

Lossless Image Compression Techniques Comparative Study

Walaa Z. Wahba¹, Ashraf Y. A. Maghari²

¹M.Sc student, Faculty of Information Technology, Islamic university of Gaza, Gaza, Palestine

²Assistant professor, Faculty of Information Technology, Islamic university of Gaza, Gaza, Palestine

Abstract- *In image compression, we can reduce the quantity of pixels used in image representation without excessively change image Visualization. Reducing image size enhance images sharing, transmitting and storing. This paper examines the performance of a set of lossless data compression algorithms which are RLE, Delta encoding and Huffman techniques on binary image, grey level images and RGB images.*

The selected algorithms are implemented and evaluated on different aspects like: compression ratio, saving storage percentage and compression time. A set of defined images are used as test bed. The performance of different algorithms are measured on the basis of different parameter and tabulated.

The results showed that delta algorithm is the best in case of compression ratio and saving storage percentage, while Huffman encoding is the best technique when evaluated by compression time.

Key Words: *compression; image; Delta; RLE; Huffman; coding; algorithm*

1. INTRODUCTION

WITH the invention of recent smart computing devices, generating, transmitting and sharing of digital images have excessively been increased. The more the small electronic devices are incorporating cameras and providing the user with technologies to share the captured images directly to the Internet, the more the storage devices are grasping the necessity of effectual storing of huge amount of image data [1].

Transmission of raw images over different types of networks required extra demand of bandwidth because images data contains more information than simple text or document files [1]. Therefore image size need to reduce before they are either stored or transmitted. Diverse studies and researches have been conducted regarding

how an image data can be best compressed apart from sacrificing the quality of the image.

Image compression methods can be classified in several ways. One of the most important criteria of classification is whether the compression algorithms remove some part of data which cannot be recovered during decompression. The algorithm which removes some part of data is called lossy data compression. And the algorithm that achieve the same what we compressed after decompression is called lossless data compression [2]. The lossy data compression algorithm is usually use when a perfect consistency with the original data is not necessary after decompression. Example of lossy data compression is compression of video or picture data. Lossless data compression is used in text file, database tables and in medical image because law of regulations. Various lossless data compression algorithm have been proposed and used. Some of main technique are Huffman Coding, Run Length Encoding, Delta encoding, Arithmetic Encoding and Dictionary Based Encoding. In this paper we examine Huffman Coding and Arithmetic Encoding and give compression between them according to their performances.

This paper examines the performance of the Run Length Encoding Algorithm (RLE), Huffman Encoding Algorithm and delta Algorithm. Performance of above listed algorithms for compressing images is evaluated and compared.

The remainder of this paper is organized as follows: Section II describes the background concepts of image compression and related work. Section III describes statement of the problem. Section IV describes used methodology. Section V describes details experiment results and discussion. Section VI describes conclusion and future work.

2. Background

2.1 IMAGE COMPRESSION

Image compression is the process intended to yield a compact representation of an image, thereby reducing the image storage and transmission requirements by reducing the amount of information required to represent a digital

image. Every image will have redundant data. Redundancy means the duplication of data in the image. Either it may be repeating pixel across the image or pattern, which is repeated more frequently in the image. The image compression occurs by taking benefit of redundant information of in the image. Reduction of redundancy provides helps to achieve a saving of storage space of an image. Image compression is achieved when one or more of these redundancies are reduced or eliminated. In image compression, three basic data redundancies can be identified and exploited. Compression is achieved by the removal of one or more of the three basic data redundancies [3].

2.1.1 Inter Pixel Redundancy

“In image neighboring pixels are not statistically independent. It is due to the correlation between the neighboring pixels of an image. This type of redundancy is called Inter-pixel redundancy. This type of redundancy is sometime also called spatial redundancy. This redundancy can be explored in several ways, one of which is by predicting a pixel value based on the values of its neighboring pixels. In order to do so, the original 2-D array of pixels is usually mapped into a different format, e.g., an array of differences between adjacent pixels. If the original image pixels can be reconstructed from the transformed data set the mapping is said to be reversible” [4].

