**Lossless Image Compression Using MATLAB**

Uditanshu Khatri

Image Processing

New Jersey Institute of Technology

Newark, NJ 07102

**ABSTRACT**

Many image processing applications, such as medical imaging, satellite imaging, and video, demand a considerable quantity of storage space or high bandwidth for transmission in their original form because the picture size or image stream size is too enormous. In these cases, image compression methods can be pretty helpful. Lossless (reversible) image compression algorithms maintain the information, allowing for accurate picture reconstruction from compressed data. In this research, we explore existing coding and lossless compression systems and present an experimental evaluation of several state-of-the-art lossless compression algorithms published in the literature. Many publicly accessible picture data sets are used in practical assessment. We compare the resulting compression using a range of linearization strategies used to picture data sets for coding algorithms. This research also proposes an approach for lossless compressing collected images to minimize picture size while maintaining image quality.

**Keywords:** image compression, lossless compression, coding.

**INTRODUCTION**

Image compression is a technique for reducing the size of an image while it is being stored and processed. Compression has become increasingly important in everyday life as picture quality and size have increased [4]. With the growing popularity of cloud storage, compression is becoming increasingly crucial for storing vast numbers of photographs online. Thus, Lossless image compression aims to encode an image signal with the fewest bits feasible without losing any information, allowing for faster transmission and lower storage needs [4]. The average bit rate, which is the average number of bits per sample for still pictures and the average number per second for video, is often used to convey the number of bits that make up the signal. In addition, because visual signals include a considerable amount of redundancy, lossless compression is achievable[4]. The level of correlation among the picture data samples determines the amount of redundancy. For example, in a natural still image, there is frequently a high degree of spatial correlation between adjacent image samples.

Furthermore, Lossless coding requires that the decoded picture data be quantitatively mathematically and qualitatively visually identical to the original encoded image. Although this condition guarantees accurate representation accuracy, it frequently restricts the amount of compression that may be done to two or three compression factors. Perceptually lossless coding methods try to eliminate superfluous as well as irrelevant data and information to obtain better compression factors; these approaches simply need that the encoded and decoded pictures be visually identical, not necessarily mathematically. Some information loss is acceptable in this situation as long as the recovered image seems to be similar to the original.

This research covers the fundamentals of lossless image coding and some more recently discovered lossless compression techniques. This research is organized as follows; existing methods, new methods, experimental results, comparisons with the existing processes and conclusion.

**EXISTING METHODS**

**Use of LZW**

The Lempel-Ziv-Weich (LZW) lossless compression method is among the oldest and most extensively used compression techniques for GIF image compression and file formats [4]. The method processes and compresses the image data by reading the symbols and arranging the characters into strings and since the space required to hold codes is minimal, the strings are transformed into codes, and compression is achieved.

The algorithm works in this way. A table with many entries is formed, with 4096 being a popular choice. In the table, single bytes are defined by the codes 0 to 255. In addition, Single bytes ranging from 0-255 are utilized to begin the encoding process, while codes ranging from 256 to 4095 are used for compression. The program finds and inserts repetitive sequences in data into a code table thus decoding is accomplished by converting the code structure to the data it defines.

Example of the algorithm;

m = NIL;

while (read a character j)

{

if mj exists in the dictionary

m= MJ;

else

add me to the dictionary;

output the code for w;

m = j;

}

**Huffman encoding**

The algorithm is designed based on the likelihood of elements or values occurring in a defined set of data. It allocates a variable-length code to the characters, with the length of the code determined by the probability of of occurrence of the character [3]. The codes for the most commonly occurring characters are shorter, whereas the codes for the less often occurring characters are longer. Two trees are produced to compress the data: The Huffman Tree and the Transverse Tree.

The Huffman Tree

The following steps are followed to acquire the Huffman tree: Construct a minimum heap of all leaf nodes, then make one for each unique character. Create a new internal node with the lowest frequency from the largest heap [3]. Create a new internal node that has the same frequency as the other two nodes. To the minimum heap, add the first extracted node's left child and the second extracted node's right child. Repeat steps 2 and 3 until the heap is empty. The tree is now complete with the exception of the root node.

The images are encoded using the images' colours, with different codes constructed for each colour. The frequency of occurrence of each colour is used to produce Huffman codes. This decreases the image's size without sacrificing the image's subjective quality. To do the compression, the image size is estimated as the total number of columns and rows in the picture, and a matrix with the same dimensions as the image is created. The histogram data counts the number of pixels in each shot for each tonal value. The red, green, and blue spectrums are separated to count the pixels. Matrix components from the image are used to generate a cell array. The ratio of the number of pixels in the histogram to the elements in the cell array is recorded in a variable. The codes created for the individual pixels are kept in the cell array that has been constructed. The binary codes are converted to decimal values, and a matrix with the same dimensions as the original image is formed.

