Efficiency Evaluation of Huffman, Lempel-Ziv, And Run-Length Algorithms in Lossless Image Compression for Optimizing Storage and Transmission Efficiency

Selumun Agber
Department of Comp. Sc.
Benue State University

Samuel Isah Odoh
Faculty of Computing
Federal University, Lafia

Barka Piyinkir Ndahi
Department of Comp Sc.
University of Maiduguri

Onuche Gideon Atabo
Department of Comp Sc.
Kogi State Collge of Education, Ankpa

Ijeoma Rufina Godwin
Department of Comp Sc.
Benue State University

Beatrice O. Akumba
Department of Comp Sc.
Benue State University

ABSTRACT
The usage of digital data has become increasingly common today, ranging from simple text documents to complex audio and image data. As the volume of data grows, the need for efficient storage solutions becomes crucial, as smaller storage reduces costs. While human memory is the cheapest storage, it is not compatible with computer data storage needs. This study investigates lossless image compression algorithms, which enable the exact reconstruction of original images from their compressed forms. Image compression is vital for reducing storage space and expediting data transmission over the Internet. This research focuses on a comparative analysis of three prominent algorithms: Lempel-Ziv, Run-length, and Huffman compression. The performance of these algorithms is evaluated based on their compression ratios, with their respective advantages and disadvantages discussed. The findings reveal that the Huffman algorithm is the most effective for compressing JPEG, PNG, and BMP image formats. Although the Lempel-Ziv algorithm is also suitable for these formats, it is less efficient than Huffman. This study underscores the importance of selecting appropriate compression algorithms to optimize storage and transmission efficiency.

General Terms
Lossless Image Compression, Data Compression, Huffman Compression Algorithm, Lempel-Ziv Compression Algorithm, Run-length Compression Algorithm

Keywords
Image Compression Algorithms, Storage Optimization, Data Transmission Efficiency, compression ratio, image compression

1. INTRODUCTION

The usage of digital data has become increasingly common today. Digital data can range from simple text data such as document files to complicated audio data and image data. As the amount of this data increases, the need to store them in a smaller space becomes more crucial. The smaller the storage, the cheaper the cost. The cheapest storage is human memory but since humans and computers are essentially different entities, data stored in human memory cannot be used all the time. That is why digital data is usually stored in storage mediums such as hard disks, compact disks, and flash memory [1].

These storage mediums have space limitations and data is meant to be stored for a long time. Due to those reasons, data compression becomes mandatory before storing data into those storage mediums. Data is compressed to save storage space and reduce data transmission and storage. Compression reduces the need for storage mediums, which saves money, and at the same time increases the speed in reading and writing data on storage mediums. Compression on data and transmission storage will reduce the amount of time to move the data from one place to another [2].

Digital image compression is a field that studies methods for reducing the total number of bits required to represent an image. This can be achieved by eliminating various types of redundancy that exist in the pixel values. Digital images become popular for transferring visual information. There are many advantages to using these images over traditional camera film images. The digital cameras produce instant images, which can be viewed without the delay of waiting for film processing. But these images are large in size [3].

The compression technique helps to reduce the cost of storage and efficient transmission of digital images. The compression techniques are mainly classified into two. Lossy and lossless compression techniques. Lossy methods are suitable for natural images such as photographs in applications where minor (sometimes imperceptible) loss of fidelity is acceptable to achieve a substantial reduction in bit rate. The lossy compression that produces imperceptible differences may be called visually lossless. Lossless compression is preferred for archival purposes and often for medical imaging, technical drawings, clip art, comics etc. The difference is that lossless data compression retains the original data when it is decompressed, while lossy data compression only retains an approximation of the original data [4].
Hence, over the years, there exists a challenge of achieving the right compression algorithm on a particular image as well as achieving the best image quality. Thus, this paper is targeted at solving these two challenges. In addition, some image compression methods use extra memory having a size

proportional to the size of the dataset thereby consuming the available memory resources this result to error such as out of memory in execution time. This tends to affect the space efficiency of the lossless compression algorithm. Comparative analysis on Lempel-Ziv-Markov chain (also known as LZMA), Run-length encoding (also known as RLE), and Huffman encoding algorithm will enable programmers as well as individuals or organizations to apply the most effective and efficient lossless image compression algorithm on a particular problem.