2.1.2 Coding Redundancy

“Consists in using variable length code words selected as to match the statistics of the original source, in this case, the image itself or a processed version of its pixel values. This type of coding is always reversible and usually implemented using lookup tables (LUTs). Examples of image coding schemes that explore coding redundancy are the Huffman codes and the arithmetic coding technique”[3].

2.1.3 Psycho Visual Redundancy

Many experiments on the psycho physical aspects of human vision have proven that the human eye does not respond with equal sensitivity to all incoming visual information; some pieces of information are more important than others. Most of the image coding algorithms in use today exploit this type of redundancy, such as the Discrete Cosine Transform (DCT) based algorithm at the heart of the JPEG encoding standard [3].

2.2 Types of Compression:

Compression can be of two types: Lossless Compression, Lossy Compression.

2.2.1 Lossless Compression:

In the process compression if no data is lost and the exact replica of the original image can be retrieved by decompress the compressed image then the compression is of lossless compression type. Text compression is generally of lossless type. Lossless compression technique can be broadly categorized in to two classes [6]:

- Entropy Based Encoding: In this compression process the algorithm first counts the frequency of occurrence of each pixel in the image. Then the compression technique replaces the pixels with the algorithm generated pixel. These generated pixels are fixed for a certain pixel of the original image; and doesn't depend on the content of the image. The length of the generated pixels is variable and it varies on the frequency of the certain pixel in the original image [6].
- Dictionary Based Encoding: This encoding process is also known as substitution encoding. In this process the encoder maintain a data structure known as 'Dictionary'. This is basically a collection of string. The encoder matches the substrings chosen from the original pixel and finds it in the dictionary; if a successful match is found then the pixels is replaced by a reference to the dictionary in the encoded file [6].

2.2.2 Lossy Compression:

Lossy Compression is generally used for image, audio, video; where the compression process neglects some less important data. The exact replica of the original file can't be retrieved from the compressed file. To decompress the compressed data we can get a closer approximation of the original file [6].

2.3 DATA COMPRESSION TECHNIQUES

Various kind of image compression algorithms have been proposed till date, mainly those algorithms is lossless algorithm. This paper examines the performance of the Run Length Encoding Algorithm (RLE), Huffman Encoding Algorithm and delta Algorithm. Performance of above listed algorithms for compressing images is evaluated and compared.

2.3.1 Run length encoding (RLE):

Run length encoding (RLE) is a method that allows data compression for information in which pixels are repeated constantly. The method is based on the fact that the repeated pixel can be substituted by a number indicating how many times the pixel is repeated and the pixel itself [7]

The Run length code for a grayscale image is represented by a sequence $\{V_i, R_i\}$ where V_i is the intensity of pixel and R_i refers to the number of consecutive pixels with the intensity V_i as shown in Figure 1

65	65	65	70	70	70	70	72	72	72
{65,3}			{70,4}				{72,3}		

Figure 1: Run Length Encoding [3]

This is most useful on data that contains many such runs for example, simple graphic images such as icons, line drawings, and animations. It is not useful with files that don't have many runs as it could greatly increase the file size. Run-length encoding performs lossless image compression. Run-length encoding is used in fax machines [3].

2.3.2 Huffman Coding

The Huffman coding algorithm [8] is named after its inventor, David Huffman, who developed this algorithm as a student in a class on information theory at MIT in 1950. It is a more successful method used for text compression. Huffman's idea is to replace fixed-length codes (such as ASCII) by variable-length codes, assigning shorter codewords to the more frequently occurring symbols and thus decreasing the overall length of the data. When using variable-length codewords, it is desirable to create a (uniquely decipherable) prefix-code, avoiding the need for a separator to determine codeword boundaries. Huffman coding creates such a code [5].

The Huffman algorithm is simple and can be described in terms of creating a Huffman code tree. The procedure for building this tree is:

- Start with a list of free nodes, where each node corresponds to a symbol in the alphabet.
- Select two free nodes with the lowest weight from the list.
- Create a parent node for these two nodes selected and the weight is equal to the weight of the sum of two child nodes.
- Remove the two child nodes from the list and the parent node is added to the list of free nodes.
- Repeat the process starting from step-2 until only a single tree remains.

