Original Article

http://napas.org.ng

An Efficient Image Compression Using Enhanced Huffman Algorithm

S. Agber[1], I. Agaji[2], B. O. Akumba[4] and S. I. Odoh[4]

[1.]Department of Mathematics and Computer Science, Benue State University, Makurdi – Nigeria. email: seseagber@gmail.com
[2.]Department of Computer Science, Joseph Sarwuan Tarkar University, Makurdi – Nigeria. email: sasemiks@gmail.com
[3.]Department of Mathematics and Computer Science, Benue State University, Makurdi – Nigeria. email: beatriceakumba@gmail.com
[4.]Department of Computer Science, Federal University Lafia – Nigeria. email: odohisahsamuel@gmail.com

OPEN ACCESS
Correspondence:
S. Agber

Emails:
Specialty Section;This article was submitted to Sciences a section of NAPAS.

Submitted: 15th October, 2024
Accepted: 10th November, 2024

Citation: S. Agber, I. Agaji, B. O. Akumba and S. I. Odoh (2025). An Efficient Image Compression Using Enhanced Huffman Algorithm

Effective Date ***7(2)*** *114 -* ***122***

Publisher:*cPrint,Nig.Ltd*
Email:*cprintpublisher@gmail.com*

Abstract

Data compression involves removing redundant bits from the source data, thereby reducing the total size of the data in a manner that allows the process to be reversed when desired. For images, this involves either removing redundant pixels or representing the pixels with a smaller number of bits. The problem addressed in this paper is the inefficiency of the traditional Huffman coding algorithm in compressing certain types of image files. To overcome this, we developed an enhanced algorithm by modifying the Huffman coding technique. Our methodology includes designing the enhanced Huffman algorithm to better handle image data by considering pixel distribution and frequency more effectively. We implemented the modified algorithm and evaluated its performance against the traditional Huffman algorithm using a variety of image formats: BMP, JPG, PNG, TIF, and GIF. The performance metrics used for evaluation were compression ratio and compression time. The results show that the Modified Huffman algorithm achieves superior compression ratios for BMP, JPG, PNG, and TIF images, with quantitative improvements of 0.91, 0.90, 0.91, 0.80 respectively, compared to the traditional Huffman algorithm. However, for GIF images, the traditional Huffman algorithm demonstrated a higher compression ratio by 0.64. These results indicate that the Modified Huffman algorithm provides a more efficient solution for compressing most image formats, except for GIF images where the traditional method remains preferable. This enhanced algorithm can be highly beneficial for applications requiring optimized image storage and transmission.

Keywords: Data compression, Image compression, Huffman coding, Compression ratio

Introduction

With the advent of the internet and increased bandwidth, images and videos are moved around the World Wide Web by millions of users almost in a nonstop basis. Also, as the world continues to move towards a digitized society, the need for memory space for storing data is on the rise and in some cases the need to use the available data for instantaneous results. Implementing memory has remained a high cost and as such the need to reduce the size of files that are stored in memory without losing their content, as well as returning the files to their original size for use by some program [17]. This has brought about the need for reducing data into smaller and more manageable sizes, both for transmission and storage.

Several compression techniques have been put forward and implemented. These algorithms vary depending on the specific application, for which they are to be used. Compression techniques differ for text data and for image/video data, the reason being that images and text are implemented differently in memory. According to [17], images are represented as two-dimensional or three-dimensional arrays if the images are black and white or coloured respectively.

Image compression seeks to reduce the redundancy of the image and store or transmit data in an efficient form. This objective is achieved using various techniques which are already in existence. The choice of technique for compression depends on the application needs or requirements [2]. In some cases it could be possible to use different techniques for the same application and a choice would need to be made for the most effective technique to use.

Different algorithms have been proposed and implemented by various researchers for compressing and decompressing images for storage, transmission and use on different devices over the years. One of these compression algorithms is the Huffman algorithm which has been neglected over the years and no attempt has been made at improving its performance, especially in terms of its compression ratio. Huffman algorithm uses a variable length code in which short code-words are assigned to more common values or symbols in the data, and longer code-words are assigned to the less frequently occurring values. A dictionary of the code-words is created and used in replacing the original data.

The major constraint of the Huffman algorithm is that the compression ratio is not certain. This can cause problems especially in applications where it is necessary to know the number of bytes needed for a data set in advance [9].

There is a need to maximize the use of available memory in the implementation of databases. This is more obvious with the daily increase in the number of databases in existence, thereby increasing the amount of data and images that need to be stored. A data compression scheme that will offer a small compression ratio will imply less storage requirement without any loss to the quality of the data, which is most desirable for most applications.

