Image compression performance comparison of RLE and LZV algorithms for effective big data management: A case study.

Şükrü Mustafa KAYA[1]
Murat Aksel AKÇAY[2]

Geliş tarihi / Received: 31.07.2022
Düzeltilerek Geliş tarihi / Received in revised form: 24.10.2022
Kabul tarihi / Accepted: 18.01.2023
DOI: 10.17932/IAU.ABMYOD.2006.005/abmyod_v17i66004

Abstract

Due to the rapid increase in digitization, the need for data storage in digital systems and network systems is increasing. As important as fixtures for public institutions and the private sector, It is vital that digital data is stored and backed up in a healthy and secure manner. At the same time, storing data on social media and similar digital platforms is becoming a necessity. As the need for data storage becomes a problem, different solutions are sought and smaller data storage possibilities are preferred. Data compression algorithms also serve this purpose in a sense. The main purpose of the studies in this field is to obtain less voluminous data by using compression methods from the raw data produced, to narrow the area covered by the data without destroying its original state, and to create data sets with low storage costs. There are many studies in the literature for this purpose. In this study, researches on the subject are carried out and different compression algorithms such as LZW, RLE are discussed. In addition, image files in different formats are compared based on their size, and image compression performances are tested based on the space occupied by image compression algorithms in memory.

Keywords: *Image Compression, Image Processing, Big Data Management, LZW, RLE*

Introduction

Different image formats are used to make the images obtained by different methods usable in the digital environment. Image compression algorithms, which are developed in order to transport and store the obtained images in personal memories, server systems and network, in a healthy and reliable way when needed, are important today as they were in the past. Especially in recent years, it is seen that the raw data transferred to the network via sensors and cameras creates very large volumes of data on the surveys. These systems, created by cameras and sensors working together, are examined within the scope of the Internet of Things (Kaya et. al., 2021). Similarly, data compression algorithms, machine learning methods and hybrid methods are used in IoT-based systems, such as data compression, preprocessing, and data filtering. The rapid spread of IoT technology and the continuous production of raw data trigger the development of data compression algorithms and ML techniques with hybrid methods (Kaya et. al., 2021).

Efforts to reduce the cost paid for functional use of files in different image formats circulating in personal memories, server systems and on the network, to provide more and healthier image storage, processing and accessibility with less cost, are increasing day by day. Image compression algorithms, developed to meet the need to store the most realistic images in a healthy and reliable way, are widely used for this purpose. Storing, processing and transporting very large files reduces the quality of service on digital systems. As the file size increases, memory costs increase and cause loss of time (Kaya Ş.M. et. al., 2022). Knowledge of which file formats should be used when creating image files is important in this respect.

In this study, different compression algorithms are run on the same images with the algorithms selected from the image compression algorithms. With an application we designed to measure the performance of algorithms, the memory sizes of the images subjected to compression process are compared and the positive and negative aspects of the algorithms are examined. In addition, the common aspects and divergent aspects of data compression algorithms and image compression algorithms are examined. A literature review is carried out on image compression methods, and RLE (Run-Lenght Encoding) and LZW (Lempel Ziv Welch) algorithms among image compression algorithms constitute the scope of the study.

Image compression algorithms

The image formats selected for comparison are just some of the commonly used image formats. The image compression algorithms used for comparison are listed as follows.

- ✓ Image compression with LZW (Lempel Ziv Welch) algorithm.
- ✓ RLE (Run-Length Encoding) algorithm.
- ✓ Image compression with TIFF (Tagged-Image File Format) algorithm that supports multiple (RLE, LZW, ZIP, JPEG) image compression algorithms.

LZW (Lempel Ziv Welch)

LZW, which are dictionary-based data compression algorithms, is considered to be a derivative of LZ77 and LZ78 developed by Abraham Lempel and Jakob Ziv. The new algorithm that emerged when Terry Welch adapted the LZ78 approach to high-performance disk units in 1984 was accepted as LZW. LZW has managed to be the best of the LZ78 family in terms of both compression and decompression performance. Since it is an algorithm that gives good results on all types of data, many subsequent algorithms are based on LZW. Has held the patent for Unisys LZW since 1985 (Altan and Cerus, 2006).