After building the Huffman tree, the algorithm creates a prefix code for each symbol from the alphabet simply by traversing the binary tree from the root to the node, which corresponds to the symbol. It assigns 0 for a left branch and 1 for a right branch. The algorithm presented above is

called as a semi adaptive or semi-static Huffman coding as it requires knowledge of frequencies for each pixel from image. Along with the compressed output, the Huffman tree with the Huffman codes for symbols or just the frequencies of pixels which are used to create the Huffman tree must be stored. This information is needed during the decoding process and it is placed in the header of the compressed file [5].

2.3.3 Delta encoding:

Delta encoding represents streams of compressed pixels as the difference between the current pixel and the previous pixel [9].

The first pixel in the delta encoded file is the same as the first pixel in the original image. All the following pixels in the encoded file are equal to the difference (delta) between the corresponding value in the input image, and the previous value in the input image [10].

In other words, delta encoding has increased the probability that each pixel value will be near zero, and decreased the probability that it will be far from zero. This uneven probability is just the thing that Huffman encoding needs to operate. If the original signal is not changing, or is changing in a straight line, delta encoding will result in runs of samples having the same value [10].

3. RELATED WORK:

[11] have reviewed state of the art techniques for lossless image compression. These techniques were evaluated experimentally using a suite of 45 image that repeated several application domains. Among these techniques, the researchers considered, the best compression ratios were achieved by CALIC.

[6] examines the performance of a set of lossless data compression algorithm, on different form of text data. A set of selected algorithms are implemented to evaluate the performance in compressing text data. A set of defined text file are used as test bed. The performance of different algorithms are measured on the basis of different parameter and tabulated in this article. The article is concluded by a comparison of these algorithms from different aspects.

[2] have reviewed lossless data compression methodologies and compares their performance. The refreshers have find out that arithmetic encoding methodology is very powerful over Huffman encoding methodology. In comparison they came to know that compression ratio of arithmetic encoding is better. And furthermore arithmetic encoding reduces channel bandwidth and transmission time.

[5] shows the comparison of different lossless compression algorithm over text data. This text data were available in the form of different kind of text file which

contain different text patterns. By considering the compression time, decompression time and compression ratio of all the algorithm, it can be derived that the Huffman Encoding can be considered as the most efficient algorithm among the selected ones.

In [3], the writers presents a survey of various types of lossy and lossless image compression techniques and analysis it.

This paper examines the performance of the Run Length Encoding Algorithm (RLE), Huffman Encoding Algorithm and delta Algorithm. Performance of above listed algorithms for compressing images is evaluated and compared on binary images. Grey level images and RGB images.

4. METHODOLOGY:

In order to test the performance of above mentioned compression algorithms e.g. the Run Length Encoding Algorithm, Huffman Encoding Algorithm and Delta Encoding, the algorithms were implemented and tested with a various set of images. Performances of the algorithm were evaluated by computing the compression ratio and compression time. The performances of the algorithms depend on the size of the source image and the organization of different images types (Binary image, Grey level images and RGB images) with different extensions (.png and .jpg extensions).

A chart is drawn in order to verify the relationship between the images sizes after compression, the compression time. An algorithm which gives an acceptable saving percentage with minimum time period for compression is considered as the best algorithm.

4.1 Experiment steps:

- A. Prepare test bed: show appendix A for test bed
- B. Run the program which developed using Java language to evaluate the algorithms shown in Figure 2.
- C. Evaluation

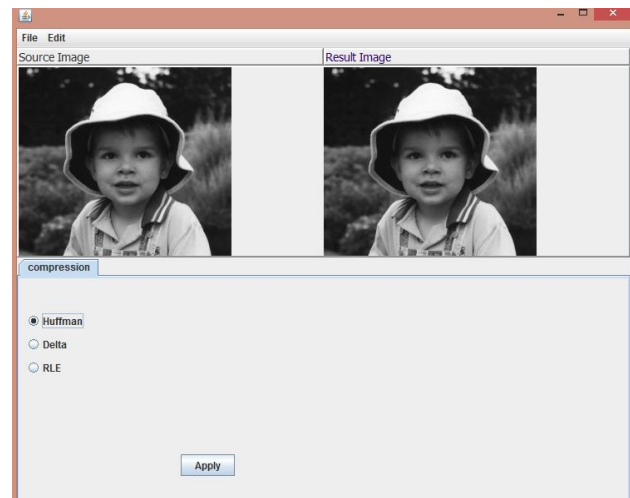


Figure 2: screenshot from used software

4.2 Measurement Parameter:

Depending on the use of the compressed file the measurement parameter can differ. Space and time efficiency are the two most important factors for a compression algorithm. Performance of the compression algorithm largely depends on the redundancy on the source data. So to generalize the test platform we used same test files for all the algorithms [6]. The parameters were as follows:

