International Journal of Engineering & Technology, 7 (2.34) (2018) 34-38

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Using Pre-Determined Patterns to Analyze the Common Behavior of Compressed Data and Their Compressibility Apeal

Kamal Al--Khayyat [1]*, Imad Fakhri Al-Shaikhli[1], Vijaykumar V[2]

[1]*Department of Computer Science, Kulliyyah of Information and Communication Technology, International Islamic University Malaysia, 53100 Gombak, Selangor, Malaysia*

[2] *School of Computing Science and Engineering, VIT University, Chennai, Tamilnadu, India*

**Corresponding author E-mail: kamal_amk@yahoo.com*

Abstract

This paper studies the behavior of compressed/uncompressed data on predetermined binary patterns. These patterns were generated according to specific criteria to ensure that they represent binary files. Each pattern is structurally unique. This study shows that all compressed data behave almost similarly when analyzing predetermined patterns. They all follow a curve similar to that of a skewed normal distribution. The uncompressed data, on the other hand, behave differently. Each file of uncompressed data plots its own curve without a specific shape. The paper confirms the side effect of these patterns, and the fact that they can be used to measure the compressibility appeal of compressed data.

Keywords: *Compressed Data, Uncompressed Data, Patterns, Compressibility, Randomness.*

1. Introduction

Data compression is a science of reducing the size of data, utilize bandwidth, quicken communication, and reduce storage space.

Despite increased technological advancement, the amount of information grows. Compression techniques simplify the handling of these information. More and more storage space is required to store information on a daily basis.

It is therefore necessary to create a compression algorithm that can compress data. Compressing compressed data is one way to reduce the file size even further.

There has been many attempts at compressing certain types of data [1, 2, 3, 4, 5, 6]. It should also be pointed out that compressed data is regarded as random data [7, 8]. The first compression revolves around redundancy, which removes the need for the next round of compression [7].

It is important that data compression does not contradict the mathematical rules and theories, such as counting theory and Kolmogorov complexity entropy coding [7]. One way to follow up a compression is to analyze the behavior of compressed data. For example, [9] studied the randomness of compressed data in the context of their validity to be regarded as a good RNG (random number generator). The same result (Bad randomness) was also a feature of LZSS [10]. The results confirmed that the compressed data are bad RNGs, meaning that they are not random enough.

[11] concluded that the behaviors of compressed data when measuring their randomness by non-parametric randomness tests remains almost similar.

They examined 49 images compressed by four compression algorithms; two were general algorithms, while the other two were image-dedicated compressors.

Their results showed that the statistical information within each compressed file are rather similar.

The information shows that 50% of the compressed data are a grouping of runs, 50% has positive signs compared to its adjacent values, 66% of the files contain turning points, and using the Cox-Stuart test, 25% of the file report positive signs, which reflects the similarity aspects of the compressed data.

In this paper, we examine and analyze compressed data. The methods used in this work differ from those reported by other works. Pre-determined patterns were selected for analyzing the behavior of compressed data. The pre-determined patterns (henceforth known as the patterns) are, in fact, composed of binary data file; compressed/uncompressed.

We intend to detail the inner structure of compressed data and how they see the patterns to understand compressed data and determine its randomness and ability to be further compressed.

The paper in section (2) describes the patterns, how to extract the patterns and some of their properties. Section (3) shows the results of analyzing compressed data based on these patterns ,whereas section (4) summarize the findings.

2. Pre-determined patterns

The specification of patterns and its use in demonstrating the behavior of compressed /uncompressed data, and a way of utilizing this concept towards examining the appeal of compressibility.

2.1. Specifications of patterns

The patterns were selected to satisfy the following conditions:

- A pattern can be a combination of ones and zeros, but should not contain two successive zeros.
- The patterns must be distinguishable.

Unlike variable-length coding, such as Huffman coding, the patterns are not distinguishable using prefix technique, but using “01”

and “10” to enclose the pattern from the start and end, respectively, which is a variation of Fibonacci code [12].
This way of structuring the pattern implies that the space between any two-adjacent patterns is implicitly known.
For example, if we have the following binary string:
01011101100110
Due to the double zero not being allowed within a pattern, two patterns can be easily distinguished:
0101110110 and 0110

2.2. Fibonacci sequence and patterns

Since there are no two conductive zeros allowed within a pattern, the possible combinations of such patterns will firmly follow the Fibonacci sequence as we increase its length.
As shown in Table 1, if we ignore exchanged order of the pattern of lengths 1 and 2, the number of variations will follow the Fibonacci sequence. Table 2 show the samples of pattern length associated with their respective number of variations.

