Australian Journal of Basic and Applied Sciences, 5(7): 32-44, 2011
ISSN 1991-8178

A Hybrid Technique for Image Compression

Hazem (Moh'd Said) Abdel Majid Hatamleh

Computer DepartmentUniversity of Al-Balqa’ Applied, Faculty of Ajloun, Jordan

Abstract: Images are used widely today at many application areas. The representation of images is space costly, so the image processing operations (e.g., storing and transmitting) are time consuming and difficult, especially when transmitting is done over the internet. This paper is targeted towards building a tool that achieves high compression rates for image files, this done by reducing the entropy of the data by using different combinations of loss-less data compression algorithms. This tool applies Huffman Encoding, LZW, and RLE algorithms on the source data in a cascading manner. The main goal of this combination is to achieve the highest average compression rate for image files in the bitmap format. The compression ratio is used as a measure for performance evaluation of the proposed tool.

Key words: Image compression, Image transmission, Image storing, and Image compression tool.

INTRODUCTION

The image compression main goal is to reduce the amount of memory space used to represent the image and its representation data, taking into consideration this large space, the image store and transmit operations are considered costly from both the time and memory space aspects.

These limitations are studied and investigated in the literature in order to reduce the number of bits that encrypts the image at different real life applications (e.g., multimedia, visual communication systems, and telecommunication network images) (Rabbani, M. and Jones, P.W., 1991) while keeping the quality of the image as consistent as possible especially during different image operations such as, storing, transmitting, and compression.

The key point following the image quality is using a good compression tool, in view of the fact that all the above image operations are based on compressing the image as a pre processing step to store or transmit the image (Pratt, 1978).

The weight of image compression tools arises with the goal of increasing the bandwidth of image data transmission by reducing the number of bits needed for presenting the image (Jerry D. Gibson *et al*. 1998). These tools have a different style of representation than using the known binary data compressing. We will show in this paper that there are some statistical properties that control the process of compressing an image (e.g., colors).

The compression tools in the literature are either used directly over the raw image data, or they are used over an already compressed image to increase the quality of the image. In later cases the probability to increase the size of the image is high comparing to the original size.

This size variant emerged as a result of using different algorithms which differ in their compression goals, for instance some existent tools are concerned in removing redundant bits in the image while other tools reduce the effects of colors by reducing the concentration of some colors and in addition some remove all colors from image and other tool use another method to do that, but all of tools concern in reducing the size of image without corrupting the quality of the image (Wallace, G.K., 1998; Shapiro, J.M., 1993; Ahmed, 1974).

Image compression algorithms are either lossy or lossless. Lossy image compression concerns in getting high compression rate by reducing the size of the image during the compression process regardless losing some representation pixels from the original image. In contrast, lossy image compression concerns in keeping the same quality of the image without losing any pixels from the original representation; regardless the compression ratio of the compression process (Gonzales, 1992).

Different using aspects (e.g., scalability, quality progressive, resolution progressive, component progressive, and region of interest coding) control choosing the type of the compression algorithm to apply in addition to the above aspects (i.e., image quality and compression-ratio),these aspects should be considered in order to

Corresponding Author: Hazem (Moh'd Said) Abdel Majid Hatamleh, Computer DepartmentUniversity of Al-Balqa' Applied, Faculty of Ajloun, Jordan
E-mail: hazim_hh@yahoo.com Tel: +962-0776-691283

have a good compression tool with hand though, this explain the diversity of different algorithms that try to meet these aspects either by playing with the data representation structures or using different quality assistance algorithms (Ahmed, 1974).

Huffman encoding, Run-Length, and LZW algorithms are investigated, studied, and used in this paper to enhance the proposed image compression approach to increase the compression ratio and the image quality, as we will explain and show each of them work and measure the compression rate for each of them in the subsections later.

The rest of the paper is organized as follows. In Section 2 we give a brief introduction to lossless compression and in section 3 about lossy compression in section four we explain the proposed method and an introduction about each of Huffman, RLE, LZW algorithms and section five contains the experimental results of proposed method .finally section six contains the conclusion and future works and section seven include the references.

2. Lossless Image Compression:

Lossless Images are those images which are able to be reconstructed so that the original image can be reproduced from decompressed image, on another word every single bit is returned as it was in the original image before compression process is done, this type of compression is used when the type of image manipulated (stored or transmitted) is very important and we don't want any change happen to the image during handling it since all details in the image if it changed cause a problems (Arps, 1994).

3. Lossy Image Compression:

Lossy image compression during the compression process reduce the information in the image by removing some pixels for ever from the creative image specifically redundant data, and when the image is reconstructed just apart of the original image is still there,lossy compression is preferably used in video where losing some details in the original file will not be reported by most users (Mamta Sharma, 2010; Tian, J. and Wells R.O., 1996).

4. Proposed Algorithm:

This research is a study of the effect of different combinations of compression algorithms on the compression rate of image files when applied in a cascading manner.

The data flow diagram of proposed method is presented in Figure 1, which shows how the compression process is done and how to choose the best sequence of compression algorithm.

The objective of this study is to find the best sequence of compression algorithms for compressing Bitmap image files. There are no similar programs written in Visual Basic (VB) make a different combination for these three algorithms (Huffman, RLE, and LZW) as this work.

The sample used for this research is made up of 1200 Bitmap Image files selected to represent all major types of images, taking into account the detail level, and the number of bits needed to represent each pixel (resolution). And here we have to note that the detail level is a different concept from resolution. The images were first categorized according to their detail level as Low-detail Images, Medium-detail Images, and High-detail Images. Then from each of these categories, we subcategorized the images According to the their resolution as 8-Bit Images,16-Bit Images,24-Bit Images,32-Bit Images. And for each of the subcategories, a sample of 100 files was selected.

Special software –written especially for this research - was used to apply all possible combinations of the three algorithms on the data sample, and then retrieve the result to spreadsheet.

4.1 Huffman Image Compression Algorithm:

The most useful compression algorithm For JPEG files is the Huffman coding algorithm. This algorithm is a lossless image compression technique, which is based on the frequency of occurrence of a pixel in the target image. The main goal of this compression algorithm is to use lower number of bits that are needed to encode the pixels that appears more frequently in the image.

In Huffman technique, a shifted-then-subtracted image is created from the targeted image, and then the Huffman histograms of the original and the created shifted images is obtained. Reversely, every bit in the compressed image is scanned sequentially to match the Huffman code, and then bits are decoded consequently. The Huffman is a lossless algorithm. In addition, it results in optimal and compact coded image. However, due to the different code lengths, the decoding process produce an overhead and consequently longer running time (Mamta Sharma, 2010).