2. LITERATURE REVIEW

In today's era, internet communication and information exchange, encompassing activities like sending emails and text messages via online platforms such as messaging apps, have become indispensable. When transmitting data, certain critical factors like message or file size require careful handling due to their importance. Additionally, the time taken for transmission is directly correlated with the size of the file; smaller files take less time. Compression techniques are utilized to reduce file size without compromising quality.

[5], in their study, focused on applying two lossless compression methods, namely Huffman encoding and run-length encoding, to images to enhance their suitability for information security measures like steganography and cryptography. The researchers aimed to decrease image sizes using these techniques and evaluated their performance based on parameters like compression ratio, compressed file size, and compression and decompression time. Consequently, from the standpoint of compression and decompression time, Huffman encoding proves to be less time-consuming than run-length encoding. Regarding compressed file size, Huffman encoding yields superior results. However, when considering the compression ratio parameter, run-length encoding demonstrates better outcomes.

The process of representing data with fewer bits is known as data compression. Lossless or lossy data compression is possible. Numerous techniques have been developed and implemented to carry out lossless or lossy compression. While lossy compression only permits an approximation of the original data to be generated, lossless compression makes it straightforward to reconstruct the original data from the compressed data. Data that needs to be compressed might be categorized as text, audio, picture, or even video material. Numerous studies are being conducted in the field of image compression [6].

In their investigation, [6], reviewed a number of publications in the field of data compression as well as methods for lossless picture compression. Additionally, they examined a few schemes that combine two or more schemes or a single strategy to compress an image. When the methods were used separately as opposed to in combination, the compression ratio in the suggested methodology was better than that of LZW, Huffman, and other methods. In summary, lossless image compression results in a small compression ratio yet preserves the original image quality after decompression.

The amount of image data generated in our daily lives is growing, making it more difficult to store and send. Because lossless image compression can lower the quantity of image data without sacrificing quality, it becomes significant for certain areas that require high fidelity [7]. The researchers suggested an enhanced lossless image compression algorithm that, in theory, combines linear prediction, integer wavelet transform (IWT) with output coefficient processing, and Huffman coding to provide an approximately quadruple compression in order to address the challenge of increasing the lossless image compression ratio. The primary contribution of their technique is a new hybrid transform that takes advantage of a new prediction template and an IWT coefficient processing. The suggested approach works better than state-of-the-art algorithms, according on the testing results on three distinct image sets. Up to 72.36%, the compression ratios are increased by at least 6.22%. At a reasonable compression speed, their approach is better suited for compressing photos with intricate textures and higher resolutions.

In the research carried out by [8], titled "Image Compression Using Run Length Encoding and Lempel Ziev Welch Method," involved utilizing various input images with diverse orientations and data sets. They assessed the compression achieved by each algorithm on every image and compared the results based on compression ratio. They also made enhancements to the conventional Run Length Encoding (RLE) algorithm to improve its efficiency, resulting in what they termed Optimized RLE. It was observed that Optimized RLE outperformed standard RLE notably, particularly when dealing with portrait images. While Lempel Ziv Welch (LZW) demonstrated superior compression compared to RLE, the execution time for the LZW algorithm was considerably longer.

[9], in their reviewed evaluated various lossless compression algorithms in terms of their effectiveness for high-resolution image compression. The Discrete Wavelet Transform (DWT) algorithm was highlighted for producing lossless images with decent compression ratios across different image types, especially high-resolution ones, while maintaining simple execution. The Discrete Cosine Transform (DCT) algorithm also demonstrates satisfactory compression but may encounter complexity issues, especially with larger high-resolution image sets. Huffman and Run Length Encoding (RLE) algorithms offer higher compression ratios but lack stability and reliability, particularly with increasing image resolution. The Lempel Ziv Welch (LZW) algorithm, though producing binary images, requires modifications for lossless compression and color image generation. Hence, the study concluded that the DWT algorithm stands out for its simplicity and effectiveness in lossless compression of high-resolution images, while other algorithms may require modifications to achieve similar results without complexity.

