	3.059
<b>Zentime</b>	837	1.380	800	1.440	494	2.330	297	3.880	307	3.762
<b>AVERAGE</b>	<b>850</b>	<b>1.39</b>	<b>811</b>	<b>1.46</b>	<b>512</b>	<b>2.28</b>	<b>360</b>	<b>3.31</b>	<b>331</b>	<b>3.54</b>

As described in Figure 4.10, the proposed CSC algorithm achieved the best compression ratio. The CSC has 0.23 better compression ratio than the RCT – BWCA and 1.26 more than the JPEG 2000 LS Cr and 2.08 more than the KMTF – BWCA and 2.15 more than the basic BWCA Cr. The CSC algorithm reach the best compression ratio due to the use of the Column Subtraction Compression function, this function provides better compression ratio with high resolution images more than the low-resolution images, since the intensity values in the high-resolution image are very close and those images are represented with large number of colours.

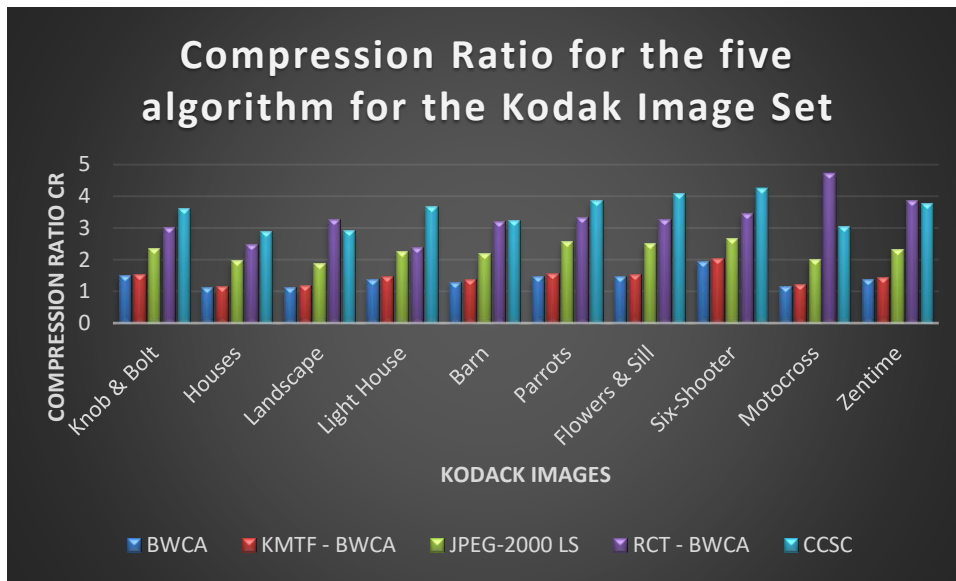


Figure 4. 10 - Compression Ratio for the Five Algorithm

#### 4.6.2.2 Second Comparison for Natural Images Compression Size

Table 4.15 lists the images compression size in KBs for the proposed CSC algorithm and the lossless benchmark compression schemes for Kodak colour images obtained from (Khan *et al.*, 2017).

Table 4. 15 - Comparison Between the CSC and various benchmark systems in Term of Compressed File Sizes of Kodak Colour Test Images (size in KBs)

#	SCHEME	KODAK TEST IMAGE										Total Size
		Knob & Bolt	Houses	Landscape	Light House	Barn	Parrots	Flowers & Sill	Six-Shooter	Motocross	Zentime	
1	ADVANCE COMP	608	422	495	493	477	666	460	468	614	451	5154
2	ALLUME	386	576	487	503	699	407	788	370	410	372	4998
3	BBWCA	381	463	352	480	308	346	350	332	244	297	3553
4	BCM	457	785	628	490	532	737	375	381	507	390	5282
5	BULK ZIP	450	710	372	441	753	613	469	751	482	603	5644
6	CAESIUM	475	422	496	518	797	500	706	407	736	411	5468
7	CSC	320	395	396	311	355	297	282	270	376	307	3310
8	C-MIX	453	510	457	497	625	520	761	666	657	456	5602
9	COMPRESSOR.IO	413	765	425	597	380	415	392	391	603	668	5049
10	CRUSH	409	692	556	536	544	795	646	612	583	376	5749
11	FILE MINIMIZER	560	758	471	521	740	681	691	596	573	495	6086

12	FILE OPTIMIZER	409	432	721	665	559	502	401	447	618	518	5272
13	HEVC (x265)	403	493	343	487	397	538	326	418	462	353	4220
14	LZ4X	370	420	412	373	402	537	465	456	444	429	4308
15	MRP	497	760	550	511	513	791	534	474	628	409	5667
16	NANOZIP	475	556	472	519	446	543	490	431	551	715	5198
17	PAQ8PXD_V4	450	490	598	607	489	655	733	596	372	569	5559
18	UPACK 0.25	661	710	379	675	503	529	371	572	587	568	5555
19	WINRK 3.1.2	598	515	398	783	674	457	374	611	456	593	5459
20	ZCM 0.92	495	631	772	542	408	714	565	597	416	450	5590

As displayed in Figure 4.11, the best compression came with the proposed CSC algorithm with a total size of 3310 KBs and a Cr with 3.54. The second-best algorithm is the BBWCA followed by the HEVC. The CSC decrease the image size with 243 KBs more than the BBWCA and 910 KBs more than the HEVC.

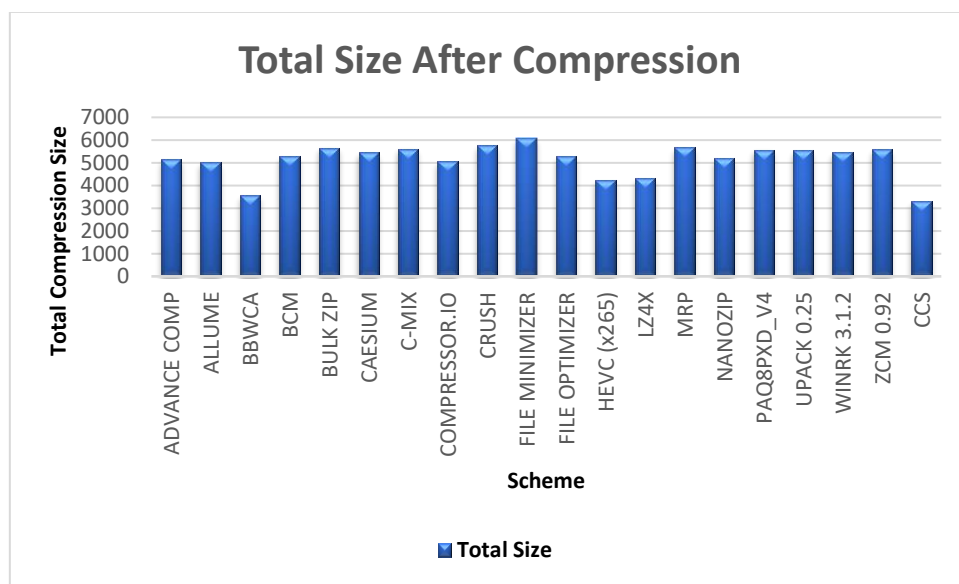


Figure 4. 11 - Total Compression Size for the Kodak Image Set

#### 4.6.2.3 Third Comparison for Raster Map Images Size

Navigation systems used the raster maps images that is mostly created by systems. This type of images is represented by a lot of redundant data between the three image channels RGB, which indicate that, the number of unique values that represent the raster map image is much less than the number of unique values that represent the high-resolution images, such as natural images. Map transmission efficiency can affect the navigation systems over wireless networks.

The use of lossless image technique is important to achieve high quality image at decompression phase. The proposed CSC algorithm reaches the best compression ratio with the high-resolution images, and less compression ratio with low resolution image.

The proposed CSC was tested, and the results were compared with another raster map benchmark scheme. The comparison is done with the JPEG-LS, PNG, GIF, Bi-level Burrows BBWCA and BLiSE algorithm. Table 4.16 lists the compression size in KBs and the compression ratio for the previous algorithms and the CSC algorithm for 10 raster map images obtained from <https://sites.google.com/site/qinzoucn/documents/>.

Table 4. 16 - The CSC Results Compared with Other Four Algorithm Results

Image	JPEG-LS		PNG		GIF		BLiSE		BBWCA		CSC	
	Size	Cr	Size	Cr	Size	Cr	Size	Cr	Size	Cr	Size	Cr
<b>Map 1</b>	235.32	3.23	29.79	25.55	20.92	36.38	12.28	61.98	11.72	64.94	149	5.11
<b>Map 2</b>	188.88	4.35	28.00	29.32	17.57	46.73	10.04	81.78	10.40	78.95	136	6.04
<b>Map 3</b>	185.36	4.32	27.15	29.50	18.39	43.55	10.94	73.21	11.22	71.38	141	5.68
<b>Map 4</b>	88.82	9.43	11.08	75.62	8.38	99.98	3.72	225.22	3.57	234.68	120	6.98
<b>Map 5</b>	238.76	3.35	34.67	23.08	22.86	35.01	16.35	48.95	16.15	49.55	151	5.30
<b>Map 6</b>	228.04	3.62	30.23	27.32	17.90	46.14	11.08	74.54	12.76	64.73	147	5.62
<b>Map 7</b>	157.13	5.33	23.59	35.52	14.21	58.96	7.37	113.69	10.41	80.49	133	6.30
<b>Map 8</b>	254.97	3.19	36.58	22.25	23.61	34.47	18.10	44.96	15.36	52.98	154	5.29
<b>Map 9</b>	265.47	2.90	41.62	18.49	25.14	30.61	19.43	39.60	17.22	44.68	158	4.87
<b>Map 10</b>	215.75	3.56	34.38	22.33	22.29	34.44	14.56	52.72	14.45	53.12	149	5.15
<b>Average</b>	<b>2058.5</b>	<b>4.32</b>	<b>297.09</b>	<b>30.89</b>	<b>191.3</b>	<b>46.62</b>	<b>123.9</b>	<b>81.66</b>	<b>123.3</b>	<b>79.55</b>	<b>143.8</b>	<b>5.63</b>

As displayed in Figure 4.12, BLiSE outperforms all other algorithms, since the BLiSE algorithm is designed for raster maps compression and the other algorithms are designed for general images type compression. The CSC algorithm results is better than the JPEG-LS and less than the other algorithms.

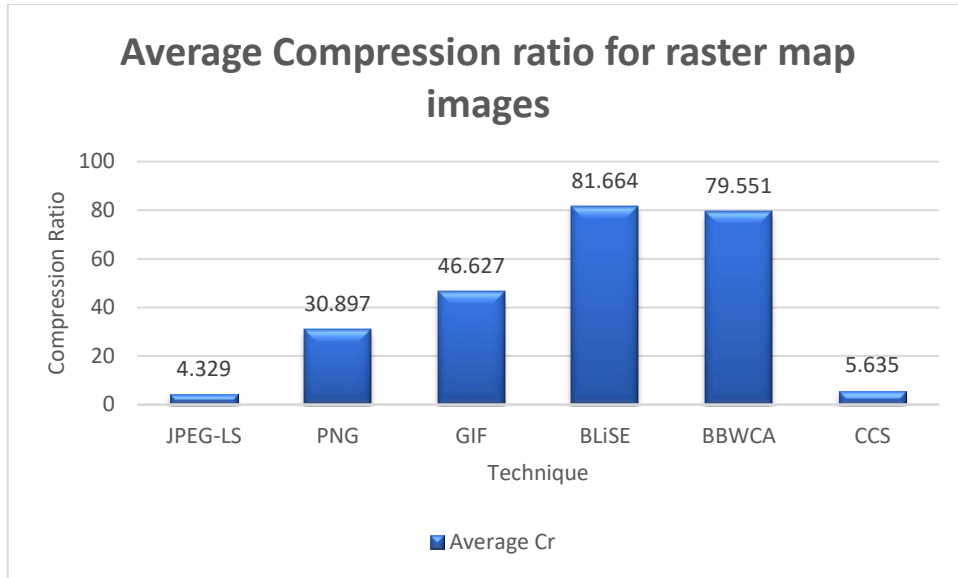


Figure 4. 12 - Average Compression Ratio for the Raster Map Image Set

#### 4.6.3 Comparison Between the CSC Execution Time and Other State of the Art Algorithms.

For the best results evaluation, the CSC algorithm was tested on the same environments as (Khan *et al.*, 2017) by using the software and hardware listed in Table 4.17 and compared with the BWCA, KMTF-BWCA and RCT\_BWCA obtained from (Khan *et al.*, 2017).

Table 4. 17 - System Requirements

Hardware	Software
Intel Core 2 Quad CPU @ 2.4 GHz	Windows XP OS
2 GB RAM	MATLAB
1 GB Virtual Memory	

Table 4.18 presents the compression needed time (CT) and the de compression needed time (DCT) for the BWCA, KMTF – BWCA, and RCT – BWCA obtained from (Khan *et al.*, 2017).

Table 4. 18 - Execution Time in Seconds for the Kodak Image Set for Different Algorithms

Image	BWCA		KMTF - BWCA		RCT - BWCA		CSC Win Xp		CSC Win 10	
	CT	DCT	CT	DCT	CT	DCT	CT	DCT	CT	DCT
Knob & Bolt	16.46	23.76	17.21	25.39	17.7	28.06	1.43	0.1	0.35	0.01
Houses	16.32	21.32	17.07	24.82	19.77	27.8	1.79	0.1	0.34	0.01
Landscape	17.37	22.77	18.12	25.55	18.18	28.66	1.99	0.1	0.38	0.01
Light House	19.17	21.46	19.92	22.63	19.48	28.95	1.85	0.1	0.35	0.01

<b>Barn</b>	16.91	21.97	17.66	22.19	19.6	22.87	2.21	0.1	0.41	0.01
<b>Parrots</b>	16.78	22.9	17.53	25.98	19.95	29.23	1.69	0.1	0.34	0.01
<b>Flowers &amp; Sill</b>	17.58	22.46	18.33	25.95	18.41	30.6	1.48	0.1	0.3	0.01
<b>Six-Shooter</b>	18.82	21.17	19.57	24.4	20.98	27.46	2.07	0.1	0.36	0.01
<b>Motocross</b>	16.07	23.85	16.82	28.09	19.14	30.5	2.03	0.1	0.42	0.01
<b>Zentime</b>	18.02	23.34	18.77	23.24	19.49	28.55	1.82	0.1	0.34	0.01
<b>Sum</b>	<b>173.5</b>	<b>225</b>	<b>181</b>	<b>248.24</b>	<b>192.7</b>	<b>282.68</b>	<b>18.36</b>	<b>1</b>	<b>3.59</b>	<b>.01</b>
<b>Total</b>	<b>398.5</b>		<b>429.24</b>		<b>475.38</b>		<b>19.36</b>		<b>3.69</b>	

The CSC algorithm achieved the best execution time by having 19.36s for compression and decompression together and save 379.14s more than the BWCA algorithm.

By running the CSC algorithm using win 10 operating system 64-bit with Intel core i7-7500U CPU @2.70GHz with 8 GB RAM the execution time is dramatically decreased to 3.69s. Figure 4.13 display the total execution time for the four algorithms in seconds.

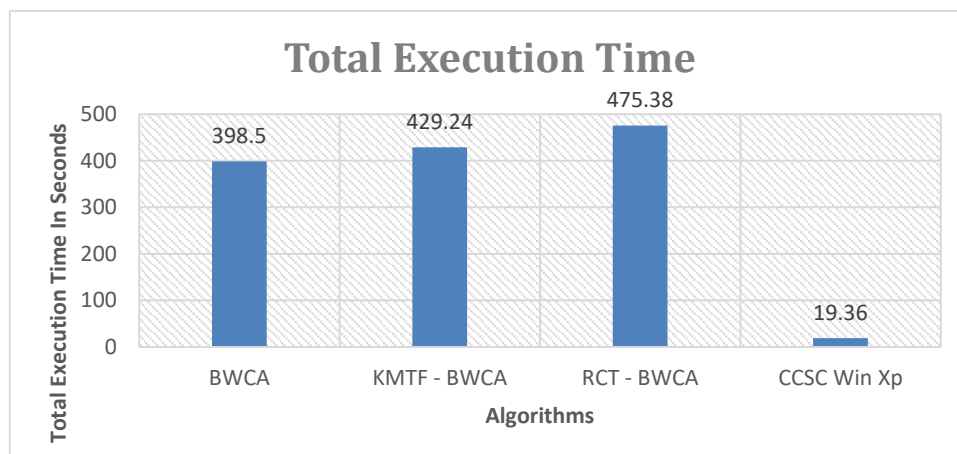


Figure 4. 13 - Total Execution Time for the Kodak Image Set

## 4.7 Chapter Summary

The aim of this chapter was to develop a lossless image compression algorithm, which enhances the current state of the art compression ratio, with zero distortion and acceptable execution time.

Many different lossless compression algorithms were created by different researchers, and all of them restored the image exactly as it was before compression. Some of the lossless approaches have high compression rates with slow performance, such as LZW, while in other

approaches, a high-performance speed with a lower compression ratio is achieved, such as the Huffman algorithm.

To reach the optimal solution between compression time and saving bits, the CSC algorithm where proposed.

By taking the best features from the fastest algorithm and the best features from the highest compression rate algorithms, and at the same time, by avoiding using any slow function or irreversible function, we created a novel lossless algorithm that enhanced the compression ratio more than the state of the art algorithm from the literature review, with an acceptable executing time and zero percentage of distortion.

- The proposed CSC algorithm is designed to work with any image format and resolution.
- The algorithm achieved the best compression ratio for natural images.
- The CSC algorithm achieved the best computational time due to its simplicity of implementation and speed of execution.

The only disadvantage of the CSC algorithm is when applying the algorithm on raster map images, the compression ratio is not the best, since the CSC algorithm gives better results for the high-resolution images.

To solve this issue, the researcher proposed a new algorithm that enhanced the compression ratio for the low-resolution images. The next chapter describe the solution in detail.



## **CHAPTER FIVE: THE PROPOSED LOSSLESS ALGORITHM FOR RASTER MAP IMAGES COMPRESSION**

---

### **5 Chapter Overview**

This chapter describes in detail the proposed lossless Low-Resolution Column Subtraction Compression algorithm (LRCSC). It starts with a detailed explanation of all the procedures used in the proposed algorithm, followed by the validation of the algorithm by testing its results, and a comparison of the newly developed algorithm with state-of-the-art algorithms.

---

#### **5.1 Introducing the LRCSC Compression Algorithm**

The lossless LRCSC algorithm is designed to work with any application and supports all image formats whether the input image is of a high resolution or low-resolution. The CSC algorithm provided the best compression ratio with the high-resolution images such as natural images and less compression ratio when compressing low resolution images such as raster map images, to solve this issue, the researcher designed an updated algorithm for the CSC algorithm, called the Low-Resolution Column Subtraction Compression algorithm (LRCSC).

Once the raster map image is compressed by using the CSC algorithm, the resulted three matrices should be represented with a smaller number of unique values with large areas of contiguous colour; where the value is repeated often (Runs).

By using the Huffman algorithm as a post-processing phase for the CSC algorithm we should represent the image unique values with a smaller weight for enhancing the current compression ratio.

Then the RLE compression algorithm should take place for enhancing the compression ratio even more by encoding the runs using their probabilities and values (value; probability).

By combining the CSC algorithm with the Huffman coding and RLE coding algorithms we enhanced the current compression ratio.

The proposed LRCSC algorithm uses the following reversible procedures:

Let **RMI** be a coloured image referred to a Raster Map Image represented with three colour matrices (R, G and B). The three matrices have the same resolution of  $m \times n$  where  $m$  and  $n$  represents the matrix dimensions (Row ( $m$ ), Columns ( $n$ )) and  $(i,j)$  refers to the elements coordinates. **Dg** refers to the transformed green matrix, **Db** refers to the transformed blue matrix and **TI** referred to the Transformed Image.

3. **Procedure transformation** is to map the input natural image **NI(R,G,B)** from the **RGB** colour space into **RDgDb** colour space. The transformation procedure output is an image with three

less coloration matrices, it maps the pixel values into new space that includes smaller values  $TI(R,Dg,Db)$ . The transformed matrices have the same dimensions of the input matrices (m x n).  $Dg$  refers to the transformed green matrix,  $Db$  refers to the transformed blue matrix and  $TI$  referred to the Transformed Image.

1. **Procedure CCS Compression** is to decrease the size of the transformed image  $TI(R,Dg,Db)$  by using the column subtraction compression. where  $CRR$  refers to the compressed red channel,  $CRG$  refers to the compressed green channel and  $CRB$  refers to the compressed blue channel. The compression procedure output is the compressed image  $CI(CRR, CRG, CRB)$ .
2. **Procedure Negative Huffman:** is to decrease the number of unique values that represent the image to enhance the compression ratio for the next phase (Huffman coding). It starts by creating a temporary dictionary file that saves the addresses for the negative values for the three matrices from the previous phase ( $CRR, CRG, CRB$ ) and multiply all of the negative values by (-1). Followed by applying the Huffman algorithm for the positive three matrices ( $Image\_R1, Image\_G1, Image\_B1$ ). Finally, restoring the negative values by using the temporary dictionary file. The output of this procedure is three matrices ( $Image\_R3, Image\_G3, Image\_B3$ ).
3. **Procedure RLE:** starts by applying the RLE algorithm on the three matrices from the previous phase ( $Image\_R3, Image\_G3, Image\_B3$ ). The output of this procedure is three matrices ( $Image\_R4, Image\_G4, Image\_B4$ ).

Algorithm 3 illustrates the raster map image compression procedures.

### Algorithm3: LRCSC For Raster Map Images Compression

#### 1: Procedure Transformation

- 2: input RMI: Raster Map Image RMI(R,G,B)
- 3: m = total rows number; n = total columns number;
- 4: // Dg, Db are (m X n) matrices of natural number;
- 5: // Dg matrix is resulted by subtracting the G matrix from the R matrix using the following formula  

$$Dg(\text{row}, \text{column}) = R(\text{row}, \text{column}) - G(\text{row}, \text{column})$$
- 6: // Db matrix is resulted by subtracting the G matrix from the B matrix using the following formula  

$$Db(\text{row}, \text{column}) = B(\text{row}, \text{column}) - G(\text{row}, \text{column})$$
- 7: output: transformed image  $TI(R,Dg,Db)$

#### 8: End Procedure Transformation

#### 1: Procedure Compression

- 2: input TI: transformed image  $TI(R,Dg,Db)$
- 3: //CRR, CRG, CRB; are matrices of natural number
- 4: // CRR matrix is resulted by the following nested for loop

```

5:     from the first column to the last column - 1
6:         for all rows in the column do
7:             CRR (row, column) = R(row, column) - R(row, column+1);
8:         end
9:     end
10: // CRG matrix is resulted by the following nested for loop
11:    from the first column to the last column - 1
12:        for all rows in the column do
13:            CRG (row, column) = Dg(row, column) - Dg(row, column+1);
14:        end
15:    end
16: // CRB matrix is resulted by the following nested for loop
17:    from the first column to the last column - 1
18:        for all rows in the column do
19:            CRB (row, column) = Db(row, column) - Db(row, column +1);
20:        end
21:    end
22:    output: compressed image CI(CRR, CRG, CRB)

```

### 23: End Procedure Compression

#### 1: Procedure Negative Huffman

```

2:  input CI: compressed image CI(CRR, CRG, CRB)
3: // convert the values of the CRR,CRG,CRB matrices to positive integer values
4:  if (CRR (row, column) < 0)
5:      Image_R1 (row, column) = CRR(row, column) * -1
6:      Dictionary_R (row, column) = 1
7:  else
8:      Dictionary_R (row, column) = 0
9:  endif
10: if (CRG (row, column) < 0)
11:     Image_G1(row, column) = CRG row, column) * -1
12:     Dictionary_G (row, column) = 1
13: else
14:     Dictionary_G (row, column) = 0
15: endif
16: if (CRB (row, column) < 0)
17:     Image_B1 (row, column) = CRB (row, column) * -1
18:     Dictionary_B (row, column) = 1
19: else
20:     Dictionary_B (row, column) = 0
21: endif
22:// Huffman Coding
23:     Image_R2 (row, column) = Huffman (Image_R1 (row, column))
24:     Image_G2 (row, column) = Huffman (Image_G1 (row, column))
25:     Image_B2 (row, column) = Huffman (Image_B1 (row, column))
26: // Restore the negative values
27:  if (Dictionary_R(row, column)=1)
28:      Image_R3 (row, column) = Image_R2 (row, column) * -1
29:  endif

```

```

30:   if (Dictionary_G (row, column) = 1)
31:       Image_G3 (row, column) = Image_G2 (row, column) * -1
32:   endif
33:   if (Dictionary_B (row, column) = 1)
34:       Image_B3 (row, column) = Image_B2 (row, column) * -1
35:   endif
36:   output: compressed matrices Image_R3, Image_G3, Image_B3
37:End Procedure Negative Huffman

```

#### **1: Procedure RLE**

```

2:   input Image_R3, Image_G3, Image_B3
3:       Image_R4 = RLE(Image_R3)
4:       Image_G4 = RLE(Image_G3)
5:       Image_B4 = RLE(Image_B3)
6:   output: compressed matrices (Image_R4, Image_G4 and Image_B4)
7: End Procedure RLE

```

By implementing the LRCSC algorithm, we expect to decrease the image size and maintain the image quality as it was before compression in a very fast time. Figure 5.1 shows the LRCSC flowchart. The decompression algorithm used the following procedures for reconstructing the compressed image as follows.