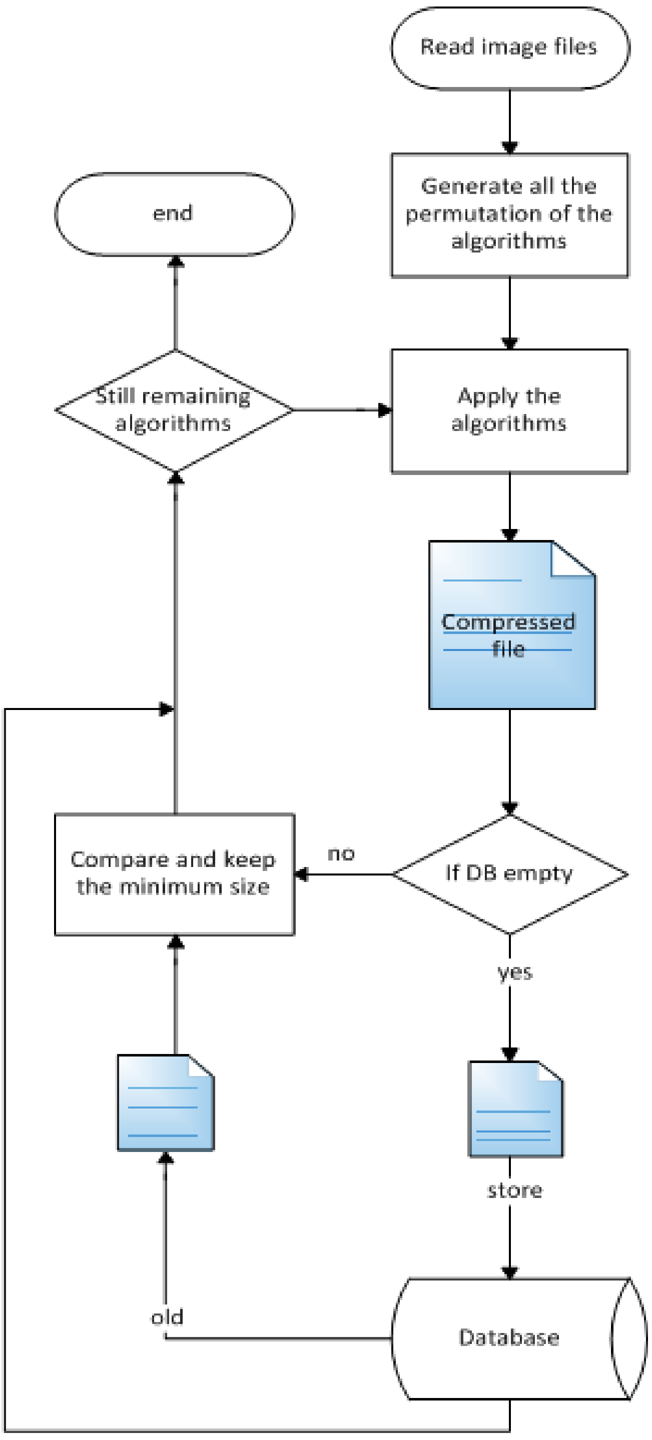
Read image files
Generate all the permutation of the algorithms
end
Still remaining algorithms
Apply the algorithms
Compressed file
Compare and keep the minimum size
no
If DB empty
yes
store
old
Database

Fig. 1: Hyprid Algorithm Flowchart

4.2 LZW Image Compression Algorithm:

One of the most common dictionary-based encoding algorithms that are used in computer graphics is the Lempel-Ziv-Welch, or LZW algorithm (Tian, J. and Wells R.O. 1996). It is used in a variety of image formats including TIFF and GIF files.

The algorithm consists of two phases, namely encoding and decoding. In encoding phase, LZW builds a data dictionary of data occurring in an uncompressed image file. As an input is read from the image, if it is already exist in the dictionary, the index number of that phrase in the dictionary is placed in the output stream of the image. Otherwise, a code phrase is created based on the data content of the input and it is stored in the dictionary. Later, when a recurring of that particular phrase is founded, the algorithm outputs the matching dictionary index instead of that phrase in the output stream.

In contrast, LZW decoding is a reverse process of encoding. In this phase, a code from the encoded image stream is examined and added to the data dictionary if it is not already there. Then, the matching data is written to the uncompressed output stream. Comparing with other dictionary based algorithms, LZW has a high compression ratio. Nevertheless, it consumes a long processing time in both of the encoding and decoding phases. Thus, LZW compression works best for files containing a lot of repetitive data (Tian, J. and Wells R.O. 1996; Ziv, J. and A. Lempel, 1978).

4.3 RLE Image Compression Algorithm:

Run-length encoding (RLE) is a data and image compression algorithm that is used with most bitmap file formats. The main features of RLE are the minimalism of implementation and execution. However, since RLE doesn’t consider the image information contents, most RLE algorithms cannot achieve the high compression ratios of the more advanced compression methods.

The RLE compression algorithm checks each pixel and identifies the equivalent object color based on a predefined convex color space (Vleuten, R.J., 2001). Consequently, consecutive pixels in a single row of the image that have the same representation value are replaced by a single pixel value and a corresponding repeating count. As a result, greater compression can be achieved in images with more symmetric and redundant consecutive pixels. On the other hand, regenerate the compressed image is a straightforward process by repeating the pixel for appropriate number of times as specified by the mentioned count. Although RLE is a simple and uncomplicated algorithm, the storage requirements for the compressed image may be significantly larger than the original image (Vleuten, R.J., 2001; Ziv, J. and Lempel, A., 1977). This negative compression takes place when groups of adjacent pixels change rapidly and less situation of pixel redundancy. Moreover, the main restriction of RLE is that it can only examine the sequential pixel redundancy.

5. Experimental Results:

In this section we ran a different combination of the proposed algorithms over an image of 1200 bitmap image files. We start measuring the compression rate over the main three algorithms (i.e., Huffman, LZW, and RLE), then we ran all the combinations of these algorithms over the same input image. Figure 2, 3 and 4 represent the compression rate average for these algorithms respectively.

5.1 Huffman Encoding:

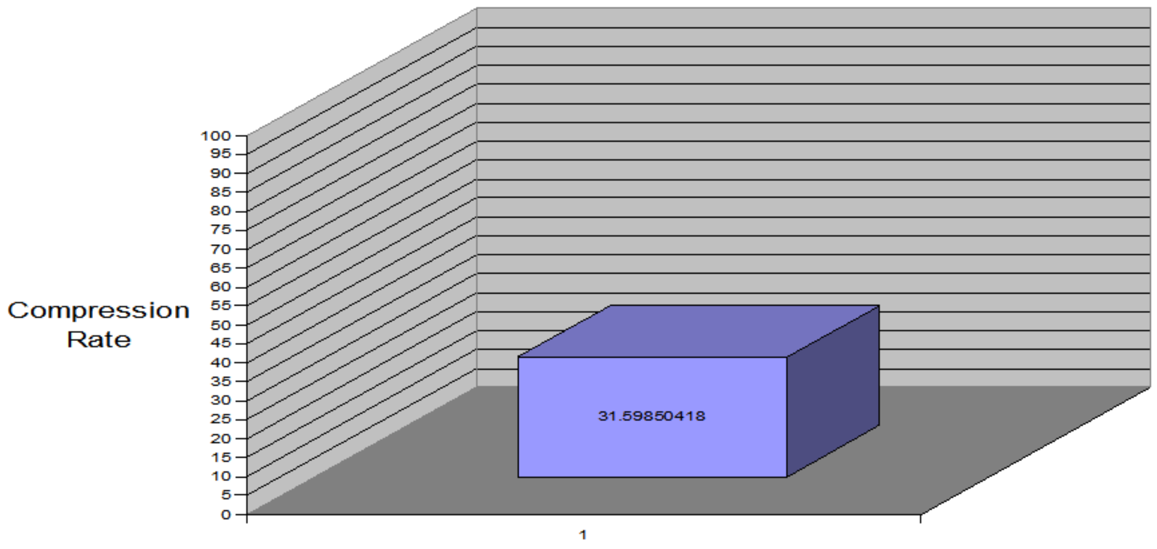

Fig. 2: Average Compression Rate of 1200 Bitmap Image Files Using Huffman Encoding