Review of Literature

Several empirical studies have focused on enhancing the Huffman algorithm to improve its performance for specific types of data, including images.

[13] proposed an improved lossless image compression algorithm that theoretically provides an approximately quadruple compression ratio by combining linear prediction, integer wavelet transform (IWT) with output coefficients processing, and Huffman coding. Their main contribution lies in a new hybrid transform exploiting a novel prediction template and a coefficient processing method for IWT. Experimental results on three different image sets showed that their algorithm outperforms state-of-the-art algorithms, with compression ratios improved by at least 6.22% up to 72.36%. This algorithm is particularly effective for compressing images with complex textures and higher resolutions at an acceptable compression speed.

[14] addressed the challenge of large digital image file sizes by proposing a new near-lossless image compression method that maintains image quality while reducing file size. Their method involves dividing the image into blocks, finding the lowest value in each block, subtracting it from the rest of the values in the same block, adjusting odd numbers, dividing all values by two, and then applying the Huffman Coding technique. Experimental results using standard evaluation measures (PSNR, MSE, and CR) demonstrated a 0.11% enhancement when using two-by-two blocks and achieved high compression rates of 25%. This approach effectively balances compression efficiency and image quality.

[11] focused on reducing image file sizes to improve device performance and data transfer efficiency. They compared various compression

techniques, including RLE, Huffman, and LZW, across different image file types. Their study utilized both lossy and lossless compression methods. For lossless compression, they achieved the best quality reduction ratio with binary images, yielding up to 99.10% compression for PNG images. They also found that using lossy compression for BMP images (such as converting BMP to JPG) provided significant size reductions, with an average compression ratio of 99%. Their results indicate that different image formats and compression techniques should be strategically chosen to optimize storage and transmission efficiency.

Data Compression

Data compression is the technique of reducing the redundancies in data representation in order to reduce data storage requirements [3]. According to [9] data compression is simply the art or science of representing information in compact form. With reference to images, compression can be referred to as reducing the size, in bytes, of a graphics file with no degradation in the quality of the image to an understandable level [12].

Categories of Compression Algorithms

Data compression methods take an input data, *D* and generate a shorter representation of the data *c(D)*. The reverse process, called decompression takes the compressed data *c(D)* and generates the data *D'*, a representation of the original data [3].

Compression algorithms are categorised based on the reversibility of the original data and the manner in which the data is manipulated. Categorization by reversibility of the original data gives us two types, lossy and lossless compression [7]. Categorizations by mode of data manipulation also give two types, predictive and transform coding. In lossy compression, the reconstructed data is not an exact replica of the original data. The general assumption here is that the data doesn't have to be stored perfectly. Lossy compression is more suitable for images, video and audio data since much information can be thrown away from them and still produce acceptable quality when uncompressed. In lossless compression the reconstructed data is an exact replica of the original data. This means that the uncompressed data is exactly as it was before compression. Text files are stored using lossless techniques because, losing a single character can make the text extremely misleading. Archival storage for images, video and audio data also need to be lossless as well.

In predictive coding, information already sent or available is used to predict future values, and the difference is coded. Since this is done in the image or spatial domain, it is relatively simple to implement and is readily adapted to local image characteristics. Transform coding first transforms the image from its spatial domain representation to a different type of representation using some well-known transform and then codes the transformed values (coefficients). This method provides greater compression compared to predictive methods albeit at the expense of greater computation.

Existing Image Compression Methods

Some image compression methods already in existence and use include:

Fractal compression: This is a compression method for digital images based on fractals. A fractal is a mathematical set that exhibits a repeating pattern displayed at every scale [4]. A fractal compression algorithm first divides an image into non-overlapping 8x8 blocks, called image range blocks and forms a domain pool containing all of probably overlapped 16x16 blocks, related with 8 isometrics from reflections and rotations, called domain blocks. For each range block, it exhaustively searches, in a domain pool, for a best matched domain block with the minimum square error after a contractive affine transform is applied to the domain block. A fractal compressed code for a range block consists of quantized contractively coefficients in the affine transform, an offset which is the mean of pixel grey levels in the range block, the location of the best matched domain block and its type of isometry [15].

Vector quantization compression: For image compression using VQ, a codebook is setup consisting of code vectors such that each vector can represent a group of image blocks of size m x m. An image is first divided into m x m non-overlapping blocks which are represented as m 2-tuple vectors (training vectors). The closest matching vector in the codebook is determined for each image vector and its index in the codebook is used to encode the original image vector [16].