#### **Algorithm4: LRCSC For Raster Map Images De-Compression**

##### **1: Procedure RLE Decompression**

```

2: input compressed image matrices Image_R4, Image_G4, Image_B4
3:   Image_R3 = De_RLE(Image_R4)
4:   Image_G3 = De_RLE(Image_G4)
5:   Image_B3 = De_RLE(Image_B4)
6: output: RLE De compressed matrices (Image_R3, Image_G3 and Image_B3)
7: End Procedure RLE Decompression

```

##### **1: Procedure Negative Huffman**

```

2:input three matrices (Image_R3, Image_G3 and Image_B3)
3:// convert the values of the Image_R3, Image_G3 and Image_B3 matrices to positive integer values
4:   if (Image_R3 (row, column) < 0)
5:       Image_R2 (row, column) = Image_R3 (row, column) * -1
6:       Dictionary_R (row, column) = 1
7:   Else
8:       Dictionary_R (row, column) = 0
9:   endif
10:  if (Image_G3 (row, column) < 0)
11:      Image_G2 (row, column) = Image_G3 (row, column) * -1
12:      Dictionary_G (row, column) = 1
13:  Else
14:      Dictionary_G (row, column) = 0
15:  endif
16:  if (Image_B3 (row, column) < 0)

```

```

17:         Image_B2 (row, column) = Image_B3 (row, column) * -1
18:         Dictionary_B (row, column) = 1
19:     Else
20:         Dictionary_B (row, column) = 0
21:     endif
22 // Huffman Decoding
23:         Image_R1 (row, column) = Huffman_Decoding (Image_R2 (row, column))
24:         Image_G1 (row, column) = Huffman_Decoding (Image_G2 (row, column))
25:         Image_B1 (row, column) = Huffman_Decoding (Image_B2 (row, column))
26: // Restore the negative values
27:     if (Dictionary_R (row, column) = 1)
28:         CRR (row, column) = Image_R2 (row, column) * -1
29:     endif
30:     if (Dictionary_G (row, column) = 1)
31:         CRG (row, column) = Image_G2 (row, column) * -1
32:     endif
33:     if (Dictionary_B (row, column) = 1)
34:         CRB (row, column) = Image_B2 (row, column) * -1
35:     endif
36: output: CI(CRR, CRG, CRB)

```

### **37: End Procedure Negative Huffman**

#### **1: Procedure LRCSC Decompression**

```

2:  input compressed image CI(CRR, CRG, CRB)
3:  m = total rows number; n = total columns number;
4: //  R,Dg,Db are (m X n) matrices of natural number;
5: // R matrix is resulted by the following nested for loop
6:  from the last column to the first column + 1
7:      for all rows in the column do
8:          R (row, column-1) = CRR(row, column) + CRR(row, column-1);
9:      end
10: end
11: // Dg matrix is resulted by the following nested for loop
12: from the last column to the first column + 1
13:     for all rows in the column do
14:         Dg (row, column-1) = CRG(row, column) + CRG(row, column-1);
15:     end
16: end
17: // Db matrix is resulted by the following nested for loop
18: from the last column to the first column + 1
19:     for all rows in the column do
20:         Db (row, column-1) = CRB(row, column) + CRB(row, column-1);
21:     end
22: end
23: output: transformed image TI(R,Dg,Db)

```

### **24: End Procedure LRCSC Decompression**

#### **1: Procedure Revers Transformation**

```

2  input R,Dg,Db; matrices of natural number
3: //  R,G,B are (m X n) matrices of natural number
4: //  G matrix is resulted by subtracting the Dg matrix from the R matrix

```

- 5:  $G(\text{row, column}) = R(\text{row, column}) - DG(\text{row, column})$
- 6: // B matrix is resulted by adding the G matrix values to the Db matrix values
- 7:  $B(\text{row, column}) = G(\text{row, column}) + DB(\text{row, column})$
- 8: output: reconstructed image RMI (R,G,B)
- 9: End Procedure Revers Transformation

## 5.2 The LRCSC Flowchart

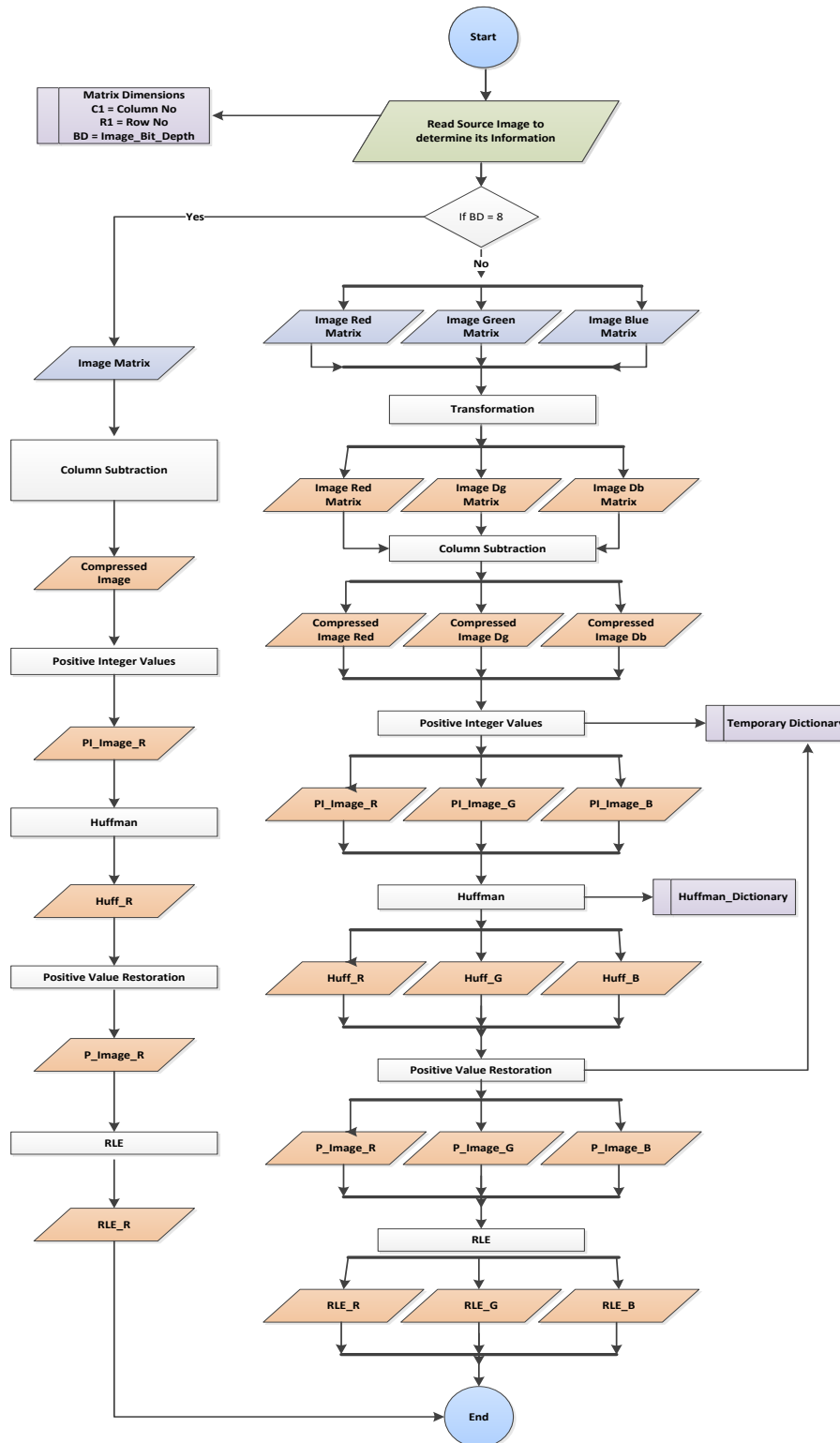


Figure 5. 1 - LRCSC Lossless Algorithm Flowchart

### 5.3 Description of the LRCSC Algorithm

The algorithm starts with loading the source image to identify the matrix dimension and the input image bit-depth, followed by a procedure to specify the suitable compression steps that meet the input image; if the image bit-depth is equal to eight then the image cannot be transformed, and will be sent to the subtraction function phase directly, and if the image bit-depth is 24 then the image will be loaded to the transformation phase as a pre-processing to the column subtraction function.

#### 5.3.1 CSC Algorithm:

The first phase of the LRCSC is using the CSC algorithm for the input image as a pre-processing phase. This phase is responsible for colour transformation and decreasing the image size by using the column subtraction compression algorithm from the previous chapter. To illustrate how the algorithm work, we used, and example as shown in Figure 5.2 which describe the transformation for the three colours space by using sample example for 8x8 block obtained from a raster map image.

Sample 8x8 for the Red Matrix (R)								=	R = Red Matrix R							
255	255	255	128	255	255	128	255		255	255	255	128	255	255	128	255
255	255	255	255	128	128	128	128		255	255	255	255	128	128	128	128
255	255	255	128	128	128	255	255		255	255	255	128	128	128	255	255
255	128	128	128	128	255	128	255		255	128	128	128	128	255	128	255
255	255	128	128	128	255	255	128		255	255	128	128	128	255	255	128
255	255	128	128	255	255	255	128		255	255	128	128	255	255	255	128
255	128	128	128	128	128	128	128		255	128	128	128	128	128	128	128
255	128	128	255	255	255	128	128		255	128	128	255	255	255	128	128

Sample 8x8 for the Green Matrix (G)								→	Dg = R - G							
255	255	255	128	255	255	128	255		0	0	0	0	0	0	0	0
255	255	255	255	128	128	128	128		0	0	0	0	0	0	0	0
255	255	255	128	128	128	255	255		0	0	0	0	0	0	0	0
255	128	128	128	128	255	128	255		0	0	0	0	0	0	0	0
255	255	128	128	128	255	255	128		0	0	0	0	0	0	0	0
255	255	128	128	255	255	255	128		0	0	0	0	0	0	0	0
255	128	128	128	128	128	128	128		0	0	0	0	0	0	0	0
255	128	128	255	255	255	128	128		0	0	0	0	0	0	0	0

Sample 8x8 for the Blue Matrix (B)								Db = B - G							
255	255	255	0	255	255	0	255	0	0	0	-128	0	0	-128	0
255	255	255	255	0	0	0	0	0	0	0	0	-128	-128	-128	-128
255	255	255	0	0	0	255	255	0	0	0	-128	-128	-128	0	0
255	0	0	0	0	255	0	255	0	-128	-128	-128	-128	0	-128	0
255	255	0	0	0	255	255	0	0	0	-128	-128	-128	0	0	-128
255	255	0	0	255	255	255	0	0	0	-128	-128	0	0	0	-128
255	0	0	0	0	0	0	0	0	0	-128	-128	-128	-128	-128	-128
255	0	0	255	255	255	0	0	0	-128	-128	0	0	0	-128	-128

Figure 5. 2 - Transformation Example

Figure 5.3 describe the invers transformation for the three colours space by using the TI(R,Dg,Db) matrix from the transformed example for the three matrices.

R Matrix								R Matrix							
255	255	255	128	255	255	128	255	255	255	255	128	255	255	128	255
255	255	255	255	128	128	128	128	255	255	255	128	128	128	255	255
255	255	255	128	128	128	255	255	255	255	128	128	128	255	255	255
255	128	128	128	128	255	255	255	128	128	128	255	255	255	128	128
255	255	128	128	255	255	255	128	255	255	128	128	128	255	255	128
255	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128
255	128	128	255	255	255	128	128	255	255	255	128	128	128	128	128

Dg Matrix								G = R - Dg							
0	0	0	0	0	0	0	0	255	255	255	128	255	255	128	255
0	0	0	0	0	0	0	0	255	255	255	255	128	128	128	128
0	0	0	0	0	0	0	0	255	255	255	128	128	128	255	255
0	0	0	0	0	0	0	0	255	128	128	128	128	255	128	255
0	0	0	0	0	0	0	0	255	255	128	128	128	255	255	128
0	0	0	0	0	0	0	0	255	255	128	128	255	255	255	128
0	0	0	0	0	0	0	0	255	128	128	128	128	128	128	128
0	0	0	0	0	0	0	0	255	128	128	255	255	255	128	128

Db Matrix								B = G + Db							
0	0	0	-128	0	0	-128	0	255	255	255	0	255	255	0	255
0	0	0	0	-128	-128	-128	-128	255	255	255	255	0	0	0	0
0	0	0	-128	-128	-128	0	0	255	255	255	0	0	0	255	255
0	-128	-128	-128	-128	0	-128	0	255	0	0	0	0	255	0	255
0	0	-128	-128	-128	0	0	-128	255	255	0	0	0	255	255	0
0	0	-128	-128	0	0	0	-128	255	255	0	0	255	255	255	0
0	-128	-128	-128	-128	-128	-128	-128	255	0	0	0	0	0	0	0
0	-128	-128	0	0	0	-128	-128	255	0	0	255	255	255	0	0

Figure 5. 3 - Invers Transformation Example



Figure 5.4 describe the CSC results for the three transformed matrix TI(R,Dg,Db).

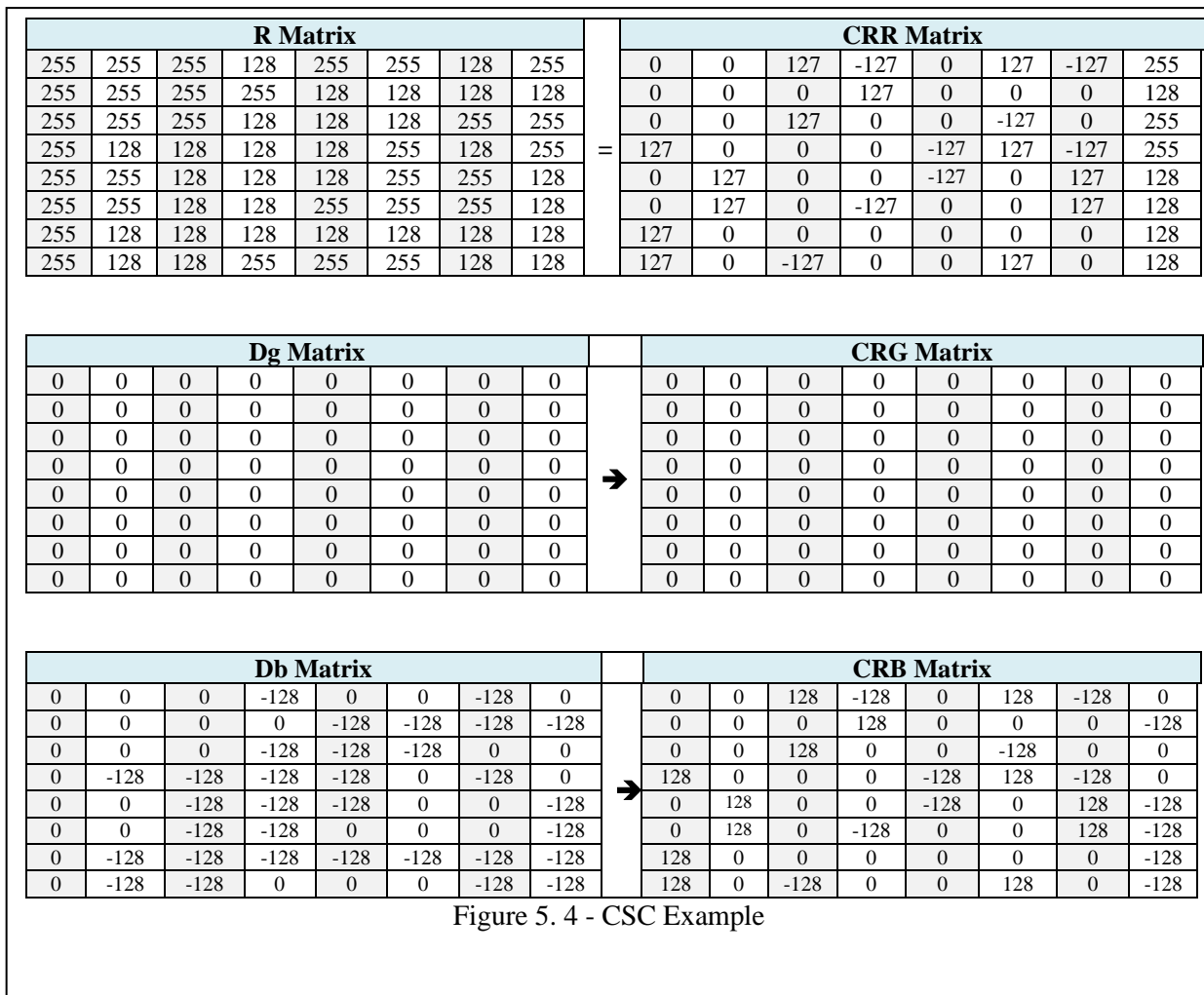


Figure 5.5 describe the CSC decompression phase for the three compressed matrices CI(CRR,CRG,CRB) to restore the TI(R,Dg,Db).

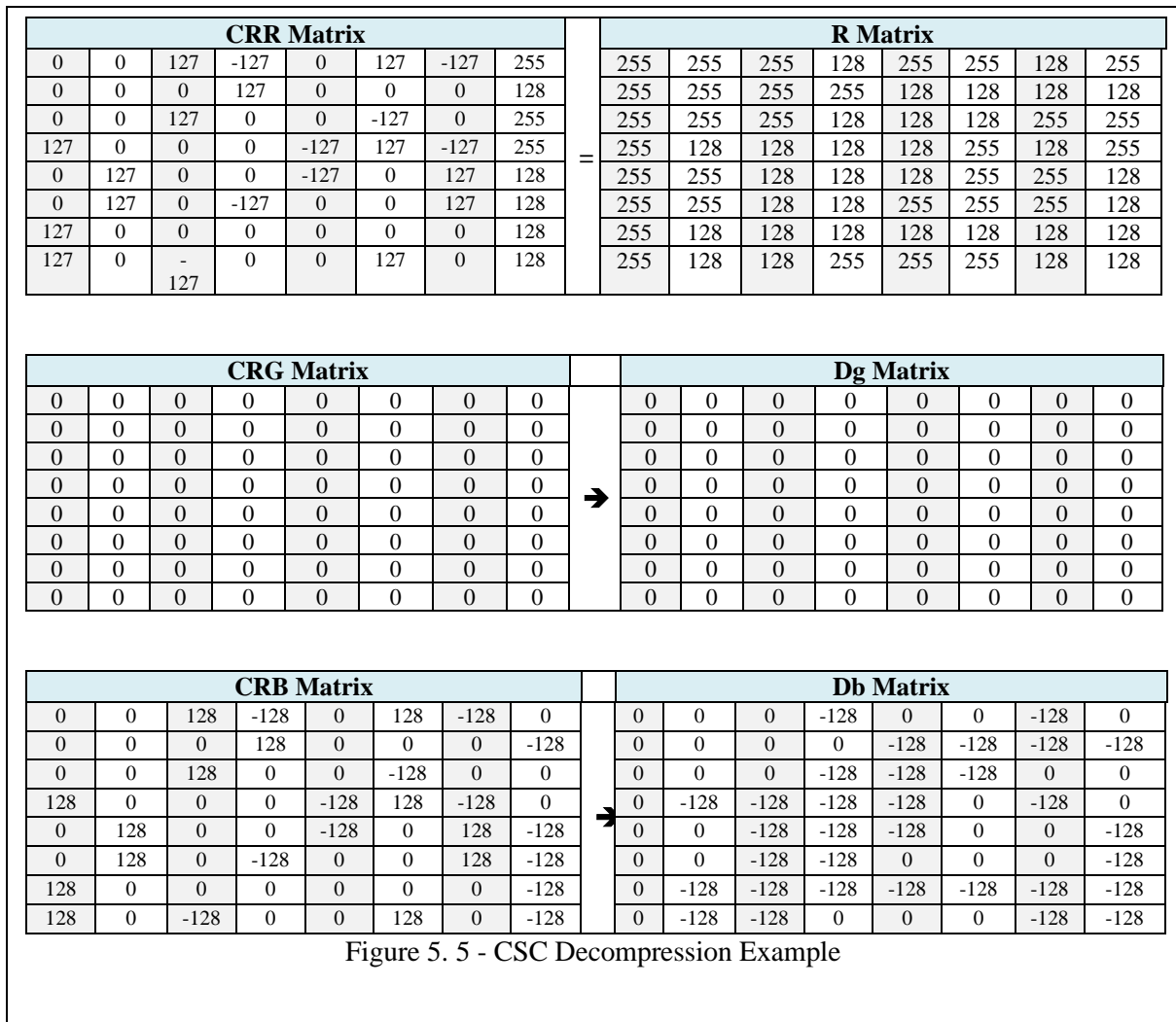


Figure 5. 5 - CSC Decompression Example

### 5.3.2 Negative Value Removing

This phase is responsible for converting the resulted matrix from the CSC algorithm phase CI(CRR,CRG,CRB) into positive integer values matrices by using a temporary dictionary file to avoid any distortion during the recovery phase. As displayed in Figure 5.6 the dictionary file is a matrix of the same sample example dimension 8x8 and represent the positive values with zero's and the negative values with one's.

CRR Temporary Dictionary File							
0	0	0	1	0	0	1	0
0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	1	1	1	0
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0

CRG Temporary Dictionary File							
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

CRB Temporary Dictionary File							
0	0	0	1	0	0	1	0
0	0	0	0	0	0	0	1
0	0	0	0	0	1	0	0
0	0	0	0	1	0	1	0
0	0	0	0	1	0	0	1
0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	1

Figure 5. 6 - Negative Values Temporary Dictionary File

Each of the negative value in (CRR, CRG and CRB) should be multiplied by (-1) to produce three positive values matrices (Image R1, Image G1 and Image B1) as displayed in Figure 5.7.

CRR								Image R1							
0	0	127	-127	0	127	-127	255	0	0	127	127	0	127	127	255
0	0	0	127	0	0	0	128	0	0	0	127	0	0	0	128
0	0	127	0	0	0	-127	255	0	0	127	0	0	127	0	255
127	0	0	0	-127	127	-127	255	127	0	0	0	127	127	127	255
0	127	0	0	-127	0	127	128	0	127	0	0	127	0	127	128
0	127	0	-127	0	0	127	128	0	127	0	127	0	0	127	128
127	0	0	0	0	0	0	128	127	0	0	0	0	0	0	128
127	0	-127	0	0	127	0	128	127	0	127	0	0	127	0	128

CRG								Image G1							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CRB								Image B1							
0	0	128	-128	0	128	-128	0	0	0	128	128	0	128	128	0
0	0	0	128	0	0	0	-128	0	128	0	0	0	0	0	128
0	0	128	0	0	0	-128	0	128	0	0	128	0	0	0	0
128	0	0	0	-128	128	-128	0	128	0	0	128	128	128	0	0
0	128	0	0	-128	0	128	-128	0	128	0	0	128	128	128	0
0	128	0	-128	0	0	128	-128	0	128	0	0	128	128	128	0
128	0	0	0	0	0	0	-128	128	0	0	0	0	0	128	128
128	0	-128	0	0	128	0	-128	128	0	128	0	0	128	0	128

Figure 5. 7 - Positive Values Matrices

The new three matrices are represented with a smaller number of unique values after eliminating the negative values and they are ready to be compressed by the Huffman algorithm.