5.2 LZW Compression:

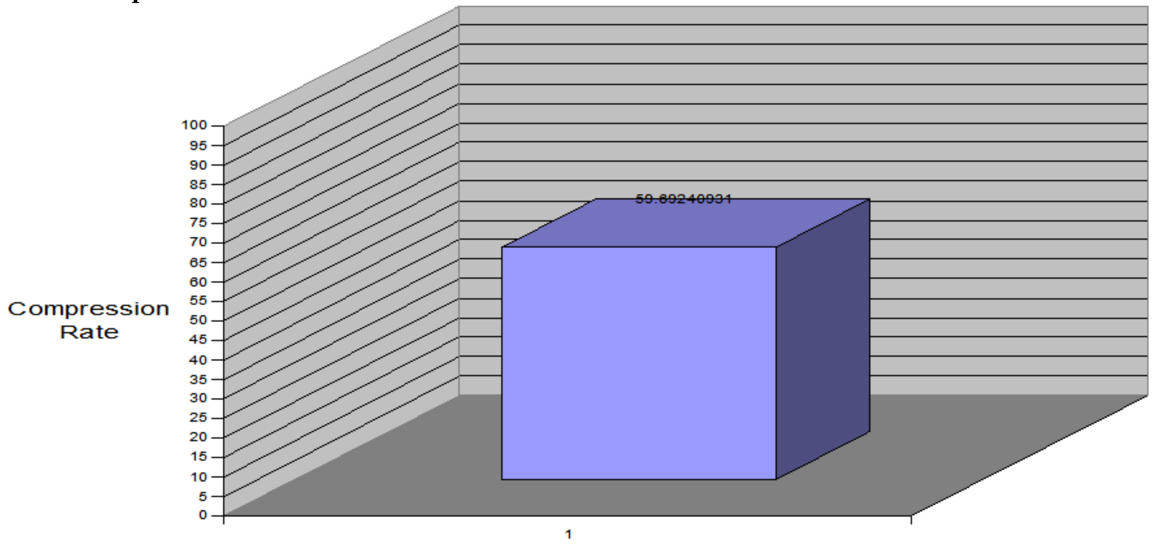

Fig. 3: Average Compression Rate of 1200 Bitmap Image Files Using LZW Encoding

5.3 RLE Encoding:

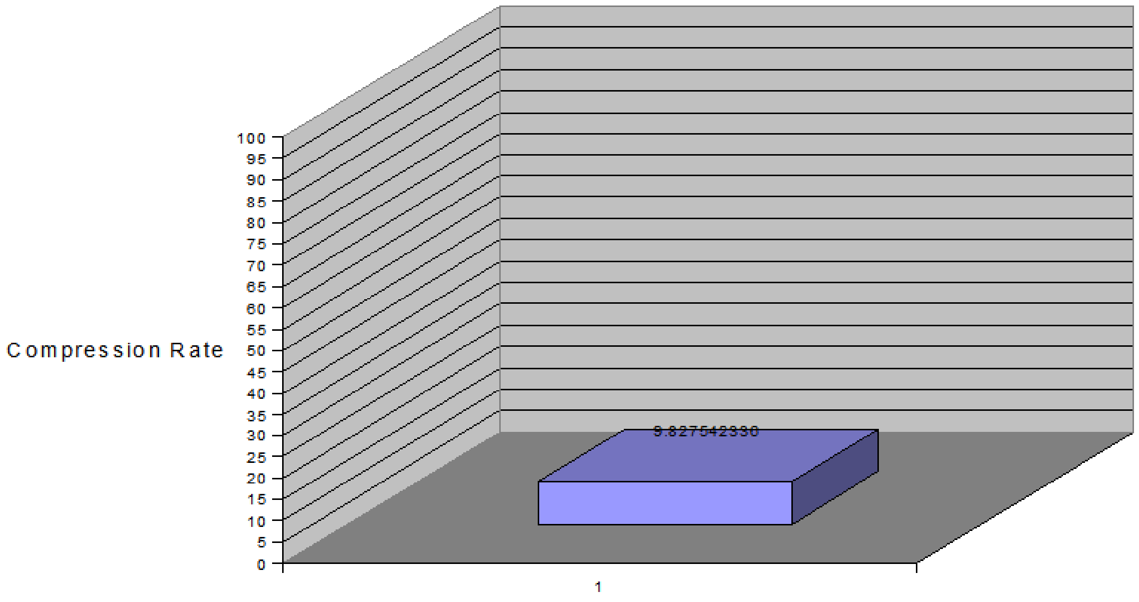

Fig. 4: Average Compression Rate of 1200 Bitmap Image Files Using RLE Encoding

5.4 Huffman-RLE Combination:

As Figure 5 shows, the average compression rate of the sample using the Huffman-RLE combination was -17.3% which is actually an increase in the image size. This outcome clearly shows that the Huffman encoding affects the order of bytes in the original image file in a scattering manner, i.e. it breaks long sequences of the same byte, thus making RLE encoding a very costly technique for the intermediate data in terms of compression rate.

5.5 LZW-RLE Combination:

Figure 6 shows a 19.7% average compression rate of the data sample, and considering the average compression rate of the data sample using the LZW algorithm by itself – which is 59.7% as Figure 2 shows – we notice that use of the RLE algorithm at the end of the sequence has reduced the average compression rate achieved by the LZW algorithm by 40%.

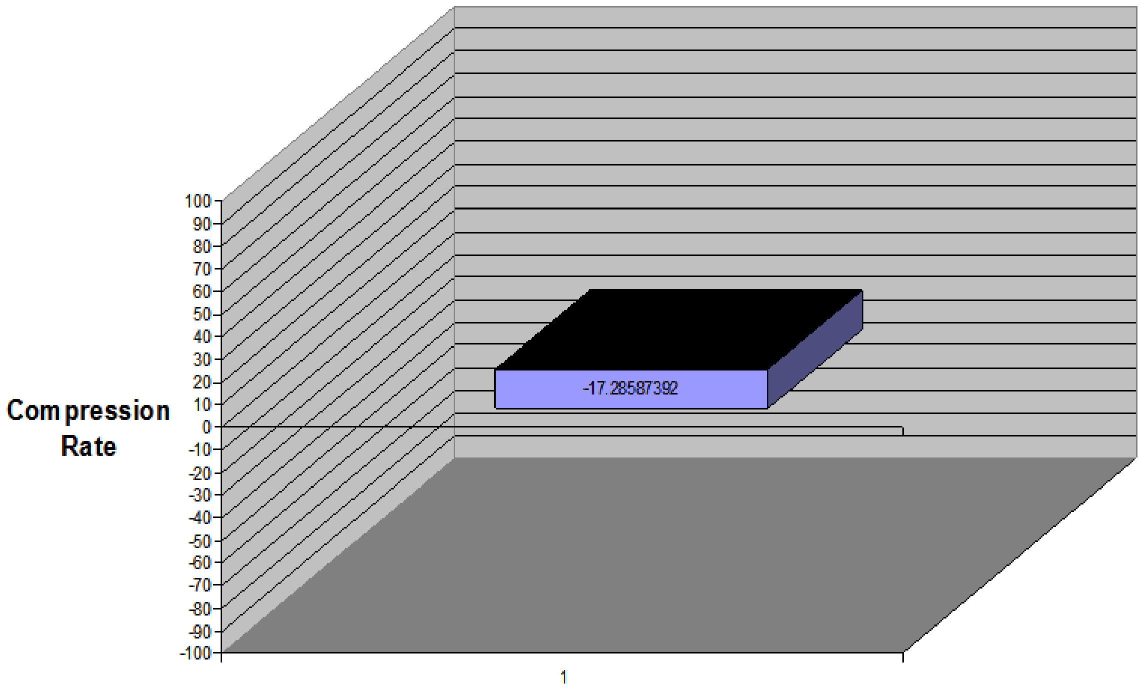

Fig. 5: The Average Compression Rate of 1200 Bitmap Image Files Using the Huffman-RLE Combination