Arithmetic coding: This is a statistical method that is similar to the Huffman method. The input file is read symbol by symbol (until the end of the

file) and the symbols tabulated along with intervals representing the probabilities of their occurrence – the order of the symbols is not important. Numeric codes are generated for the symbols using these intervals and used to represent the symbols in the new file [1]. Compression is achieved by replacing frequently occurring symbols with shorter codes while the less occurring symbols are replaced with longer codes.

Run-Length coding: Run Length encoding is one of the simplest data compression methods. The technique is useful in the case of repetitive data. The idea behind this technique is this: if there is a data item *d* which occurs *n* consecutive times in the input stream, then the occurrences of *d* are replaced with a single pair $[d, n]$, where *d* is the data item itself and *n* is the number of consecutive occurrences of *d* [16].

LZW coding: Abraham Lempel and Jacob Ziv proposed a compression algorithm in 1978 and named it LZ-78, it was refined in 1984 by Terry Welch and renamed LZW [1]. LZW is a dictionary-based coding technique. The dictionary can either be static or dynamic (the static dictionary is fixed for the encoding and decoding process while the dynamic dictionary is updated as the encoding and decoding processes are carried out). The algorithm constructs a dictionary of words or parts of words in a file and then uses pointers to the words in the dictionary to represent the file [6][8].

Methodology

Huffman coding is an entropy encoding algorithm used for lossless data compression. Entropy coding refers to the use of a variable-length code table for encoding a source symbol, where the code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbols. Huffman coding utilizes a variable length code in which short code-words are assigned to more common values or symbols in the data, and longer code-words are assigned to the less frequently occurring values. These code-words are generated by creating a binary tree using the symbols as leaves, according to their frequencies and the path to each symbol is taken as the code-word for that symbol. A dictionary of the code-words is created and used in replacing the original data

The advantage of this method is that it generally produces good codes and is simple to implement. It has a number of disadvantages though, the most obvious being that the compression ratio is not certain [9]. The aim of this research is to develop an improved image compression algorithm by modifying the traditional Huffman coding algorithm. The proposed modifications are intended to enhance the compression ratio without compromising the quality of the image.

Model Architecture

The proposed image compression algorithm follows a systematic process consisting of the following steps:

1. **Image Preprocessing**: The input image is first pre-processed to prepare it for compression. This involves converting the image into a suitable format (e.g., grayscale for certain types of analysis).
2. **Frequency Analysis**: The frequency of each pixel value (or symbol) in the image is calculated.
3. **Modified Huffman Tree Construction**: Using the frequency data, a modified Huffman tree is constructed. This step involves the following modifications to the traditional Huffman algorithm:

i. **Block-based Processing**: Instead of processing the entire image as a single data stream, the image is divided into smaller blocks. Each block is processed independently, which allows for more localized optimization of the compression.

ii. **Enhanced Symbol Mapping**: Additional symbol mapping techniques are applied within each block to better capture the redundancy and structure of the image data.

4. **Encoding**: The modified Huffman coding algorithm is applied to each block to generate the compressed bitstream.
5. **Image Reconstruction**: During decompression, the image blocks are decoded and reassembled to reconstruct the original image.

Modification of the Huffman Algorithm for improved Compression

The modifications to the traditional Huffman algorithm are designed to improve the compression ratio for various image formats (BMP, JPG, PNG, and TIF). The key modifications include:

1. **Block-Based Processing**: By dividing the image into smaller blocks, the algorithm

can more effectively handle local variations in pixel intensity and structure. This leads to better compression ratios for images with complex textures and patterns. Since, images often contain regions with different statistical properties. Block-based processing allows the algorithm to adapt to these local properties more efficiently than a global approach.

2. **Adaptive Frequency Calculation**: The frequency of pixel values is calculated adaptively for each block rather than for the entire image. This approach captures local redundancies more effectively, leading to improved compression performance.
3. **Enhanced Symbol Mapping**: Additional symbol mapping techniques, such as run-length encoding (RLE) within each block, are integrated with the Huffman coding process. By combining RLE with Huffman coding, the algorithm can more efficiently compress sequences of repeated values, which are common in many images.

Considering the fact that compression involves reducing redundancies present in the data [3], we will attempt to further reduce these redundancies from the output of the traditional Huffman algorithm. To do this, we will add four steps to the Huffman algorithm as in Figure 1. We will separate the data stream obtained from the Huffman algorithm into nibbles and consider the value of each of these nibbles. We will then use two data sets to represent the data based on the resulting values of the nibbles. If the nibble value is greater than zero, the nibble will be added to *file I* (which contains the first data set). The nibbles whose values are equal to zero will be discarded.