LZW algorithm is a dictionary-based data compression algorithm. Instead of creating a code tree by calculating the number of repetitive characters as in Huffman coding, one of the accepted data compression methods, it performs the encoding process by referring to a dictionary. A dictionary is created with the strings contained in the data to be compressed, and then the dictionary order of this string is encoded in any character repetition. As the number of repeating characters increases, the compression ratio also increases (Bulut , 2016).

The LZW algorithm performs the encoding process through the dictionary. The LZW algorithm tries to add the word containing as many characters as possible to the dictionary by proceeding step by step in the data to be compressed, and in the meantime, it replaces the word with its equivalent in the dictionary. If a four-character text is encoded in the dictionary with a symbol, this four-character message is represented by a single symbol. Initially, the dictionary contains all ASCII characters. All characters after 256 characters are encoded into the dictionary by the LZW algorithm (Shrividhiya et al., 2021). Another important feature of the algorithm is that it does not require the transmission of the encoded dictionary to the other party. The dynamically created dictionary is

in the compressed data and when it is transmitted to the other party, it is opened and the dictionary is created (Şişman, 2014).

The working principle of the algorithm can be expressed in six stages as follows.

1. Take letters from text
2. Search letter in dictionary
3. If the letter from the text is equal to one of the entries in the dictionary, continue to get letters from the text
4. If the value in the text is equal to the value in the dictionary, keep the value in the dictionary as the result
5. If the values are not equal, add the matching new letter and the matching dictionary entry as a new dictionary entry.
6. If the text is not finished, continue with the second step with the new letter that does not fit (Cerus and Altan, 2006).

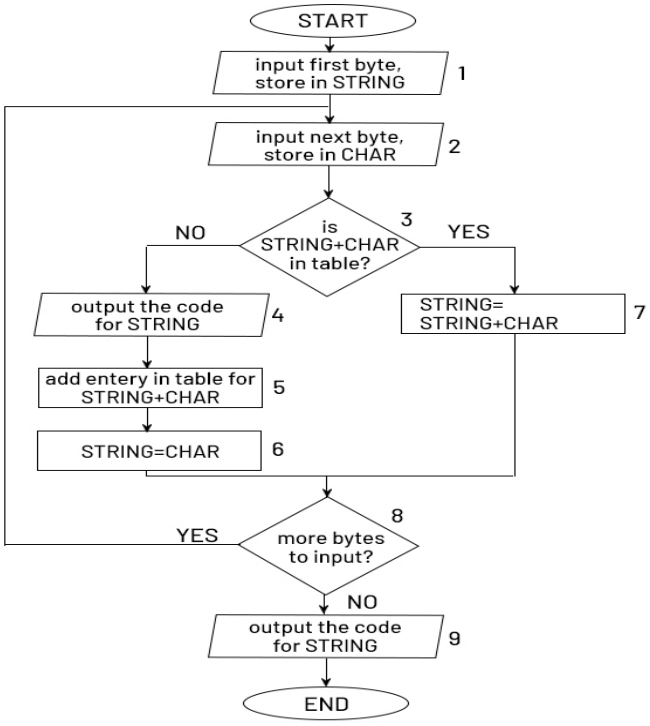

Figure 1: *LZW Algorithm Flowchart.*

CHAR variable is sign bit. STRING variables are byte length arrays. The data is read as a single byte in the input-1 and input-2 boxes and written as a compressed 12-bit code in the input-4 box. (Uçar E ,2011).

RLE (Run-Length Encoding)

Although it is an algorithm that can be used for any type of data, it is often used in image compression because it provides a good compression ratio when the same symbol is repeated many times in succession. BMP, PCX and TIFF image file formats can compress with RLE (Albahadilye et al., 2016).

To give an example of the working principle of the Run Length Encoding compression algorithm, let's assume that the data we want to compress is AAAABBCCCCCDD. AAAABBCCCCCDD – 13 characters If we re-encode the characters in this data with the repeating number information, we get 4A2B5C2D – 8 characters. In this method, which is used depending on the number of characters of the data, when two data are compared, it can be said that it has decreased from 13 characters to 8 characters and the space it occupies has decreased. However, since this situation depends on the number of repeating characters in the data, it can be said that the size of the data to be compressed can naturally increase. If "BABA" data is to be compressed, BABA-4 characters will be 1A1B1A1B-8 characters when compressed with RLE method (Şişman Ç.,2014). In this case, the size of the data grows more. If the data to be compressed contains only 0- and 1-bit values and these values are repeated one after the other, the RLE method may be more effective (Wijaya M.C., 2021). In addition, when the flow diagram of the algorithm presented in Figure 3 is examined, it will become clear how the data compression process is handled.