But we also have to note that the result was an improvement to the average compression rate achieved by the RLE algorithm by itself by 10%, which means that the LZW affects the byte sequences of the original data by making those sequences longer, thus allowing a higher compression rate by the RLE algorithm.

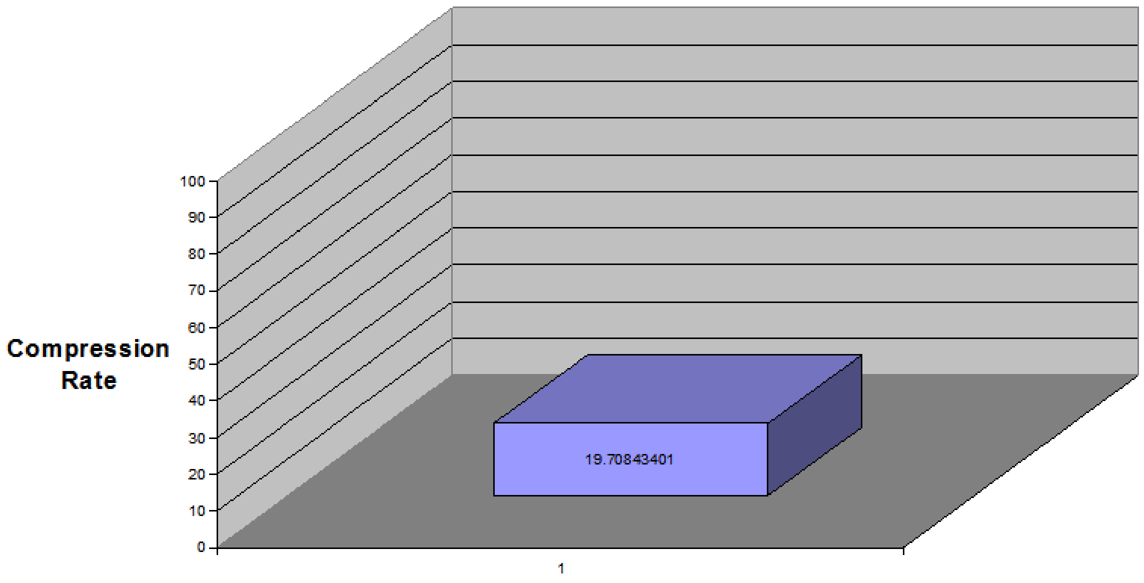

Fig. 6: The Average Compression Rate of 1200 Bitmap Image Files Using the LZW-RLE Comibination

5.6 Huffman-LZW-RLE Combination:

Figure 7 shows a -17.1% average compression rate of the data sample, and considering the average compression rate of the sample using the LZW-RLE combination – which is 19.7% - we notice a decrease in the average compression rate by 36.8% when using the Huffman encoding algorithm on the original sample data before passing the result to the LZW-RLE algorithm pair. This result shows that the Huffman encoding affects the nature of the original data by generating a large number of different byte string combinations, which is an effect which reduces the efficiency of the LZW algorithm in generating long byte sequences, and instead, it generates very short byte sequences, thus making the RLE encoding a very costly technique for the intermediate data in terms of compression rate.

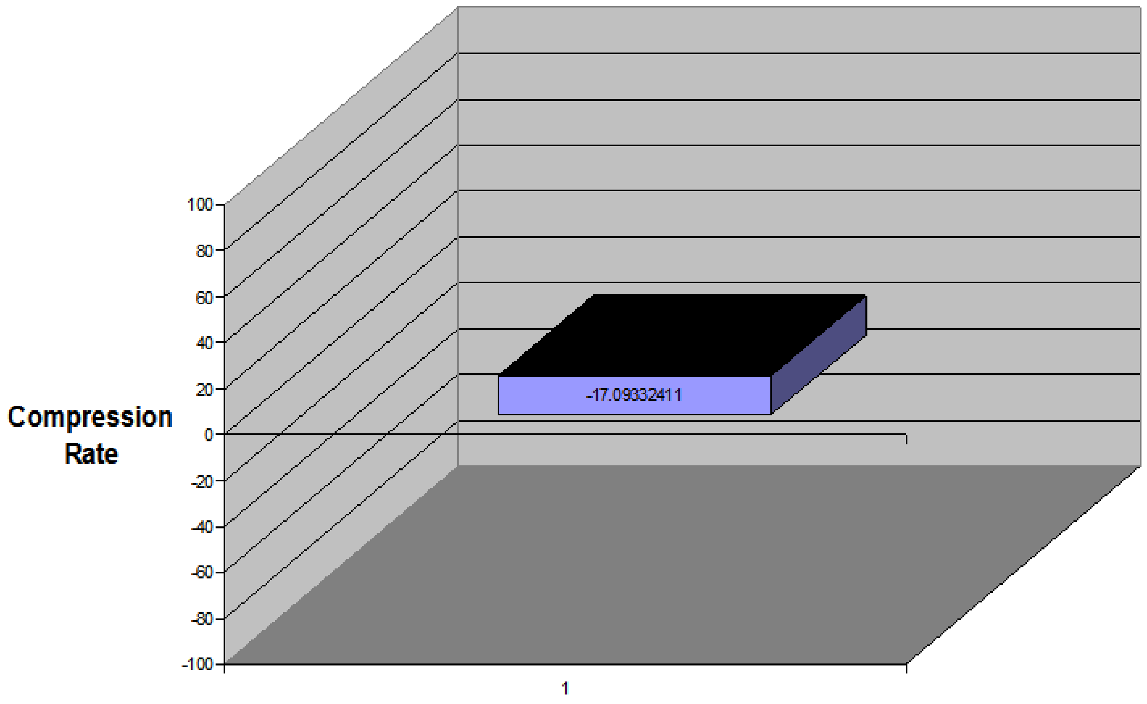

Fig. 7: Average Compression Rate of 1200 Bitmap Image Files Using the Huffman-LZW-RLE Combination

5.7 LZW-Huffman-RLE Combination:

Figure 8 shows a 22.3% average compression rate of the data sample, and considering the average compression rate of the sample using the LZW-RLE combination – which is 19.7% - we notice a slight increase in the average compression rate by 1.6% when using the Huffman encoding algorithm on the intermediate data before passing the result to the RLE algorithm. This result shows that the Huffman encoding affects the nature of the intermediate data by generating longer identical byte sequences, which increases the efficiency of the RLE algorithm in terms of compression rate.