Compression Ratio: The ratio between the compressed file and the original file.

$$\text{Compression Ratio} = \frac{\text{compressed file size}}{\text{original file size}}$$

Compression Factor: The ratio between the original file and the compressed file. This is basically the inverse of the Compression Ratio.

$$\text{Compression Factor} = \frac{\text{original file size}}{\text{compressed file size}}$$

OR

$$\text{Compression Factor} = \frac{1}{\text{compression ratio}}$$

Saving Percentage: The percentage of the size reduction of the file after the compression.

$$\text{Saving Percentage} = \frac{\text{original file size} - \text{compressed file size}}{\text{original file size}} \%$$

Compression Time: The time taken by the algorithm to compress the file calculated in milliseconds (ms).

5. RESULTS AND DISCUSSION:

The result found by implementing the three algorithms (RLE, delta and Huffman) over the 7 test images are shown in tables 1,2,3, the tables show compressed images size in pixels, compression ratio, compression factor, saving percentage and compression time in ms.

5.1 Results:

5.1.1 RLE algorithm:

Table 1: RLE algorithm results

Image No.	Image type	original image size "pixels"	compressed image size "pixels"	compression ratio	compression factor	saving percentage	compression time "ms"
1	Binary	8100	550	0.07	14.73	93%	0.25
2		16384	1938	0.12	8.45	88%	0.5
3	Grey level	157960	274634	1.74	0.58	-74%	5.25
4		65536	111066	1.69	0.59	-69%	2
5	RGB	157960	928728	5.88	0.17	-488%	15.5
6		157960	928728	5.88	0.17	-488%	14.75
7		65536	239156	3.65	0.27	-265%	3.75

5.1.2 Delta Encoding algorithm:

Table 2: delta encoding results

Image No.	Image type	original image size "pixels"	compressed image size "pixels"	compression ratio	compression factor	saving percentage	compression time "ms"
1	Binary	8100	267	0.03	30.34	97%	0.5
2		16384	125	0.01	131.07	99%	0.5
3	Grey level	157960	124095	0.79	1.27	21%	15.5
4		65536	51450	0.79	1.27	21%	6.5
5	RGB	473880	344912	0.73	1.37	27%	34.5
6		473880	344912	0.73	1.37	27%	33
7		196608	113774	0.58	1.73	42%	12.5

5.1.3 Huffman Encoding:

Table 3: Huffman encoding results

Image No.	Image type	original image size "pixels"	compressed image size "pixels"	compression ratio	compression factor	saving percentage	compression time "ms"
1	Binary	8100	1225	0.15	6.61	85%	0.25
2		16384	2984	0.18	5.49	82%	0.5
3	Grey Level	157960	137025	0.87	1.15	13%	6.5
4		65536	57293	0.87	1.14	13%	2.75
5	RGB	473880	419885	0.89	1.13	11%	20.25
6		473880	419885	0.89	1.13	11%	19.75
7		196608	154124	0.78	1.28	22%	7.75

5.2 Discussion:

5.2.1 Compression ratio comparison:

Table 4: Compression ratio comparison

Image No.	RLE compression Ratio	Delta compression Ratio	Huffman compression Ratio
1	0.07	0.03	0.15
2	0.12	0.01	0.18
3	1.74	0.79	0.87
4	1.69	0.79	0.87
5	5.88	0.73	0.89
6	5.88	0.73	0.89
7	3.65	0.58	0.78