[10], presented the performance evaluation of the four lossless compression algorithms. The accelerometer data is used as an input to the four lossless algorithms such as Delta encoding, Run Length, Huffman and Lempel-Ziv. Additionally, the accelerometer data is used to calculate a number of measurement parameters, including the compression ratio, space saving, compression factor, compression gain, and elapsed time. According to the performance study, Lempel Ziv compression offers a better compression ratio of about 4:1, which reduces redundant data by removing statistical redundancy. Thus, by lowering the data redundancy, the Lempel-Ziv method's efficiency is increased. Additionally, the Lempel ziv compression approach can save 76.9% of the available space. However, the Lempel-Ziv method is a sophisticated algorithm that requires a significant amount of compression time during code execution. In contrast to Run Length and Huffman, the Delta encoding provides somewhat better measuring parameters.

3. METHODOLOGY

This section outlines the approaches adopted and the tools used in the comparative analysis of Lempel Ziv- Welch, Huffman encoding and Run-length Algorithms.

3.1 Lempel Ziv-Welch Algorithm

A brief implementation of Lempel Ziv-Welch has been shown below in tables 1

Table 1. Algorithm for Lempel Ziv-Welch

```
*   PSEUDOCODE
 1   Initialize table with single character
strings
 2   P = first input character
 3   WHILE not end of input stream
 4       C = next input character
 5       IF P + C is in the string table
 6         P = P + C
 7       ELSE
 8         output the code for P
 9       add P + C to the string table
 10        P = C
 11      END WHILE
 12   output code for P
```

3.2 Performance Analysis of Lempel-Ziv Compression Algorithm

Lempel Ziv's algorithm is a dictionary-based compression algorithm that maintains an explicit dictionary. This dictionary has to be built both at the encoding and decoding side and they must follow the same rules to ensure that they use an identical dictionary. LZ78 algorithm has the ability to capture patterns and hold them indefinitely but it also has a serious drawback. The dictionary keeps growing forever without bound. There are various methods to limit dictionary size; the easiest way is to stop adding entries and continue like a static dictionary coder or to throw the dictionary away and start from scratch after a certain number of entries has been reached. LZW is a general compression algorithm capable of working on almost any type of data. LZW compression creates a table of strings commonly occurring in the data being compressed, and replaces the actual data with references into the table. The table is formed during compression at the same time which the data is encoded and during decompression at the same time as the data decoded. The Lempel-Ziv-Welch (LZW) compression algorithm is widely used because it achieves an excellent compromise between compression performance and speed.

3.3 Huffman Encoding Algorithm

A brief implementation of the Huffman encoding algorithm has been shown below in table 2

Table 2. Algorithm for Huffman Encoding

1. Create a leaf node for each symbol and add it to the priority queue.
2. While there is more than one node in the queue:
    1. Remove the node of highest priority (lowest probability) twice to get two nodes.
    2. Create a new internal node with these two nodes as children and with probability equal to the sum of the two nodes' probabilities.
    3. Add the new node to the queue.
3. The remaining node is the root node and the tree is complete.

3.4 Performance analysis of Huffman Compression Algorithm

The technique works by creating a binary tree of nodes. These can be stored in a regular array, the size of which depends on the number of symbols, *n*. A node can be either a leaf node or an internal node. Initially, all nodes are leaf nodes, which contain the **symbol** itself, the **weight** (frequency of appearance) of the symbol and optionally, a link to a **parent** node which makes it easy to read the code (in reverse) starting from a leaf node. Internal nodes contain symbol **weight**, links to **two child nodes** and the optional link to a **parent** node. As a common convention, bit '0' represents following the left child and bit '1' represents following the right child. A finished tree has up to n leaf nodes and n-1 internal nodes. A Huffman tree that omits unused symbols produces the most of optimal code lengths. The process essentially begins with the leaf nodes containing the probabilities of the symbol they represent, and then a new node whose children are the 2 nodes with probability is created such that the new node's probability is equal to the sum of the children's probability. With the previous 2 nodes merged into one node (thus not considering them anymore), and with the new node being now considered, the procedure is repeated until only one node remains, the Huffman tree. Huffman`s procedure creates the optimal code for a set of symbols and probabilities' subject to the constraints that the symbols be coded one at a time After the code has been created coding or Decoding. Is accomplished in a simple look up table manner. The code itself is an instantaneous uniquely decodable block code. It is called a block code because each source symbol is mapped into a fixed sequence of code symbols.