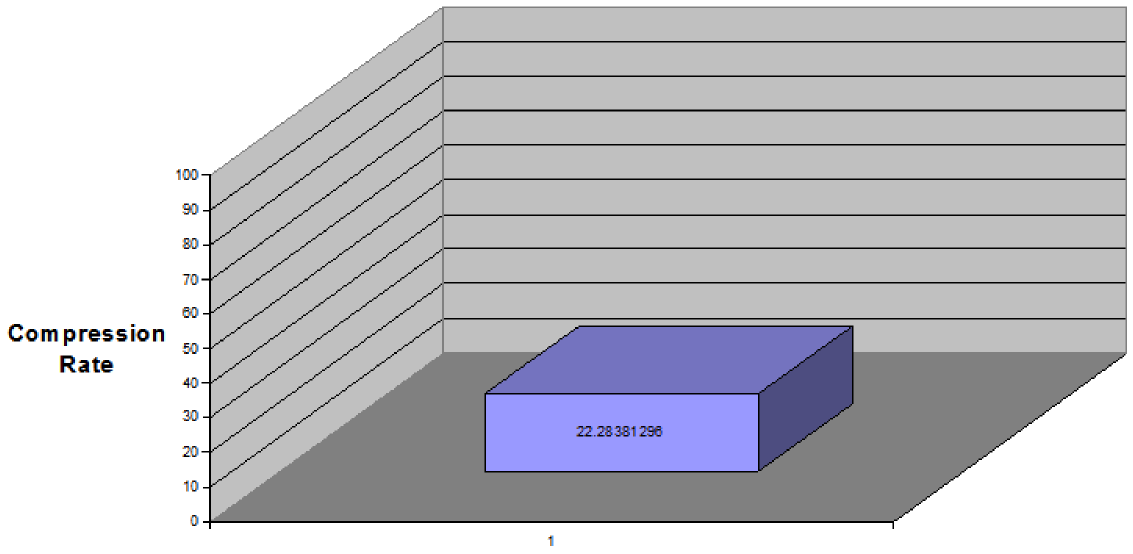

Fig. 8: Average Compression Rate of 1200 Bitmap Image Files Using the LZW-Huffman-RLE Combination

5.8 Huffman-LZW Combination:

Figure 9 show a 41.2% average compression rate of the data sample, and considering the average compression rate of the sample using the LZW and the Huffman algorithms separately, we notice a decrease in the average compression achieved by the LZW by 18% when using the Huffman encoding algorithm on the original sample data before passing the result to the LZW algorithm. This result shows that the Huffman encoding affects the nature of the original data by generating a large number of different byte string combinations, which reduces the efficiency of the LZW algorithm in terms of compression rate.

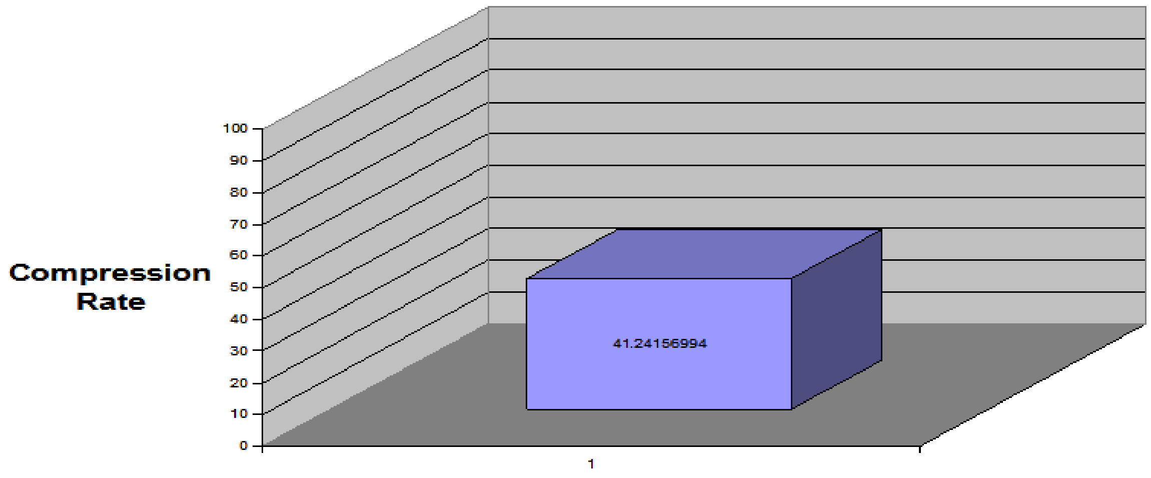

Fig. 9: Average Compression Rate of 1200 Bitmap Image Files Using the Huffman-LZW Combination

5.9 RLE-LZW Combination:

Figure 10 show a 55.9% average compression rate of the data samp

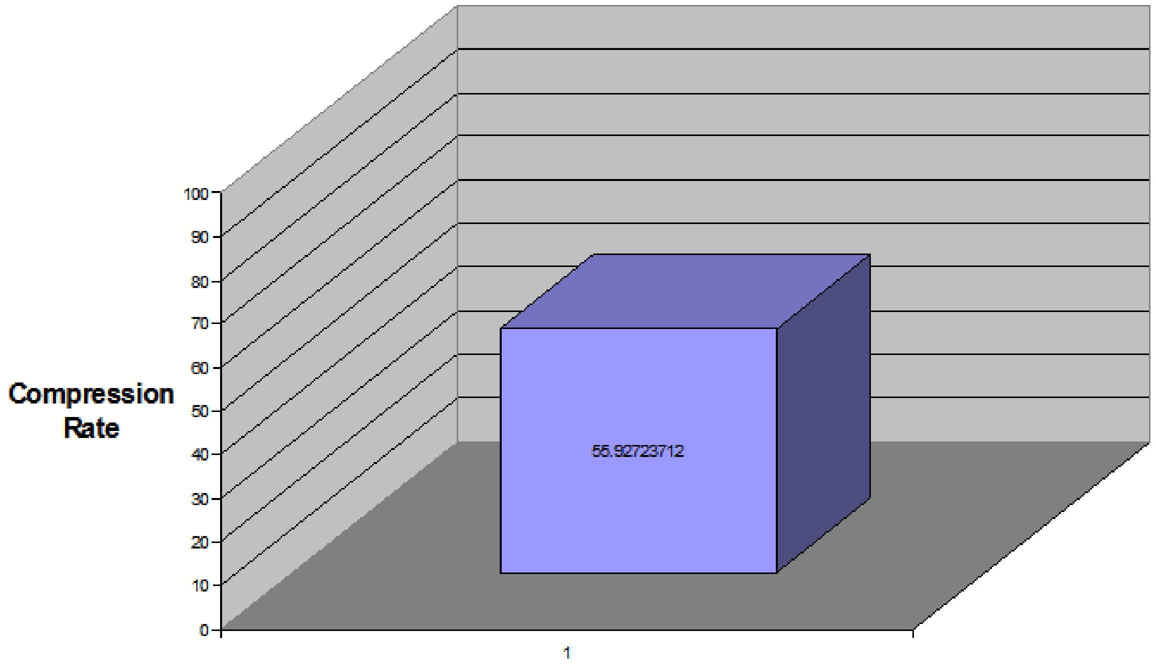

Fig. 10: Average Compression Rate of 1200 Bitmap Image Files Using the RLE-LZW Combination.