**Using the Run Length Encoding (RLE)**

Run Length Encoding scans the image data and looks for similar values. The values may appear in a sequence [4]. The data value for these repeated values is kept as a single value. The redundancy is eliminated by storing similar data as single values. The file's size shrinks as the number of unnecessary values is reduced. If the file has many redundancy values, the technique will be effective [4]. The technique will enlarge the file size for data that does not have recurring values.

In addition, the approach reduces the physical size of a run, which is a string of repeated data[4]. Two bytes are used to represent the run. The run count (number of values in the run) is stored in the first byte. If the encoded run has n characters, the run count will be n-1. The run value, is represented by the second byte. It will represent a number between 0 and 255. An RLE Packet is made up of two bytes concatenated and if the amount of characters in a run exceeds the limit, the RLE Packets are created. RLE compresses data using three different approaches: encoding along the X-axis, Encoding along the Y-axis, and Zig-zag encoding.

Along the X-axis

Sequential processing is the name for this encoding method[4]. Here, the one-dimensional lines are scanned from a two-dimensional data map by simply encoding a bitmap from the utmost top left corner to the very end of the right corner for each line.

Along the Y-axis

This encoding is likewise sequential, however it is done along the Y-axis, that is, for the column.

Zig-zag encoding

Instead of scanning only one dimension, this encoding method is utilized for encoding a bitmap into two dimensions. This encoding technique scans the map diagonally. The scanning starts in the utmost upper left corner and proceeds in a diagonal Zig-zag pattern until it reaches the bottom right corner [4]. Text data, including character strings or sequences, are best compressed using sequential encoding. Encoding in two dimensions is required to compress picture files, and Zig-zag encoding is the best option.

In lossless image compression, the algorithm is implemented by keeping records of the relationship of the words used and the code elements [4]. When code is used in place of words and the file is compressed, the algorithm's efficiency increases if the data has a high degree of redundancy. The input image data is compressed in a single pass, and no prior knowledge of the input data is required. The RLE technique is used with wavelet decomposition to generate a lossless compressed picture. The standard implementation of RLE for colour pictures yields greyscale images; hence, the image is divided into wavelets, and the image data from the wavelets is then quantized. Using Zig-Zag encoding, the RLE method is applied to these pictures.

**IMPROVED METHODS**

**Discrete Cosine Transform (DCT)**

In this algorithm, a square-matrix is used for masking process of the picture matrix, which compresses the image. essentially, the more masking there is, the more image compression and quality loss there is [2]. The compression may be made lossless and the picture size reduced by using correct masking settings in the matrix. The masking matrix is specified and may be modified according to the needs.

To calculate the DCT coefficients for compressing pictures using the DCT technique, an 8 by 8-pixel block example is built. In this case the pixel block case's entries are represented as 0 or 1. The picture matrix is masked using this pixel block case. The DCT matrix and its respective transpose are cross-multiplied by the presented image matrix. In addition, the DCT matrix, which is an (x n) discrete matrix, is also included in the image processing in MATLAB. The output of this matrix multiplication is afterwards multiplied by a respective pixel block case, namely the masking matrix, and finally the resulting matrix is then multiplied by the DCT matrix and its respective transpose. As a result, the compressed picture matrix has a resultant larger energy co-efficient in the top left corner of the DCT matrix. The masking matrix is shown below.

Mask = [ 1 1 1 1 0 0 0 0

1 1 1 0 0 0 0 0

1 1 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0];

Furthermore, because all of the masking matrix components except for the upper left corner which is represented by 1’s is defined by zeros, then only the resultant higher energy co-efficient is compressed, resulting in a lossless compression. Finally, when paired with the coding of the residual error sequence, this form of DCT execution yields a lossless image.

**Discrete Wavelet Transform (DWT)**

The Discrete Wavelet Transform is a compression method that uses discrete wavelet sampling. The orthonormal series formed in any picture is used to describe a square-integral function, and wavelets are the orthonormal series produced in any picture [2]. The wavelet is made up of time and data-related information. The Wavelet transform allows you to adjust the length of the function but not the shape. Wavelet transformations provide the same data as short-time Fourier transformations, plus wavelet features [2]. The DWT compression may be used at many levels of wavelet decomposition in the picture. Haar wavelet decomposition and compression were used for this investigation because Haar wavelet requires no multiplications and has a quick calculation time. Haar wavelet is easy to add elements to since it has a lot of zero elements.