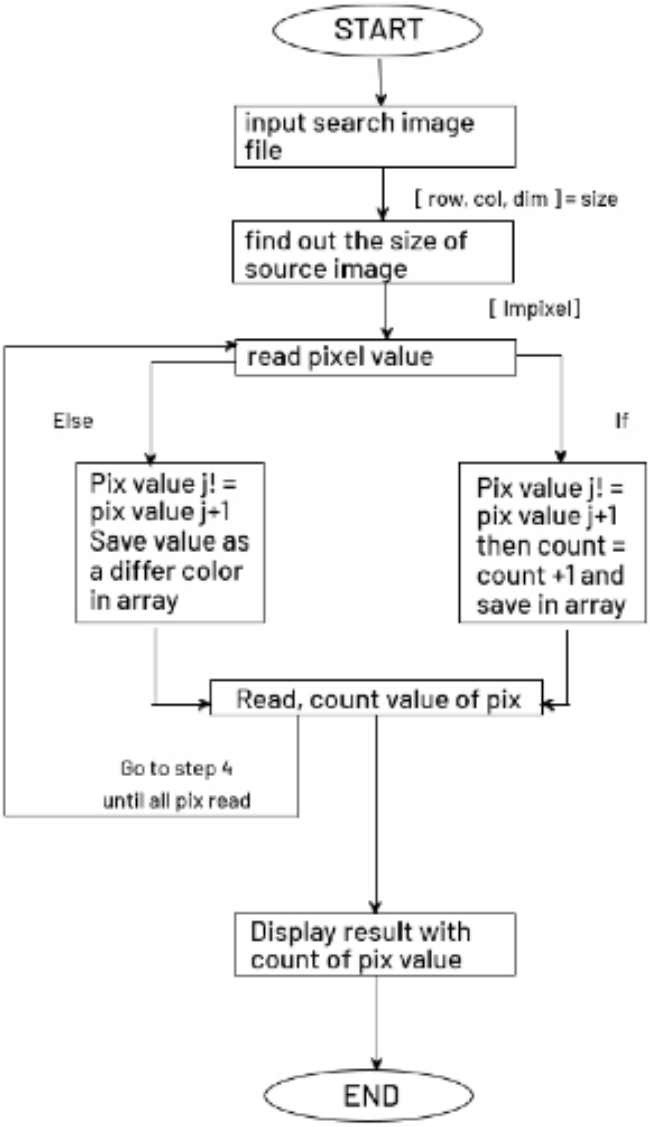

Figure 2: *RLE Algorithm Flowchart.*

When 32-bit data is compressed with the RLE method, the data consisting of 0 and 1 values, the number of consecutive bit sets are coded in the binary system and expressed as 0 and 1 bits again. As can be seen in Figure 3, the compression ratio of 32-bit data is 37.5%, since it is 20-bit in size (Ibrahim et. al.,2015).

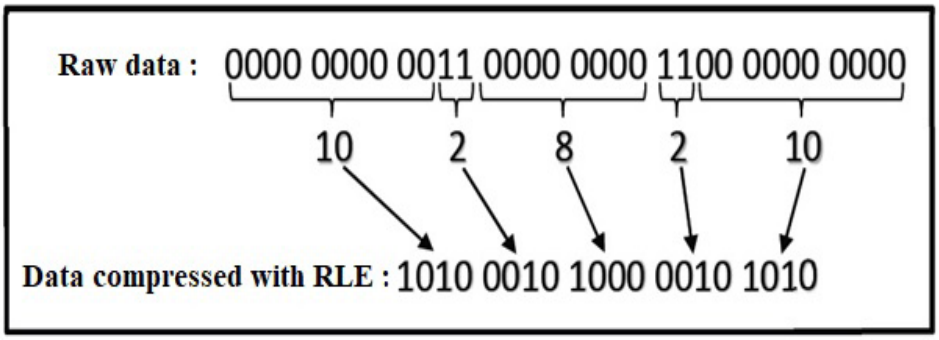

Figure 3: *RLE Algorithm compression example.*

Case study

RLE and LZW image compression algorithm functions are used in the application developed using C# programming language in Visual Studio environment. In the application, any image file in any format is moved from its location to the designed form. After the image is transferred to the form, the image compression algorithm that is desired to be applied is preferred with the radio buttons in the form, as shown in figure4.

