An Efficient Data Compression Approach based on Entropic Coding for Network Devices with Limited Resources

Elie Tagne Fute, Hugues Marie Kamdjou, Alain Bertrand Bomgni, and Armand Nzeukou

Abstract—The expansion of sensitive data deriving from a variety of applications has required the need to transmit and/or archive them with increased performance in terms of quality, transmission delay or storage volume.

However, lossless compression techniques are almost unacceptable in the application fields where data does not allow alterations because of the fact that loss of crucial information can distort the analysis.

This paper introduces MediCompress, a lightweight lossless data compression approach for irretrievable data like those from the medical or astronomy fields. The proposed approach is based on entropic Arithmetic coding, Run-length encoding, Burrows-wheeler transform and Move-to-front encoding. The results obtained on medical images have an interesting Compression Ratio (CR) in comparison with the lossless compressor SPIHT and a better Peak Signal to Noise Ratio (PSNR) and Mean Squared Error (MSE) than SPIHT and JPEG2000.

Index Terms—Big Data; Entropic Coding; Image Compression; Lossless Compression; MSE; PSNR; Telemedicine

I. INTRODUCTION

For the World Health Organization specialized group, telematics is defined as a form of cooperative medical practice connecting from a distance a patient and a doctor (or several health professionals) through information communication technology [1]. In some of its applications such as medical imaging, telemonitoring or remote monitoring of a patient, remote control, telemedicine and telesurgery [2]. The amount of digital medical images is steadily increasing, which requires a crucial treatment on quality and size of the images to be stored because of their nature or because of the context in which the compression takes place (because of regulations or legislations). These methods aim at reducing images to a more compact form at the cost of compression and decompression effort [3].

This paper proposes an efficient lossless data compression technique for resource-limited devices. We name this approach as MediCompress and apply to images compression while optimizing the processing time and guaranteeing the integrity of the original image. MediCompress is based on entropic coding and does not allow alterations. The main benefit of this approach is its low complexity. The data is reduced without any quality loss, while reducing the image data size by removing unnecessary meta-data from the submitted files. Nevertheless, the original data can be restored and reconstructed from the compressed file data. Our lossless approach will only temporarily delete the file data, this allows it to be transferred quickly, to optimize loading and processing speeds in memory. While the amount of space that will save is not as much as if we were to use lossy compression, it does give us higher quality images and the option to fully restore.

The remainder of this work is organized as follows: in section 2, we review relevant existing research on data compression; in section 3, we present our approach MediCompress and its main components; in section 4, we present the implementations and interpretation of results. Finally section 5 provides conclusion and future directions.

II. RELATED REVIEW

A. Theoretical framework

Data compression is considered the starting point for Claude Shannon’s 1948 publication of "The Mathematical Theory of Communication" [4]. In this theory he particularly introduced the notion of quantity of information I(X = x) of an event X = x, and the entropy H (X) of a discrete variable which defines the limit of X compression. Image compression technique basically divided into two parts: lossy technique and lossless technique. Lossless compression algorithm involves no loss of information. If data have been losslessly compressed the original data must be recovered exactly from the compressed data. Lossy compression algorithm involves some loss of information and data that have been compressed using lossy technique generally cannot be recovered or reconstructed exactly.

B. Variable length coding, adaptive Huffman coding

The general idea of this type of code is to assign a shorter code to the most frequent characters, and a longer code to the least frequent characters. Adaptive Huffman encoding requires prior knowledge of the probability of symbols appearing, and then the encoding procedure is as follows: we identify a prefix code from a discrete source X to a
binary tree with $|X|$ labeled sheets by the letters of $X$. Suppose we have already read $n$ characters in the text, corresponding to $K$ separate letters. We make $X_a$ a source of $(K+1)$ letters consisting of $K$ letters already appeared to which we assign a probability proportional to the number of occurrences, and a $(K+1)^{th}$ letter (empty) to which we attribute the probability $0$; we build a Huffman code $T_a$ from this source; the $(n+1)^{th}$ letter is read and is coded by its codeword if it exists, by the $(K+1)^{th}$ codeword followed by the ASCII code otherwise. For this purpose, Huffman uses a tree based on the basic notions of information theory (Shannon-Fano) namely the notion of quantity of information and entropy [5].

C. Adaptive arithmetic coding

This encoding replaces the set of symbols read by a single real number which is between $[0; 1)$. At the first iteration, the interval is divided into $n$ segments of equal length. Each time a symbol is received; the algorithm calculates the new probability distribution and updates the symbols. On the other hand, if we really consider all the possible sequences, this number can include an infinite number of digits in its binary representation [6].

D. Run-length encoding

Its principle is to replace several appearances of the same symbol by a copy of this symbol and the number of times that it appears consecutively in the data [7].

E. Burrows-Wheeler Transform

Transformed from Burrows-Wheeler's principle is to rearrange a string of characters into a series of similar characters. The result contains exactly the same characters as at the input, the only difference is the order in which they appear [8].