Apart from the knowledge base that is a major characteristic of any expert system for storing rules and facts about the object, the proposed system makes use of a conventional database table. This table serve for the of storing information about patients whose health status has been diagnosed and the test result thereof, this can be useful in further researches, for example one may decide to query out from the database the number of patients with mouth diseases to assess the rate of the pandemic, or to use the previous results of patients as Case base reasoning for further diagnosis or treatment.

3.5 Run-length encoding Algorithm

A brief implementation of the run-length Algorithm has been shown below in table 3

Table 3. Run-length Algorithm

1. Pick the first character from source string.
2. Append the picked character to the destination string.
3. Count the number of subsequent occurrences of the picked character and append the count to destination string.
4. Pick the next character and repeat steps **2 3** and **4** if end of string is NOT reached.

3.6 Performance analysis of Run-length Compression Algorithm

Run Length Encoding (RLE) is a simple and popular data compression algorithm. It is based on the idea to replace a long sequence of the same symbol by a shorter sequence and is a good introduction into the data compression field for newcomers. The RLE algorithm performs a lossless compression of input data based on sequence of identical values (runs). In this algorithm is represents explicitly by a pair (v, l) where v is the value and l is the length of the value. The basic problem that degrades the performance of run length encoding technique is sometimes a data may contain a very large sequence of consecutive ones or zeros. In such sequences as the largest sequence of consecutive ones/zeros decides the number of bits to represent the length of the run. As a result, the length of the run in all other sequences is also represented by the same number of bits. This in turn increases the size of memory stack and decreases the transmission speed of data.

3.7 Tool Used

MATLAB: Also known as Matrix laboratory is a multipurpose application used in programming, performing mathematical operations and statistical analysis; was used to implement the lossless compression algorithm, plot the graph of the compression ratios obtained from the application to illustrate the comparative analysis of Lempel-Ziv, Huffman Encoding, and Run length algorithms.

3.8 Approach Used

The MATLAB software was used to import images of different extensions (i.e. JPG, BMP, and PNG) from the system in order to analyze the compression algorithm. The imported image (i.e. the original image) was then converted into byte and the system stores the value obtained from the image, the selected algorithms were then implemented on the original images which compressed the original images and the result of the compressed image was also stored. After executing the algorithms, the size of both the original and the compressed image was collected on all image formats (i.e. JPG, BMP, and PNG). The compression ratio was obtained by dividing the size of the compressed image by the size of the original image. And the result of the compression ratio was used to plot a corresponding graph for each of the afore-mentioned image formats.

4. RESULTS

This section present results and discussion obtained from the algorithms explained in chapter three. It also presents the graphical comparison of the lossless image compression algorithms. In the previous section, we have discussed the implementation of the lossless compression algorithms considered for this study. In this chapter we have presented practical results of the experiments. According to the theoretical study, we have conducted and compared the size efficiency of the various lossless algorithms on all the different image formats, and also a graphical comparison is presented.
The simulated experiments were carried out five times (on different image) for each file format. This is because of the variable conditions that may affect the running of the experiment. The size of all image file formats of all algorithms was then taken for evaluation. Evaluating the geometric mean of the various simulation of each image file format makes it easier for any algorithm to be rated high or low on that particular file format, rather than evaluating the algorithm on the score of just one experimental run. Thereby making the overall rating a better indication of the performance.

The table below shows the result obtained from the Joint Photographic Experts Group (JPG) for all three algorithms with their various compression ratios. And their geometric mean was also taken for all the five images that were collected.

Table 4. Simulation Results for JPG Image

Image Compressed	**Run Original Size**	**Run Compressed Size**	**LZW Original Size**	**LZW Compressed Size**	**HUFF Original Size**	**HUFF Compressed Size**
A	235591	2767	20615	7447	85100	6120
M	332266	4159	33488	12737	91600	6120
APC	2588904	6402	240000	25905	113600	6120
Anthony	447405	7024	45000	18897	75300	6120
Beauty	394587	18805	50432	30957	113100	6120

From the simulated result obtained on table 4 for the .JPG images, for each of the algorithms, the respective compression ratio is as shown on table 5.

Table 5. Compression Ratios of the Simulated Result for JPG

Image	**Run-Length**	**LZW**	**HUFF**
A	0.0117	0.3612	0.0719
M	0.0125	0.3808	0.0668
APC	0.0025	0.1079	0.0539
Anthony	0.0157	0.4199	0.0813
Beauty	0.0477	0.6138	0.0541
Average	**0.01802**	**0.3767**	**0.0656**

From the result obtained in table 5, below is a bar graph showing the compression ratios of all the algorithms (i.e. Huffman, Lempel-Ziv, and Run-length) on figure 1.