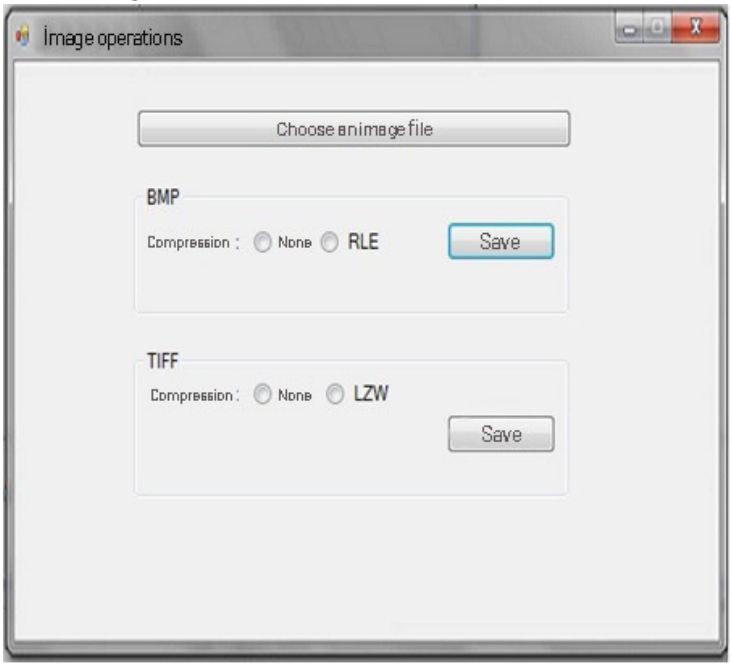

Figure 4: *Application form.*

Figure 5: Application form 2.

The image files used in the comparison were selected in JPEG format. Image files in different formats can also be preferred, but the preference of JPEG format is that the difference in file sizes can be seen clearly when the compression process is performed with RLE and LWW algorithms.

The image files used in the comparison are shown in figure 6. Below each image are the name of the file to which that image belongs, width and height pixel values. The image files named in Figure 6 were chosen as two-color and multi-color. In order to avoid duplication of the study, two image files that were tested in total were added to Figure 6. By testing the compression process with image files of different file formats and different sizes, results close to the results obtained from the image files in figure 6 were obtained.

Figure 6: Tested image files.

The image files in Figure 6 were used to compare the RLE and LZW compression algorithms. Two image files in JPEG format have been compressed. The changes in the file sizes after and before the compression process are included in the tables created by using numerical data in the conclusion part of the study.

4. Discussion and conclusion

In the study, two image files in JPEG format were compressed with RLE and LZW compression algorithms, and the file sizes as a result of the compression process were compared. The numerical data about the size of the image files obtained as a result of the compression process are shown in table1 for examining the changes in the space of the files on the disk as a whole.

Name Of the File	File Format	Size On Disk	Algorithm	Pixel Values
1-two color jpg	JPEG	604 KB	JPEG	545X255
2-multi color jpg	JPEG	20 KB	JPEG	470x470
1-two color jpg	BMP	404 KB	RLE	545X255
1-two color jpg	TIFF	588 KB	LZW	545X255
2-multi color jpg	BMP	652 KB	RLE	470x470
2-multi color jpg	TIFF	316 KB	LZW	470x470