5.9 RLE-LZW Combination:

Average compression rate of the sample using the LZW algorithm by itself, we notice a decrease in the average compression achieved by the LZW by 3% when using the RLE encoding algorithm on the original sample data before passing the result to the LZW algorithm. This result shows that the RLE encoding slightly affects the nature of the original data by generating a larger number of different byte string combinations than the number of combinations existing in the original data, which slightly reduces the efficiency of the LZW algorithm in terms of compression rate.

5.10 Huffman-RLE-LZW Combination:

Figure 11 shows a 34.8% average compression. 10 Huffman-RLE-LZW Combinationonsidering the average compression rate of the sample using the LZW and the Huffman algorithms separately, we notice a decrease in the average compression achieved by the LZW algorithm by 25.2%, and a increase in the average compression rate achieved by the Huffman encoding algorithm by 3.2% when using the RLE algorithm as the second phase of the compression process. This result shows that the RLE encoding affects the nature of the intermediate data generated by the Huffman algorithm in such a way that the number of different byte combinations increases in the intermediate data before being sent to the final step which is the LZW algorithm. That effect reduces the efficiency of the LZW algorithm in terms of compression rate.

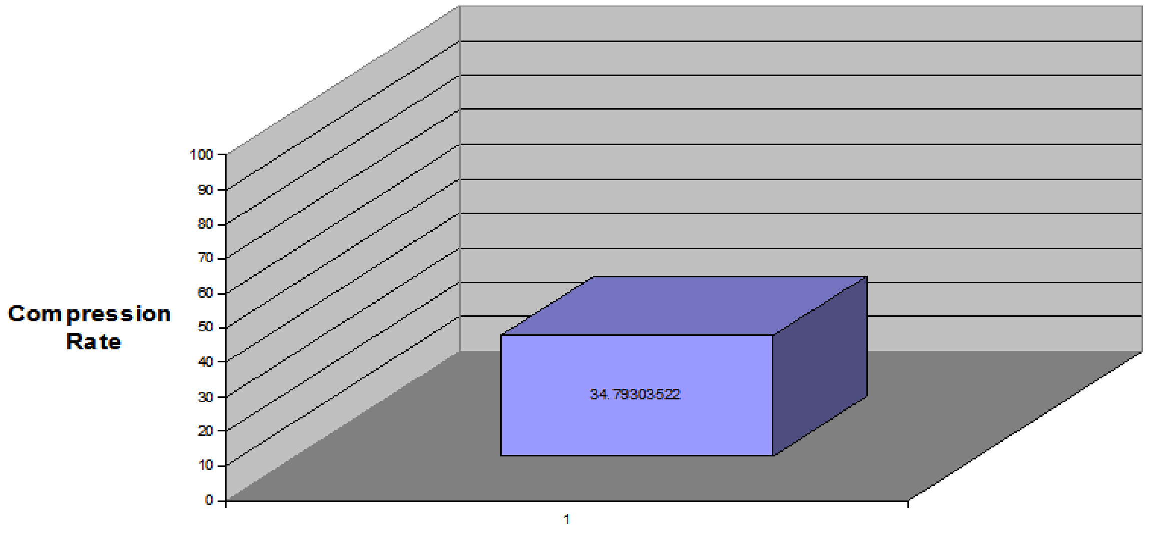

Fig. 11: Average Compression Rate of 1200 Bitmap Image Files Using the Huffman-RLE-LZW Combination

5.11 RLE-Huffman-LZW Combination:

Figure 12 show a 40.7% average compression rate of the data sample, and considering the average compression rate of applying the Huffman-LZW pair on the sample data, we notice a slight decrease of 0.5% in the average compression rate. This insignificant difference show that the RLE has a negligible effect on the probability of occurrence of bytes in the original data, thus it does not affect the entropy of the intermediate data which will be sent to the Huffman-LZW pair. We also notice that the number of different bytes combinations in the intermediate data which will be sent to the LZW algorithm does not significantly change.

5.12 LZW-Huffman Combination:

Figure 13 shows a 65.1% average compression rate of the data sample, and considering the average compression rate of the sample using the LZW and the Huffman algorithms separately, we notice an increase in the average compression achieved by the LZW algorithm by 6.1%, and a increase in the average compression rate achieved by the Huffman encoding algorithm by 33.5%. This result shows that the LZW encoding reduces the entropy of the original data sample, thus, the Huffman encoding when applied on the intermediate data will provide a much better average compression rate than if either the LZW or the Huffman algorithms were applied separately on the original data sample.

5.13 RLE-Huffman Combination:

Figure 14 shows a 45. % average compression rate of the data sample, and considering the average compression rate of the sample using the Huffman algorithm, we notice an increase in the average compression achieved by the Huffman algorithm by 13.4%. This result shows that the RLE encoding reduces the entropy of the original data sample, thus, the Huffman encoding when applied on the intermediate data will provide a much better average compression rate than if it were applied alone on the original data sample.

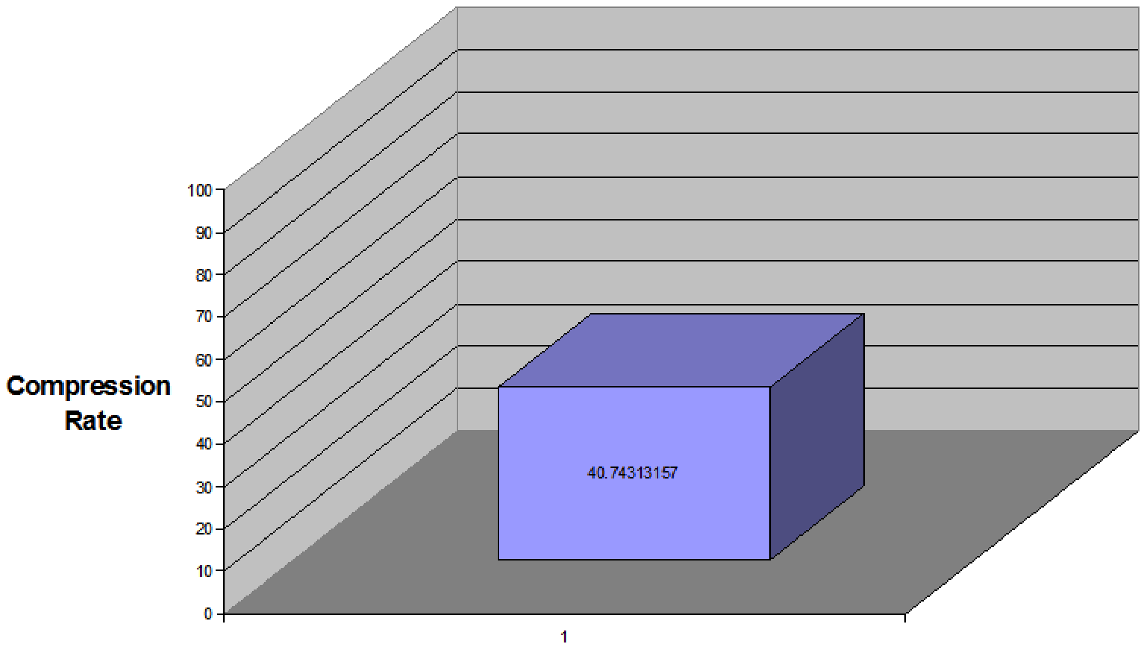

Fig. 12: Average Compression Rate of 1200 Bitmap Image Files Using the RLE-Huffman-LZW Combination