### 5.3.3 Huffman Algorithm

This phase is responsible for decreasing the intensities values that represents the image by applying Huffman algorithm on the positive matrixes. Huffman algorithm starts by measuring the frequency of occurrence for each symbol (giving for each intensity its weight), and then gives prefix codes to those symbols according to their probabilities (creates a frequency Table of the symbols). Shorter codes will be assigned to the more frequently occurring symbols while larger codes will be assigned to the less frequently occurring symbol c. Figure 5.8 displays the Huffman dictionary files for the three matrices.

Image R1								→	Image R1 Dictionary File		
0	0	127	127	0	127	127	255		Unique Values	frequency	weight
0	0	0	127	0	0	0	128		0	35	0
0	0	127	0	0	127	0	255		127	21	1
127	0	0	0	127	127	127	255		128	5	2
0	127	0	0	127	0	127	128	255	3	3	
0	127	0	127	0	0	127	128				
127	0	0	0	0	0	0	128				
127	0	127	0	0	127	0	128				

Image G1								→	Image G1 Dictionary File		
0	0	0	0	0	0	0	0		Unique Values	frequency	weight
0	0	0	0	0	0	0	0		0	64	0
0	0	0	0	0	0	0	0				
0	0	0	0	0	0	0	0				
0	0	0	0	0	0	0	0				
0	0	0	0	0	0	0	0				
0	0	0	0	0	0	0	0				
0	0	0	0	0	0	0	0				

Image B1								→	Image B1 Dictionary File		
0	0	128	128	0	128	128	0		Unique Values	frequency	weight
0	0	0	128	0	0	0	128		0	38	0
0	0	128	0	0	128	0	0		128	26	1
128	0	0	0	128	128	128	0				
0	128	0	0	128	0	128	128				
0	128	0	128	0	0	128	128				
128	0	0	0	0	0	0	128				
128	0	128	0	0	128	0	128				

Figure 5. 8 - Huffman Dictionary Files

The Huffman dictionary files are used to replace each of the three matrices values with their related weight to produce a new smaller size matrix (Image R2, Image G2 and Image B2) as displayed in Figure 5.9.

Image R1								→	Image R2							
0	0	127	127	0	127	127	255		0	0	1	1	0	1	1	3
0	0	0	127	0	0	0	128		0	0	0	1	0	0	0	2
0	0	127	0	0	127	0	255		0	0	1	0	0	1	0	3
127	0	0	0	127	127	127	255		1	0	0	0	1	1	1	3
0	127	0	0	127	0	127	128		0	1	0	0	1	0	1	2
0	127	0	127	0	0	127	128		0	1	0	1	0	0	1	2
127	0	0	0	0	0	0	128		1	0	0	0	0	0	0	2
127	0	127	0	0	127	0	128		1	0	1	0	0	1	0	2

Image G1								→	Image G2							
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

Image B1								→	Image B2							
0	0	128	128	0	128	128	0		0	0	1	1	0	1	1	0
0	0	0	128	0	0	0	128		0	0	0	1	0	0	0	1
0	0	128	0	0	128	0	0		0	0	1	0	0	1	0	0
128	0	0	0	128	128	128	0		1	0	0	0	1	1	1	0
0	128	0	0	128	0	128	128		0	1	0	0	1	0	1	1
0	128	0	128	0	0	128	128		0	1	0	1	0	0	1	1
128	0	0	0	0	0	0	128		1	0	0	0	0	0	0	1
128	0	128	0	0	128	0	128		1	0	1	0	0	1	0	1

Figure 5. 9 - Huffman Results

### 5.3.4 Negative Value Restoration

After decreasing the number of unique values by using Huffman algorithm from the previous phase, the algorithm should restore the negative values by using the temporary dictionary files, to avoid any distortion. Figure 5.10 displays the resulted matrices (Image R3, Image G3 and Image B3).

Image R2								→	Image R3							
0	0	1	1	0	1	1	3		0	0	1	-1	0	1	-1	3
0	0	0	1	0	0	0	2		0	0	0	1	0	0	0	2
0	0	1	0	0	1	0	3		0	0	1	0	0	-1	0	3
1	0	0	0	1	1	1	3		1	0	0	0	-1	1	-1	3
0	1	0	0	1	0	1	2		0	1	0	0	-1	0	1	2
0	1	0	1	0	0	1	2		0	1	0	-1	0	0	1	2
1	0	0	0	0	0	0	2		1	0	0	0	0	0	0	2
1	0	1	0	0	1	0	2		1	0	-1	0	0	1	0	2

Image G2								→	Image G3							
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	

Image B2								→	Image B3							
0	0	1	1	0	1	1	0		0	0	1	-1	0	-1	0	
0	0	0	1	0	0	0	1		0	0	0	0	0	-1	-1	
0	0	1	0	0	1	0	0		0	0	-1	0	0	0	0	
1	0	0	0	1	1	1	0		1	0	0	-1	1	-1	0	
0	1	0	0	1	0	1	1		0	1	0	-1	0	1	-1	
0	1	0	1	0	0	1	1		0	1	0	-1	0	1	-1	
1	0	0	0	0	0	0	1		0	0	0	0	0	0	-1	
1	0	1	0	0	1	0	1		0	0	-1	0	0	1	0	
1	0	1	0	0	1	0	1		0	0	-1	0	0	1	0	

Figure 5. 10 - Restoring the Negative Values

For Huffman decompression, we need to create a temporary dictionary file for the negative values for the three matrices (Image R3, Image GR3 and Image B3) to convert the three matrices values into positive values. Figure 5.11 display the temporary dictionary files for the three matrices.

CRR Temporary Dictionary File							
0	0	0	1	0	0	1	0
0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	1	1	1	0
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0

CRG Temporary Dictionary File							
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

CRB Temporary Dictionary File							
0	0	0	1	0	0	1	0
0	0	0	0	0	0	0	1
0	0	0	0	0	1	0	0
0	0	0	0	1	0	1	0
0	0	0	0	1	0	0	1
0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	1

Figure 5. 11 - Negative Values Temporary Dictionary File

After creating the dictionary files, we can convert each of the negative value in (Image R3, Image GR3 and Image B3) by multiplying by (-1) to produce three positive values matrices (Image R1\_P, Image G1\_P and Image B1\_P) as displayed in Figure 5.12.

Image R3								→	Image R3_P							
0	0	1	-1	0	1	-1	3		0	0	1	1	0	1	1	3
0	0	0	1	0	0	0	2		0	0	0	1	0	0	0	2
0	0	1	0	0	-1	0	3		0	0	1	0	0	1	0	3
1	0	0	0	-1	1	-1	3		1	0	0	0	1	1	1	3
0	1	0	0	-1	0	1	2		0	1	0	0	1	0	1	2
0	1	0	-1	0	0	1	2		0	1	0	1	0	0	1	2
1	0	0	0	0	0	0	2		1	0	0	0	0	0	0	2
1	0	-1	0	0	1	0	2		1	0	1	0	0	1	0	2

Image G3								→	Image G3_P							
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

Image B3								→	Image B3_P							
0	0	1	-1	0	1	-1	0		0	0	1	1	0	1	1	0
0	0	0	1	0	0	0	-1		0	0	0	0	1	0	0	1
0	0	1	0	0	-1	0	0		0	0	1	0	0	1	0	0
1	0	0	0	-1	1	-1	0		1	0	0	1	1	1	1	0
0	1	0	0	-1	0	1	-1		0	1	0	1	0	1	1	1
0	1	0	-1	0	0	1	-1		0	1	0	0	1	1	1	1
1	0	0	0	0	0	0	-1		1	0	0	0	0	0	1	1
1	0	-1	0	0	1	0	-1		1	0	1	0	1	0	1	1

Figure 5. 12 - Positive Values matrices

The three matrices are ready to be decoded by Huffman decompression phase, by using the Huffman dictionary files we can replace each weight by its original value for the three matrices as displayed in Figure 5.13.

Image R3_P								→	Image R3_PHD							
0	0	1	1	0	1	1	3		0	0	127	127	0	127	127	255
0	0	0	1	0	0	0	2		0	0	0	127	0	0	0	128
0	0	1	0	0	1	0	3		0	0	127	0	0	127	0	255
1	0	0	0	1	1	1	3		127	0	0	0	127	127	127	255
0	1	0	0	1	0	1	2		0	1	0	0	127	0	127	128
0	1	0	1	0	0	1	2		0	1	0	127	0	0	127	128
1	0	0	0	0	0	0	2		127	0	0	0	0	0	0	128
1	0	1	0	0	1	0	2		127	0	127	0	0	127	0	128

Image G3_P									Image G3_PHD									
0	0	0	0	0	0	0	0	→	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0

Image B3_P									Image B3_PHD									
0	0	1	1	0	1	1	0	→	0	0	128	128	0	128	128	0	0	
0	0	0	1	0	0	0	1		0	0	0	128	0	0	0	0	128	0
0	0	1	0	0	1	0	0		128	0	0	0	0	128	128	128	0	0
1	0	0	0	1	1	1	0		0	128	0	0	128	0	128	0	128	128
0	1	0	0	1	0	0	1		0	128	128	0	0	128	0	128	128	128
0	1	0	1	0	0	0	1		128	0	0	128	0	0	0	128	128	128
1	0	0	0	0	0	0	1		0	0	0	0	0	0	0	0	128	128
1	0	1	0	0	1	0	1		128	0	128	0	0	0	128	0	0	128
1	0	1	0	0	1	0	1		128	0	128	0	0	0	128	0	0	128

Figure 5. 13 - Huffman Decompression Results

The final step of Huffman decompression phase is to restore the negative values by using the temporary dictionary file for each matrix as displayed in Figure 5.14.

Image R3_PHD									CRR Matrix									
0	0	127	127	0	127	127	255	→	0	0	127	-127	0	127	-127	255	0	
0	0	0	127	0	0	0	128		0	0	0	127	0	0	0	0	128	0
0	0	127	0	0	127	0	255		0	0	127	0	0	-127	0	255	0	255
127	0	0	0	127	127	127	255		127	0	0	0	-127	127	-127	255	255	255
0	1	0	0	127	0	127	128		0	1	0	0	-127	0	127	128	128	128
0	1	0	127	0	0	127	128		0	1	0	-127	0	0	127	128	128	128
127	0	0	0	0	0	0	128		127	0	0	0	0	0	0	0	128	128
127	0	127	0	0	127	0	128		127	0	-127	0	0	127	0	128	128	128
127	0	127	0	0	127	0	128		127	0	-127	0	0	127	0	128	128	128

Image G3_PHD									CRG Matrix									
0	0	0	0	0	0	0	0	→	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0

Image B3_PHD									CRB Matrix									
0	0	128	128	0	128	128	0	→	0	0	128	-128	0	128	-128	0	0	
0	0	0	128	0	0	0	128		0	0	0	128	0	0	0	0	-128	0
0	0	128	0	0	128	0	0		0	0	128	0	0	-128	0	0	0	0
128	0	0	0	128	128	128	0		128	0	0	0	-128	128	-128	0	0	0
0	128	0	0	128	0	128	128		0	128	128	0	0	-128	0	128	-128	0
0	128	0	128	0	0	128	128		0	128	128	0	-128	0	0	128	-128	0
128	0	0	0	0	0	0	128		128	0	0	0	0	0	0	0	-128	0
128	0	128	0	0	128	0	128		128	0	-128	0	0	128	0	128	-128	0
128	0	128	0	0	128	0	128		128	0	-128	0	0	128	0	128	-128	0

Figure 5. 14 - Huffman Positive Values matrices



### 5.3.5 RLE

Since the resulted matrices from the previous phase are represented by less unique values, the RLE compression algorithm is applied to decrease the image size dramatically by scanning the image to find the runs (pixels with the same value); the runs should be encoded by their probabilities and values (value; probability) this value with its probability is called a unit. The best RLE results come with the images that have large areas of contiguous colour (where the value is repeated often) (Husseen, Mahmud and Mohammed, 2017). Figure 5.15 display the RLE results for the three matrices (Image R4, Image G4 and Image B4).

Image R3							
0	0	1	-1	0	1	-1	3
0	0	0	1	0	0	0	2
0	0	1	0	0	-1	0	3
1	0	0	0	-1	1	-1	3
0	1	0	0	-1	0	1	2
0	1	0	-1	0	0	1	2
1	0	0	0	0	0	0	2
1	0	-1	0	0	1	0	2

Image R4																																
3	1	2	2	4	2	2	1	1	1	4	2	1	3	1	5	2	3	1	1	1	1	3	1	1	2	1	2	2	1	1	2	4
0	1	0	1	0	1	0	1	0	1	0	-1	1	0	-1	0	-1	0	1	0	-1	1	0	1	-1	0	-1	1	0	3	2	3	2

Image G3							
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Image G4
64
0

Image B4																															
3	1	2	2	4	2	2	1	1	1	4	2	1	3	1	5	2	3	1	1	1	1	3	1	1	2	1	2	3	1	2	4
0	1	0	1	0	1	0	1	0	1	0	-1	1	0	-1	0	-1	0	1	0	-1	1	0	1	-1	0	-1	1	0	-1	0	-1

Figure 5. 15 - RLE Results

## 5.4 LRCSC Time Complexity

A simplified time-complexity analysis for the LRCSC algorithm is elaborated based on Big-O notation.

### 5.4.1 LRCSC algorithm Time Complexity Analyses

The LRCSC algorithm starts by loading the input (R,G,B) image to identifies the matrix size  $n$ . Then the transformation is used to map the (R,G,B) image into new colour space (R,Dg,Db) by using the transformation equations as a second phase. followed by the Huffman encoding as the third compression phase and the RLE compression algorithm as a final phase.

Table 5.1 lists the time complexity for each of the LRCSC algorithm phases by using the O notation.

Table 5.1 - LRCSC Algorithm Complexity

Line	Description	O
1	<b>Procedure Transformation</b>	-
2	input RMI: Raster Map Image RMI(R,G,B)	$O(n^2)$
3	M = total rows number; N = total columns number;	$O(1)$
5	$Dg(\text{row}, \text{column}) = R(\text{row}, \text{column}) - G(\text{row}, \text{column})$	$O(n^2)$
6	$Db(\text{row}, \text{column}) = B(\text{row}, \text{column}) - G(\text{row}, \text{column})$	$O(n^2)$
1	<b>Procedure Compression</b>	-
2	input TI: transformed image TI(R,Dg,Db)	$O(n^2)$
7	$CRR(\text{row}, \text{column}) = R(\text{row}, \text{column}) - R(\text{row}, \text{column}+1);$	$O(n^2)$
13	$CRG(\text{row}, \text{column}) = Dg(\text{row}, \text{column}) - Dg(\text{row}, \text{column}+1);$	$O(n^2)$
19	$CRB(\text{row}, \text{column}) = Db(\text{row}, \text{column}) - Db(\text{row}, \text{column} + 1);$	$O(n^2)$
1	<b>Procedure Negative Huffman</b>	-
2	input CI: compressed image CI(CRR, CRG, CRB)	$O(n^2)$
5	$\text{Image\_R1}(\text{row}, \text{column}) = CRR(\text{row}, \text{column}) * -1$	$O(n^2)$
6	Create the Dictionary_R	$O(n^2)$
11	$\text{Image\_G1}(\text{row}, \text{column}) = CRG(\text{row}, \text{column}) * -1$	$O(n^2)$
12	Create the Dictionary_G	$O(n^2)$
17	$\text{Image\_B1}(\text{row}, \text{column}) = CRB(\text{row}, \text{column}) * -1$	$O(n^2)$
18	Create the Dictionary_B	$O(n^2)$
23	$\text{Image\_R2}(\text{row}, \text{column}) = \text{Huffman}(\text{Image\_R1}(\text{row}, \text{column}))$	$O(n \log n)$
24	$\text{Image\_G2}(\text{row}, \text{column}) = \text{Huffman}(\text{Image\_G1}(\text{row}, \text{column}))$	$O(n \log n)$
25	$\text{Image\_B2}(\text{row}, \text{column}) = \text{Huffman}(\text{Image\_B1}(\text{row}, \text{column}))$	$O(n \log n)$

28	$\text{Image\_R3 (row, column)} = \text{Image\_R2 (row, column)} * -1$	$O(n^2)$
31	$\text{Image\_G3 (row, column)} = \text{Image\_G2 (row, column)} * -1$	$O(n^2)$
34	$\text{Image\_B3 (row, column)} = \text{Image\_B2 (row, column)} * -1$	$O(n^2)$
1	<b>Procedure RLE</b>	-
2	input Image_R3, Image_G3, Image_B3	$O(n^2)$
3	$\text{Image\_R4} = \text{RLE}(\text{Image\_R3})$	$O(n)$
4	$\text{Image\_G4} = \text{RLE}(\text{Image\_G3})$	$O(n)$
5	$\text{Image\_B4} = \text{RLE}(\text{Image\_B3})$	$O(n)$
1	<b>Procedure RLE Decompression</b>	-
2	input compressed image matrices Image_R4, Image_G4, Image_B4	$O(n^2)$
3	$\text{Image\_R3} = \text{De\_RLE}(\text{Image\_R4})$	$O(n)$
4	$\text{Image\_G3} = \text{De\_RLE}(\text{Image\_G4})$	$O(n)$
5	$\text{Image\_B3} = \text{De\_RLE}(\text{Image\_B4})$	$O(n)$
1	<b>Procedure Negative Huffman Decoding</b>	-
2	input three matrices (Image_R3, Image_G3 and Image_B3)	$O(n^2)$
5	$\text{Image\_R2 (row, column)} = \text{Image\_R3 (row, column)} * -1$	$O(n^2)$
6	Create the Dictionary_R	$O(n^2)$
11	$\text{Image\_G2 (row, column)} = \text{Image\_G3 (row, column)} * -1$	$O(n^2)$
12	Create the Dictionary_G	$O(n^2)$
17	$\text{Image\_B2 (row, column)} = \text{Image\_B3 (row, column)} * -1$	$O(n^2)$
18	Create the Dictionary_B	$O(n^2)$
23	$\text{Image\_R1 (row, column)} = \text{Huffman\_Decoding}(\text{Image\_R2 (row, column)})$	$O(n \log n)$
24	$\text{Image\_G1 (row, column)} = \text{Huffman\_Decoding}(\text{Image\_G2 (row, column)})$	$O(n \log n)$
25	$\text{Image\_B1 (row, column)} = \text{Huffman\_Decoding}(\text{Image\_B2 (row, column)})$	$O(n \log n)$
28	$\text{CRR (row, column)} = \text{Image\_R2 (row, column)} * -1$	$O(n^2)$
31	$\text{CRG (row, column)} = \text{Image\_G2 (row, column)} * -1$	$O(n^2)$
34	$\text{CRB (row, column)} = \text{Image\_B2 (row, column)} * -1$	$O(n^2)$
1	<b>Procedure LRCSC Decompression</b>	-
2	input compressed image CI(CRR, CRG, CRB)	$O(n^2)$
3	M = total rows number; N = total columns number;	$O(1)$
8	$\text{R (row, column-1)} = \text{CRR}(\text{row, column}) + \text{CRR}(\text{row, column-1})$	$O(n^2)$
14	$\text{Dg (row, column-1)} = \text{CRG}(\text{row, column}) + \text{CRG}(\text{row, column-1})$	$O(n^2)$

20	$Db(\text{row}, \text{column}-1) = CRB(\text{row}, \text{column}) + CRB(\text{row}, \text{column}-1)$	$O(n^2)$
1	<b>Procedure Revers Transformation</b>	-
2	input R,Dg,Db; matrices of natural number	$O(n^2)$
5	$G(\text{row}, \text{column}) = R(\text{row}, \text{column}) - DG(\text{row}, \text{column})$	$O(n^2)$
7	$B(\text{row}, \text{column}) = G(\text{row}, \text{column}) + DB(\text{row}, \text{column})$	$O(n^2)$

The growth rate function in terms of time for the LRCSC components is analysed as:

5- The growth rate function of the transformation component is

$f(n) = 3O(n^2) + O(1)$  Therefore, the overall rate of growth for this component is

$f(n) = O(n^2)$  after removing the constants.

6- The growth rate function of the LRCSC component is

The second phase of the algorithm is to compress the R,Dg,Db image by using the CSC function. This phase is to apply the CSC function for each of the three matrices individually. For each colour space we used nested two loops. The outer loop runs  $n$  times and the inner loop runs  $n$  times for each iteration of the outer loop; this indicate that, this function will be running for  $n^2$  total times, thus the function is running  $O(n^2)$  time for each colure and the complexity of this phase is

$f(n) = 4 O(n^2)$  Therefore, the overall rate of growth for this component is

$f(n) = O(n^2)$  after removing the constants.

7- The growth rate function of the Negative Huffman component is

$f(n) = 10 O(n^2) + 3 O(n \log n)$  Therefore, the overall rate of growth for this component is

$f(n) = O(n^2) + O(n \log n)$  after removing the constants.

8- The growth rate function of the RLE component is

$f(n) = 3O(n) + O(n^2)$  Therefore, the overall rate of growth for this component is

$f(n) = O(n) + O(n^2)$  after removing the constants.

The decompression phase is to reconstruct the image by using four reversable procedures.

1- The growth rate function of the RLE decompression component is

$f(n) = 3O(n) + O(n^2)$  Therefore, the overall rate of growth for this component is

$f(n) = O(n) + O(n^2)$  after removing the constants.

9- The growth rate function of the Negative Huffman decoding component is

$f(n) = 10 O(n^2) + 3 O(n \log n)$  Therefore, the overall rate of growth for this component

is  $f(n) = O(n^2) + O(n \log n)$  after removing the constants.

10- The growth rate of the LRCSC decompression component is

$f(n) = 4On^2 + O(1)$  Therefore, the overall rate of growth for this component is

$f(n) = O(n^2)$  after removing the constants.

11- The growth rate of the invers transformation component is

$f(n) = 3 O(n^2)$  Therefore, the overall rate of growth for this component is

$f(n) = O(n^2)$  after removing the constants.

After approximating the computation complexity for the individual components of the system, the overall complexity is calculated by summing up the overhead for the individual parts.

$$f(n) = [O(n^2)] + [O(n^2)] + [O(n^2) + O(n \log n)] + [O(n) + O(n^2)] + [O(n) + O(n^2)] + [O(n^2) + O(n \log n)] + [O(n^2)] + [O(n^2)].$$

$$f(n) = 2 O(n) + 8 O(n^2) + 2 O(n \log n)$$

Therefore, the overall complexity of the algorithm is:

$$f(n) = O(n) + O(n^2) + O(n \log n) \quad \text{after removing the constants.}$$

#### 5.4.2 Proof

Assume that  $g(n) = n$ , the time complexity of  $f(n)$  is  $O(n)$ . To proof that  $f(n)$  in Equation 3.1 is  $O(g(n))$ , we will apply the limit to find a constant  $c > 0$ .

We have  $f(n) = O(n) + O(n^2) + O(n \log n)$  and  $g(n) = n$ . That is

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)}$$

$$\lim_{n \rightarrow \infty} \frac{n + n^2 + n \log n}{n}$$

$$\lim_{n \rightarrow \infty} (n + \log n)$$

As the proof shows, there is a constant  $c > 0$  that satisfy the limit in the proof theorem. Since  $n_0$  must be positive integer, we can say the  $f(n)$  in Equation 3.1 is  $O(n + \log n)$ , for  $n \geq n_0$ .