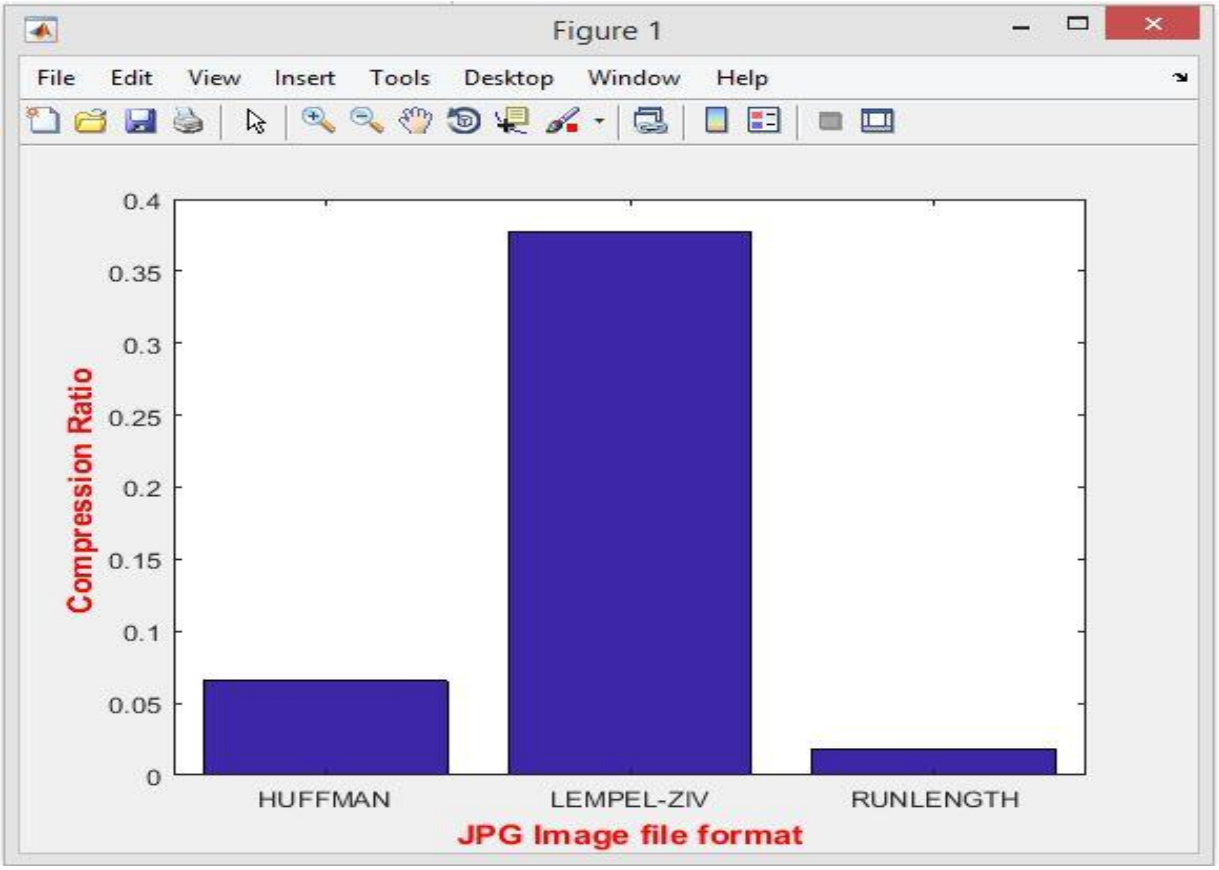

Figure 1. Graph of JPG image Compression ratio analysis

The graph on figure 1 explains the compression ratio of the various algorithms on JPG image file format. The result on the also shows that the Lempel-Ziv algorithm compresses JPG images more followed by the Huffman algorithm and then the Run-length algorithm.

The table 6 shows the result obtained from the portable network graphics **(**PNG**),** for all three algorithms with their various compression ratios. And their geometric mean was also taken for all the five images that were collected.