Table 1: Sample of variations of each pattern

Pattern length	Variations	# of variations
1	0	1
2	-	0
3	010	1
4	0110	1
5	01110 01010	2
6	011110 011010 010110	3
7	0111110 0101110 0110110 0111010 0101010	5

Table 2: Number of variations for each pattern length

Pattern length	Number of variations
1	1
2	0
3	1
4	1
5	2
6	3
7	5
8	8
9	13
10	21
11	34

2.3. Binary data and patterns

The most special thing about the patterns is the fact that any binary data file is composed entirely of a subset of these patterns.
This made the pattern a valid standard that can be used in many ways to measure certain features in a binary file, as we will see shortly.
To determine the subset of the patterns, which is used to represent a binary file, an extraction pattern algorithm is introduced.
The algorithm will use the second zero as the delimiter, which goes along with the specification of the aforementioned patterns.
The algorithm extracts the patterns in a file in such a way that if we put all the extracted patterns next to each other, we will obtain the original binary file. A suitable data structure can be used to store the patterns to simplify getting statistics out of it.

2.3.1. The patterns extraction algorithm

```
N← all the binary bits of the file
i ← 0
while n not empty
pattern ← empty vector
if (N[i]==0 && N[i+1] != 0)OR(N[i] == 1)
        pattern ← append(N[i])
else if (N[i] == 0 && N[i+1] ==0 )
        list_of_pattern ← append (pattern)
        pattern ← empty vector
i ← i+1
```

2.4. Pattern statistics

Post-extraction, all patterns made up of a binary file, the next step is to obtain the following statistics:

- The number of frequencies of each pattern
- The number of unique patterns

2.4.1. The number of frequencies of each pattern

This number will show, for example, whether the bigger patterns are repeated more than its smaller counterparts. This information is critical if we want to know the compressibility of a file in the context of the patterns, for example, consider the following statistics: pattern of length 5 appears 21,000 times, whereas a pattern of length 7 appears 25,000 times.
This suggests going further for each variation of both lengths and check the frequencies for further chances of compression. Suppose the patterns with length 5, which has variations with the following frequencies:
01110 repeated 11000
01010 repeated 10000
Whereas the pattern of length 7 has the following statistics:
0111110 repeated 12000
01011110 repeated 2000
0110110 repeated 5000
0111010 repeated 3000
0101010 repeated 3000
This suggests that a pattern of the first variation of patterns of length 7 has more frequency than one or both of the variation of a pattern of length 5.
One may suggest a simple swapping to gain compressing.
Length (7), variation (1), will be represented by the length (5), variation (2):
0111110 will be changed to 01010
The total gain will be:
12,000 x 2 - updating_header_file bits
Updating header file is information required for the decoder, this information, in this case, are just little bits.
If we neglect this small information, we gain approximately:
24000/8 =3000 bytes
For just a simple swapping.

2.4.2. The number of unique patterns

This number reflects the existence of patterns according to the Fibonacci sequence of variations, as per Table 2.
When, for example, the number of variations of length (9) is 13, but 12 is present, we know that a pattern is missing, and we know exactly which.
Again, we may use these missing patterns to judge the compressibility appeal of a file. For example, if a variation of length 9 is missing, we may use it to replace the bigger patterns.
Metadata, or information needed at the beginning of the decoded file must be considered.

2.5. Stock market forecasting and analytical techniques

Compressed data is considered as a random or random-like data. It is generally difficult to recompress compressed data because the first compressor got rid of redundancy, and so further working on redundancy is likely to fail.
Analyzing compressed data in the context of the patterns can lead to us knowing more about the behavior of compressed data.

2.5.1. Compressibility of compressed data depending on the patterns

We mentioned in (2.4.1) that if any bigger pattern repeated more any smaller one, a simple swapping can be beneficial.
If a longer pattern, denoted by A, and the smaller one is denoted by B, we get:

[Length (A) - length (B)] * freq(A) > metadata (1)

Eq. (1) shows the practical benefits of these patterns.
When compressed data are behaving randomly enough, Eq. (1) is always unsatisfied.
To measure the compressibility of a compressed data numerically, Eq. (1) can be adapted to:

[[length(A)- length(B)] * freq(A) - meta_data]/ size(file) (2)

When Eq. (1) is fulfilled, Eq. (2) will be applied to give, exactly in percentage, the compression gains of swapping.