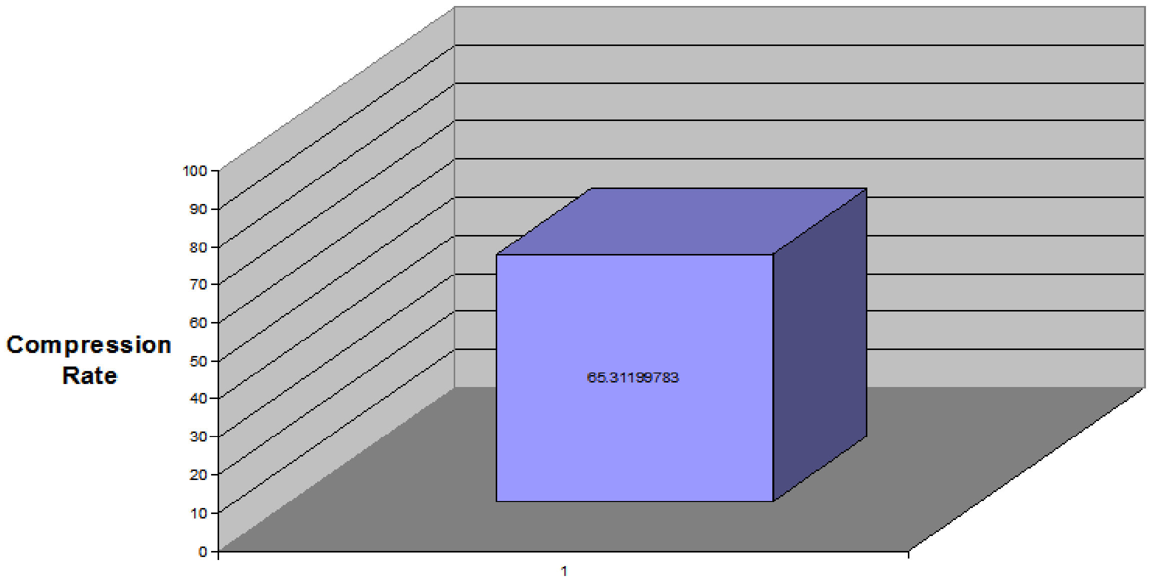

Fig. 13: Average Compression Rate of 1200 Bitmap Image Files Using the LZW-Huffman Combinations

5.14 LZW-RLE-Huffman Combination:

Figure 15 shows a 50.9% average compression rate of the data sample, and considering the average compression rate of the sample using the LZW and the Huffman algorithms separately, we notice an decrease in the average compression achieved by the LZW algorithm by 8.1%, and a increase in the average compression rate achieved by the Huffman encoding algorithm by 19.4%. This result shows that the LZW encoding reduces the entropy of the original data sample, thus, the Huffman encoding when applied on the intermediate data will provide a much better average compression rate than if either the LZW or the Huffman algorithms were applied separately on the original data sample. But we must also note that applying the RLE algorithm as the second step in the compression process reduced the efficiency of the process, and we can see that from the results of applying the LZW-Huffman combination –average compression rate of 65.1%- on the sample data.

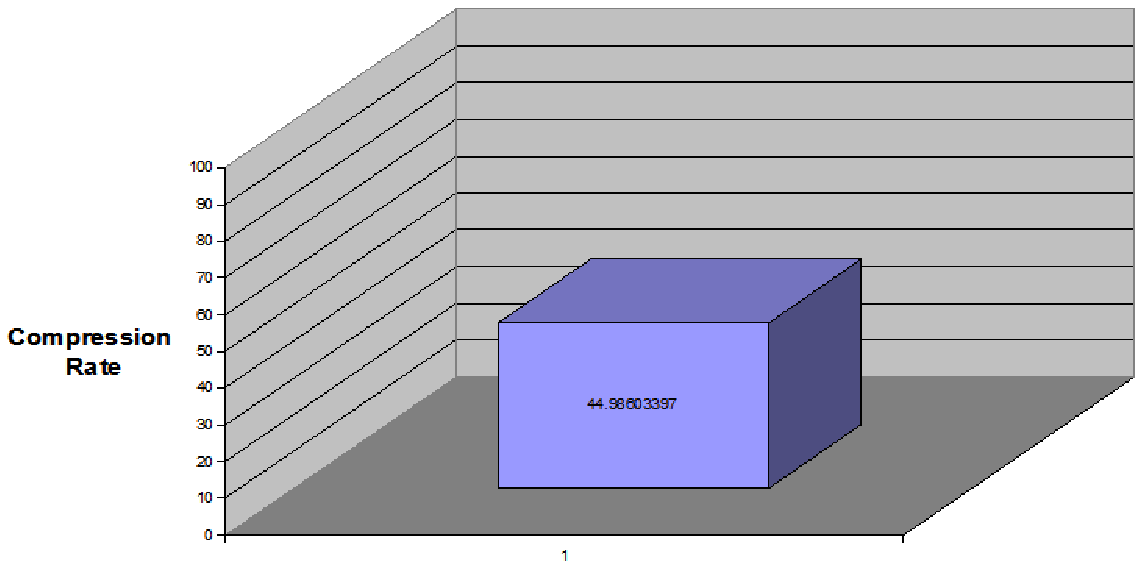

Fig. 14: Average Compression Rate of 1200 Bitmap Image Files Using the RLE-Huffman Combination

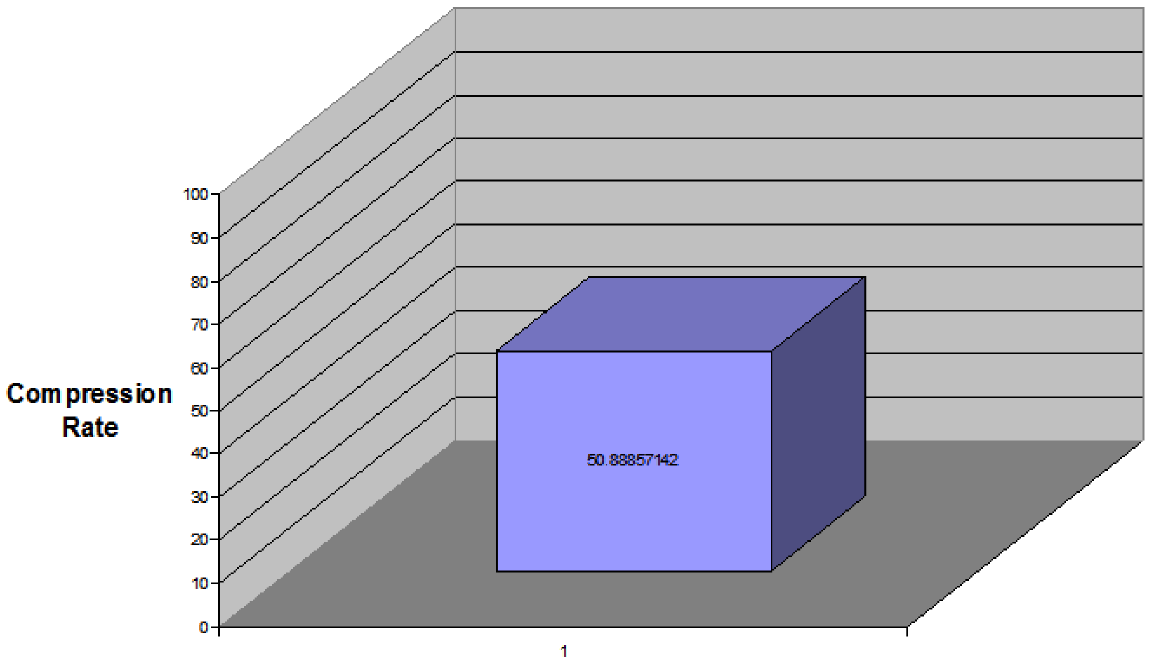

Fig. 15: 1Average Compression Rate of 1200 Bitmap Image Files Using the LZW-RLE-Huffman Combination