The second data set (contained in *file II*) will contain a bit stream of 0's and 1's corresponding with the nibbles that are evaluated. A 1 bit will be added if the nibble value is greater than zero and a 0 bit will be added if the nibble value is equal to zero.

The content of *file I* will then be appended to that of *file II* for storage or transmission. The number of bits contained in *file II* (before adding *file I*) will be added to the header of the stored file this detail will be used when the need to reverse the process arises (at the receiving end of a transmission or when the file is retrieved from memory).

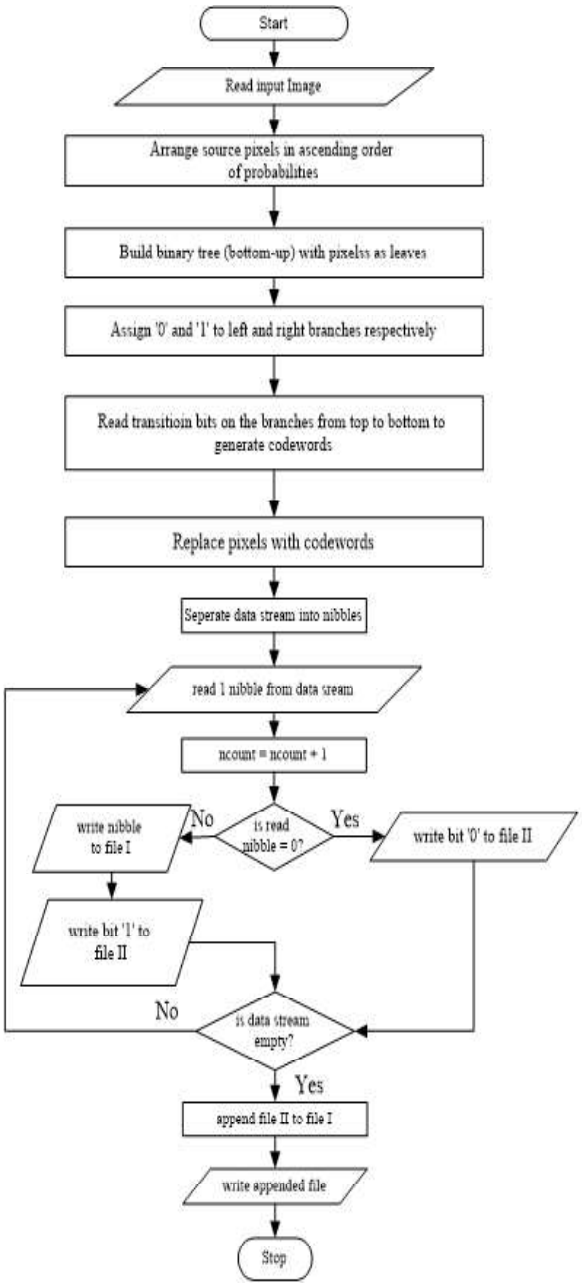

Figure 1: Flowchart of the Modified Huffman Algorithm

Assumptions

Several assumptions are made in the development and evaluation of the modified Huffman algorithm:

i. **Image Characteristics**: It is assumed that the input images are either grayscale or have been converted to a format suitable for block-based processing.
ii. **Block Size**: A fixed block size is used for all images, with the assumption that this size provides a good balance between compression efficiency and computational complexity.
iii. **Redundancy Patterns**: The algorithm

assumes that redundancy patterns within blocks are similar enough to be captured effectively by the modified symbol mapping techniques.

Evaluation Metrics

To evaluate the performance of the proposed algorithm, the following metrics are used:

i. **Compression Ratio (CR)**: The ratio of the original image size to the compressed image size.

ii. **Peak Signal-to-Noise Ratio (PSNR)**: A measure of the reconstructed image quality compared to the original image.

iii. **Mean Squared Error (MSE)**: The average squared difference between the original and reconstructed image pixel values.

Simulation Design

The simulation program was written using MATLAB programming language. The program is command driven and has two modules, each of the which implements one of the algorithms we wish to investigate. The program creates files stored in a specified location in the system on which it is run, where results of each simulation are recorded. At the beginning of execution, the program requests for an image as input. This image is passed to each module where the specified operations are carried out and the result recorded.