Example:
Suppose a file of size 1,000,000 bits contains a pattern of length (9) repeated 12,000 times, whereas a pattern of length (7) is repeated 10,000 times. Assuming that a metadata is 40 bits, according to Eq. (1), we found:
(9-7) * 12,000 > 40 is true
Eq. (2) will follow:
[(9-7) * 12,000 – 40] / 1,000,000 = 0.2396
Which means that ~2.4% of the file will be reduced in the case of swapping.
If Eq. (1) is fulfilled more than once, the results of Eq. (2) will be acculturated, which means that the percentage will increase.
In terms of the missing patterns, if the compressed data behave randomly enough, the missing patterns will not occur at early lengths. As the experimental results show, there is always a peak when drawing unique pattern numbers, and after the falling down out of the peak, we must be able to see the missing patterns. If a missing pattern appears before this peak, compressibility becomes more likely.

For example:
If a pattern of length (11) is missing (denoted by M), a search for a bigger pattern denoted by X must be committed until the following condition is met:

Length(X)-length(M) * freq(X) > metadata (3)

All patterns, X, that are longer than M, must be experimented upon according to Eq. (3). At the end, the pattern that resulted in maximum gain will be used as per below:

Max[length(X)-length(M) * freq(X)- meta_data] /size(file) (4)

For example:
A pattern of length (15) with frequency of 1,000, and another of length (17) with a frequency of 700, assuming that the meta-data is 60 bits, then the gain for a pattern of length (15) becomes:
(15-11) * 1,000 – 60 = 3,940 bits
And for the pattern of length (17) is:
(17-11) * 700 – 60 = 4,140 bits
Here, if we use the missing pattern M instead of a pattern of length 17, we gain 4140 bits reduction via equation (4):
[(17-11) * 700-60]/ 1,000,000 =
0.41 % of the file can be reduced by swapping.

3. Experimental study and analysis

Ten grey images of raw format (PGM) were used as inputs to three types of compression algorithm working as a sample of compressors. The images were selected to be diverse, as per Fig. 7.
These compressors are:

- jpeg-ls: is a lossless image compression. It works by predicting the next pixel value basing on the MED (Median Edge Detection) technique, and uses Glomb-Rice for encoding [13][14][15].
- bz2: Bzip2 (Bz2 for short) concatenates RLE, Burrows-Wheeler transform, and Huffman coding. 7-zip can be used for bz2 format.
- jpeg2000: is a wavelet-based lossy-to-lossless transform coder[16], irfanview can be used for jpeg2000 and jpeg-ls formats.

Along with the original files in the (PGM) format, the output of these compressors was tested using our analysis method. Fig. 1 shows the diagram of the full process.
As per Fig. 1, it was suggested that the binary files will be examined, and the patterns found will be saved within an appropriate data structure.
When the patterns stored (or extracted into) data structures, counting operations will be applied to obtain the required statistics.
Section (2.4) explained the use of such statistics.

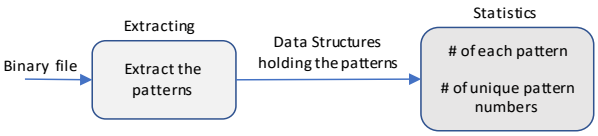

Fig. 1: The process of getting statistics of the patterns

3.1. Behaviour of compressed data

Different types of compressed data (jpeg-ls, bz, jp2000) behave similarly when their unique pattern numbers are plotted.
Regardless of the compression ratio, the pattern behaves similar to a skewing normal distribution.
Table 3 shows the size of the experimental files in the (PGM) format and all other compressed data.
After its peak, the number of unique patterns starts to degenerate.
The shape of the graph, in this case, serves as a fingerprint of the compressed data regarding the patterns. Fig. 2 shows the graph of jpeg-ls, only the first seventy patterns, while Fig. 3 shows one file of the jpeg-ls, which has a long tail, but has the general shape of other jpeg-ls files. Long tails not always a characteristics of the curve's shape; some have long tails and some do not.