## 5.5 Validation and Testing

After implementing the algorithm, the compression performance needed to be evaluated and then compare the proposed algorithm results with other state-of-the-art results regarding using the main three parameters that may affect any compression algorithm, namely: compression size, image quality and execution time. Using our testbed, we are going to evaluate the CSC algorithm based on:

- 1- The compressed image size by calculating the compression ratio.

- 2- The image quality by calculating the (MSE and PSNR).
- 3- The algorithm execution time by using the Tic-Toc matlab function.

### 5.5.1 The LRCSC Algorithm Compression Size

Table 5.2 lists the compression ratio Cr, Compression rate and the algorithm storage saving percentage resulting from applying the LRCSC algorithm on the five image sets. The compression algorithm should provide a compressed image size smaller than the original image.

Table 5.2 - The Lossless LRCSC Algorithm Compression Size

Image Set 1							
Image	Format	Resolution	Original Size in KBs	New Size in KBS	Compression Ratio (CR)	Compression Rate	Storage Saving %
Baboon	GIF	512*512	225	94	2.394	0.418	58.22
Barbara	PNG	512*512	230	154	1.494	0.670	33.04
Boats	PNG	512*512	239	148	1.615	0.619	38.08
Boats	BMP	720*576	368	201	1.831	0.546	45.38
Camera Man	BMP	256*256	60	35	1.714	0.583	41.67
Camera Man	GIF	256*256	56	32	1.750	0.571	42.86
House	PNG	256*256	59	30	1.967	0.508	49.15
Lena	PNG	256*256	59	32	1.844	0.542	45.76
Lena	JPG	512*512	226	112	2.018	0.496	50.44
Image Set 2							
Image	Format	Resolution	Original Size in KBs	New Size in KBS	Compression Ratio (CR)	Compression Rate	Storage Saving %
Lena	PNG	330*330	290	159	1.824	0.548	45.17
Lena	BMP	220*220	129	49	2.633	0.380	62.02
Lena	JPG	225*225	135	51	2.647	0.378	62.22
Airplane	BMP	512*512	751	354	2.121	0.471	52.86
Baboon	BMP	500*480	599	451	1.328	0.753	24.71
Barbara	BMP	720*576	1064	474	2.245	0.445	55.45
Boats	BMP	787*576	1148	543	2.114	0.473	52.70
Goldhill	BMP	720*576	1031	452	2.281	0.438	56.16
Pepper	BMP	512*512	651	451	1.443	0.693	30.72
Image Set 3							
Image	Format	Resolution	Original Size in KBs	New Size in KBS	Compression Ratio (CR)	Compression Rate	Storage Saving %
Medic	JPG	168*90	32	10	3.200	0.313	68.75
Medic1	JPG	160*90	24	9	2.667	0.375	62.50
Butterfly	JPG	128*85	20	16	1.250	0.800	20.00
Mountain	JPG	128*96	28	14	2.000	0.500	50.00
swarm	JPG	128*85	25	14	1.786	0.560	44.00
Lake_jpg	JPG	128*85	25	15	1.667	0.600	40.00
Saturn_jpg	JPG	128*100	19	11	1.727	0.579	42.11
Earth_jpg	JPG	225*225	78	46	1.696	0.590	41.03
boat_jpg	JPG	128*85	25	13	1.923	0.520	48.00

<b>Waterfall</b>	JPG	128*96	26	16	1.625	0.615	38.46
<b>Eagle</b>	JPG	128*96	17	8	2.125	0.471	52.94
<b>Grand_Sone</b>	JPG	128*96	28	15	1.867	0.536	46.43
<b>Car</b>	JPG	128*85	28	11	2.545	0.393	60.71
<b>Shape</b>	JPG	128*95	31	19	1.632	0.613	38.71
<b>Image Set 4</b>							
<b>Image</b>	<b>Format</b>	<b>Resolution</b>	<b>Original Size in KBs</b>	<b>New Size in KBS</b>	<b>Compression Ratio (CR)</b>	<b>Compression Rate</b>	<b>Storage Saving %</b>
<b>Knob &amp; Bolt</b>	PNG	768*512	1155	455	2.536	0.394	60.606
<b>Houses</b>	PNG	768*512	1149	535	2.149	0.466	53.438
<b>Landscape</b>	PNG	768*512	1153	548	2.106	0.475	52.472
<b>Light House</b>	PNG	768*512	1150	469	2.450	0.408	59.217
<b>Barn</b>	PNG	768*512	1149	502	2.290	0.437	56.310
<b>Parrots</b>	PNG	768*512	1153	438	2.632	0.380	62.012
<b>Flowers &amp; Sill</b>	PNG	768*512	1154	418	2.763	0.362	63.778
<b>Six-Shooter</b>	PNG	768*512	1152	337	3.416	0.293	70.747
<b>Motocross</b>	PNG	768*512	1150	519	2.216	0.451	54.870
<b>Zentime</b>	PNG	768*512	1155	440	2.627	0.381	61.905
<b>Image Set 5</b>							
<b>Image</b>	<b>Format</b>	<b>Resolution</b>	<b>Original Size in KBs</b>	<b>New Size in KBS</b>	<b>Compression Ratio (CR)</b>	<b>Compression Rate</b>	<b>Storage Saving %</b>
<b>Map 1</b>	BMP	600*480	761	35	21.47	0.05	95.40
<b>Map 2</b>	BMP	600*480	821	30	27.53	0.04	96.35
<b>Map 3</b>	BMP	600*480	801	28	28.37	0.03	96.50
<b>Map 4</b>	BMP	600*480	838	11	78.39	0.01	98.69
<b>Map 5</b>	BMP	600*480	800	40	19.91	0.05	95.00
<b>Map 6</b>	BMP	600*480	826	43	19.07	0.05	94.79
<b>Map 7</b>	BMP	600*480	838	28	29.85	0.03	96.66
<b>Map 8</b>	BMP	600*480	814	40	20.59	0.05	95.09
<b>Map 9</b>	BMP	600*480	769	49	15.77	0.06	93.63
<b>Map 10</b>	BMP	600*480	768	36	21.46	0.05	95.31

As listed in Table 5.3, the LRCSC algorithm dramatically decreases the image size for all the image sets. In image sets 1 and 2, the average compression ratios were 1.847 and 2.071 respectively, which indicates that the algorithm saved in terms of storage 44.95% from the first image set and 49.1% from the original size of the second image set. By observing the third image set results, the algorithm achieved the lowest compression ratio of 1.979, since this image set is already compressed by the JPEG algorithm; however, the algorithm decreases the JPEG images from the third image set as well, by saving 46.68% from the original image's size. The Cr resulted from the fourth image set is 2.52 and saved 59.54% from the original image size. The best Cr came with compressing the fifth image set by having 28.245 Cr and saves 95.74% from the original image set size, since this algorithm as designed for low resolution images such as raster map images.

Table 5.3 - The Lossless LRCSC Algorithm Average Compression Size

Image Sets	Average Compression Ratio CR	Average Compression Rate	Average Storage Saving %
Image Set 1	1.847	0.550	44.956
Image Set 2	2.071	0.509	49.113
Image Set 3	1.979	0.533	46.688
Image Set 4	2.52	0.40	59.54
Image Set 5	28.245	0.043	95.744
<b>Average</b>	<b>7.33</b>	<b>0.4</b>	<b>59.28</b>

Figure 5.16 shows the compression storage saving for the proposed lossless algorithm LRCSC. By observing the results for each of the image set, we conclude that the LRCSC algorithm decreases the image size by different percentages for different images types and different images resolution. The best results came with the fifth image set, since the LRCSC algorithm is designed for low resolution images.

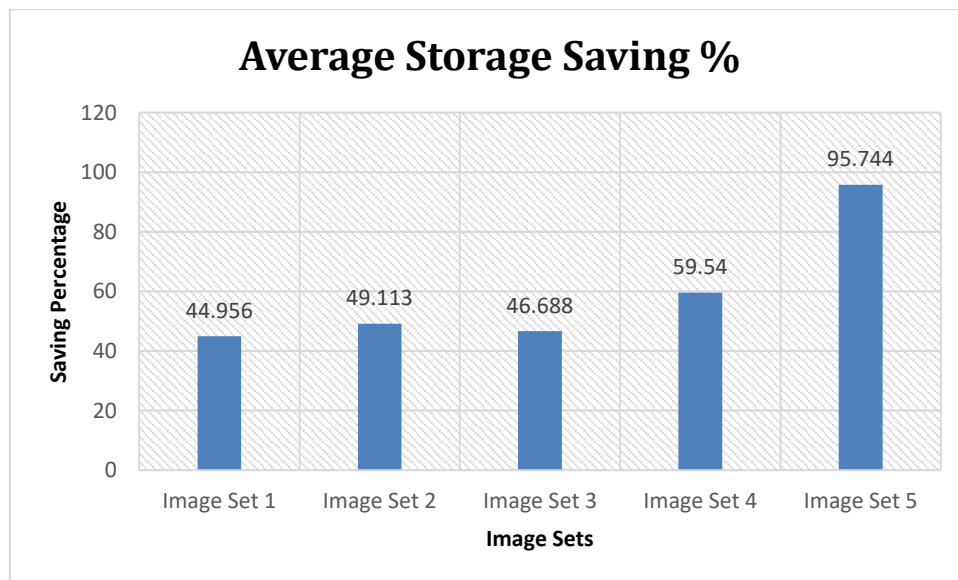


Figure 5. 16 - The LRCSC Algorithm Storage Saving



### 5.5.2 The LRCSC Algorithm Image Quality.

For distortion assessment, we used the following two metrics:

- i. Mean Squared Error (MSE).
- ii. Peak Signal to Noise Ratio (PSNR).

Table 5.4 lists the average MSE and PSNR for the five image sets after applying the LRCSC algorithm.

Table 5.4 - Image Quality Average Results for the Lossless LRCSC Algorithm

Image Sets	Average (MSE)	Average (PSNR)
Image Set 1	0	Inf
Image Set 2	0	Inf
Image Set 3	0	Inf
Image Set 4	0	Inf
Image Set 5	0	Inf
<b>Average</b>	<b>0</b>	<b>---</b>

After averaging the distortion value for the five image sets, the results prove that the proposed image compression algorithm is a lossless algorithm. Figure 5.17 shows that the MSE value is zero for all the decompressed images for the five image sets.

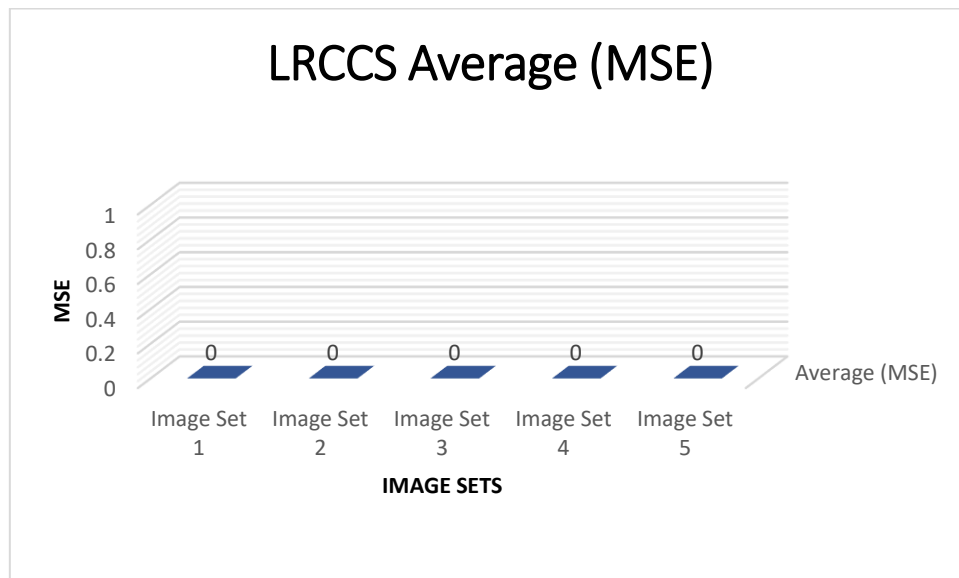





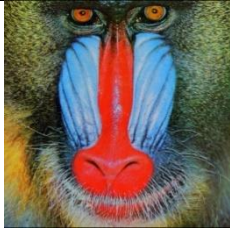
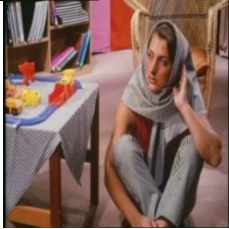





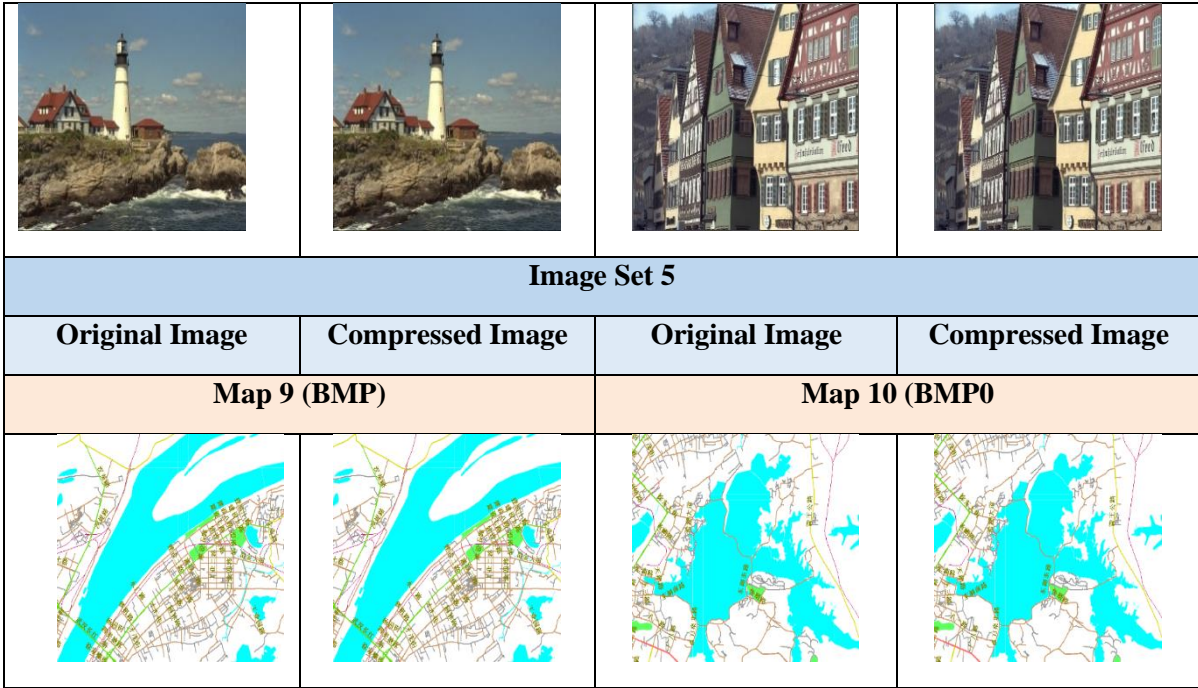


Figure 5. 17 - The Average MSE Results for the Lossless LRCSC Algorithm

All the images from the five image sets have been perfectly restored after decompression with zero distortion, since the MSE value is zero for all the test images. Table 5.5 displayed two image samples from the five image sets.

Table 5.5 - Sample Images for the Five Image Sets Before and After Compression by LRCSC

Image Set 1			
Original Image	Compressed Image	Original Image	Compressed Image
Baboon (GIF)		Boats (PNG)	
			
Image Set 2			
Original Image	Compressed Image	Original Image	Compressed Image
Baboon (BMP)		Barbara (BMP)	
			
Image Set 3			
Original Image	Compressed Image	Original Image	Compressed Image
Medic (JPEG)		Lake (JPEG)	
			
Image Set 4			
Original Image	Compressed Image	Original Image	Compressed Image
Light House (PING)		Houses (PING)	



**5.5.3 The LRCSC Algorithm Execution Time.**

Image compression algorithm execution time can be calculated by measuring the compression speed and the decompression speed in seconds. Compression speed is the time needed for compressing the image while the decompression speed is the time needed for decompressing the image. Table 5.6 displays the needed execution time for each image from the five sets in seconds.

Table 5.6 - The LRCSC Algorithm Compression Time in Seconds

Image Set 1			
Image	Compression Time	Decompression Time	Total Time
Baboon	0.56	0.22	0.78
Barbara	1.1	0.34	1.44
Boats	1	0.35	1.35
Boats	1	0.34	1.34
Camera Man	0.29	0.08	0.37
Camera Man	0.29	0.08	0.37
House	0.24	0.07	0.31
Lena	0.28	0.09	0.37
Lena	0.74	0.34	1.08
Image Set 2			
Image	Compression Time	Decompression Time	Total Time
Lena	0.8	0.27	1.07
Lena	0.93	0.14	1.07

Lena	0.37	0.14	0.51
Airplane	2.2	0.09	2.29
Baboon	2.3	0.8	3.1
Barbara	3.6	1.2	4.8
Boats	2.8	0.9	3.7
Goldhill	2.7	1.1	3.8
Pepper	2.8	0.9	3.7
<b>Image Set 3</b>			
<b>Image</b>	<b>Compression Time</b>	<b>Decompression Time</b>	<b>Total Time</b>
Medic	0.27	0.06	0.33
Medic1	0.20	0.05	0.25
Butterfly	0.27	0.03	0.30
Mountain	0.23	0.04	0.27
swarm	0.15	0.05	0.20
Lake_jpg	0.18	0.04	0.22
Saturn_jpg	0.22	0.04	0.26
Earth_jpg	0.40	0.12	0.52
boat_jpg	0.19	0.03	0.22
Waterfall	0.19	0.04	0.23
Eagle	0.22	0.06	0.28
Grand_Sone	0.20	0.04	0.24
Car	0.17	0.04	0.21
Shape	0.27	0.07	0.34
<b>Image Set 4</b>			
<b>Image</b>	<b>Compression Time</b>	<b>Decompression Time</b>	<b>Total Time</b>
Knob & Bolt	2.1	0.8	2.9
Houses	2.6	1.2	3.8
Landscape	2.7	1	3.7
Light House	2.4	1.1	3.5
Barn	2.6	1.1	3.7
Parrots	2.2	1	3.2
Flowers & Sill	2	0.8	2.8
Six-Shooter	2.3	0.9	3.2
Motocross	2.4	1.1	3.5
Zentime	2.4	0.9	3.3
<b>Image Set 5</b>			
<b>Image</b>	<b>Compression Time</b>	<b>Decompression Time</b>	<b>Total Time</b>
Map 1	0.45	0.27	0.72
Map 2	0.4	0.22	0.62
Map 3	0.66	0.44	1.1
Map 4	0.44	0.22	0.66

<b>Map 5</b>	0.45	0.22	0.67
<b>Map 6</b>	0.37	0.19	0.56
<b>Map 7</b>	0.35	0.18	0.53
<b>Map 8</b>	0.57	0.29	0.86
<b>Map 9</b>	0.48	0.29	0.77
<b>Map 10</b>	0.45	0.25	0.7

One of the main parameters for measuring the compression algorithm performance is the computation time. Table 5.7 lists the total average resulted by calculating the average compression time and the average decompression time for the five image sets.

The LRCSC algorithm needs on average 0.82 seconds to compress and decompress images in the first image set, while the second image set needs 2.67 seconds for both compression and decompression. The third image set has the best computation time with 0.28 seconds, because all the images in the third image set are JPEG images (JPEG format represent images with small intensities values). The fourth image set execution time is 3.36 seconds and the fifth image sets needed 0.71 second.

Table 5.7 - The LRCSC Algorithm Average Compression Time

<b>Image Sets</b>	<b>Average (Compression Time)</b>	<b>Average (De-Compression Time)</b>	<b>Average (Total Time)</b>
<b>Image Set 1</b>	0.61	0.21	0.82
<b>Image Set 2</b>	2.06	0.62	2.67
<b>Image Set 3</b>	0.23	0.05	0.28
<b>Image Set 4</b>	2.37	0.99	3.36
<b>Image Set 5</b>	0.462	0.257	0.71
<b>Average</b>	<b>1.15</b>	<b>0.43</b>	<b>1.57</b>

Figure 5.18 represents the execution time for each image set. For a better understanding of the results, we display the algorithm's results in bar-charts, where each column represents the value for the needed time for the image sets.

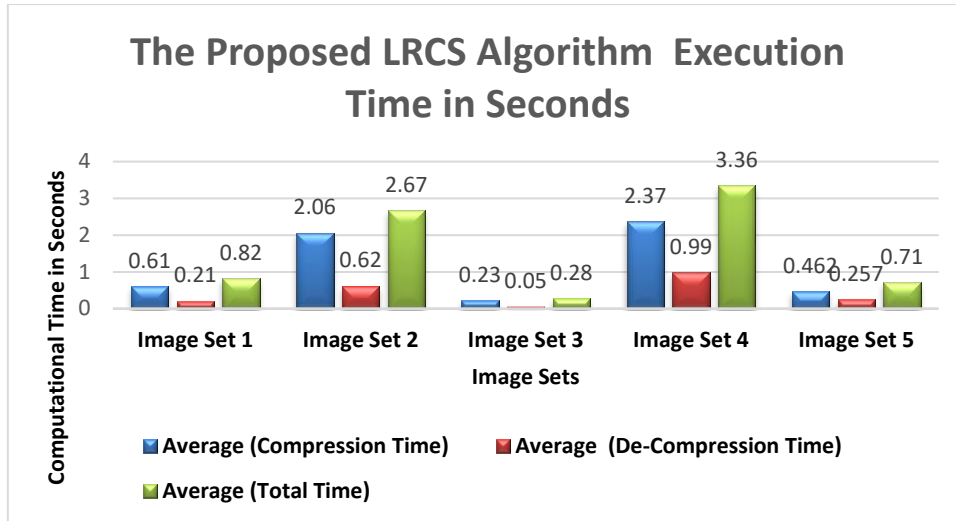


Figure 5. 18 - Compression and Decompression Time for the LRCSC Algorithm

## 5.6 Evaluations, Results and Observations

To describe the algorithm contribution, we investigated the results regarding the compression size, image quality and execution time. To reach the best conclusion from the investigations' results, we need to compare our proposed lossless LRCSC algorithm results with the most common state of the art lossless algorithms and describe the analytical results to reach the best conclusion.

### 5.6.1 Comparison Between the LRCSC Results and Huffman Algorithm Results.

To compare the proposed LRCSC lossless algorithm results with Huffman algorithm results, we need to compare both algorithm results using the three main metrics (image size, image quality and execution time).

#### 5.6.1.1 Comparison Between the LRCSC Algorithm and Huffman Algorithm in Terms of Image Size

Table 5.8 lists the compression rate and space saving for LRCSC and the Huffman algorithms.