Table 6: Simulation Results for PNG Image

Image	**Run Original Size**	**Run Compressed Size**	**LZW Original Size**	**LZW Compressed Size**	**HUFF Original Size**	**HUFF Compressed Size**
B	125964	4415	61504	12011	176800	2040
AB	426912	4965	38950	15465	73200	6120
Revolution	1273359	15198	178848	36681	87700	6120
Apps	1904414	4939	160000	18127	112500	6120
PDP	102868	3989	50176	10460	176800	2040

From the simulated result obtained on table 6 for the .PNG images, for each of the algorithms, the respective compression ratio is as shown on table 7.

Table 7. Compression Ratios of the Simulated Result for PNG

Image	**Run-Length**	**LZW**	**HUFF**
B	0.0350	0.1953	0.0115
AB	0.0116	0.3970	0.0836

Revolution	0.0119	0.2051	0.0698
Apps	0.0026	0.1133	0.0544
PDP	0.0388	0.2085	0.0115

Average	**0.0120**	**0.2238**	**0.04616**

From the result obtained in table 7, below is a bar graph showing the compression ratios of all the algorithms (i.e. Huffman, Lempel-Ziv, and Run-length).

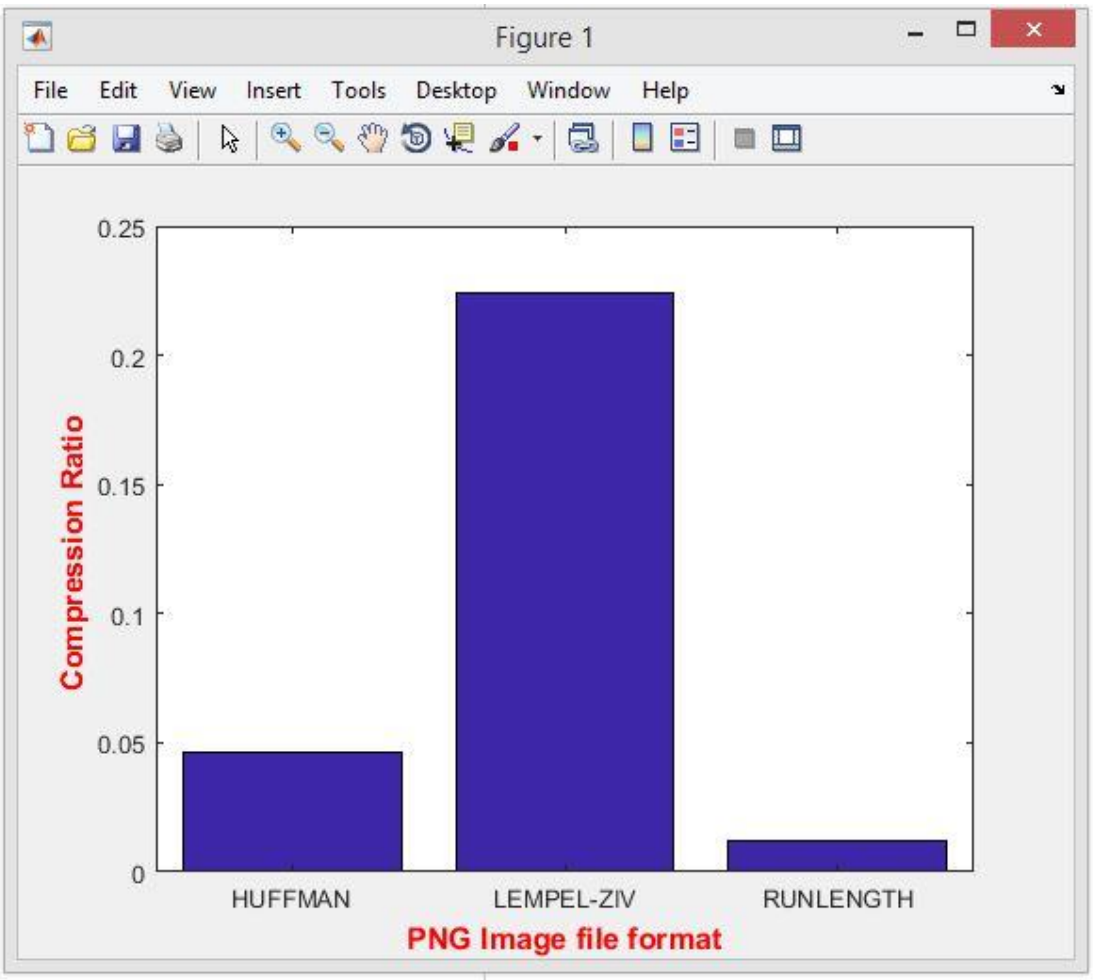

Figure 2. Graph of PNG image Compression ratio analysis

The graph above explains the compression ratio of the various algorithms on portable network graphics (PNG) image file format. The result shows that the Lempel-Ziv algorithm compresses portable network graphics (PNG) images more followed by the Huffman algorithm and then the Run-length algorithm.