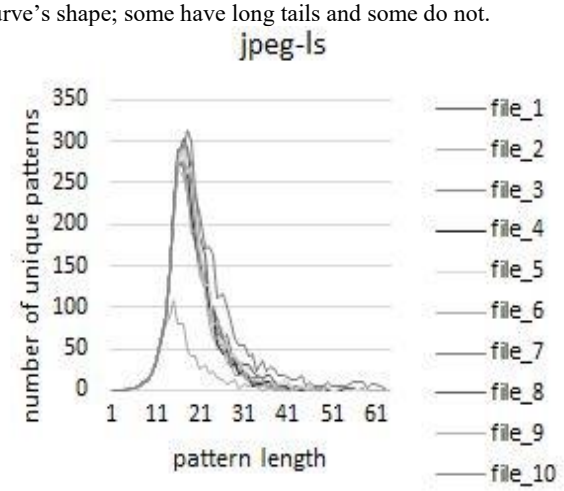

Fig. 2: Unique patterns of jpeg-ls files

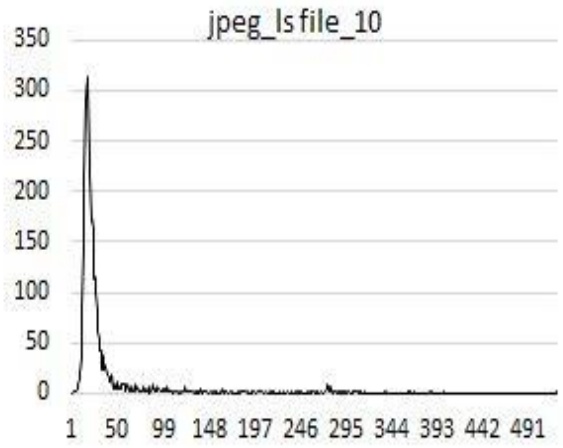

Fig.3: Unique patterns of jpeg-ls file number (10)

Figs. 4 and 5 show the graph of bz2 and jp2000 files, respectively. Unlike jpeg-ls and bz2, jp2000 curves are shown to be more coincidental and in harmony. jpeg-ls curves were, irrespective of general shape, jagged. The peaks also show different magnitudes, unlike jp2000.

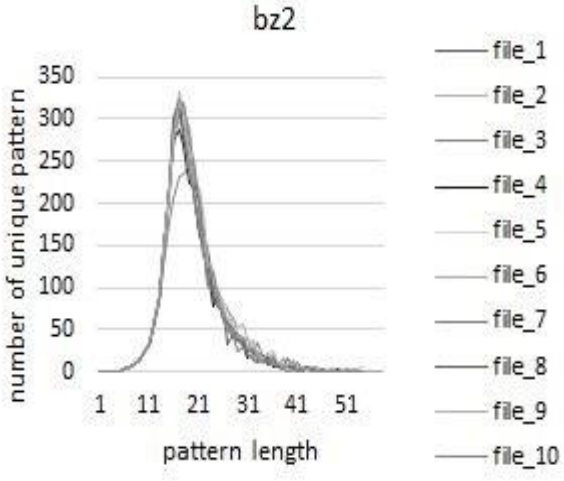

Fig. 4: Unique patterns of Bz2 files

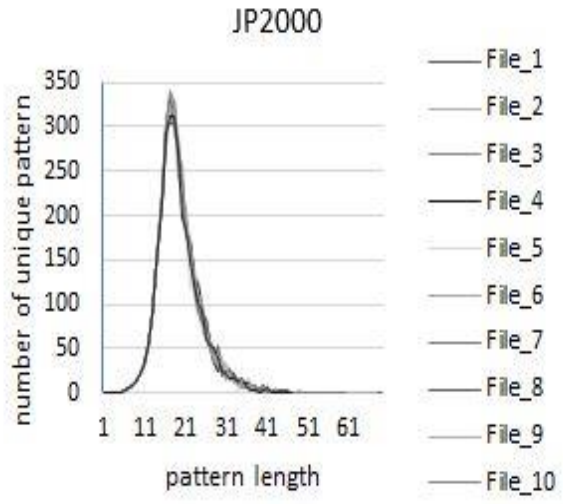

Fig. 5: Unique patterns of Jp2000 files

It is important to note that the details on the curves could change for the other samples. Only the general shape is the common behaviors for all the compressed data used in the experiment.