Table 5.8 - The Lossless LRCSC Compression Size and Huffman Compression Size

The Proposed Lossless LRCSC Algorithm Results						Huffman Results		
Image Set 1						Image Set 1		
Image	Original Size in KBs	New Size in KBs	Cr	Rate	Storage Saving %	New Size in KBs	Cr	Space Saving %
1	225	94	2.394	0.418	58.22	104	2.163	53.78
2	230	154	1.494	0.670	33.04	204	1.127	11.30
3	239	148	1.615	0.619	38.08	174	1.374	27.20
4	368	201	1.831	0.546	45.38	276	1.333	25.00
5	60	35	1.714	0.583	41.67	40	1.5	33.33
6	56	32	1.750	0.571	42.86	43	1.302	23.21

7	59	30	1.967	0.508	49.15	36	1.639	38.98
8	59	32	1.844	0.542	45.76	49	1.204	16.95
9	226	112	2.018	0.496	50.44	194	1.165	14.16
<b>The Proposed Lossless LRCSC Algorithm Results</b>						<b>Huffman Results</b>		
<b>Image Set 2</b>						<b>Image Set 2</b>		
Image	Original Size in KBs	New Size in KBs	Cr	Rate	Storage Saving %	New Size in KBs	Cr	Storage Saving %
1	290	159	1.824	0.548	45.17	234	1.239	19.31
2	129	49	2.633	0.380	62.02	104	1.24	19.38
3	135	51	2.647	0.378	62.22	110	1.227	18.52
4	751	354	2.121	0.471	52.86	459	1.636	38.88
5	599	451	1.328	0.753	24.71	426	1.406	28.88
6	1064	474	2.245	0.445	55.45	955	1.114	10.24
7	1148	543	2.114	0.473	52.70	898	1.278	21.78
8	1031	452	2.281	0.438	56.16	933	1.105	9.51
9	651	451	1.443	0.693	30.72	526	1.238	19.20
<b>The Proposed Lossless LRCSC Algorithm Results</b>						<b>Huffman Results</b>		
<b>Image Set 3</b>						<b>Image Set 3</b>		
Image	Original Size in KBs	New Size in KBs	Cr	Rate	Storage Saving %	New Size in KBs	Cr	Storage Saving %
1	32	10	3.200	0.313	68.75	32	1	0.00
2	24	9	2.667	0.375	62.50	22	1.091	8.33
3	20	16	1.250	0.800	20.00	19	1.053	5.00
4	28	14	2.000	0.500	50.00	26	1.077	7.14
5	25	14	1.786	0.560	44.00	24	1.042	4.00
6	25	15	1.667	0.600	40.00	24	1.042	4.00
7	19	11	1.727	0.579	42.11	17	1.118	10.53
8	78	46	1.696	0.590	41.03	75	1.04	3.85
9	25	13	1.923	0.520	48.00	24	1.042	4.00
10	26	16	1.625	0.615	38.46	25	1.04	3.85
11	17	8	2.125	0.471	52.94	14	1.214	17.65
12	28	15	1.867	0.536	46.43	27	1.037	3.57
13	28	11	2.545	0.393	60.71	21	1.333	25.00
14	31	19	1.632	0.613	38.71	25	1.24	19.35
<b>The Proposed Lossless LRCSC Algorithm Results</b>						<b>Huffman Results</b>		
<b>Image Set 4</b>						<b>Image Set 4</b>		
Image	Original Size in KBs	New Size in KBs	Cr	Rate	Storage Saving %	New Size in KBs	Cr	Space Saving %
1	1155	455	2.536	0.394	60.606	544	2.123	52.90
2	1149	535	2.149	0.466	53.438	878	1.309	23.59
3	1153	548	2.106	0.475	52.472	844	1.366	26.80
4	1150	469	2.450	0.408	59.217	754	1.525	34.43
5	1149	502	2.290	0.437	56.310	809	1.42	29.59
6	1153	438	2.632	0.380	62.012	829	1.391	28.10
7	1154	418	2.763	0.362	63.778	773	1.493	33.02

<b>8</b>	1152	337	3.416	0.293	70.747	632	1.823	45.14
<b>9</b>	1150	519	2.216	0.451	54.870	851	1.351	26.00
<b>10</b>	1155	440	2.627	0.381	61.905	749	1.542	35.15
<b>The Proposed Lossless LRCSC Algorithm Results</b>						<b>Huffman Results</b>		
<b>Image Set 5</b>						<b>Image Set 5</b>		
<b>Image</b>	<b>Original Size in KBs</b>	<b>New Size in KBs</b>	<b>Cr</b>	<b>Rate</b>	<b>Storage Saving %</b>	<b>New Size in KBs</b>	<b>Cr</b>	<b>Space Saving %</b>
<b>1</b>	761	35	21.47	0.05	95.40	119	6.395	84.36
<b>2</b>	821	30	27.53	0.04	96.35	115	7.139	85.99
<b>3</b>	801	28	28.37	0.03	96.50	118	6.788	85.27
<b>4</b>	838	11	78.39	0.01	98.69	110	7.618	86.87
<b>5</b>	800	40	19.91	0.05	95.00	123	6.504	84.63
<b>6</b>	826	43	19.07	0.05	94.79	118	7	85.71
<b>7</b>	838	28	29.85	0.03	96.66	112	7.482	86.63
<b>8</b>	814	40	20.59	0.05	95.09	119	6.84	85.38
<b>9</b>	769	49	15.77	0.06	93.63	121	6.355	84.27
<b>10</b>	768	36	21.46	0.05	95.31	117	6.564	84.77

As listed in Table 5.9, the proposed LRCSC algorithm decreases the image size more than the Huffman algorithm for all of the image sets. The LRCSC algorithm results decrease the image size more than the Huffman algorithm by saving 17.73% for the first image set, 21.33% from the second image set, 38.48% from the third image set, 36.34% from the fourth image sets and 10.34% from the fifth image set. By averaging the five test image results for both algorithms, the LRCSC algorithm saved 24.97% more than the Huffman algorithm.

Table 5.9 - The Average LRCSC Approach Compression Size with Huffman Average Compression Size

<b>The LRCSC Algorithm Average Results</b>			<b>Huffman Average Results</b>	
<b>Image Sets</b>	<b>Average Compression Ratio CR</b>	<b>Average Storage Saving %</b>	<b>Average Compression Ratio CR</b>	<b>Average Storage Saving %</b>
<b>Image Set 1</b>	1.847	44.95	1.425	27.22
<b>Image Set 2</b>	2.072	49.13	1.448	27.8
<b>Image Set 3</b>	1.979	46.688	1.096	8.2
<b>Image Set 4</b>	2.518	59.536	1.325	23.2
<b>Image Set 5</b>	28.041	95.742	6.87	85.4
<b>Average</b>	<b>7.291</b>	<b>59.20</b>	<b>2.440</b>	<b>34.98</b>

Figure 5.19 describes the compression storage saving of the proposed LRCSC algorithm and Huffman algorithm for each of the five image sets. For a better result understanding, we



displayed the algorithm results in bar-charts, where each column represents the image set and the value for the bits-saving percentage. The proposed lossless algorithm has a better compression ratio.

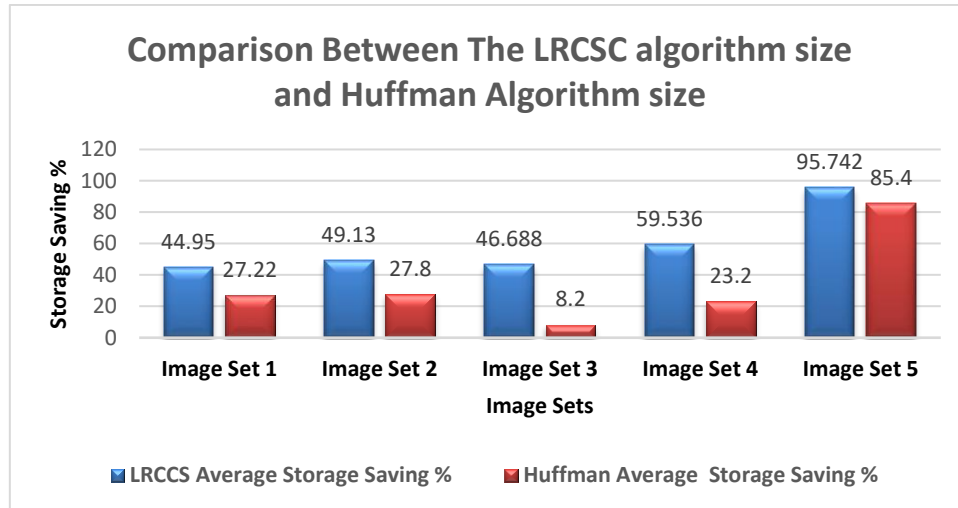


Figure 5.19 - The Average Compression size for the LRCSC and Huffman algorithm

### 5.6.1.2 Comparison Between the LRCSC Algorithm and Huffman in Terms of Image Quality

Since both the proposed LRCSC algorithm and Huffman algorithm are lossless techniques, the two algorithm results should have zero distortion after decompressing the tested image. All the images MSE value are zero.

### 5.6.1.3 Comparison Between the LRCSC and Huffman in Terms of Execution Time

Table 5.10 lists the execution time (compression and decompression time) for the proposed algorithm and the execution time for the Huffman algorithm, for all the five image sets.

Table 5.10 - The Lossless LRCSC Algorithm Execution Time with Huffman Execution Time

LRCSC Algorithm Execution Time				Huffman Algorithm Execution Time
Image Set 1				
Image	Compression Time	Decompression Time	Total Time	Huffman Total Time
1	0.56	0.22	0.78	0.4
2	1.1	0.34	1.44	0.8
3	1	0.35	1.35	0.06
4	1	0.34	1.34	0.4
5	0.29	0.08	0.37	0.4
6	0.29	0.08	0.37	0.4
7	0.24	0.07	0.31	0.4
8	0.28	0.09	0.37	0.6
9	0.74	0.34	1.08	0.6

LRSC Algorithm Execution Time				Huffman Algorithm Execution Time
<b>Image Set 2</b>				
Image	Compression Time	Decompression Time	Total Time	Huffman Total Time
1	0.8	0.27	1.07	0.2
2	0.93	0.14	1.07	0.3
3	0.37	0.14	0.51	0.06
4	2.2	0.09	2.29	0.8
5	2.3	0.8	3.1	0.55
6	3.6	1.2	4.8	1.6
7	2.8	0.9	3.7	1.6
8	2.7	1.1	3.8	1.8
9	2.8	0.9	3.7	1.8
LRSC Algorithm Execution Time				Huffman Algorithm Execution Time
<b>Image Set 3</b>				
Image	Compression Time	Decompression Time	Total Time	Huffman Total Time
1	0.27	0.06	0.33	0.12
2	0.20	0.05	0.25	0.12
3	0.27	0.03	0.30	0.1
4	0.23	0.04	0.27	0.08
5	0.15	0.05	0.20	0.08
6	0.18	0.04	0.22	0.06
7	0.22	0.04	0.26	0.06
8	0.40	0.12	0.52	0.34
9	0.19	0.03	0.22	0.06
10	0.19	0.04	0.23	0.06
11	0.22	0.06	0.28	0.06
12	0.20	0.04	0.24	0.02
13	0.17	0.04	0.21	0.06
14	0.27	0.07	0.34	0.15
LRSC Algorithm Execution Time				Huffman Algorithm Execution Time
<b>Image Set 4</b>				
Image	Compression Time	Decompression Time	Total Time	Huffman Total Time
1	2.1	0.8	2.9	2.34
2	2.6	1.2	3.8	2.3
3	2.7	1	3.7	2.2
4	2.4	1.1	3.5	2.2
5	2.6	1.1	3.7	2.2
6	2.2	1	3.2	2.2
7	2	0.8	2.8	2.1
8	2.3	0.9	3.2	2.1
9	2.4	1.1	3.5	2.2

10	2.4	0.9	3.3	2.1
LRSC Algorithm Execution Time				Huffman Algorithm Execution Time
Image Set 5				
Image	Compression Time	Decompression Time	Total Time	Huffman Total Time
1	0.45	0.27	0.72	0.14
2	0.4	0.22	0.62	0.16
3	0.66	0.44	1.1	0.16
4	0.44	0.22	0.66	0.15
5	0.45	0.22	0.67	0.2
6	0.37	0.19	0.56	0.15
7	0.35	0.18	0.53	0.16
8	0.57	0.29	0.86	0.16
9	0.48	0.29	0.77	0.16
10	0.45	0.25	0.7	0.15

Table 5.11 lists the average total time for compression and decompression both algorithms for each test image.

Table 5.11 - The Average Execution Time for Both Algorithms in Seconds

The Proposed Lossless LRSC Algorithm Average Execution Time				Huffman Average Time
Image Sets	Average Compression Time	Average De-Compression Time	Total Execution Time	Total Execution Time
Image Set 1	0.61	0.21	0.82	0.45
Image Set 2	2.06	0.62	2.67	0.97
Image Set 3	0.23	0.05	0.28	0.10
Image Set 4	2.37	0.99	3.36	2.19
Image Set 5	0.462	0.257	0.719	0.16
Average	1.15	0.43	1.57	0.77

Figure 5.20 shows the total execution time needed for compression and decompression for each of the image sets for the two algorithms. The Figure shows that the Huffman algorithm has better execution time for all image sets.

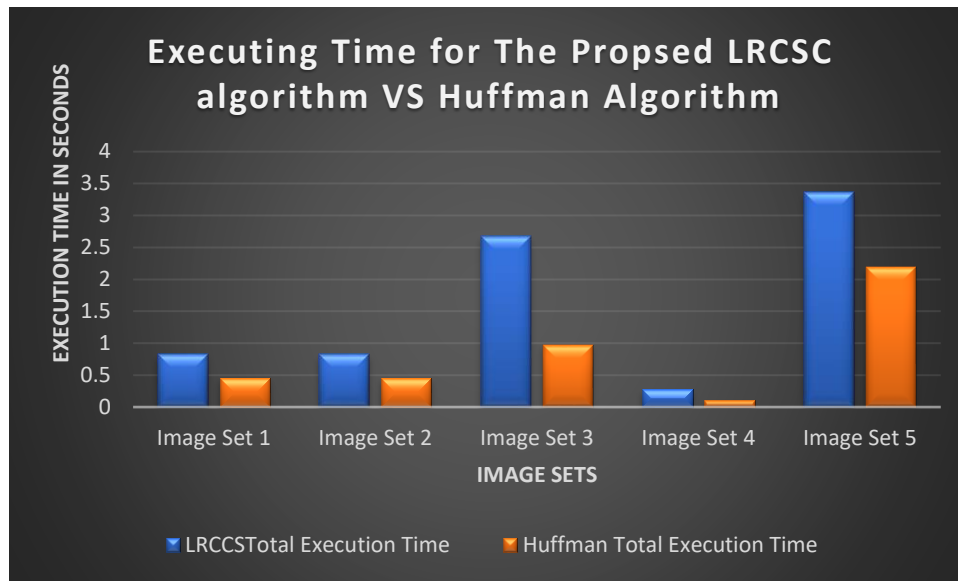


Figure 5. 20 - The Average Execution Time for Both Algorithms

### 5.6.2 Comparison Between the LRCS Compression Size and Other State of the Art Algorithm.

The LRCS algorithm was tested as lossless compression techniques and compared against other benchmark scheme for natural images compression obtained from (Khan *et al.*, 2017), the comparison with the most recent state of the art algorithms is needed for a better evaluation for the proposed algorithm results.

#### 5.6.2.1 First Comparison for Natural Images Compression Size

Table 5.12 shows the compression sizes in KBs and the compression ratios Cr, for the basic BWCA, KMTF based BWCA, JPEG 2000 LS, RCT-BWCA algorithm obtained from (Khan *et al.*, 2017) and the proposed LRCS algorithm results.

Table 5.12 - The LRCS Algorithm Results Compared with Other Four Algorithm Results

Image	BWCA		KMTF - BWCA		JPEG-2000 LS		RCT - BWCA		LRCS	
	Size	Cr	Size	Cr	Size	Cr	Size	Cr	Size	Cr
Knob & Bolt	765	1.510	750	1.540	487	2.370	381	3.020	455	2.536
Houses	1008	1.140	981	1.170	578	1.990	463	2.490	535	2.149
Landscape	1020	1.130	965	1.190	612	1.880	352	3.270	548	2.106
Light House	827	1.390	783	1.470	509	2.260	480	2.400	469	2.450
Barn	891	1.290	839	1.370	525	2.190	358	3.220	502	2.290
Parrots	791	1.460	739	1.560	447	2.580	346	3.330	438	2.632
Flowers & Sill	780	1.480	743	1.550	457	2.520	350	3.290	418	2.763

<b>Six-Shooter</b>	591	1.950	560	2.060	433	2.660	332	3.470	337	3.416
<b>Motocross</b>	991	1.160	947	1.220	574	2.010	244	4.720	519	2.216
<b>Zentime</b>	837	1.380	800	1.440	494	2.330	297	3.880	440	2.627
<b>AVERAGE</b>	<b>850</b>	<b>1.39</b>	<b>811</b>	<b>1.46</b>	<b>512</b>	<b>2.28</b>	<b>360</b>	<b>3.31</b>	<b>466</b>	<b>2.518</b>

As described in Figure 5.21, the LRCSC has 1.28 better compression ratio than the BWCA and 1.05 more than the KMTF-BWCA Cr and 0.24 more than the JPEG-2000 LS. The compression ratio is less than RCT-BWCA algorithm with 0.79.

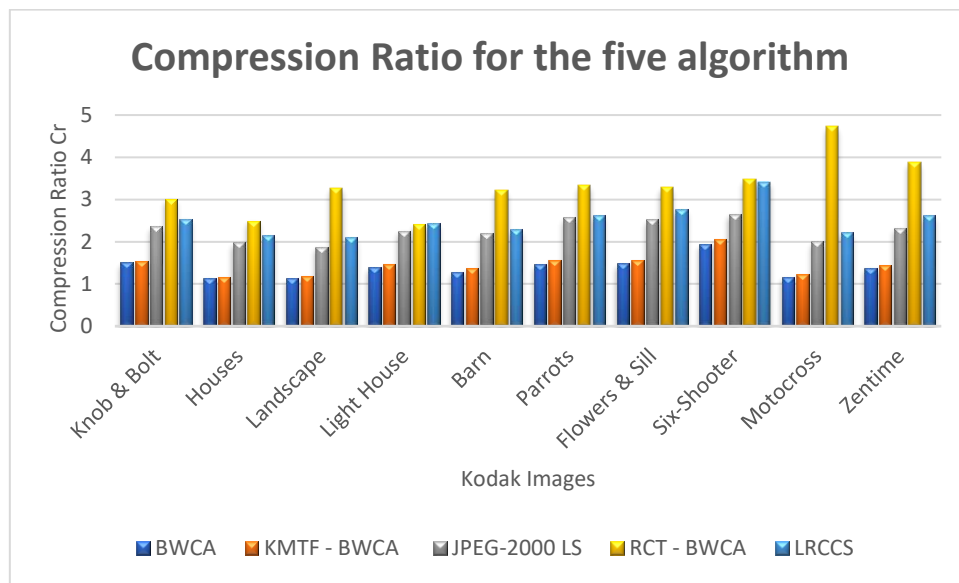


Figure 5. 21 - Compression Ratio for the Five Algorithm

### 5.6.2.2 Second Comparison for Natural Images Compression Size

Table 5.13 lists the images compression size in KBs for the proposed LRCSC algorithm and the lossless benchmark compression schemes for Kodak colour images obtained from (Khan *et al.*, 2017).

Table 5.13 - Comparison Between the LRCSC and various benchmark systems in Term of Compressed File Sizes of Kodak Colour Test Images (size in KBs)

S.No	SCHEME	KODAK TEST IMAGE										Total Size
		Knob & Bolt	Houses	Landscape	Light House	Barn	Parrots	Flowers & Sill	Six-Shooter	Motocross	Zentime	
1	ADVANCE COMP	608	422	495	493	477	666	460	468	614	451	5154
2	ALLUME	386	576	487	503	699	407	788	370	410	372	4998
3	BBWCA	381	463	352	480	308	346	350	332	244	297	3553
4	BCM	457	785	628	490	532	737	375	381	507	390	5282
5	BULK ZIP	450	710	372	441	753	613	469	751	482	603	5644
6	CAESIUM	475	422	496	518	797	500	706	407	736	411	5468
7	C-MIX	453	510	457	497	625	520	761	666	657	456	5602
8	COMPRESSOR.IO	413	765	425	597	380	415	392	391	603	668	5049
9	CRUSH	409	692	556	536	544	795	646	612	583	376	5749
10	FILE MINIMIZER	560	758	471	521	740	681	691	596	573	495	6086
11	FILE OPTIMIZER	409	432	721	665	559	502	401	447	618	518	5272
12	HEVC (x265)	403	493	343	487	397	538	326	418	462	353	4220
13	LZ4X	370	420	412	373	402	537	465	456	444	429	4308
14	LRCSC	455	535	548	469	502	438	418	337	519	440	4660
15	MRP	497	760	550	511	513	791	534	474	628	409	5667
16	NANOZIP	475	556	472	519	446	543	490	431	551	715	5198
17	PAQ8PXD_V4	450	490	598	607	489	655	733	596	372	569	5559
18	UPACK 0.25	661	710	379	675	503	529	371	572	587	568	5555
19	WINRK 3.1.2	598	515	398	783	674	457	374	611	456	593	5459
20	ZCM 0.92	495	631	772	542	408	714	565	597	416	450	5590

As displayed in Figure 5.22, the LRCSC algorithm provide good compression ratio. The best compression came with the BWCA with a total size of 3553 KBs. The second-best algorithm is the HEVC followed by the LZ4X by having 4220 and 4308 respectively. The LRCSC decrease the image size with 4660 KBs.

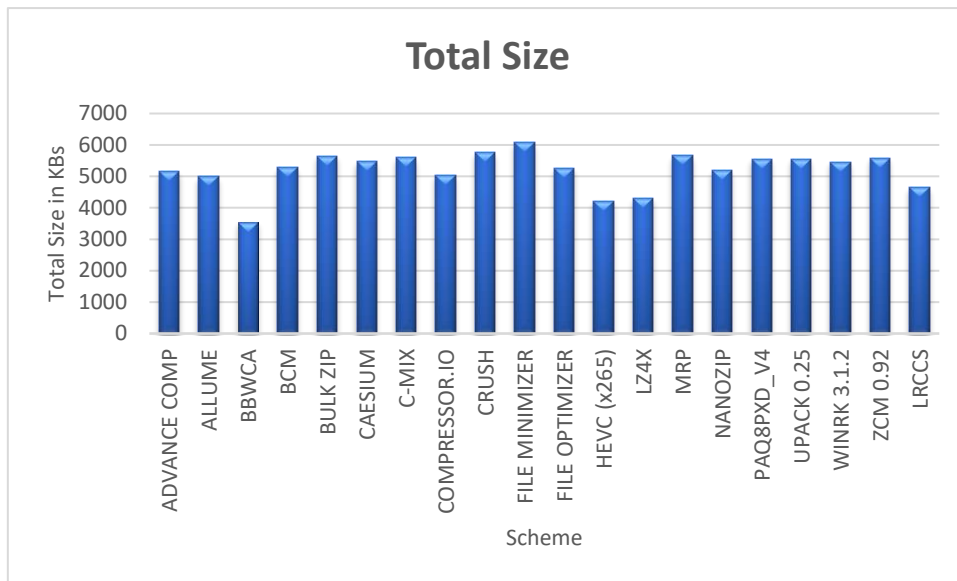


Figure 5.22 - Total Compression Size for the Kodak Image Set

### 5.6.2.3 Third Compression for Raster Map Images Size

The proposed LRCS C was tested, and the results were compared with another raster map benchmark scheme. The comparison is done with the JPEG-LS, PNG, GIF, Bi-level Burrows BBWCA and BLiSE algorithms as obtained from (Khan *et al.*, 2017). Table 5.14 lists the compression size in KBs and the compression ratio for the previous algorithms and the LRCS C algorithm for 10 raster map images obtained from <https://sites.google.com/site/qinzoucn/documents/>.

Table 5.14 - The LRCS C Results Compared with Other Four Algorithm Results

Image	JPEG-LS		PNG		GIF		BLiSE		BBWCA		LRCS C	
	Size	Cr	Size	Cr	Size	Cr	Size	Cr	Size	Cr	Size	Cr
<b>Map 1</b>	235.32	3.23	29.79	25.55	20.92	36.38	12.28	61.98	11.72	64.94	35	21.47
<b>Map 2</b>	188.88	4.35	28.00	29.32	17.57	46.73	10.04	81.78	10.40	78.95	30	27.53
<b>Map 3</b>	185.36	4.32	27.15	29.50	18.39	43.55	10.94	73.21	11.22	71.38	28	28.37
<b>Map 4</b>	88.82	9.43	11.08	75.62	8.38	99.98	3.72	225.22	3.57	234.68	11	78.39
<b>Map 5</b>	238.76	3.35	34.67	23.08	22.86	35.01	16.35	48.95	16.15	49.55	40	19.91

<b>Map 6</b>	228.04	3.62	30.23	27.32	17.90	46.14	11.08	74.54	12.76	64.73	43	19.07
<b>Map 7</b>	157.13	5.33	23.59	35.52	14.21	58.96	7.37	113.69	10.41	80.49	28	29.85
<b>Map 8</b>	254.97	3.19	36.58	22.25	23.61	34.47	18.10	44.96	15.36	52.98	40	20.59
<b>Map 9</b>	265.47	2.90	41.62	18.49	25.14	30.61	19.43	39.60	17.22	44.68	49	15.77
<b>Map 10</b>	215.75	3.56	34.38	22.33	22.29	34.44	14.56	52.72	14.45	53.12	36	21.46
<b>Average</b>	<b>2058.5</b>	<b>4.32</b>	<b>297</b>	<b>30.89</b>	<b>191.27</b>	<b>46.62</b>	<b>123.87</b>	<b>81.66</b>	<b>123.26</b>	<b>79.55</b>	<b>34</b>	<b>28.243</b>

As displayed in Figure 5.23, BLiSE outperforms all other algorithm, since the BLiSE algorithm is designed for raster maps compression and the other techniques are designed for general images type compression. The LRCSC algorithm results are better than the JPEG-LS but is less than the other techniques; although, the LRCSC algorithm saved 95.7% from the total image sets size.