The simulation experiments were carried out using five images (arbitrary value) for each image format. The geometric mean of the compression ratios was taken and used to evaluate the performance of the algorithms for each image. This is because a single extreme value has less impact on the geometric mean of a series than on arithmetic mean [5]. Evaluating the geometric mean makes it harder for one algorithm to be rated high or low for a particular group by achieving a high or low score on just one image in the group, thereby making the algorithm's overall rating a better indicator of its performance

Results

The simulations were run using images of the most common formats (*bmp*, *gif*, *jpg*, *png* and *tif*). This is so as to enable us observe the performance of the algorithms for various categories of images. Five images of different sizes and dimensions stored using the formats stated above were used. The resulting compression ratios are shown in Figures 3(a) – 3(d) while the mean compression ratios obtained are shown in Figure 4.

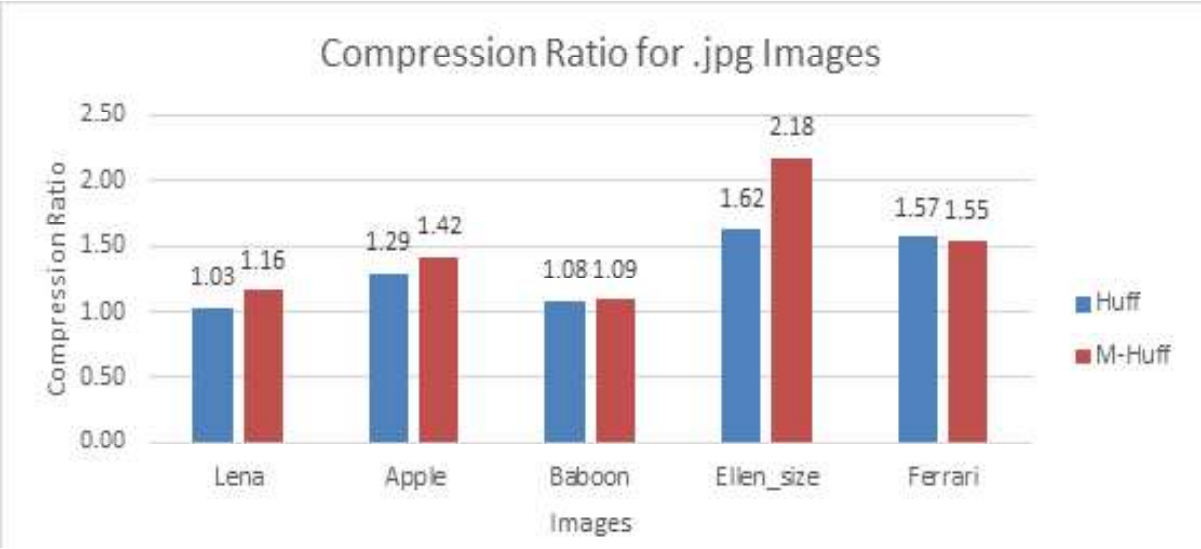

Figure 3(a): Compression Ratio for .jpg images

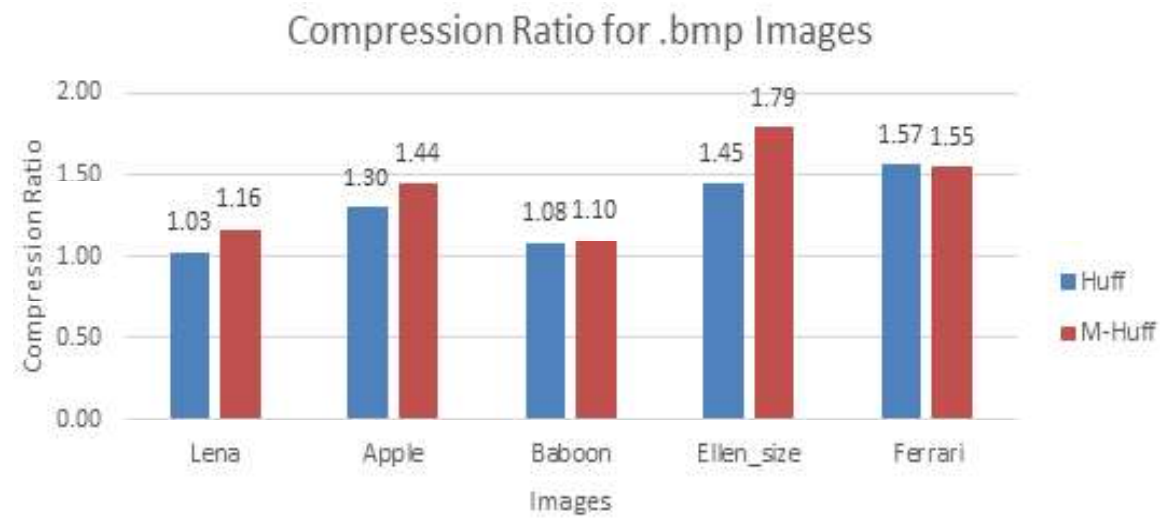

Figure 3(b): Compression Ratio for .bmp images

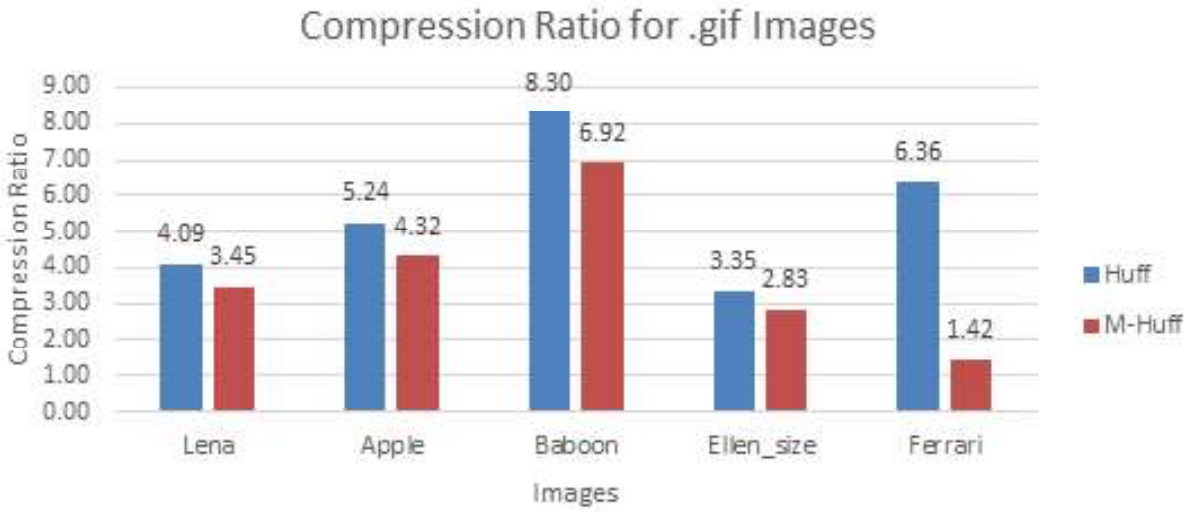

Figure 3(c): Compression Ratio for .gif images

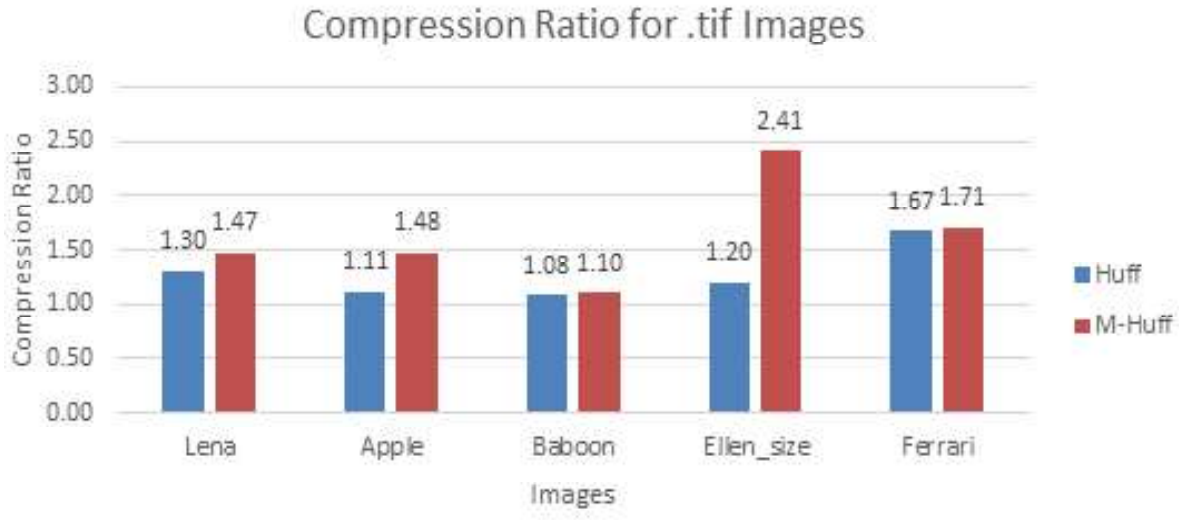