The table below shows the result obtained from the portable network graphics (BMP), for all three algorithms with their various compression ratios. And their geometric mean was also taken for all the five images that were collected.

Table 8: Simulation Results for BMP Image

Image	**Run Original Size**	**Run Compressed Size**	**LZW Original Size**	**LZW Compressed Size**	**HUFF Original Size**	**HUFF Compressed Size**
C	212366	5550	20584	13683	77100	6120
Word2	59498	1348	5022	3206	126300	6120
Land1	195972	3768	5022	3116	112000	6120
Land1	195972	3768	19200	9712	107500	6120
Tiger11	199714	5494	19200	13111	79400	6120

From the simulated result obtained on table 8 for the .BMP images, for each of the algorithms, the respective compression ratio is as shown on table 9.

Table 9. Compression Ratios of the Simulated Result for BMP

Image	**Run-Length**	**LZW**	**HUFF**
C	0.0794	0.6647	0.0261
Word2	0.0485	0.6384	0.0239
Land1	0.0546	0.3342	0.0244
Land1	0.0569	0.5058	0.0192
Tiger11	0.0771	0.6829	0.0276
Average	**0.0633**	**0.5652**	**0.0242**

From the result obtained in table 9, below is a bar graph showing the compression ratios of all the algorithms (i.e. Huffman, Lempel-Ziv, and Run-length).

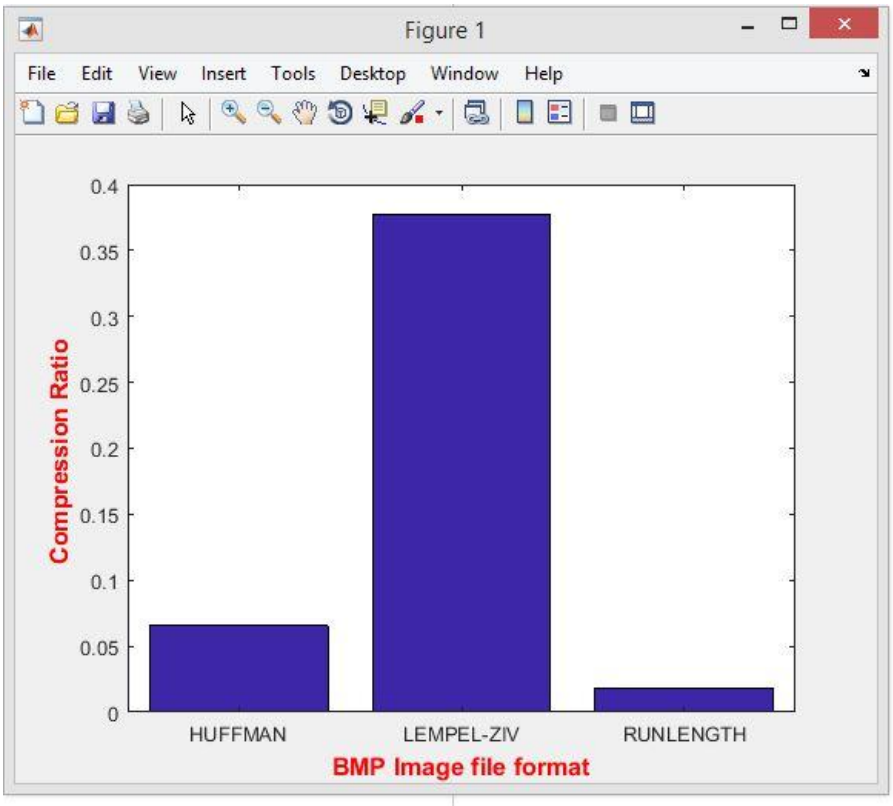

Figure 3. Graph of BMP image Compression ratio analysis

The graph above explains the compression ratio of the various algorithms on portable network graphics (BMP) image file format. The result shows that the Lempel-Ziv algorithm compresses portable network graphics (BMP) images more followed by the Huffman algorithm and then the Run-length algorithm.