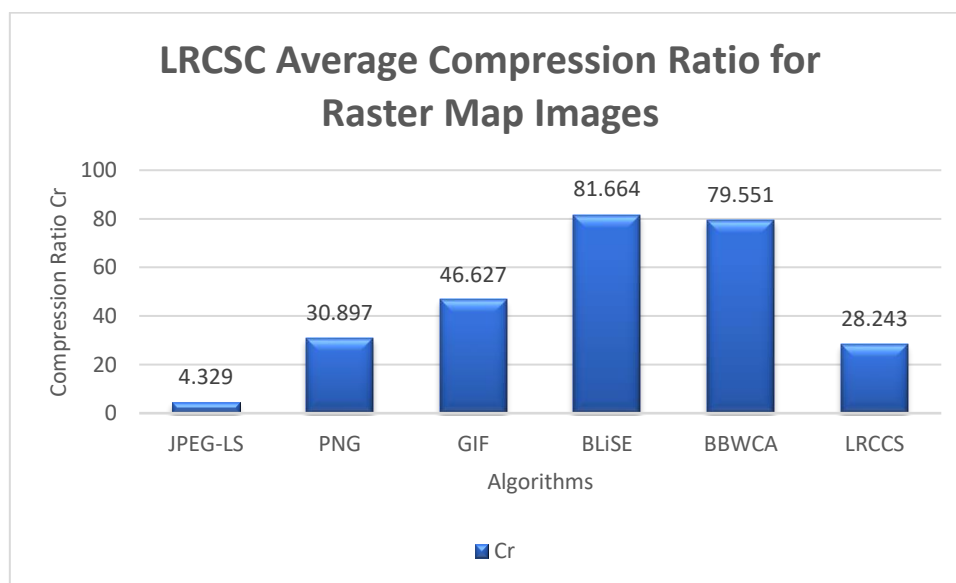


Figure 5. 23 - LRCSC Average Compression Ratio for the Raster Map Image Set



### 5.6.3 Comparison Between the LRCSC Execution Time and Other State of the Art Algorithm.

The LRCSC algorithm was tested in the same environments as described and used in (Khan *et al.*, 2017) by using the same hardware and software and compared with the BWCA, KMTF-BWCA and RCT\_BWCA. Table 5.15 presents the compression needed time (CT) and the decompression needed time (DCT) for the LRCSC, BWCA, KMTF – BWCA, and RCT – BWCA obtained from (Khan *et al.*, 2017).

Table 5.15 - Execution Time in Seconds for the Kodak Image Set for Different Algorithms

Image	BWCA		KMTF - BWCA		RCT - BWCA		LRCSC Win Xp		LRCSC Win 10	
	CT	DCT	CT	DCT	CT	DCT	CT	DCT	CT	DCT
<b>Knob &amp; Bolt</b>	16.46	23.76	17.21	25.39	17.7	28.06	5.43	1.29	2.1	0.8
<b>Houses</b>	16.32	21.32	17.07	24.82	19.77	27.8	6.24	1.53	2.6	1.2
<b>Landscape</b>	17.37	22.77	18.12	25.55	18.18	28.66	5.61	1.34	2.7	1
<b>Light House</b>	19.17	21.46	19.92	22.63	19.48	28.95	7.61	1.44	2.4	1.1
<b>Barn</b>	16.91	21.97	17.66	22.19	19.6	22.87	6.91	1.85	2.6	1.1
<b>Parrots</b>	16.78	22.9	17.53	25.98	19.95	29.23	5.74	1.23	2.2	1
<b>Flowers &amp; Sill</b>	17.58	22.46	18.33	25.95	18.41	30.6	5.01	1.84	2	0.8
<b>Six-Shooter</b>	18.82	21.17	19.57	24.4	20.98	27.46	6.20	2	2.3	0.9
<b>Motocross</b>	16.07	23.85	16.82	28.09	19.14	30.5	5.45	2.01	2.4	1.1
<b>Zentime</b>	18.02	23.34	18.77	23.24	19.49	28.55	7.55	1.52	2.4	0.9
<b>Sum</b>	<b>173.5</b>	<b>225</b>	<b>181</b>	<b>248.24</b>	<b>192.7</b>	<b>282.68</b>	<b>61.75</b>	<b>16.05</b>	<b>23.7</b>	<b>9.9</b>
<b>Total</b>	<b>398.5</b>		<b>429.24</b>		<b>475.38</b>		<b>77.8</b>		<b>3.69</b>	

The LRCSC algorithm achieve the best execution time by having 77.8 s for compression and decompression together and save 320.7 s more than the BWCA algorithm.

By running the LRCSC algorithm in win 10 operating system (64-bit) with Intel core i7-7500U CPU @2.70GHz with 8 GB RAM the execution time is 3.69 s. Figure 5.24 display the total execution time for the four algorithms in seconds.

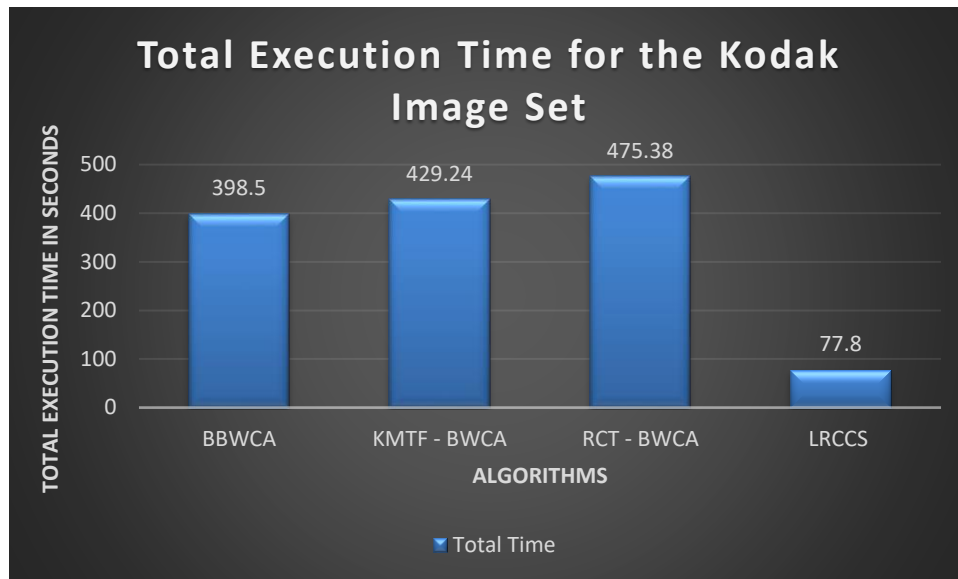


Figure 5. 24 - Total Execution Time for the Kodak Image Set

## 5.7 Chapter Summary

The aim of this chapter was to develop a lossless image compression algorithm, to enhance the CSC compression ratio for the low-resolution images, with zero distortion and acceptable execution time.

To reach the optimal solution between compression time and saving bits, the LRCSC algorithm was proposed.

- The proposed LRCSC algorithm is designed to work with low resolution images.
- The LRCSC algorithm achieved high compression ratio for raster map images.
- The LRCSC algorithm achieved the best computational time due to its simplicity of implementation and speed of execution.

The only disadvantage of the LRCSC algorithm is when applying the algorithm on natural images, the compression rate is decreased, since the LRCSC algorithm is designed for low-resolution images.

To solve this issue, the researcher adapted an artificial intelligence system that is responsible for classifying the images before compression to enhance the compression ratio for the CSC and LRCSC algorithms. The next chapter describe the solution in detail.

## ***CHAPTER SIX: AUTOMATED SYSTEM FOR IMAGE COMPRESSION***

---

### **6 Chapter Overview**

This chapter describes in detail the need for an automated compression system and the need for a classification system, followed by detailed description of a possible automated compression system.

---

#### **6.1 Introduction**

This research developed two image compression algorithms, the first algorithm is the CSC algorithm, which was designed to compress any image format with any resolution and achieved the best compression rate when compressing high resolution images (HRI) such as natural images. The only disadvantage of the CSC algorithm is when applying it on low resolution images such as synthetic images E.g. raster map images where the compression ratio decreased. The second algorithm is the LRCSC which was developed to enhance the compression ratio for the low-resolution images (LRI). The only disadvantage of the LRCSC algorithm is when compressing HRI such as natural images where the compression ratio decreases. Choosing the suitable compression algorithm (CSC or LRCSC) that gives the best compression results with the input image without requiring human intervention would be a better approach. This approach could be implemented by running both compression algorithms (CSC and LRCSC) for the same input image to find out the best compression performance in terms of compression ratio or by using an image classification system.

Human beings can distinguish easily between natural and synthetic images in a very short time. Unfortunately, computer can't distinguish between images without having an image classification system to perform this task. Many features can be derived from the input images, such as number of colours, edge map and energy level. In order to build a solid classifier, we need to combine those features classification values, since if we used them separately, they would lead to wrong classification results. An ideal image classification system will classify images into different classes with no hesitation such as the human intelligence do (Khalsa and Gudadhe 2014). Many Artificial Intelligence (AI) systems have been developed to mimic the human brain functionality of image classification. This chapter is to enhance the compression ratio of the proposed two algorithms by developing an automated compression system for choosing the suitable algorithm with the input image.

The proposed automated compression system is developed to run both compression algorithms (CSC and LRCSC) for the same input image to find out the best compression performance in terms of compression ratio. The system will start by loading the input image to determine its details followed by compressing the image by the CSC algorithm to calculate the compression ratio, and then the system compress the input image for the second time using the LRCSC algorithm to calculate its compression ratio. Both algorithm compression ratios are then compared to identifies the best algorithm for the input image. The final phase is to restore the image according to the suitable compression algorithm that achieved the best compression ratio.

## 6.2 The Fully Automated System Compression Size

Table 6.1 lists the fully automated compression system results in term of compression ratio.

The system should provide the best compression ratio when comparing with the CSC and the LRCSC.

Table 6. 1 -Fully Automated System Compression Ratio

Image Sets	CSC Average Compression Ratio	LRCSC Average Compression Ratio	automated system Compression Ratio
Image Set 1	2.35	1.85	2.35
Image Set 2	2.57	2.07	2.57
Image Set 3	2.22	1.98	2.22
Image Set 4	3.54	2.52	3.54
Image Set 5	5.63	28.25	28.25
<b>Average</b>	<b>3.26</b>	<b>7.33</b>	<b>7.79</b>

As listed in Table 6.1, the fully automated compression system dramatically decreases the image size for the five image sets. In image sets (1,2,3,4) the system achieved the best compression ratio by using the CSC algorithm. By observing the fifth image set results, the fully automated compression system achieved the best compression ratio by using the LRCSC algorithm, since this image sets includes low resolution images.

## 6.3 The Fully Automated System Execution Time

The fully automated system execution time can be calculated by measuring the system compression speed in seconds for the CSC and the LRCSC, and the decompression speed for

the chosen compression algorithm. Table 6.2 displays the average execution time for each image sets in seconds.

Table 6. 2 - The Fully Automated System Average Compression Time





Image Sets	CSC Average (Execution Time)	LRCSC Average (Execution Time)	Automated System Average (Execution Time)
Image Set 1	0.32	0.82	0.93
Image Set 2	0.44	2.67	2.5
Image Set 3	0.05	0.28	0.28
Image Set 4	0.37	3.36	2.74
Image Set 5	0.14	0.71	0.84
<b>Average</b>	0.264	1.568	1.449




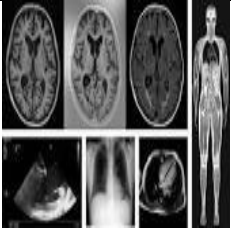
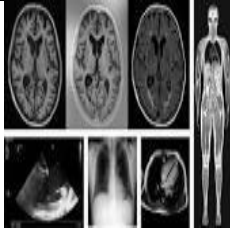






The fully automated system execution time is increased when comparing with the CSC or LRCSC. In the first, second, third and fourth image sets, the system needs 0.61, 2.06, 0.23, 2.37 seconds respectively more than the CSC algorithm. Were in the fifth image sets the system needed 0.13 seconds more than the LRCSC algorithm.

#### 6.4 The Fully Automated System Image Quality

The fully automated system uses the two lossless image compression algorithm and restore the original image as a 100% perfect match from the decompressed image. The fully automated system MSE value is zero for all the images in the five image sets. Table 6.3 display two image samples for each image set.

Table 6. 3 - Sample Images from the Five Image Sets Before and After Compression

Image Set 1			
Original Image	Compressed Image	Original Image	De-Compressed Image
Camera Man (GIF)		Lena (PNG)	
			
Image Set 2			
Original Image	Compressed Image	Original Image	De-Compressed Image

Lena (PNG)		Goldhill (BMP)	
			
<b>Image Set 3</b>			
Original Image	Compressed Image	Original Image	De-Compressed Image
Medic (JPEG)		Boat (JPEG)	
			
<b>Image Set 4</b>			
Original Image	Compressed Image	Original Image	De-Compressed Image
Parrots (PNG)		Motocross (PNG)	
			
<b>Image Set 5</b>			
Original Image	Compressed Image	Original Image	De-Compressed Image
Map 1 (BMP)		Map 2 (BMP)	
			

Although the fully automated system is running well, one of the system limitations is every time the user run the system to compress an image; the system run both algorithms. One way of improving the current system is to develop an intelligence system that will classify the input

image first into (low-resolution or high- resolution images) and then chose the appropriate algorithm to compress the image. Indeed, there are many AI modules that are developed to classify images.

## **6.5 Chapter Summary**

This chapter described the fully automated compression system to enhance the image compression ratio by choosing the suitable image compression algorithm (CSC or LRCSC) for compressing the image regarding its type (low-resolution or high- resolution images) images. By adopting the previous fully automated compression system, the researcher solved the problem of the low compression ratio when compressing low-resolution images such as synthetic images by the CSC algorithm, and also, solved the problem of the low compression ratio when compressing high- resolution images such natural images by LRCSC algorithm.

## **CHAPTER SEVEN: CONCLUSIONS AND FUTURE WORK**

---

### **7 Chapter Overview**

The entire research is summarized in this chapter. Starting with reviewing the research activities, followed by the research contribution to the knowledge and the limitation encountered by the researcher, and finally, some recommendation for future work in the domain is described.

---

### **7.1 Research Summary**

This research aims to address the challenge of saving and transmitting large size images by developing two lossless image compression algorithms. The first chapter covered the research background and motivation; the research aim and objectives were described, followed by the research methodology and the expected contribution to knowledge. After establishing the research foundation from the first chapter, the second chapter concentrated on reviewing the literature for the two image compression techniques (lossless and lossy) to provide a comprehensive state-of-the-art literature and analysed them by focusing on the research aim and objectives to provide the research gaps for the lossless and lossy image compression techniques.

Based on the research gap identified from chapter two, the third chapter describes the proposed solutions and the research requirements that are needed to develop the proposed solutions, such as the development tools, the used programming language and the tested image sets.

Chapter four described the first solution of enhancing the lossless image compression rate and execution time by proposing the CSC algorithm for high resolution images such as natural images. It starts with a detailed explanation for all the procedures used in the proposed algorithm, followed by the validation of the algorithm and described some of its limitation. Finally, the algorithm was compared to the state-of-the-art algorithms.

Chapter five described the solution for enhancing the lossless image compression rate and execution time by proposing the LRCSC algorithm for the low-resolution images such as synthetic images and raster map images, It starts with a detailed explanation for all the procedures used in the proposed technique, followed by the validation of the algorithm by testing its results, and a comparison of the newly developed algorithm with state-of-the-art algorithms. Furthermore, the limitation of the LRCSC algorithm is described as well.



Chapter six provided solutions for the limitations from chapter four and five. By a fully automated compression system for choosing the suitable compression algorithm.

## **7.2 Research Contributions and Review of the Research Objectives**

The following sub sections describes in detail the research contributions to the body of knowledge in the domain of lossless image compression, by achieving the research objectives.

### **7.2.1 Lossless and Lossy Image Compression State-of-the-Art**

Conducting a comprehensive critical review of existing literature in the domain of lossless and lossy image compression techniques to provide a critical review for leveraging existing image compression techniques and summarising the research findings to provide a contribution to the body of knowledge. This part is to provide a rich knowledge for researchers in the domain with a guide for the gaps in the current research.

### **7.2.2 Addressing the Challenges of Lossless Image Compression Techniques**

The research gaps from the literatures review is used to set a path for developing a new solution to enhance the compression performance. A novel solution is developed by avoiding the limitations from the current literatures and by adopting the advantages from some of the literatures.

### **7.2.3 Developing a Lossless Image Compression Algorithm for Natural Images**

Natural images mostly defined as high resolution images. Such images suffer from the low compression rate and execution time regarding the reasons of having many colours to represent it. Natural images share the fact of having a high coloration between neighbouring pixels; this fact is the main feature that where considered in developing the CSC compression algorithm.

The proposed algorithm was developed by using MATLAB programming language and the results was investigated regarding the compression size, image quality and execution time. To reach the best conclusion from the results investigation, the results where compared with the most common state of the art lossless algorithms.

The CSC lossless image compression algorithm enhanced the current state of the art compression rate, with zero distortion and acceptable execution time. The CSC algorithm is designed to work with any application and support all image formats whether the input image is a high resolution or low-resolution image. The CSC algorithm is suitable for natural image compression and can be used as stand-alone algorithm or as a pre-processing phase for any lossless or lossy techniques.

The limitation of the CSC compression algorithm is when applying the algorithm on raster map images, the compression rate is decreased, since the CSC algorithm gives better results for the high-resolution images. To solve this issue, the researcher proposed a new algorithm that enhanced the compression rate for the low-resolution images. The next section describes the solution contribution.

#### **7.2.4 Developing a Lossless Image Compression Algorithm for Synthetic Images**

The LRCSC algorithm is designed to solve the low compression rate problem with low resolution images in the CSC algorithm. The LRCSC algorithm is capable to work with low resolution images and achieved higher compression rate than the CSC compression algorithm for synthetic images such as raster map images.

The proposed algorithm was developed by using MATLAB programming language and the results was investigated regarding the compression size, image quality and execution time. To reach the best conclusion from the results investigation, the results where compared with the most common state of the art lossless algorithms.

The LRCSC lossless image compression algorithm achieved high compression rate with zero distortion and acceptable execution time. The LRCSC algorithm is designed to work with any application and support all image formats whether the input image is a high resolution or low-resolution image. The LRCSC achieved the best results when compressing low resolution images and its suitable for raster map image compression and can be used as stand-alone algorithm or as a pre-processing phase for any lossless or lossy techniques.

The limitation of the LRCSC compression algorithm is when applying the algorithm on natural images, the compression rate is decreased, since the LRCSC algorithm gives better results for the low-resolution images.

#### **7.2.5 Develop a Fully Automated System for Choosing the Suitable Algorithm for Compressing the Images Regarding its Type.**

The CSC algorithm provide the best compression rate with natural images and resulted less compression efficiency when compressing synthetic images, while, the LRCSC algorithm give better results when applying it on synthetic images and less compression rate when dealing with natural images.

The researcher designed a fully automated system to run both compression algorithms (CSC and LRCSC) for the same input image to find out the best compression performance in terms of compression ratio, and regarding the image type the system will choose to compress the

input image with the CSC or LRCSC algorithm. The fully automated system enhanced the compression ratio by combining the two proposed algorithms in one system.

### **7.3 Research Limitations**

After achieving the research aim and objectives, the researcher recognised the following limitation which can be solved in the future work.

- The CSC and LRCSC algorithms dramatically decrease the execution time when comparing with other algorithm from the literature review, while the enhancement in compression rate is very small.
- Using the fully automated system enhanced the compression rate and increase the execution time; the classification execution time should be added to the compression and decompression time for measuring the system performance.
- Nowadays, images have larger size than before, finding a compression algorithm results that uses new images in their research is to hard; hence, a comparison using new image data sets is not applicable.

### **7.4 Recommendations for Future Work**

Enhancing the compression rate and execution time while preserving the image quality as 100% perfect match with the original image is the main three parameters for evaluating any lossless image compression algorithm. Further research in these regards are described as follows:

- Adopting pre-processing phase before the CSC algorithm may enhance the compression rate, the pre-processing phase should result in matrices with smaller values and maintain the high correlation between intensities.
- The CSC algorithm can be used as a pre-processing phase for any other lossless or lossy image compression techniques to enhance the compression rate.
- Creating a dynamic application that includes the most popular and effective image compression algorithms by programming each of the algorithms to compare the algorithms results for any input image. This application should be controlled and tested correctly. Having such application, may help the researchers in making comparison easily for any images (old or new) without the need for benchmark system that confines the researcher to specific images.

- Implementing the GF-FSAE algorithm to provide details of the classification algorithm, complexity and added execution time to calculate the classification system performance.

#### 7.4.1 The Proposed AI Algorithm (GF-FSAE) for Image Classification

Before compressing the image, an artificial intelligence system is used to classify the input image into (low-resolution or high-resolution images) by using an efficient deep learning algorithm. The research problem is to choose the suitable compression algorithm (CSC or LRCSC) that gives better compression results with the input image.

To reach the best results from the two proposed algorithms, the researcher adopted a fully automated system by using a deep learning technique for classifying the input image to determine its type (low-resolution or high-resolution images), and based on the image type, the system will choose to compress the input image with the CSC or LRCSC algorithm. Figure 7.1 shows the fully automated system flowchart.

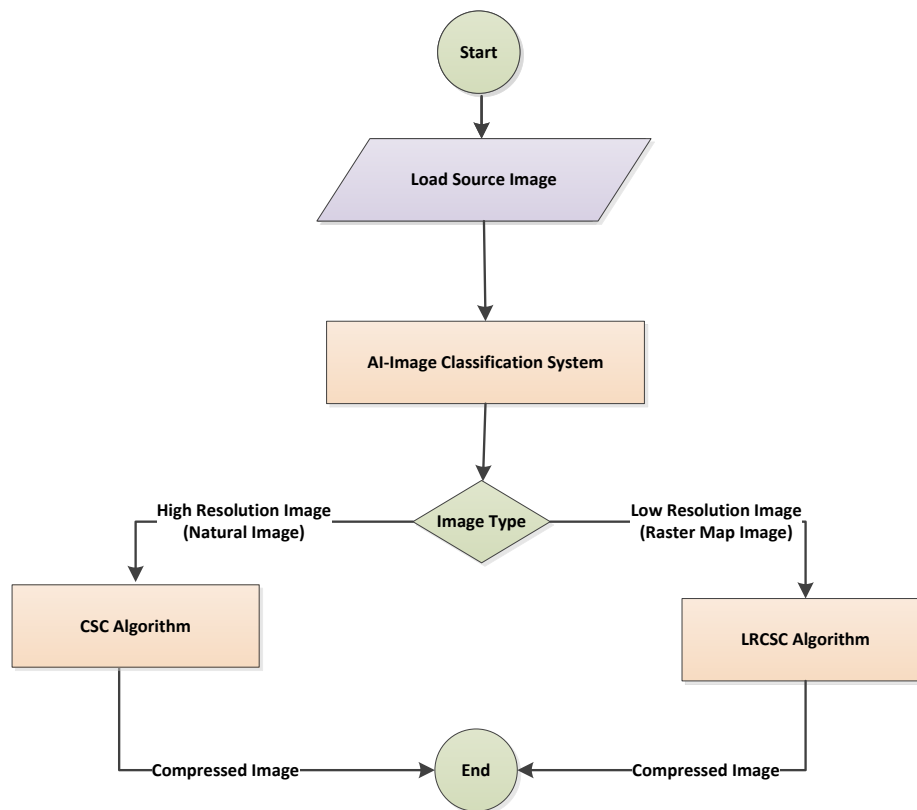


Figure 7. 1 - The Fully Automated AI system Flowchart

### 7.4.2 Artificial Intelligence

Artificial intelligence (AI) is a computer science and engineering domain interested in developing systems that mimic the intelligence from human behaviours, such as learning and adaptation, image observation and classification, natural language processing, problem solving and planning (Tecuci, 2012). AI is a way of making a computer or a software think intelligently, like intelligent humans do. The main goal of AI is to simulate the intelligent environment and principles in the human and animal behaviour (Chassagnon, Vakalopoulou and Paragios, 2020).