Table 3: The sizes of ten images and before and after compression

# File	PGM	Jpeg-ls	Jp2000	Bz2
1	262,182	159,340	156,885	202,148
2	262,182	154,391	158,497	183,466
3	262,182	169,199	193,443	97,885
4	262,182	142,043	155,694	128,530
5	262,182	200,485	207,917	187,280
6	262,182	176,355	184,987	157,711
7	262,182	208,227	212,781	154,920
8	262,182	106,880	127,875	95,715
9	262,182	234,748	241,057	119,044
10	262,182	52,331	128,644	47,729

3.2. Behaviour of uncompressed data

To observe the behavior of testing uncompressed data, the (PGM) files can be used. Fig. 6 shows the result, which totally differ from that of the compressed data.
This result confirms the unique behavior of compressed data.

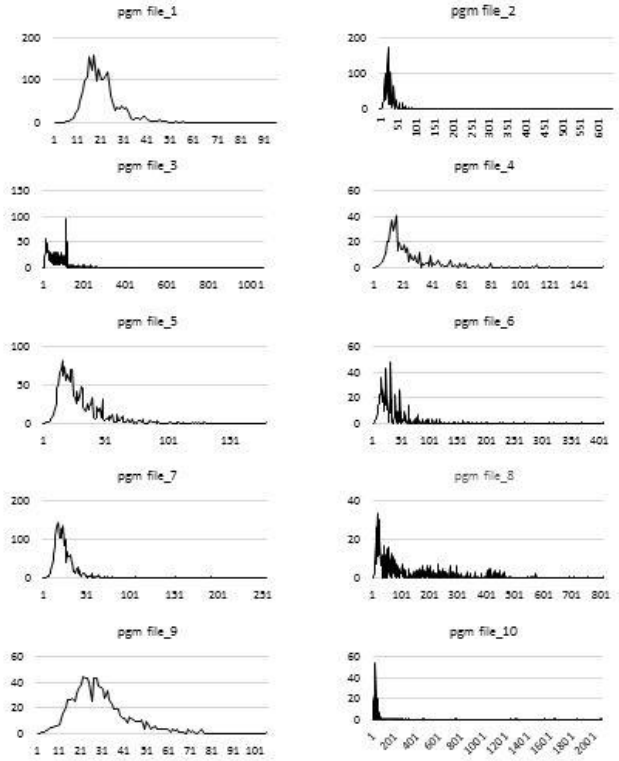

Fig. 6: Unique patterns of ten images of PGM format

4. Conclusion and future work

We confirmed that compressed data of three different data compressors behave almost similarly when analyzing their inner structures as per the predetermined binary patterns. The fingerprint of compressed data was shown as a curve similar to a skewed normal distribution curve. These patterns follow the Fibonacci sequence, and could help us to understand compressed data and how they differ from that of uncompressed data. Each file in the uncompressed data behaves differently from each other. When a curve is plotted for any data, one can decide whether it is random (or compressed data) or non-random (or uncompressed data) just by looking at the shape of the curve.
The patterns not only serves to draw the common fingerprint of compressed data, and they have another use.
Since the patterns are known and follow the Fibonacci sequence, the compressed data can show compressibility appeal when a smaller length pattern is repeated less than its longer counterpart, and when a short pattern is missing from the file. In the case of the former, a simple suggestion is to swap when a certain criterion was satisfied. In the case of the latter, the missing short pattern can play a key role in compression. It can be used to re-

place longer patterns in a manner that maximizes the gain of compression.

This paper addressed the theory of compressibility appeal and provide equations and examples of how it can be determined. Its practical implementation is currently being explored by the authors.