5.15 RLE-LZW-Huffman Combinations:

Figure 16 shows a 56.1% average compression rate of the data sample, and considering the average compression rate of the sample using the LZW and the Huffman algorithms separately, we notice an decrease in the average compression achieved by the LZW algorithm by 2.1%, and an increase in the average compression rate achieved by the Huffman encoding algorithm by 24.5%. This result shows that the LZW encoding reduces the entropy of the original data sample, thus, the Huffman encoding when applied on the intermediate data will provide a much better average compression rate than if either the LZW or the Huffman algorithms were applied separately on the original data sample. But we must also note that applying the RLE algorithm as the first step in the compression process reduced the efficiency of the process, and we can see that from the results of applying the LZW-Huffman combination –average compression rate of 65.1%- on the sample data. We must also note that the use of the RLE algorithm as the first phase of the compression process enhances the outcome of the next phase, but that enhancement does not hold for the final phase.

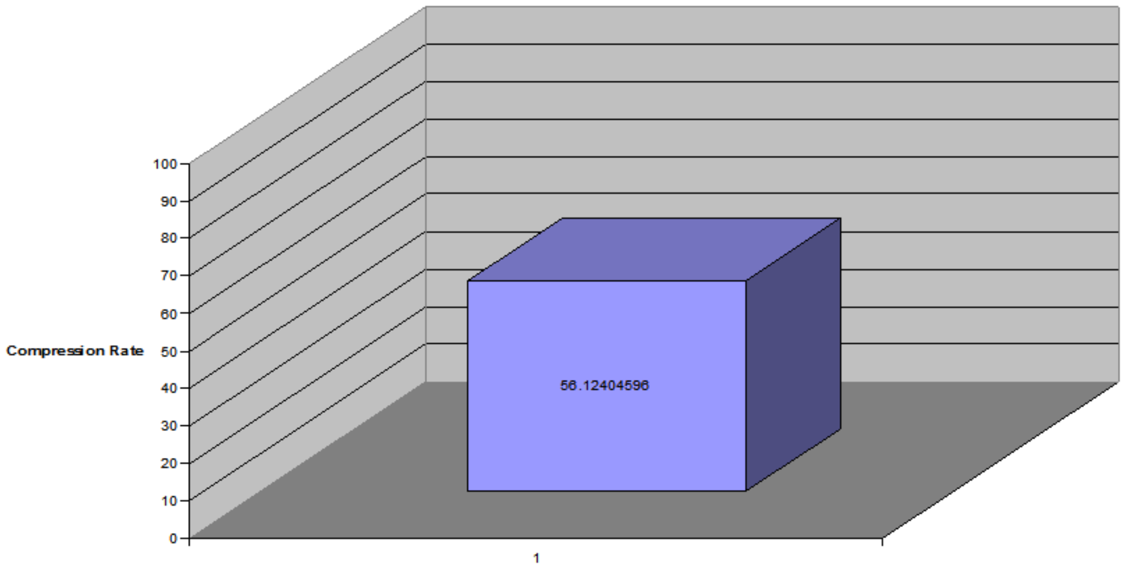

Fig. 16: Average Compression Rate of 1200 Bitmap Image Files Using the RLE-LZW-Huffman Combination

6. Conclusion and Futurework:

The following table summarizes the best and the worst average compression rate for the different compression techniques used.

Table 1: Compression Rate Comparison

File Type	Best Average	Worst Average
*.BMP	*LZW-Huffman* *(60.5%)*	*Huffman-RLE* *(–17.3%)*
*.EXE	*LZW-Huffman* *(24.1%)*	*Huffman-LZW-RLE* *(-100.6%)*
*.TXT	*LZW* *(60.76%)*	*RLE* *(-66.3%)*

From the table above we see that the best technique for compressing .BMP files would be to apply the LZW algorithm and then apply the Huffman algorithm on the result. We can also achieve very good compression rates if we apply the LZW algorithm only.

The best technique for compressing .EXE files would be to apply the LZW algorithm and then apply the Huffman algorithm on the result, and the best technique for compressing .TXT files would be to apply the LZW algorithm only. We can also achieve very good compression rates if we apply the LZW algorithm then apply the Huffman algorithm on the result.

In general, the best technique for compressing any file type is to apply the LZW algorithm first then apply the Huffman algorithm on the result and the worst compression technique for any file type would be a sequence which ends with the RLE algorithm.

In the future our concern is to study the effect of more compression algorithms on compression rate, to find the best compression ratio.

REFERENCES

Ahmed, N., T. Natarajan and K.R. Rao, 1974. On image processing and a discrete cosine transform. IEEE Transactions on Computers, C-23(1): 90-93.

Arps, R.B. and T.K. Troung, 1994. Comparison of International Standards for Lossless Still Image Compression. *Proceeding of the IEEE*, 82: 889-899.

Gonzales, R.C. and R.E. Wood 1992. Digital Image Processing. MA: Addison Wesley.

Jerry D. Gibson *et al.*, 1998. Digital Compression for Multimedia.Morgan Kaufman Publishers, California.

Mamta Sharma, 2010. Compression Using Huffman Coding. IJCSNS International Journal of Computer Science and Network Security, 10(5).

Pratt, W.K., 1978. Digital Image Processing. New York: Wiley-Interscience.

Rabbani, M. and P.W. Jones, 1991. Digital Image Compression Techniques, volume TT7 of Tutorial Texts Series. Belligham, WA: SPIE Optical Engineering Press.

Shapiro, J.M., 1993. Embedded Image Coding Using Zerotrees of Wavelet Coefficient. IEEE Trans. on Signal Processing, 41: 3445-3462.

Tian, J. and R.O. Wells, 1996. A lossy image codec based on index coding. IEEE Data Compression Conference, 456-457.

Vleuten, R.J., 2001. Low-Complexity Lossless and Fine- Granularity Scalable Near-Lossless Compression of Color Images. Data Compression Conference, (DCC '02), Snowbird, Utah, USA.

Wallace, G.K., 1998. The JPEG Still Picture Compression Standard, Communication of the ACM, 34: 30-44.

Ziv, J. and A. Lempel, 1978. Compression of Individual Sequences via Variable-Rate Coding. IEEE Transactions on Information Theory, IT-24(5): 530-536.

Ziv, J. and A. Lempel, 1977. A Universal Algorithm for Data Compression. IEEE Transactions on Information Theory, IT-23(3): 337.