**Machine learning (ML):** is part of AI which gives machines the ability of automatically learn and improve from experience without being explicitly programmed to do so, it allows the machine to solve problems by giving it the ability of thinking (Han, Liu. Mihaela, 2017).

Samuel was one of the first who identified ML in 1959, where the ML affected the development of technological progress significantly since that day (Samuel., 1959). Since the ML revolution starts to the current day, a massive amount of data is created. It is reported that the estimated amount of the created data in the year 2020 will be approximately 1.7 MB per second for every person (Anon, 2019). ML provides a path to create predictive models to analyse this huge data to provide better and accurate results (Lu *et al.*, 2014); (Stoyanov, Taylor and Hutchison, 2018).

**Drawbacks of the Machine Learning:** (Edureka, 2019) demonstrate the relationship between ML and Deep learning (DL) as shown in Figure 7.2.

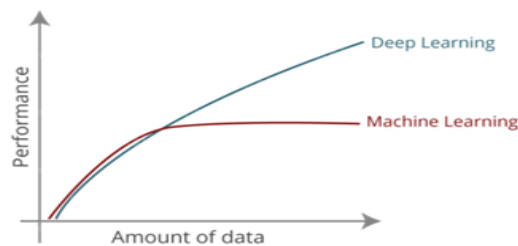


Figure 7. 2 - Relationship Between ML and Deep Learning (DL) (Edureka, 2019)

- ML cannot process high dimensional data, since it contains a large set of variables.
- Not ideal for performing object detection and image processing:  
The traditional ML algorithm is not effective in predicting when dealing with high dimensional data, since it has too many inputs and outputs. E.g. in case of image recognition, to reach the best prediction, the algorithm input should cover most of the images features by having a huge input image sets with different image type.
- ML is not effective with feature extraction.

feature extraction is to give the computer the search parameters that distinguish the images from each other's and helps the algorithm to classify images. The main objective of feature extraction is to help the algorithm in predicting the outcome to have a better accuracy. The limitation of feature extraction comes when loading raw data to the algorithm where the feeding rarely worked, and that is why ML is struggling with feature extraction. Therefore, the programmer faces a new challenge without using feature extraction, since the algorithm efficiency will depend on the programmer skills of narrow down only the significant predictors from the input predictor by manually studying the relationship between the input data. Hence, complex problems are difficult to solve with ML algorithms. Deep Learning by using Neural Networks is part of ML that focus on solving complex problem with a huge data set (He *et al.*, 2015).

**Deep Learning (DL):** is a sub-field of ML, created to solve the problem with analysing huge data that needs a lot of computational resources, by using the Neural Networks (NN) concepts to automate the feature extraction process and minimize the human interference as possible (Pan, Shi and Xu, 2018) (Traore, Kamsu-foguem and Tangara, 2018).

### **7.4.3 The Chosen Artificial Intelligence Technique for Image Classification**

Before compressing the image, an artificial intelligence system should be used to classify the input image into (synthetic images or natural images) by using an efficient deep learning algorithm. One of the leading image classification algorithms is the Guided Filtering for Fine-Tuning Stacked Autoencoder (GF-FSAE) algorithm. The (GF-FSAE) achieved high image classification accuracy rate by having a 99% accuracy (Wang *et al.*, 2017). The GF-FSAE algorithm is proposed to be implemented as an image classification system, since it has a very high accuracy.

The proposed classification system should load the input image and classify it into (low-resolution or high-resolution images) and send the original input image to the related compression algorithm (CSC or LRCSC) to compress it according to its type.

The proposed AI technique for image classification (GF-FSAE) is not implemented in this research and could be implemented in the future as future work. This section is to propose a solution for classifying the image before compressing it to enhance the compression ratio and to decrease the human interference as possible.

## References

- Abdalla, S. H. and Osman, S. E. (2016) ‘Digital Image Processing Technology based on MATLAB’, *International Journal of Advanced Research in Computer Science*, 7(3), pp. 216–221.
- Abo-Zahhad, M. *et al.* (2015) ‘Huffman Image Compression Incorporating DPCM and DWT’, *Journal of Signal and Information Processing*, 06(02), pp. 123–135. doi: 10.4236/jsip.2015.62012.
- Abubaker, A., Eshtay, M. and AkhoZahia, M. (2016) ‘Comparison Study of Different Lossy Compression Techniques Applied on Digital Mammogram Images’, *International Journal of Advanced Computer Science and Applications (IJACSA)*, 7(12), pp. 149–155.
- Agustsson, E. *et al.* (2019) ‘Generative Adversarial Networks for Extreme Learned Image Compression’, *In Proceedings of the IEEE International Conference on Computer Vision*, 1, pp. 221–231.
- Al-azawi, S. *et al.* (2011) ‘LOW COMPLEXITY IMAGE COMPRESSION ALGORITHM USING AMBTC AND BIT PLANE SQUEEZING’, *International Workshop on Systems, Signal Processing and their Applications, WOSSPA. IEEE*, pp. 131–134. doi: 10.1109/WOSSPA.2011.5931432.
- Al-Laham, M. and Emary, I. (2007) ‘Comparative Study Between Various Algorithms of Data Compression Techniques’, *International Journal of Computer Science and Network Security(IJCSNS)*, 7(4), pp. 281–291. Available at: [http://paper.ijcsns.org/07\\_book/200704/20070440.pdf](http://paper.ijcsns.org/07_book/200704/20070440.pdf).
- Al-Wahaib, M. and Wong, K. (2010) ‘A lossless Image Compression Algorithm Using Duplication Free Run-Length Coding’, in *Second International Conference on Network Applications, Protocols and Services (NETAPPS), IEEE*, pp. 245–250. doi: 10.1109/NETAPPS.2010.51.
- Alan, H. and Samir, C. (2012) *Integrated Series in Information Systems, Design Research in Information Systems, Springer*. New York. doi: 10.1007/978-1-4419-6108-2.
- Anon (2019) ‘World’s First General Technology Platform for Borderless Data Sharing Revealed.’, *PR Newswire Europe Including UK Disclose*, pp. 2019–2021. Available at: <https://www.prnewswire.com/news-releases/worlds-first-general-technology-platform-for-borderless-data-sharing-revealed-300811548.html>.
- Ballé, J., Laparra, V. and Simoncelli, E. P. (2017) ‘End-to-End Optimized Image Compression’, in *ICLR arXiv preprint arXiv*. doi: 1611.01704.
- Begum, S. and Aygun, R. S. (2012) ‘Analyzing the Performance of Hierarchical Binary Classifiers for Multi-Class Classification Problem Using Biological Data’, in *International Conference on Machine Learning and Applications IEEE*, pp. 145–150.
- Burrows, M. and Wheeler, D. J. (1994) ‘A Block-Sorting Lossless Data Compression Algorithm’, *Digital Systems Research*.
- Carreto-Castro, M. *et al.* (1993) ‘Comparison of Lossless Compression Techniques’, *Proceedings of 36th Midwest Symposium on Circuits and Systems IEEE*, pp. 1268–1270. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=343329>.
- Chassagnon, G., Vakilopoulou, M. and Paragios, N. (2020) ‘Artificial Intelligence

- Applications for Thoracic Imaging’, *European Journal of Radiology*, 123(November 2019).
- Chawla, S., Beri, M. and Mudgil, R. (2014) ‘Image Compression Techniques : A Review’, *International Journal of Computer Science and Mobile Computing*, 3(8), pp. 291–296.
- Chou, C-H., K.-C. L. (2008) ‘Colour image compression based on the measure of just noticeable colour difference’, *IET Image Processing*, 2(6), pp. 295–303. doi: 10.1049/iet-ipr.
- Dadgostar, H. and Afsari, F. (2016) ‘Image Steganography Based on Interval-Valued Intuitionistic Fuzzy Edge Detection and Modified LSB’, *Journal of Information Security and Applications*. Elsevier, 30, pp. 94–104. Available at: <http://dx.doi.org/10.1016/j.jisa.2016.07.001>.
- Edureka (2019) *Machine Learning Process*. Available at: <https://www.edureka.co/blog/what-is-a-neural-network/>.
- Ernawan, F., Abu, N. A. and Suryana, N. (2013) ‘TMT Quantization Table Generation Based on Psychovisual Threshold for Image Compression’, *International Conference of Information and Communication Technology, ICoICT*. IEEE, pp. 202–207.
- Gagie, T., Gawrychowski, P. and Puglisi, S. J. (2015) ‘Approximate Pattern Matching in LZ77-Compressed Texts’, *Journal of Discrete Algorithms*. Elsevier, 32, pp. 64–68. Available at: <http://dx.doi.org/10.1016/j.jda.2014.10.003>.
- Gogoi, M. and Ahmed, M. (2016) ‘Image Quality Parameter Detection : A Study’, *International Journal of Computer Sciences and Engineering*, 4(7), pp. 110–116.
- Gregor, S. and Hevner, A. R. (2013) ‘Positioning and Presenting Design Science Research for Maximum Impact’, *MIS Quarterly*, 37(2), pp. 337–355.
- Gupta, A., Bansal, A. and Khanduja, V. (2017) ‘Modern Lossless Compression Techniques: Review, Comparison and Analysis’, *Proceedings of the 2nd International Conference on Electrical, Computer and Communication Technologies, ICECCT, IEEE*, pp. 1–8.
- Han, Liu. Mihaela, C. (2017) ‘Granular Computing Based Machine Learning’, in *Big Data Processing Approach*. Warsaw, Poland: Springer, pp. 4–7.
- Hanid, M. (2014) *Design Science Research as an Approach to Develop Conceptual Solutions for Improving Cost Management in Construction, Doctoral dissertation, University of Salford*.
- Hazarika, D., Nath, V. K. and Bhuyan, M. (2015) ‘A lapped Transform Domain Enhanced Lee Filter with Edge Detection for Speckle Noise Reduction in SAR Images’, *IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS)*, pp. 243–248.
- He, X. *et al.* (2015) ‘Intelligence Science and Big Data Engineering. Big Data and Machine Learning Techniques’, *5th International Conference, IScIDE Springer*. Suzhou, China, Vol(3243).
- Hevner, A., Chatterjee, S. and Juhani, I. (2010) ‘Twelve Theses on Design Science Research in Information Systems’, in *Design Research in Information Systems*. Boston, MA, pp. 43–62. Available at: <http://www.springerlink.com/index/10.1007/978-1-4419-5653-8>.
- Huffman, D. A. (1951) ‘A method for the Construction of Minimum-Redundancy Codes’, *Proceedings of the IRE*, 40(9), pp. 1098–1101.
- Hussain, A. and Al-Fayadh, A. (2018) ‘Image Compression Techniques: A Survey in



Lossless and Lossy Algorithms’, *Neurocomputing*. Elsevier B.V., 300, pp. 44–69. Available at: <https://doi.org/10.1016/j.neucom.2018.02.094>.

Hussein, A. H., Mahmud, S. S. and Mohammed, R. J. (2017) ‘Image Compression Using Proposed Enhanced Run Length Encoding Algorithm’, *Ibn AL-Haitham Journal For Pure and Applied Science*, 24(1).

Ibrahim, R. A., Youssef, S. M. and Elkaffas, S. M. (2015) ‘An Enhanced Fractal Image Compression Integrating Quantized Quadrees and Entropy Coding’, in *11th International Conference on Innovations in Information Technology, IIT*, pp. 190–195.

Jallouli, S. *et al.* (2017) ‘A Preprocessing Technique for Improving the Compression Performance of JPEG 2000 for Images with Sparse or Locally Sparse Histograms’, in *25th European Signal Processing Conference, EUSIPCO*. IEEE, pp. 1912–1916.

John, F. P. and Joe, L. (2005) ‘Making Better Use of Bandwidth: Data Compression and Network Management Technologies.’, *RAND ARROYO CENTER SANTA MONICA CA*, p. 52.

Kale, V. U. and Deshmukh, S. M. (2010) ‘Visually Improved Image Compression by Combining EZW Encoding with Texture Modeling Using Huffman Encoder’, *International Journal of Computer Science Issues (IJCSI)*, 7(3), pp. 10–28.

Karimi, N. *et al.* (2015) ‘Use of Symmetry in Prediction-Error Field for Lossless Compression of 3D MRI Images’, *Multimedia Tools and Applications*, 74(24), pp. 11007–11022.

Karri, C. and Jena, U. (2016) ‘Fast Vector Quantization Using a Bat Algorithm for Image Compression’, *Engineering Science and Technology, an International Journal*. Elsevier B.V., 19(2), pp. 769–781. Available at: <http://dx.doi.org/10.1016/j.jestch.2015.11.003>.

Kaur, R. and Kaur, M. (2017) ‘A Survey of Medical Image Compression Techniques’, *International Journal of Advanced Research in Computer Science*, 8(4), pp. 2015–2018.

Kavitha, S. and Anandhi, R. (2015) ‘A Survey of Image Compression Methods for Low Depth-of-Field Images and Image Sequences’, *Multimedia Tools and Applications*, 74(18), pp. 7943–7956.

Ken Peffers *et al.* (2007) ‘A Design Science Research Methodology for Information Systems Research’, *Journal of Management Information Systems*, 24(3), pp. 45–77.

Khalsa, N., Gudadhe, P. and Ingole, V. (2014) ‘Advance Image Classification System’, *International Journal of Computer Science and Information Technology*, 5(3), pp. 3210–3214.

Khan, Aftab *et al.* (2017) ‘Lossless Image Compression : Application of Bi-Level Burrows Wheeler Compression Algorithm ( BBWCA ) to 2-D data’, *Multimedia Tools and Applications*. Multimedia Tools and Applications, 76(10), pp. 12391–12416.

Kodituwakku, S. and Amarasinghe, U. (2010) ‘Comparison of Lossless Data Compression Algorithms for Text Data’, *Indian Journal of Computer Science and Engineering*, 1(4), pp. 416–425.

Kuechler, W. and Vaishnavi, V. (2012) ‘A Framework for Theory Development in Design Science Research: Multiple Perspectives’, *Journal of the Association for Information Systems*, 13(6), pp. 395–423. Available at: <http://aisel.aisnet.org/jais/vol13/iss6/3>.

Kumar, G. *et al.* (2015) ‘A Review: DWT-DCT Technique and Arithmetic-Huffman Coding based Image Compression’, *International Journal of Engineering and Manufacturing*, 5(3), pp. 20–33. Available at: <http://www.mecs-press.org/ijem/ijem-v5-n3/v5n3-3.html>.

Kumar, R., Kumar, A. and Singh, G. K. (2016) ‘Hybrid Method based on Singular Value Decomposition and Embedded Zero Tree Wavelet Technique for ECG Signal Compression’, *Computer Methods and Programs in Biomedicine*. Elsevier Ireland Ltd, 129, pp. 135–148. Available at: <http://dx.doi.org/10.1016/j.cmpb.2016.01.006>.

Kuppusamy, K. and Mehala, R. (2013) ‘Sparse Transform Matrix at Low Complexity for Color Image Compression’, *International Journal of Computer Trends and Technology (IJCTT)*, 4(6), pp. 2–7.

Lakhani, G. (2004) ‘Optimal Huffman Coding of DCT Blocks’, *IEEE Transactions on Circuits and Systems for Video Technology*, 14(4), pp. 522–527.

Li-Hui, L. and Chen, T.-J. (2017) ‘Mutual Information Correlation with Human Vision in Medical Image Compression’, *Current Medical Imaging Reviews*, 14(1), pp. 64–70. Available at: <http://www.eurekaselect.com/156137/article>.

Li, M. *et al.* (2018) ‘Efficient Trimmed Convolutional Arithmetic Encoding for Lossless Image Compression’, *arXiv preprint arXiv*. Available at: <http://arxiv.org/abs/1801.04662>.

Li, S., Li, M. and Jiang, C. (2018) ‘Semantic Enhanced Deep Learning for Image Classification’, *Concurrency and Computation*, 30(23), pp. 1–10.

Li, Z.-N., Drew, M. . and Liu, J. (2014) *Fundamentals of Multimedia*. Second Edi. Edited by D. Gries and F. Schneider. Switzerland: Springer. doi: 10.1007/978-3-319-05290-8.

Lifewire (2018) *JPEG Files*, <https://www.lifewire.com/jpg-jpeg-file-4139913>.

Lu, R. *et al.* (2014) ‘Toward Efficient and Privacy-Preserving Computing in Big Data Era’, *IEEE Network*. IEEE, 28(August), pp. 46–50.

Lucas, L. *et al.* (2017) ‘Lossless Compression of Medical Images’, *IEEE Transactions on Medical Imaging*, 36(11), pp. 2250–2260.

Lyon, R. (2006) ‘A Brief History of “Pixel”’, *International Society for Optics and Photonics, In Digital Photography II*, 6069, pp. 15–19. Available at: <http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=729181>.

Madhu and Dalal, S. (2017) ‘Review Paper on Image Compression Using Lossless and Lossy Technique’, *International Journal of Advance Research , Ideas and Innovations in Technology*, 3(2), pp. 873–878.

Maheshwari, S. *et al.* (2019) ‘Automated Glaucoma Diagnosis Using Bit-Plane S and Local Binary Pattern Techniques’, *Computers in Biology and Medicine*. Elsevier Ltd, 105, pp. 72–80. Available at: <https://doi.org/10.1016/j.combiomed.2018.11.028>.

Manjinder, K. and Gaganpreet, K. (2013) ‘A Survey of Lossless and Lossy Image Compression Techniques’, *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(2), pp. 323–326.

Masmoudi, Atef, Puech, W. and Masmoudi, Afif (2015) ‘An Improved Lossless Image Compression based Arithmetic Coding Using Mixture of Non-Parametric Distributions’, *Multimedia Tools and Applications*, 74(23), pp. 10605–10619.

- Mehanna, A. (2013) *Novel Entropy Coding and its Application of the Compression of 3D Image and Video Signals*. Doctoral dissertation, Brunel University.
- Meyer, B. and Tischer, P. (1997) 'TMW-a New Method for Lossless Image Compression', *Proc. Int. ITG FACHBERICHT*, 2, pp. 533–540.
- Mofreh, A., Barakat, T. and Refaat, A. (2016) 'A New Lossless Medical Image Compression Technique using Hybrid Prediction Model', *Signal Processing: An International Journal (SPIJ)*, 10(3), pp. 20–30.
- Motta, G., Storer, J. a. and Carpentieri, B. (2000) 'Lossless Image Coding via Adaptive Linear Prediction and Classification', *Proceedings of the IEEE*, 88(11), pp. 1790–1796. Available at: [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=892714](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=892714).
- Muntean, M., Căbulea, L. and Vălean, H. (2014) 'A New Text Clustering Method Based on KSEP', in *IEEE International Conference Journal of Software In Automation, Quality and Testing, Robotics*, pp. 1–6. Available at: <http://ojs.academypublisher.com/index.php/jsw/article/view/6197>.
- Novikov, D., Egorov, N. and Gilmutdinov, M. (2016) 'Local-Adaptive Blocks-Based Predictor for Lossless Image Compression', in *XV International Symposium 'Problems of Redundancy in Information and Control Systems' (REDUNDANCY)*. Saint-Petersburg, Russia: University of Aerospace Instrumentation, pp. 92–99.
- Nunamaker, J. F., Chen, M. and Purdin, T. D. M. (1990) 'Systems Development in Information Systems Research', *Journal of Management Information Systems*, 7(3), pp. 89–106.
- Offermann, P. *et al.* (2009) 'Outline of a Design Science Research Process', *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology, DESRIST '09*, (January), pp. 1–11.
- Oliveira, F. D. V. R. *et al.* (2013) 'CMOS Imager with Focal-Plane Analog Image Compression Combining DPCM and VQ', *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(5), pp. 1331–1344.
- Ososkov, G. and Goncharov, P. (2017) 'Shallow and Deep Learning for Image Classification', *Optical Memory and Neural Networks (Information Optics)*, 26(4), pp. 221–248.
- Otaïr, M. A. and Shehadeh, F. (2016) 'Lossy Image Compression by Rounding the Intensity Followed by Dividing (RIFD)', *Research Journal of Applied Sciences, Engineering and Technology*, 12(6), pp. 680–685. Available at: <http://maxwellsci.com/jp/mspabstract.php?jid=RJASET&doi=rjaset.12.2716>.
- Ouyang, J., Coatrieux, G. and Shu, H. (2015) 'Robust Hashing for Image Authentication Using Quaternion Discrete Fourier Transform and Log-Polar Transform', *Digital Signal Processing*. Elsevier Inc., 41, pp. 98–109. Available at: <http://dx.doi.org/10.1016/j.dsp.2015.03.006>.
- Padmavathi, K. and Thangadurai, K. (2016) 'Implementation of RGB and Grayscale Images in Plant Leaves Disease Detection - Comparative Study', *Indian Journal of Science and Technology*, 9(6), pp. 4–9.
- Pan, B., Shi, Z. and Xu, X. (2018) 'MugNet: Deep Learning for Hyperspectral Image Classification using Limited Samples', *ISPRS Journal of Photogrammetry and Remote*

- Sensing*. International Society for Photogrammetry and Remote Sensing, Inc. (ISPRS), 145, pp. 108–119. Available at: <https://doi.org/10.1016/j.isprsjprs.2017.11.003>.
- Paul, A. *et al.* (2015) ‘Iris Image Compression using Wavelets Transform Coding’, *IEEE 2nd International Conference on Signal Processing and Integrated Networks, SPIN*, (February 2015), pp. 544–548.
- Paul, S. and Kumar, S. (2003) ‘Subsethood Based Adaptive Linguistic Networks for Pattern Classification’, *IEEE Transactions on Systems, Man and Cybernetics*, 33(2), pp. 248–258.
- Poobal, S. and Ravindran, G. (2014) ‘The Performance of Fractal Image Compression on Different Imaging Modalities Using Objective Quality Measures’, *International Journal of Engineering Science and Technology (IJEST)*, 3(August), pp. 2085–2087.
- Praisline Jasmi, R., Perumal, B. and Pallikonda Rajasekaran, M. (2015) ‘Comparison of Image Compression Techniques using Huffman Coding, DWT and Fractal Algorithm’, in *IEEE International Conference on Computer Communication and Informatics, ICCCI. COMPARISON,INDIA*, pp. 1–5.
- Raghavendra, C., Sivasubramanian, S. and Kumaravel, A. (2018) ‘Improved Image Compression Using Effective Lossless Compression Technique’, *Cluster Computing*. Springer, 22(2), pp. 3911–3916. Available at: <http://link.springer.com/10.1007/s10586-018-2508-1>.
- Rusyn, B. *et al.* (2016) ‘Lossless Image Compression in the Remote Sensing Applications’, *IEEE First International Conference on Data Stream Mining & Processing 23-27*, (August), pp. 195–198.
- Sanchez, V. (2015) ‘Lossless Screen Content Coding in HEVC Based on Sample-Wise Median and Edge Prediction’, *IEEE International Conference In Image Processing (ICIP)*, pp. 4604–4608.
- Scotland, J. (2012) ‘Exploring the Philosophical Underpinnings of Research: Relating Ontology and Epistemology to the Methodology and Methods of the Scientific, Interpretive, and Critical Research Paradigms’, *English Language Teaching*, 5(9), pp. 9–16.
- Sengupta, A. and Roy, D. (2018) ‘Intellectual Property-Based Lossless Image Compression for Camera Systems’, *IEEE Consumer Electronics Magazine*, 7(1), pp. 119–124.
- Senturk, A. and Kara, R. (2016) ‘An Analysis of Image Compression Techniques in Wireless Multimedia Sensor Networks’, *Tehnicki vjesnik - Technical Gazette*, 23(6), pp. 1863–1869. Available at: <http://hrcak.srce.hr/169375>.
- Sharma, M. (2010) ‘Compression Using Huffman Coding’, *IJCSNS International Journal of Computer Science and Network Security*, 10(5), pp. 133–141.
- Shaw, L., Rahman, D. and Routray, A. (2018) ‘Highly Efficient Compression Algorithms for Multichannel EEG Laxmi’, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 26(5), pp. 957–968.
- Shinde, B. and Dani, A. (2011) ‘The Origins of Digital Image Processing & Application areas in Digital Image Processing Medical Images’, *IOSR Journal of Engineering (IOSRJEN)*, 1(1), pp. 66–71.
- Shukla, J., Alwani, M. and Tiwari, A. K. (2010) ‘A Survey on Lossless Image Compression Methods’, in *2nd International Conference on Computer Engineering and Technology*. IEEE.