Figure 3(d): Compression Ratio for .tif images

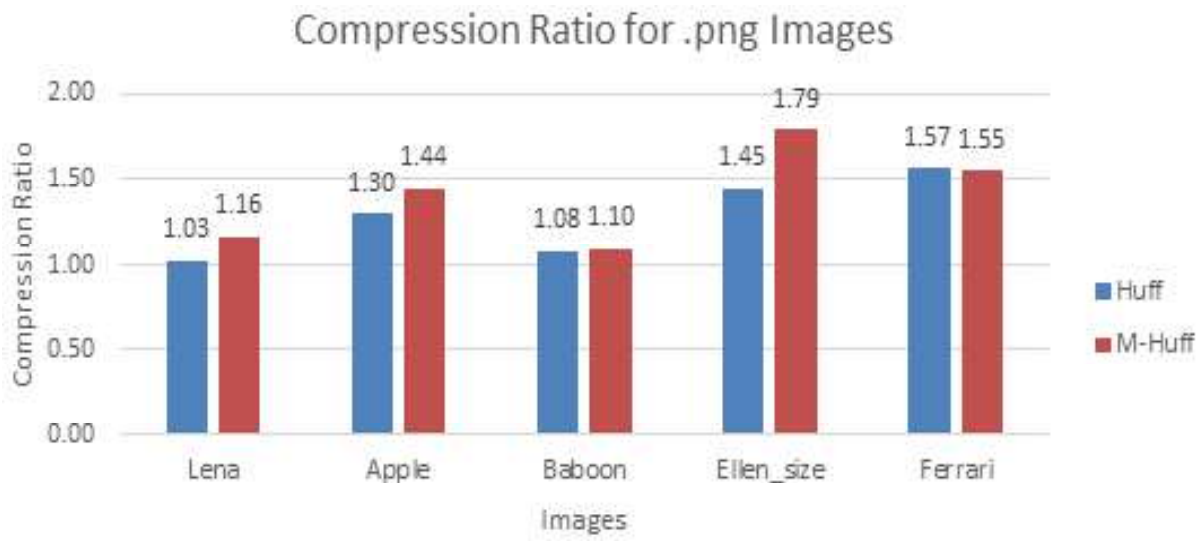

Figure 3(e): Compression Ratio for .png images

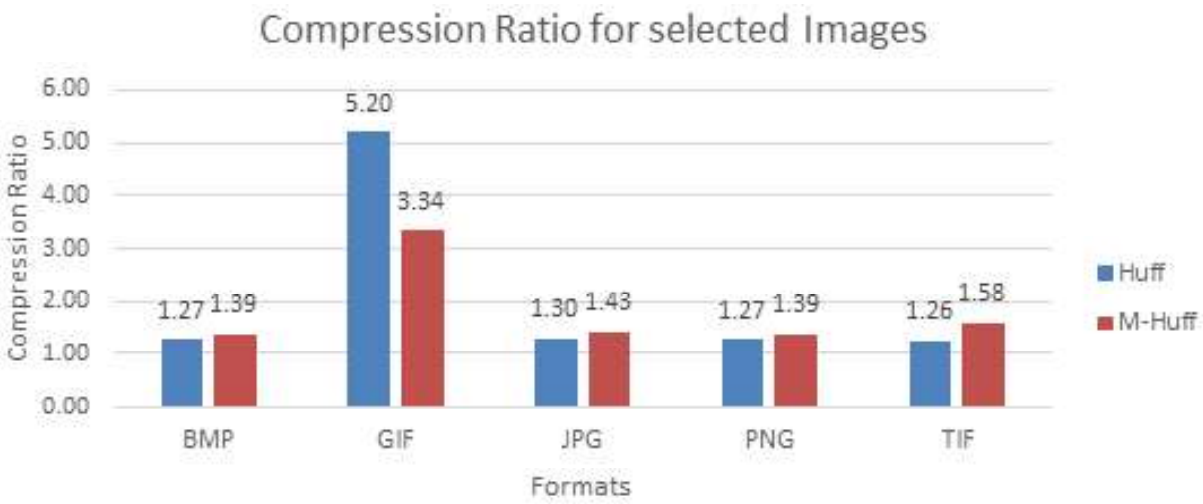

Figure 4: Comparison of Compression Ratios for selected Images

Figure 4 shows that the Modified Huffman algorithm produces a better compression than the original algorithm for the .bmp, .jpg, .png and .tiff formats. The .gif format is the only exception, where the original Huffman algorithm has better compression ratio. This could be because gif storage algorithm compresses images before storing, which causes negative compression in the attempt to further manipulate the Huffman output.