Table 1: Sizes of image files after compression

When Table 1 is examined, it can be seen that the JPEG image file named 1-two color jpeg has a disk size of 604 kilobytes, and the JPEG image file named 2-multi color jpeg has a disk size of 20 kilobytes. When we compress these two image files with RLE and LZW algorithms, 1-two-color jpeg file consisting of two colors, black and white, is initially in JPEG format and 604 KB, when compressed with the RLE algorithm, the area occupied by the file on the disk is reduced by about 33%, up to 404 KB, LZW When it is compressed with the algorithm, it has decreased by about 1%. This shows that even an image file in JPEG format, which is one of the most advanced compression algorithms, should be compressed with the RLE algorithm if it consists of two colors, and BMP, etc., which supports the RLE algorithm. Storing or processing in file formats is more advantageous in terms of file volume. In the same way, more advantageous results were obtained with the RLE compression algorithm than the LZW compression algorithm in two-color image files. When the RLE algorithm is compared to the compression of multi-colored image files with the LZW algorithm, the situation is reversed, as can be seen in table 1. 2- When the multicolor image file named multi color jpeg is compressed with the RLE algorithm, the floppy size is 652 KB, while when it is compressed with the LZW algorithm, the floppy disk size is reduced to 316 KB. has fallen so far. Compression operations show that the volume of big data created by all instantaneous visual data must be controlled. In order to fulfill the data integrity rule, which is one of the conditions necessary for the effective management of big data, it is necessary to store the original data in the least amount of space without damaging it.

REFERANCES

[1]Albahadiliy H. K., Tsviatkou V. U., Altaay A. A., Kanapelka V. K., (2016). New Modified RLE Algorithms to Compress Grayscale Images with Lossy and Lossless Compression, International *Journal of Advanced Computer Science and Applications, vol:7, no:7,* DOI: 10.14569/IJACSA.2016.070734

[2]Altan M., (2006). *Veri Sıkıştırmada Yeni Yöntemler,* Trakya University Institute of Science and Technology, Department of Computer Engineering, PhD Thesis, Edirne, 2006

[3]Bulut F., (2016). Huffman Lossless Fast Text Compression by Al-Jazari Algorithm. *Journal of Science and Engineering* 3(2): 287-296.

[4]Ibrahim, A. M. A., & Mustafa, M. E. (2015). Comparison between (rle and huffman) algorithmsfor lossless data compression. *International Journal of Innovative Technology and Research,* 3(1), 1808-1812.

[5]Kaya Ş. M., Erdem A ., & Güneş A., (2022). Anomaly Detection and Performance Analysis by Using Big Data Filtering Techniques For Healthcare on IoT Edges. *Sakarya University Journal of Science Institute* 26 (1), 1-13

[6]Kaya, Ş. M., Erdem A., & Güneş A., (2021). A Smart Data Pre-Processing Approach to Effective Management of Big Health Data in IoT Edge. Smart Homecare Technology and TeleHealth, 8, 9-21.

[7]Kaya, Ş. M., (2021). *A smart data pre-processing approach for effective management of healthcare big data on IoT edges,* Istanbul Aydın University, Graduate School of Natural and Applied Sciences, Department of Computer Engineering, PhD Thesis.

[8]Kaya, Ş. M., Güneş A., & Erdem A. (2021). A Smart Data Pre-Processing Approach by Using ML Algorithms on IoT Edges: A Case Study. In 2021 *International Conference on Artificial Intelligence of Things* (ICAIoT) (pp. 36-42). IEEE.

[9]Shrividhiya, G., Srujana, K. S., Kashyap, S. N., & Gururaj, C. (2021, March). Robust data compression algorithm utilizing LZW framework based on huffman technique. In 2021 *International Conference on Emerging Smart Computing and Informatics (ESCI)* (pp. 234-237). IEEE.

[10]Şişman Ç., (2014). *Investigation of Data Compression and Data Encryption Algorithms Together on Network and Performance,* Trakya University, Institute of Science and Technology, Department of Computer Engineering, Master's Thesis, Edirne.

[11]Uçar E. (2011). *Investigation of the Effects of Compressed Raster Images on Quality and Accuracy in Photogrammetric Automation,* Selcuk University, Institute of Science and Technology, PhD Thesis, Konya.

[12]Wijaya M. C., (2021). Comparative Analysis of Performance Run Length (RLE) Data Compression Design by VHDL and Design by Microcontroller, I.J. *Modern Education and Computer Science,* 6, 11-24, DOI: 10.5815/ijmecs.2021.06.02.