Shukla, R. and Gupta, N. K. (2015) 'Image Compression through DCT and Huffman Coding Technique', *International Journal of Current Engineering and Technology*, 5(3), pp. 1942–1946.

Siddeq, M. M. and Rodrigues, M. A. (2015) 'A Novel 2D Image Compression Algorithm Based on Two Levels DWT and DCT Transforms with Enhanced Minimize-Matrix-Size Algorithm for High Resolution Structured Light 3D Surface Reconstruction', *3D Research*, 6(3), pp. 1–26.

Singh, A., Potnis, A. and Kumar, A. (2016) 'A Review on Latest Techniques of Image Compression', *International Research Journal of Engineering and Technology (IRJET)*, 3(7), pp. 262–271.

Singh, S. and Pandey, P. (2016) 'Enhanced LZW Technique for Medical Image Compression', in *3rd International Conference In Computing for Sustainable Global Development (INDIACom)*. INDIA: IEEE, pp. 1080–1084.

Singhal, V. *et al.* (2017) 'Discriminative Robust Deep Dictionary Learning for Hyperspectral Image Classification', *IEEE Transactions on Geoscience and Remote Sensing*, 55(9), pp. 5274–5283.

Solomon, C. and Breckon, T. (2011) *Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab*. First Edit, Wiley's global Scientifi. First Edit. Oxford. doi: 10.1002/9780470689776.

Sood, A., Bhathal, V. and Singh, S. (2018) 'Image Compression Techniques : A Review', *International Research Journal of Engineering and Technology (IRJET)*, 5(4), pp. 3855–3857.

Starosolski, R. (2007) 'Simple Fast and Adaptive Lossless Image Compression Algorithm', *Software - Practice and Experience*, 37(1), pp. 65–91.

Starosolski, R. (2014) 'New Simple and Efficient Color Space Transformations for Lossless Image Compression', *Journal of Visual Communication and Image Representation*, 25(5), pp. 1056–1063. doi: 10.1016/j.jvcir.2014.03.003.

Stoyanov, D., Taylor, Z. and Hutchison, D. (2018) *Deep Learning in Medical and Multimodal Learning*. Granada, Spain: Springer. doi: 10.1007/978-3-030-00889-5.

Sudhakar, R., Karthiga, M. R. and Jayaraman, S. (2005) 'Image Compression using Coding of Wavelet Coefficients – A Survey', *Icgst-Gvip*, 5(6), pp. 25–38.

Szoke, I., Lungeanu, D. and Holban, S. (2015) 'Image Compression Techniques Using Local Binary Pattern', *IEEE 13th International Symposium on Applied Machine Intelligence and Informatics*, pp. 139–143.

Tajne, A. S. and Kulkarni, P. P. S. (2015) 'A Survey on Medical Image Compression Using Hybrid Technique', *International Journal of Computer Science and Mobile Computing*, 4(2), pp. 18–23.

Talukder, K. H. and Harada, K. (2010) 'Haar Wavelet Based Approach for Image Compression and Quality Assessment of Compressed Image', *IAENG International Journal of Applied Mathematics*, 36(1), pp. 1–9. Available at: <http://arxiv.org/abs/1010.4084>.

Tecuci, G. (2012) *Advanced Neural Network Clustering Techniques for Liquid Crystal Texture Classification*, Doctoral dissertation, Kent State University).

- Tomar, R. R. S. and Jain, K. (2015) ‘Lossless Image Compression Using Differential Pulse Code Modulation and its Application’, *IEEE, Fifth International Conference on Communication Systems and Network Technologies, In Computational Intelligence and Communication Networks (CICN)*, pp. 397–400. Available at: <http://ieeexplore.ieee.org/document/7279977/>.
- Traore, B. B., Kamsu-foguem, B. and Tangara, F. (2018) ‘Deep Convolution Neural Network for Image Recognition’, *Ecological Informatics*. Elsevier, 48(September), pp. 257–268. Available at: <https://doi.org/10.1016/j.ecoinf.2018.10.002>.
- Uzair, M. *et al.* (2018) ‘Representation Learning with Deep Extreme Learning Machines for Efficient Image Set Classification’, *Neural Computing and Applications*. Springer London, 30(4), pp. 1211–1223.
- Varnan, C. S. *et al.* (2011) ‘Image Quality Assessment Techniques in Spatial’, *International Journal of Computer Science and Technology*, 2(3), pp. 177–184. Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?rep=rep1&type=pdf&doi=10.1.1.219.5535>.
- Vidal, M. and Amigo, J. M. (2012) ‘Pre-Processing of Hyperspectral Images. Essential Steps Before Image Analysis’, *Chemometrics and Intelligent Laboratory Systems*. Elsevier B.V., 117, pp. 138–148. Available at: <http://dx.doi.org/10.1016/j.chemolab.2012.05.009>.
- Vidhya, K. *et al.* (2016) ‘A Review of Lossless and Lossy Image Compression Techniques’, *International Research Journal of Engineering and Technology (IRJET)*, 3(4), pp. 616–617.
- Vijay, V., Bill, K. and Stacie, P. (2015) ‘Design Science Research in Information Systems’, in *Design research in information systems*. Boston, MA: Springer, pp. 1–66. Available at: <http://www.desrist.org/design-research-in-information-systems/>[Accessed 11 may 2017].
- Vijayaran, S. and Sakila, A. (2016) ‘Document Image Compression using Hybrid Compression Technique’, *International Journal of Engineering Science*, 6(11), pp. 3441–3445.
- Vijayvargiya, G., Silakari, S. and Pandey, R. (2013) ‘A Survey : Various Techniques of Image Compression’, *International Journal of Computer Science and Information Security (IJCSIS)*, 11(10).
- Wang, K. *et al.* (2017) ‘Cost-Effective Active Learning for Deep Image Classification’, *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12), pp. 2591–2600.
- Wang, L. *et al.* (2017) ‘Spectral–Spatial Multi-Feature-based Deep Learning for Hyperspectral Remote Sensing Image Classification’, *Soft Computing*. Springer Berlin Heidelberg, 21(1), pp. 213–221.
- Wang, Z. and Li, Q. (2011) ‘Information Content Weighting for Perceptual Image Quality Assessment’, *IEEE Transactions on Image Processing*, 20(5), pp. 1185–1198.
- Wei, Y. (2008) ‘An Introduction to Fractal Image Compression’, *National Taiwan University*, (October), p. 20. doi: 10.1016/0143-8166(91)90068-5.
- Wiseman, Y. (2015) ‘The Still Image Lossy Compression Standard - JPEG’, in *In Encyclopedia of Information Science and Technology*. Third. Encyclopedia of Information Science and Technology, pp. 295–305.
- Wozniak, M. *et al.* (2015) ‘A Multiscale Image Compressor with RBFNN and Discrete Wavelet Decomposition’, in *International Joint Conference on Neural Networks (IJCNN)*.

IEEE, pp. 1–7.

Yan-li, Z. *et al.* (2010) ‘Improved LZW Algorithm of Lossless Data Compression for WSN’, *3rd IEEE International Conference In Computer Science and Information Technology (ICCSIT)*, 4, pp. 523–527.

Yang, M. and Bourbakis, N. (2005) ‘An Overview of Lossless Digital Image Compression Techniques’, *IEEE, 48th Midwest Symposium on Circuits and Systems*, p. Vol. 2 PP.1099-1102. Available at:  
<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1594297>.

Yuan, J. *et al.* (2019) ‘Multi-Criteria Active Deep Learning for Image Classification’, *Knowledge-Based Systems*. Elsevier B.V., 172, pp. 86–94. Available at:  
<https://doi.org/10.1016/j.knosys.2019.02.013>.

Zaineldin, H., Elhosseini, M. A. and Ali, H. A. (2015) ‘Image Compression Algorithms in Wireless Multimedia Sensor Networks: A Survey’, *Ain Shams Engineering Journal*. Faculty of Engineering, Ain Shams University, 6(2), pp. 481–490. Available at:  
<http://dx.doi.org/10.1016/j.asej.2014.11.001>.

Zhang *et al.* (2015) ‘Acoustic Emission Detection of Rail Defect Based on Wavelet Transform and Shannon Entropy’, *Journal of Sound and Vibration*. Elsevier, 339, pp. 419–432. Available at: <http://dx.doi.org/10.1016/j.jsv.2014.11.021>.

## Appendix A LRCSC Compression

```
%-----  
%----- Read The Image and determine its Details -----  
%-----  
Img_Original = imread('Image_1.bmp'  
imshow(Img_Original)  
  
Img_Original_1 = Img_Original(:,:,1);  
Img_Original_2 = Img_Original(:,:,2);  
Img_Original_3 = Img_Original(:,:,3);  
  
Img_Red    = double(Img_Original_1);  
Img_Green  = double(Img_Original_2);  
Img_Blue   = double(Img_Original_3);  
  
Row_Num   = size(Img_Red,1);  
Col_Num   = size(Img_Red,2);  
%-----  
%----- Transformation Phase RGB-YUV -----  
%-----  
U = Img_Red - Img_Green;  
V = Img_Blue - Img_Green;  
  
Image_R1 = Img_Red;  
Image_G1 = U ;  
Image_B1 = V;  
%-----  
%----- Column Subtraction Phase -----  
%----- Red -----  
Image_R2 = Image_R1;  
for j=1:Col_Num - 1  
    for i=1:Row_Num  
        Image_R2(i,j) = Image_R1(i,j) - Image_R1(i,j+1);  
    end  
end  
%----- Green -----  
Image_G2 = Image_G1;  
for j=1:Col_Num - 1  
    for i=1:Row_Num  
        Image_G2(i,j) = Image_G1(i,j) - Image_G1(i,j+1);  
    end  
end  
%----- Blue -----  
Image_B2 = Image_B1 ;  
for j=1:Col_Num - 1  
    for i=1:Row_Num  
        Image_B2(i,j) = Image_B1(i,j) - Image_B1(i,j+1);  
    end  
end  
  
Image_R3 = Image_R2;  
Image_G3 = Image_G2;  
Image_B3 = Image_B2;
```



```

%-----
%----- Positive Compression -----
%----- Positive Image Red -----
-----
Bin_R    = Image_R3;
Image_R4 = Image_R3;
for i=1:Row_Num
    for j=1:Col_Num
        if Image_R3(i,j)<=0
            Bin_R(i,j) = 0;
            Image_R4(i,j)= Image_R3(i,j)*(-1);
        else
            Bin_R(i,j) = 1;
            Image_R4(i,j)= Image_R3(i,j);
        end
    end
end
end
%----- Positive Image Green -----
Bin_G    = Image_G3;
Image_G4 = Image_G3;
for i=1:Row_Num
    for j=1:Col_Num
        if Image_G3(i,j)<=0
            Bin_G(i,j) = 0;
            Image_G4(i,j)= Image_G3(i,j)*(-1);
        else
            Bin_G(i,j) = 1;
            Image_G4(i,j)= Image_G3(i,j);
        end
    end
end
end
%----- Positive Image Blue -----
Bin_B    = Image_B3;
Image_B4 = Image_B3;
for i=1:Row_Num
    for j=1:Col_Num
        if Image_B3(i,j)<=0
            Bin_B(i,j) = 0;
            Image_B4(i,j)= Image_B3(i,j)*(-1);
        else
            Bin_B(i,j) = 1;
            Image_B4(i,j)= Image_B3(i,j);
        end
    end
end
end
%-----
%----- Huffman Coding -----
%----- Red -----
[Image_R5, Image_R5_dic] = Func_My_Huff_1(Image_R4);
Image_R5_dic = Image_R5_dic(:,3);

BtC_Image_R5    = Fun_Bit_Count_PM(Image_R5);
BtC_Image_R5_dic = Fun_Bit_Count_PM(Image_R5_dic);
%----- Green -----
[Image_G5, Image_G5_dic] = Func_My_Huff_1(Image_G4);
Image_G5_dic = Image_G5_dic(:,3);

BtC_Image_G5    = Fun_Bit_Count_PM(Image_G5);
BtC_Image_G5_dic = Fun_Bit_Count_PM(Image_G5_dic);

```

```

%----- Blue -----
[Image_B5, Image_B5_dic] = Func_My_Huff_1(Image_B4);
Image_B5_dic = Image_B5_dic(:,3);

BtC_Image_B5 = Fun_Bit_Count_PM(Image_B5);
BtC_Image_B5_dic = Fun_Bit_Count_PM(Image_B5_dic);

%----- Restore the Negative Value -----
%----- Red -----
Image_R6 = Image_R5;
for i=1:Row_Num
    for j=1:Col_Num
        if Bin_R(i,j)==0
            Image_R6(i,j)= Image_R5(i,j) * (-1);
        else
            Image_R6(i,j)= Image_R5(i,j);
        end
    end
end

%----- Green -----
Image_G6 = Image_G5;
for i=1:Row_Num
    for j=1:Col_Num
        if Bin_G(i,j)==0
            Image_G6(i,j)= Image_G5(i,j) * (-1);
        else
            Image_G6(i,j)= Image_G5(i,j);
        end
    end
end

%----- Blue -----
Image_B6 = Image_B5;
for i=1:Row_Num
    for j=1:Col_Num
        if Bin_B(i,j)==0
            Image_B6(i,j)= Image_B5(i,j) * (-1);
        else
            Image_B6(i,j)= Image_B5(i,j);
        end
    end
end
Dic_1 = BtC_Image_R5_dic + BtC_Image_G5_dic + BtC_Image_B5_dic;

BtC_Image_R6 = Fun_Bit_Count_PM(Image_R6);
BtC_Image_G6 = Fun_Bit_Count_PM(Image_G6);
BtC_Image_B6 = Fun_Bit_Count_PM(Image_B6);
A3 = Dic_1 + BtC_Image_R6 + BtC_Image_G6 + BtC_Image_B6 ;

%----- RLE Compression Phase -----
[RLE_Image_R6]=My_Func_RLE(Image_R6);
[RLE_Image_G6]=My_Func_RLE(Image_G6);
[RLE_Image_B6]=My_Func_RLE(Image_B6);

[Btc_RLE_Image_R6] = Fun_Bit_Count_PM(RLE_Image_R6);
[Btc_RLE_Image_G6] = Fun_Bit_Count_PM(RLE_Image_G6);
[Btc_RLE_Image_B6] = Fun_Bit_Count_PM(RLE_Image_B6);
A3 = Dic_1 + Btc_RLE_Image_R6 + Btc_RLE_Image_G6 + Btc_RLE_Image_B6;
toc;

```

## Appendix B LRCSC De-Compression

```
%-----  
%----- Decompression -----  
%----- De_RLE -----  
tic;  
  
[De_RLE_Image_R5]=Fun_De_RLE(RLE_Image_R6);  
De_Image_R5 = reshape(De_RLE_Image_R5, [Row_Num, Col_Num]);  
  
[De_RLE_Image_G5]=Fun_De_RLE(RLE_Image_G6);  
De_Image_G5 = reshape(De_RLE_Image_G5, [Row_Num, Col_Num]);  
  
[De_RLE_Image_B5]=Fun_De_RLE(RLE_Image_B6);  
De_Image_B5 = reshape(De_RLE_Image_B5, [Row_Num, Col_Num]);  
  
%-----  
%----- Positive Value De_Compression -----  
%----- Red -----  
Bin_R1      = De_Image_R5;  
De_Image_R  = De_Image_R5;  
for i=1:Row_Num  
    for j=1:Col_Num  
        if De_Image_R5(i,j)<=0  
            Bin_R1(i,j) = 0;  
            De_Image_R(i,j)= De_Image_R5(i,j)*(-1);  
        else  
            Bin_R1(i,j) = 1;  
            De_Image_R(i,j)= De_Image_R5(i,j);  
        end  
    end  
end  
%-----  
%----- Green -----  
Bin_G1      = De_Image_G5;  
De_Image_G  = De_Image_G5;  
for i=1:Row_Num  
    for j=1:Col_Num  
        if De_Image_G5(i,j)<=0  
            Bin_G1(i,j) = 0;  
            De_Image_G(i,j)= De_Image_G5(i,j)*(-1);  
        else  
            Bin_G1(i,j) = 1;  
            De_Image_G(i,j)= De_Image_G5(i,j);  
        end  
    end  
end  
%-----  
%----- Blue -----  
Bin_B1      = De_Image_B5;  
De_Image_B  = De_Image_B5;  
for i=1:Row_Num  
    for j=1:Col_Num  
        if De_Image_B5(i,j)<=0  
            Bin_B1(i,j) = 0;  
            De_Image_B(i,j)= De_Image_B5(i,j)*(-1);  
        else  
            Bin_B1(i,j) = 1;  
            De_Image_B(i,j)= De_Image_B5(i,j);  
        end  
    end  
end  
end
```

```

%-----
%----- De_Huffman Code -----
%----- Red -----
[ DE_Huff_Image_R4 ] = Func_My_Huf_De_1( De_Image_R,Image_R5_dic );
[ DE_Huff_Image_G4 ] = Func_My_Huf_De_1( De_Image_G,Image_G5_dic );
[ DE_Huff_Image_B4 ] = Func_My_Huf_De_1( De_Image_B,Image_B5_dic );

%-----
%----- Restore the Negative Value -----
%----- Red -----
De_Image_R3 = DE_Huff_Image_R4;
for i=1:Row_Num
    for j=1:Col_Num
        if Bin_R1(i,j)==0
            De_Image_R3(i,j)= DE_Huff_Image_R4(i,j) * (-1);
        else
            De_Image_R3(i,j)= DE_Huff_Image_R4(i,j) ;
        end
    end
end

%----- Green -----
De_Image_G3 = DE_Huff_Image_G4;
for i=1:Row_Num
    for j=1:Col_Num
        if Bin_G1(i,j)==0
            De_Image_G3(i,j)= DE_Huff_Image_G4(i,j)* (-1) ;
        else
            De_Image_G3(i,j)= DE_Huff_Image_G4(i,j);
        end
    end
end

%----- Blue -----
De_Image_B3 = DE_Huff_Image_B4;
for i=1:Row_Num
    for j=1:Col_Num
        if Bin_B1(i,j)==0
            De_Image_B3(i,j)= DE_Huff_Image_B4(i,j)* (-1) ;
        else
            De_Image_B3(i,j)= DE_Huff_Image_B4(i,j);
        end
    end
end

%-----
%----- CSC Decompression Column -----
%----- Red -----
D_R2 = De_Image_R3;
for j=Col_Num:-1:2
    for i=1:Row_Num
        D_R2(i,j-1) = D_R2(i,j) + D_R2(i,j-1);
    end
end

%----- Green -----
D_G2 = De_Image_G3;
for j=Col_Num:-1:2
    for i=1:Row_Num
        D_G2(i,j-1) = D_G2(i,j) + D_G2(i,j-1);
    end
end
end

```

```

%----- Blue -----
D_B2 = De_Image_B3;
for j=Col_Num:-1:2
    for i=1:Row_Num
        D_B2(i,j-1) = D_B2(i,j) + D_B2(i,j-1);
    end
end
%----- Reverse Transformation -----
%-----
U1 = D_R2 - D_G2;
V1 = U1 + D_B2 ;

toc;
%----- Reconstruct the Image -----
%-----
De_Compressed_IMG_R = D_R2;
De_Compressed_IMG_G = U1;
De_Compressed_IMG_B = V1;
De_Comp(:, :, 1) = De_Compressed_IMG_R;
De_Comp(:, :, 2) = De_Compressed_IMG_G;
De_Comp(:, :, 3) = De_Compressed_IMG_B;
%----- End Decompression -----
%-----

```

## Appendix C Huffman Coding Function

```

%----- Huffman Coding Function -----
%-----
function [ Mat_My_Huff, Dic ] = My_Huff_1( Mat_1 )
Size_Row=size(Mat_1,1);
Size_Clm=size(Mat_1,2);

Mat_1= double(Mat_1);
Compressed_IMG_x = Mat_1;

symbols_1 = unique(Compressed_IMG_x);
s1=size(symbols_1,1);

for i=1 : s1
    UQ_Count_1 = sum(sum(Compressed_IMG_x==symbols_1(i)));
    UQ_Count_11(i,1) = UQ_Count_1;
end
%----- Sort the Matrix in descending order with respect to the first two column -----
%-----
A1 = [UQ_Count_11,symbols_1];

Dict_11= sortrows(A1,[1 2]);
Dict_12=sortrows(Dict_11,[-1 2]);

for i=1:s1
    Dict_13(i,:)= i;
end

Dict_1=[Dict_13,Dict_12];

for i=1:s1
    Mat_My_Huff(Mat_1 == Dict_1(i,3))= Dict_1(i,1);
end

```

```

end

Mat_My_Huff1 = reshape(Mat_My_Huff,[Size_Row,Size_Clm]);
Mat_My_Huff = Mat_My_Huff1;
Dic=Dict_1;

end

%-----
%----- Huffman De-Coding Function -----
%-----
function [ Img_2 ] = Func_My_Huf_De_1( Huff_1,Huff_1_Dic ,Row_Num,Cols_Num)

Huff_55 = Huff_1;
L4=length(Huff_1_Dic);

New_Row_Num = size(Huff_1,1);
New_Col_Num = size(Huff_1,2);

for i=1:L4
    Huff_1_Dic(i,2)=i;
end

for i=1:New_Row_Num
    for j=1:New_Col_Num
        for x=1:L4

            if(Huff_1(i,j) == Huff_1_Dic(x,2))
                Huff_55(i,j)=Huff_1_Dic(x,1);
            end
        end
    end
end
end
end
Img_2=double(Huff_55);

end

```

## Appendix D RLE Coding Function

```

%-----
%----- RLE Coding Function -----
%-----
function [ output_args ] = My_Func_RLE( Mat_x )

ImageArray = Mat_x(:).';
j=1;
a=length(ImageArray);
count=0;
for n=1:a
    b=ImageArray(n);
    if n==a
        count=count+1;
        c(j)=count;
        s(j)=ImageArray(n);
    elseif ImageArray(n)==ImageArray(n+1)
        count=count+1;
    elseif ImageArray(n)==b
        count=count+1;
    end
end
end

```

```

c(j)=count;
s(j)=ImageArray(n);
j=j+1;
count=0;
end
end
output_args=[c;s];

```

```
end
```

```

%-----
%----- RLE De-Coding Function -----
%-----

```

```

function [ v ] = Fun_De_RLE( input_args )
c=input_args(1,:);
s=input_args(2,:);
g=length(s);

```

```

j=1;
l=1;
for i=1:g
    v(l)=s(j);
    if c(j)~=0
        w=1+c(j)-1;
        for p=1:w
            v(l)=s(j);
            l=l+1;
        end
    end
    j=j+1;
end
ReconstructedImageArray=v;
v=v.';

```

```
end
```