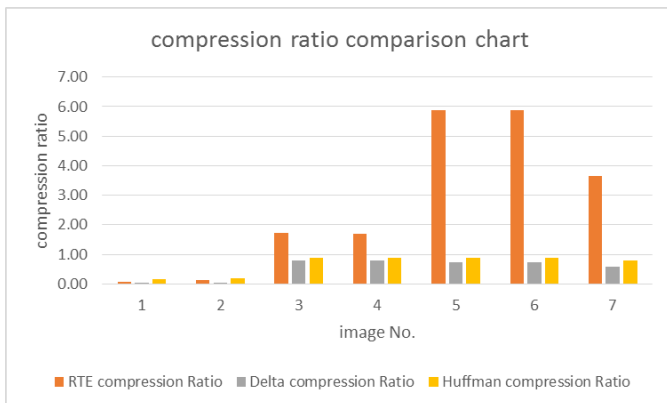


Figure 3: compression ratio comparison chart

From compression ratio comparison which presented in table 4 and shown in Figure 3, we can conclude that the Delta encoding is the best algorithm in all types of images (Binary, Grey and RGB images), which similar to the results mentioned in [12] in case of grey level images.

6 Saving percentage storage comparison:

Table 5: Saving percentage storage

image No.	RTE saving percentage	Delta saving percentage	Huffman saving percentage
1	93%	97%	85%
2	88%	99%	82%
3	-74%	21%	13%
4	-69%	21%	13%
5	-488%	27%	11%
6	-488%	27%	11%
7	-265%	42%	22%

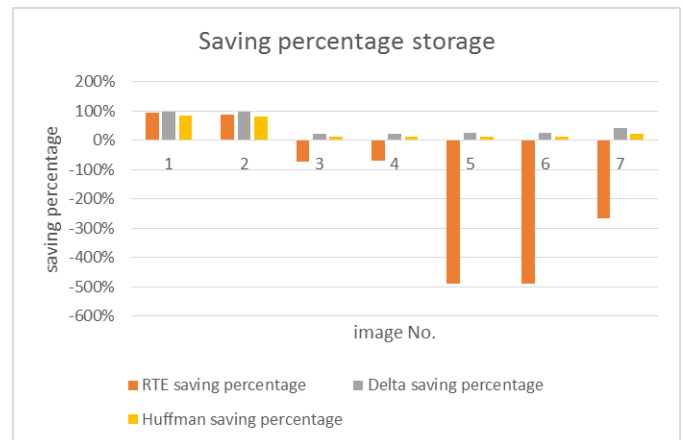


Figure 4: Saving percentage storage

From saving percentage comparison which presented in table 5 and displayed in Figure 4, we can show that:

- Delta encoding is the best compression algorithm because it has the highest percentage saving value
- RTE algorithm is not suitable for grey level images and RGB images, it just suitable for binary images because the size of compressed images is raised in grey and RGB images.

7 Compression time comparison:

Table 6: Compression time comparison

image No.	RLE compression time "ms"	Delta compression time "ms"	Huffman compression time "ms"
1	0.25	0.5	0.25
2	0.5	0.5	0.5
3	5.25	15.5	6.5
4	2	6.5	2.75
5	15.5	34.5	20.25
6	14.75	33	19.75
7	3.75	12.5	7.75

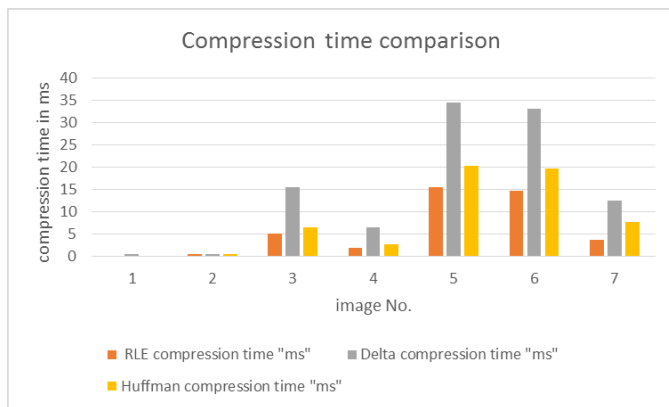


Figure 5: Compression time comparison

From compression time comparison which presented in table 6 and shown in Figure 5, we can notice that:

- In case of binary image, the compression time was very small in all techniques.
- In case of grey level and RGB images, Huffman encoding is the best technique then RLE and the delta encoding was the worst one.