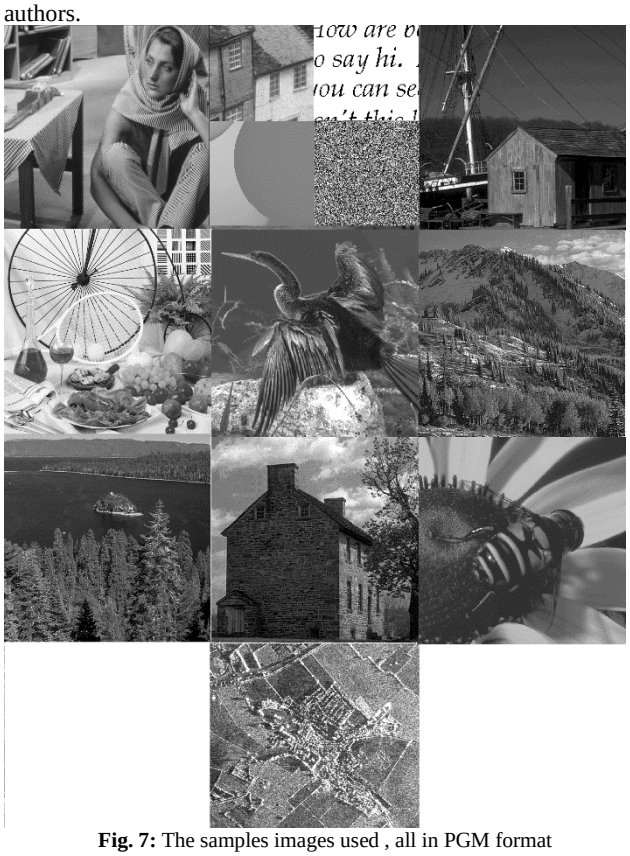

Fig. 7: The samples images used , all in PGM format

Acknowledgements

This material is based upon work supported
by the IIUM under Grant No. RIGS16-366-0530

References

[1] I. Bauermann and E. Steinbach, "Further lossless compression of Jpeg Images," in In Proceedings of PCS 2004 – Picture Coding Symposium, CA, 2004.

[2] N. Ponomarenko, K. Egiazarian, V. Lukin, and J. Astola, "Additional lossless compression of JPEG images," in Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on, 2005, pp. 117–120.

[3] M. Stirner and G. Seelmann, "Improved redundancy reduction for JPEG files," in Proc. Picture Coding Symposium (PCS 2007), 2007.

[4] I. Matsuda, Y. Nomoto, K. Wakabayashi, and S. Itoh, "Lossless reencoding of JPEG images using block-adaptive intra prediction," in Signal Processing Conference, 2008 16th European, 2008, pp. 1–5.

[5] I. Matsuda, K. Wakabayashi, Y. Ikeda, and S. Itoh, "A lossless reencoding scheme for MPEG-1 video," in Signal Processing Conference, 2009 17th European, 2009, pp. 1834–1838.

[6] M. Hasan, K. M. Nur, and H. Bin Shakur, "An Improved JPEG Image Compression Technique based on Selective Quantization," Int. J. Comput. Appl., vol. 55, no. 3, 2012.

[7] D. Salomon, A guide to data compression methods. Springer Science & Business Media, 2013.

[8] B. Waggoner, Compression for great video and audio: master tips and common sense. Taylor & Francis, 2010.

[9] W. Chang, B. Fang, X. Yun, S. Wang, X. Yu, and M. Ethodology, "Randomness Testing of Compressed Data," vol. 2, no. 1, pp. 44–52, 2010.

[10] W. Chang, X. Yun, N. Li, and X. Bao, "Investigating Randomness of the LZSS Compression Algorithm," in Computer Science & Service System (CSSS), 2012 International Conference on, 2012, pp. 2001–2006.

[11] V. V. Kamal A. Al-Khayyat, Imad F. Al-Shaikhli, "ON THE RANDOMNESS OF NON-PARAMETRIC RANDOMNESS TESTS AND THEIR STATISTICAL," Bull. Electr. Eng. Informatics Univ. Ahmad Dahlan., vol. Vol 7, No, p. 63~69, 2018.

[12] D. Salomon, Variable-length codes for data compression. Springer Science & Business Media, 2007.

[13] M. J. Weinberger, G. Seroussi, and G. Sapiro, "LOCO-I: A low complexity, context-based, lossless image compression algorithm," in Data Compression Conference, 1996. DCC'96. Proceedings, 1996, pp. 140–149.

[14] M. J. Weinberger and G. Seroussi, "From LOCO-I to the JPEG-LS Standard," 1999.

[15] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standandization into JPEG-LS," IEEE Trans. Image Process., vol. 9, no. 8, pp. 1309–1326, 2000.

[16] D. Tabuman and M. Marcellin, "JPEG2000: Image Compression Fundamentals, Standards and Parctice." Norwell, MA: Kluwer, 2002.