In implementing this algorithm, the wavelet transformation is carried out using four coefficient matrices, cA, cD, cH, and cV. This corresponds to level 1 approximation, vertical and diagonal horizontal details. Similarly, the level 2 decomposition may be achieved using level 1 approximations. A vector is used to hold the data from level 2 breakdown as well as level 1 decomposition. Following multilevel decomposition, the original picture is synthesized or recreated using the coefficients from the vector containing the decomposed data. In addition, the image may be compressed using the wavelet tools in MATLAB using the commands 'ddencmp' and 'wpdencmp' based on this data decomposition data. The pictures are recreated using the coefficients obtained after compressing the deconstructed data saved as vectors. To minimize subjective degradation of the picture, the data acquired after level 1 decomposition is then compressed along with the data received after level-2 decomposition in the vector while conducting image compression utilizing decomposed.

**Implementation**

The algorithms are implemented in MATLAB code. The data set was used to evaluate lossless compression methods such as Lempel-Ziv-Weich (LZW), Huffman, Run Length Encoding (RLE), Discrete Cosine Transform (DCT), and Discrete Wavelet Transform (DWT). Each of the techniques technique will compress the photographs in the dataset, which includes images from the following categories: dog, cat, tree, portrait, cup, and truck. The resolution of these images used is 256 × 256. MATLAB is chosen for execution because the processing toolbox in MATLAB will build the algorithms faster. In addition to this, the functions in MATLAB will give a superior technique for processing and compressing the photos. The experimental results of the algorithms are discussed below.

**EXPERIMENTAL RESULTS**

In this chapter, the algorithms have been implemented and compared. For each approach, compression ratios were obtained for the same images. It is important to note that the subjective image quality was considered only to ensure that the methods were correctly implemented and that no image data was lost, as actual lossless image compression is expected to yield the identical image after decompression. The following equation was used to calculate the compression ratio.

**Compression Ratio = Original File Size/Compressed File Size**

**Results of Run Length Encoding**

It employs Zig-zag encoding for compression, and the output is a completely rebuilt picture with compression ratios similar to those illustrated below. In terms of subjective picture quality, the reconstructed image is lossless. The lossless compression is returned for all picture categories in the dataset. Between 2.5 and 3.2, compression ratios are used.

|  |  |
| --- | --- |
| image | Compression ratio |
| Dog | 2.77 |
| Car | 2.91 |
| Tree | 2.83 |
| Portrait | 3.13 |
| Cup | 3.06 |
| Truck | 2.53 |

*Lossless image compression ratios for the RLE*

In addition, the perceived visual quality is comparable to that of the compressed image. With rising picture size and resolution, the time required to compile the code rises.

**Results for Huffman**

After compression, Huffman encoding creates a lossless picture. Huffman encodes and compresses using a probability function. In terms of subjective visual quality, the compression output is lossless. The Huffman algorithm compiles quicker than the RLE method. The ratios used are 2.8- 3.8, compression ratios are used. When this is compared to the RLE, the compression ratios are finer.

|  |  |
| --- | --- |
| image | Compression ratio |
| Dog | 3.27 |
| Car | 2.89 |
| Tree | 3.10 |
| Portrait | 3.73 |
| Cup | 3.66 |
| Truck | 3.23 |

*The ratio for the Huffman algorithm*

**Results for the DCT**

The algorithm provides a lossless image by adjusting its masking matrix to produce an image with no quality loss. The compression however is efficient for specific photos as shown in the table below. As the subjective image quality is similar, the compression results are lossless. The ratios are from 2.1-3.1. Notably the masking matrix can be changed to enhance the ratios; however, this may lead to data loss, which is not lossless compression.

|  |  |
| --- | --- |
| Image | Compression ratio |
| Dog | 2.17 |
| Car | 2.61 |
| Tree | 2.43 |
| Portrait | 3.09 |
| Cup | 3.11 |
| Truck | 2.83 |

*Compression ratio for DCT algorithm*

It is important to note that the Fourier coefficients with the highest energy co-efficients are specifically compressed to maintain picture quality. To affirm that the compression is lossless and prevents data loss, the low energy co-efficients are either maintained or adjusted to yield zero.