Summary, Conclusion and Recommendations

In this paper, we evaluated and modified the Huffman algorithm with the aim of improving its level of compression for image data. A simulation program was written to implement the algorithms in order to ascertain their performance using the compression ratios of the algorithms. The results obtained show an overall improved compression ratio by the Modified Huffman algorithm.

Further work can be carried out to ascertain why the algorithm's performance varies for different images, first for images of the same format and then for different formats.

References

[1] Al-laham, M., & El Emary, I. M. (2007). Comparative Study Between Various Algorithms of Data Compression Techniques. *Proceedings of the World Congress on Engineering and Computer Science.* San Francisco.

[2] Al-Mahmood, H., & Al-Rubaye, Z. (2014). Lossless Image Compression based on Predictive Coding and Bit Plane Slicing. *International Journal of Computer Applications, 93*(1), 1 - 6. doi:10.5120/

16176-2068

[3] Arun, K. P. (2009). *Implementtion of Image Compression Algorithm Using Verilog with Area Power and Timing Constraints.* National Institute of Technology, Rourkela: Thesis Submitted to Department of Electronics and Communications Engineering.

[4] Boeing, G. (2016). Visual Analysis of Non-Linear Dynamical Systems: Chaos, Fractals, Self-Similarity and Limits of Prediction. *Systems, 4(4)*(37). doi:10.3390/systems4040037

[5] Carter, N. (2002). *Computer Architecture: Schaums Outline Series.* New Delhi: Tata McGraw Hill.

[6] Devi, M., & Mehta, U. (2016). A Review on Variouis Techniques of Image Compression. *International Journal of Engineering and Computer Science, 5*(7), 17127 - 17129. doi: 10.18535/ijecs/v5i7.01

[7] Dhawan, S. (2011). Review of Image Compression and Comparison of its Algorithms. *International Journal of Electronics and Communication Technology, Vol 2(1)*, 22 - 26.

[8] Kaur, R., & Choudhary, P. (2016). A Review of Image Compression Techniques. *International Journal of Computer Applications, 142*(1), 8 - 11. doi:10.5120/ijca2016909658

[9] Khalid, S. (2006). *Introduction to Data Compression, Third Edition.* San Francisco: Elsevier Inc.

[10] Kodituwakku, S. R., & Amarasinghe, S. U. (2010). Comparison of Lossless Data Compression Algorithms for Text Data. *Indian Journal of Comoputer Science and Engineering. Vol 1, No. 4*, 416 - 425.

[11] Lantana, D. A., Sholihati, I. D., Sari, R. T., & Hendrik, B. (2023). An Extensive Analysis of Digital Image Compression Techniques Using Different Image Files and Color Formats. *nternational Journal on Advanced Science, Engineering and Information Technology, 13*(5). doi:10.18517/ijaseit.13.5.19319

[12] Lata, A., & Singh, P. (2013). Review of Image Compression Techniques. *International Journal of Emerging Technology and Advanced Engineering. Vol 3, No. 7*, 461 - 464.

[13] Liu, X., An, P., Chen, Y., & Huang, X. (2022). An improved lossless image compression algorithm based on Huffman coding. *Multimedia Tools and Applications, 81*, 4781 - 4795. doi:10.1007/s11042-021-11017-5

[14] Otair, M., Abualiga, L., & Quawaqzeh, M. K. (2011). Improved near-lossless technique using the Huffman coding for enhancing the quality of image compression. *Multimedia Tools and Applications, 81*, 28509 - 28529. doi: 10.1007/s11042-022-12846-8

[15] Parwe, P. R., & Mandaogade, N. N. (2015). A Review on Image Compression Techniques. *International Journal of Computer Science and Mobile Computing, Vol. 4, No 2*, 198 - 201.

[16] Singh, A. P., Potnis, A., & Kumar, A. (2016). A Review on Latest Techniques of Image Compression. *International Research Journal of Engineering and Technology, Vol 3, No 7.*, 727 - 734.

[17] Thyagarajan, K. S. (2011). *Still Image and Video Compression with Matlab.* Hoboken, New Jersey: John Wiley & Sons Inc.

[18] Zhou, H., Wu, J., & Zhang, J. (2010). *Digital Image Processing: Part 1.* Retrieved August 17, 2016, from bookboon.com: http://bookboon.com/en/digital-image-processing-part-one-ebook