5. CONCLUSION


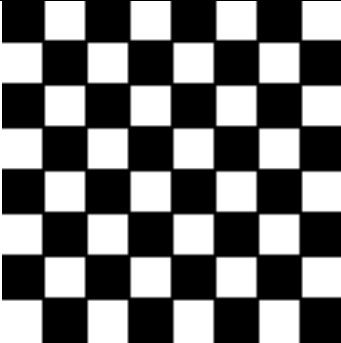


In this paper we have compared three lossless data compression algorithm RLE, Delta and Huffman encoding using our test bed which consists of variant images types and extension(binary images, grey level and RGB images) and we evaluate the algorithms using different aspects like, compression ratio, saving percentage storage and compression time. We found that Delta encoding is the best algorithm in case of compression ratio and saving percentage storage, while Huffman encoding is the best algorithm in case of compression time. In addition to that, we found that RLE is not suitable for grey level and RGB images.



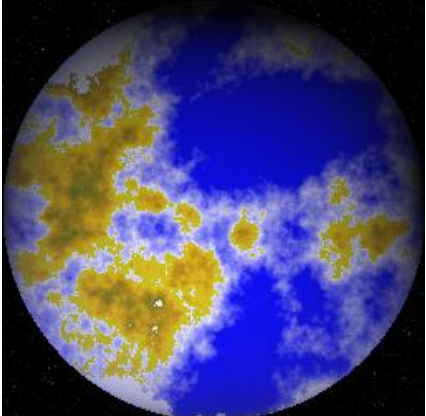
In the future, more compression techniques will used and compared over a large data set of images and video files until reach to the best compression technique.

6. REFERENCES:

- [1] M. Hasan and K. Nur, "A Lossless Image Compression Technique using Location Based Approach," vol. 1, no. 2, 2012.
- [2] S. Porwal, Y. Chaudhary, J. Joshi, and M. Jain, "Data Compression Methodologies for Lossless Data and Comparison between Algorithms," vol. 2, no. 2, pp. 142-147, 2013.
- [3] S. Gaurav Vijayvargiya and R. P. Pandey, "A Survey : Various Techniques of Image Compression," vol. 11, no. 10, 2013.
- [4] T. Europe, "An Introduction to Fractal Image Compression," *Outbro*, no. October, p. 20, 1997.
- [5] M. Grossberg, I. Gladkova, S. Gottipati, M. Rabinowitz, P. Alabi, T. George, and A. Pacheco, "A comparative study of lossless compression algorithms on multi-spectral imager data," *Proc. - 2009 Data Compression Conf. DCC 2009*, 2009.
- [6] Arup Kumar Bhattacharjee, Tanumon Bej, and Saheb Agarwal, "Comparison Study of Lossless Data Compression Algorithms for Text Data \n," *IOSR J. Comput. Eng.*, vol. 11, no. 6, pp. 15-19, 2013.
- [7] F. Semiconductor, "Using the Run Length Encoding Features on the MPC5645S," pp. 1-8, 2011.
- [8] P. O. F. T. H. E. I. R. E, "A Method for the Construction of Minimum-Redundancy Codes *,"
- [9] M. Dipperstein, "Adaptive Delta Coding Discussion and Implementation." [Online]. Available: <http://michael.dipperstein.com/delta/index.html>.
- [10] S. W. Smith, "Data Compression," *Sci. Eng. Guid. to Digit. Signal Process.*, pp. 481-502, 1997.
- [11] B. C. Vemuri, S. Sahni, F. Chen, C. Kapoor, C. Leonard, and J. Fitzsimmons, "Lossless image compression," *Igarss 2014*, vol. 45, no. 1, pp. 1-5, 2014.
- [12] N. Efford, "Digital image processing: a practical introduction using java." 2000.

Appendix A: Test bed

No.	Image	Image name	extension	Image size	Image type
1		char_a	.png	8100 pixels	Binary image
2		chess	.png	16384 pixels	Binary image
3		mattgrey	.jpg	157960 pixels	Grey level image
4		matthead	.png	65536 pixels	Grey level image

No.	Image	Image name	extension	Image size	Image type
5		matthew1	.jpg	157960 pixels	RGB
6		matthew1	.png	157960 pixels	RGB
7		frplanet	.png	65536 pixels	RGB