**Results for the DWT**

After compression, the Discrete Wavelet Transform yields a lossless picture. The Haar wavelet is well compressed and quantized to get the highest quality image. The ratios for DWT are as shown in the table below. The compression ratios vary from 2.4-3.7, more significant than DCT but less than Huffman. The algorithm's implementation is made more accessible by MATLAB's existing tools for wavelet decomposition and picture synthesis. The Haar wavelet may also be used to enhance the number of stages of deterioration in the wavelet decomposition.

|  |  |
| --- | --- |
| Image | Compression ratio |
| Dog | 2.17 |
| Car | 2.61 |
| Tree | 2.43 |
| Portrait | 3.09 |
| Cup | 3.11 |
| Truck | 2.83 |

*Compression ratio for DWT*

**COMPARISONS WITH EXISTING METHODS**

With the exception of the LZW method, the rest of the algorithms used in this study were capable of lossless compression. Varied techniques have different compression ratios and compilation rates in MATLAB. Although the Huffman method produced superior compression ratios, it took longer to compile. The RLE yielded decent compression ratios, but the technique is complicated, and for photos with higher resolutions, the implementation may fail. Lossless pictures with good compression ratios and compilation rates were obtained using the DCT and DWT algorithm implementations, quicker than Huffman and RLE. When comparing DCT with DWT compression, DWT compression is more straightforward to develop and execute.

**Discussion**

Initially, the RLE and Huffman algorithms returned greyscale pictures. In addition, the algorithms were changed by quantizing the data and then encoding it using the RLE technique. The approach merely encodes the image matrix's rows, and it was not converted into a two-dimensional picture with a colour map during implementation. The picture could not be reconstructed at first since there were no specifications on how the image should be generated. The process was subsequently expanded to include columns, and the two-dimensional picture was encoded and decoded using Zig-Zag encoding. After then, the program was able to produce a colour picture.

Furthermore, Initially, the Huffman method was unable to produce colour pictures that were comparable to RLE. Since the process are dependent on probabilities, the final codes had to be decoded before being transformed to picture coordinates. The conversion was not completed effectively; thus, several functions were used to recreate the lossless picture. After that, the decoded codes were transformed into a cell array, which is then utilized to reassemble the image.

Notably, by altering the masks that were used to encode the image for compression, the DCT technique was converted to lossless compression. Only one level of wavelet processing was employed in the DWT. It was then tweaked to reconstruct the image without sacrificing quality.

In addition, the DWT method delivers lossless pictures with reasonable compression ratios for all types of images utilized in this investigation. Images of the more excellent resolution were also reduced, with no delays in the algorithm's compilation. The DWT technique is more dependable for high-resolution picture compression than Huffman encoding, despite the lower compression ratios. Compared to the other algorithms employed in this study, the DWT algorithm's execution is likewise straightforward, with no substantial difficulties or complexity. When compressing many high-quality photos, the DCT technique also provides lossless compressed images with decreasing compression ratios. Still, the matrix conversion and coefficient compression might get complicated. Changing the masking matrix in the DCT technique is also tricky since lossless compression may not be achieved.

In addition, although the compression ratios for Huffman and RLE algorithms are slightly greater than for DWT and DCT techniques, the compilation is not steady for rising picture resolution. During the collection of Huffman encoding, there may be a brief halt or termination without any output. As a result, the Huffman method is untrustworthy for lossless picture compression at high resolutions. The RLE algorithm's execution is highly complicated, requiring several functions, encoding, and decoding approaches to be implemented to create lossless color pictures. Moreover, The RLE technique fails to compress high-resolution photos in most circumstances. Because the LZW technique generates binary pictures as output, it cannot be termed lossless. It will need to be modified to accomplish lossless compression and generate color images.

**CONCLUSIONS**

As a result of the comparison of lossless compression algorithms in this study, it can be stated that the DWT method is best fit for lossless compression of the high-resolved photos due to its simplicity of execution and implementation in MATLAB. After lossless compression, the file size can be lowered with good compression ratios. After lossless compression and file size reduction, the subjective image quality stays the same as before. In addition, without complicating the implementation, the DCT method may be integrated with other algorithms or altered matrix implementation. Also, both the Huffman and the RLE can be integrated to execute lossless compression for the high resolved images without complexities and interruptions.

**References**

[1] K. S. Thyagarajan, *Still image and video compression with MATLAB*. Hoboken: Wiley-Blackwell, 2011.

[2] N. Joshi, “Optimization of Image Compression algorithms using DWT- DCT method,” *International Journal Of Engineering And Computer Science*, Jun. 2017, doi: 10.18535/ijecs/v6i6.18.

[3] Gabriel Iwasokun, “Lossless JPEG-Huffman Model for Digital Image Compression,” *Advances in Image and Video Processing*, vol. 7, no. 1, Feb. 2019, doi: 10.14738/aivp.71.5837.

[4] B. Khan, Ms Poonam, and Mr M. Talib, “Matlab Based Image Compression Using Various Algorithm,” *International Journal of Trend in Scientific Research and Development*, vol. Volume-2, no. Issue-4, pp. 1638–1642, Jun. 2018, doi: 10.31142/ijtsrd14394.