5. CONCLUSION

Lossless image compression algorithm is a class of data compression algorithm that allows the original image to be perfectly reconstructed from the compressed image. Image compression algorithms are important because they can be used to reduce the amount of space needed to store data. Using compressed images can free up valuable space on any storage device, or media. Also, the amount of time it takes to send something over the Internet depends on the size of the transmitted image. Compressing image before sending them over the Internet can reduce the number of resources needed by a considerable margin. This study made an analysis of the Lempel-Ziv, Run-length, and Huffman compression algorithms using images. The performance of these algorithms based on their respective compression ratios was compared. Discussion was carried out concerning their advantages and disadvantages. This study has shown that the Huffman algorithm is the best and suitable algorithm for the compression of joint photographic experts' group **(**JPEG**)**, Portable network graphics (PNG), and bitmap (BMP) image format. The

Lempel-Ziv encoding algorithm is also suitable for joint photographic experts' group **(**JPEG**)**, Portable network graphics (PNG) formats, and bitmap (BMP) image format; it is not as efficient as the Lempel-Ziv algorithm. Further work can be done on the comparative analysis of lossless algorithms for best performance image quality of both the original and the compressed images. Also, more study can be done to determine the time complexities of the lossless algorithm. More Study can also be done to determine the peak signal to noise ratio (PSNR) of the algorithms.

6. ACKNOWLEDGMENTS

Our thanks to the experts who have contributed towards development of the template.

7. REFERENCES

[1] M. Al-khassaweneh and O. AlShorman, "Frei-Chen bases based lossy digital image compression technique," *Appl.*

Comput. Informatics, vol. 20, no. 1–2, pp. 105–118, 2024, doi: 10.1016/j.aci.2019.12.004.

[2] E. W. Abood *et al.*, "Provably secure and efficient audio compression based on compressive sensing," *Int. J. Electr. Comput. Eng.*, vol. 13, no. 1, pp. 335–346, 2023, doi: 10.11591/ijece.v13i1.pp335-346.

[3] Z. Lu, "Analyzing the Trade-offs in Lossless Image Compression Techniques:Insights for Computer Science Research," *Sci. Technol. Eng. Chem. Environ. Prot.*, vol. 1, no. 7, pp. 1–5, 2024, doi: 10.61173/99t0ga22.

[4] A. Ijaz, "Fine-Tuning Audio Compression : Algorithmic Implementation and Performance Metrics," vol. 6, no. 1, pp. 220–236, 2024.

[5] W. A. Awadh, A. S. Alasady, and A. K. Hamoud, "Hybrid information security system via combination of compression, cryptography, and image steganography," *Int. J. Electr. Comput. Eng.*, vol. 12, no. 6, pp. 6574–6584, 2022, doi: 10.11591/ijece.v12i6.pp6574-6584.

[6] R. N. Hussain, A, Al-Fayad A, *Image Compression Techniques : A Survey in Lossless and*. 2018.

[7] Y. Hu, W. Yang, Z. Ma, and J. Liu, "Learning End-to-End Lossy Image Compression: A Benchmark," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 8, pp. 4194–4211, 2022, doi: 10.1109/TPAMI.2021.3065339.

[8] A. Birajdar, H. Agarwal, M. Bolia, and V. Gupte, "Image Compression Using Run Length Encoding and Lempel Ziev Welch Method," *2019 Glob. Conf. Adv. Technol. GCAT 2019*, pp. 1–6, 2019, doi: 10.1109/GCAT47503.2019.8978408.

[9] M. A. Rahman, M. Hamada, and J. Shin, "The impact of state-of-the-art techniques for lossless still image compression," *Electron.*, vol. 10, no. 3, pp. 1–40, 2021, doi: 10.3390/electronics10030360.

[10] A. Gopinath and M. Ravisankar, "Comparison of Lossless Data Compression Techniques," *Proc. 5th Int. Conf. Inven. Comput. Technol. ICICT 2020*, pp. 628–633, 2020, doi: 10.1109/ICICT48043.2020.9112516.