F. Dictionary-based method: PPM Compression, LZMA, LZ77, LZ78, LZO, 7-Zip, WinRar, WinZip, Bzip2 and Gzip

7-Zip is the compression algorithm that integrates the following methods: LZMA (Enhanced and Optimized Version of LZ77), LZMA2 (Enhanced Version of LZMA), PPM (partial recognition prediction algorithm which consists of to predict the next symbol to be encoded based on the previous $n$ symbols), BCJ, BCJ2 (Converter for 32-bit x86 executables), Bzip2 (compression using the BWT transform, LZMA and Huffman coding) and Deflate or Gzip (similar to LZO is a standard algorithm based on LZ77 and Huffman). Unlike LZO, the LZ77 no longer uses the sliding window but a dictionary containing the byte strings already encountered. WinRar uses LZ77 and Huffman while WinZip uses LZ77, LZMA, BWT and Huffman [9].

G. The discrete cosine transforms (DCT)

DCT is a mathematical function used for destructive compression of data, especially for sounds, images and videos according to JPEG and MPEG standards. During the data compression operation in these different formats, the algorithm transforms the pixels of the image or the samples of the audio sequence into frequencies, while eliminating the frequencies that do not correspond to the relevant data for the eye or the human ear [10].

H. Joint Photographic Experts Group (JPEG) compression

JPEG compression encodes the colors of the image in any format, however the best compression ratios are achieved with luminance/chrominance color coding because the human eye is quite sensitive to the luminance (brightness) but not very sensitive to chrominance (the hue) of an image. In order to exploit this property, the algorithm converts the original image from its initial color model (usually RGB (Red, Green, Blue)) to the YIQ chrominance/luminance type (the Y, I, and Q represent luminance, interpolation and quadrature respectively [11].

I. Fractal compression

According to fractal compression, any image is a finite set of geometric transformations (rotations, translations, enlargements, reductions) applied to subsets of identical patterns and variable sizes that compose it. This method consists to detect on the one hand the recurrence of the patterns at different scales, and on the other hand to eliminate the redundancy of information in the image [12].

J. Discrete Wavelet Transform (DWT)

DWT is a method of image compression based on mathematical theory of signal analysis. Wavelets are sets of elementary signals from which a complex signal is reconstructed. The idea of this transform is to determine the correlation of several wavelets (compressed or dilated mother wavelet) with the signal, while highlighting the details and the overall look [13]. DWT is a multi-resolution/multi-frequency representation. It is a tool that splits data, functions or operators into frequency components at a scale resolution.

K. Coding of distributed sources

The coding of distributed sources do not allow communication between encoders (decoding is always done jointly) (see Fig. 1) [14].

L. Pipelined In-Network Compression

In this technique, the data collected by the sensor is stored in the buffer of the aggregation node for a certain time lapse [15]. During this time, the data packets are combined into one packet while suppressing the redundancies (see Fig. 2).

M. Analysis of above methods

In view of the above techniques, we realize that some are general while others are specialized in data of a certain nature, for example: images, videos, sounds, text. Some perform a lossless encoding and others, with a loss of information. The former are mandated to reconstruct the original data without any difference after they undergo the compression and decompression steps. The latter may alter the data after compression and decompression provided that the reconstructed data is relatively similar to the original data [16].

![Fig. 1. Example of a Distributed Source Coding System](image_url)
The DCT-based methods, such as the JPEG algorithm, aim to nullify low value coefficients that represent little information in the image, and reduce the dynamics of others. Thus, it introduces “artifacts” ripple due to the truncation of high frequency coefficients; which generates a loss of information. Despite the fact that the DWT is a global transformation, the block effect affects the quality of the image, because of the error due to the quantization of the coefficients. The disadvantage of the dictionary methods is that the size of the dictionary is limited by the available memory [17].

III. THE PROPOSED APPROACH OF COMPRESSION: MEDICOMPRES

MediCompress is a lossless compression approach of data that does not allow alterations. Our aims consist to reduce the amount of information represented by a pixel, without losing the originality of the pixel. We begin by splitting an input image into 8x8 rows and columns, and then compress each block with our algorithm. At final stage sum of all individual compressed bloc which not only provide better result but also the information content will be kept secure.

The architecture (Fig. 3) of this approach is based on the codings: ARITHmetic revised (ARI), Run-Length Encoding revised (RLE), Burrows-Wheeler Transform revised (BWT) and Move-To-Front Encoding revised (MTF). The reason why these codings have been chosen is that they are particularly valuable in constrained resource applications (energy and memory) because their implementations involve simple instructions for adding and offsetting integer values [18].

The encoding inspired by adaptive arithmetic coding consists of replacing the set of symbols read by a single real number that is in the interval [0, 1]. Either to code \( x \in \mathbb{E}^n \) to the order \( k \) chosen beforehand; we proceed by iteration, for \( t \geq 0 \), put \( x^t = x_1; \ldots; x_n \) and suppose treated the first \( t \) symbols, \( t \geq 0 \); \( t = 0 \) meaning that the coding has not started yet. To treat the \( (t+1)^{th} \), we update the probabilities of transitions as follows:

\[
y(t)(i|j) = \frac{n^{(i)(j)}+1}{n^{(i)}+m}, \text{ With } i \in \mathbb{E}; \ j \in \mathbb{E}; \ n^{(i)}(j)\text{ and } n^{(i)}(j)\text{ which are respective occurrence numbers of } i \text{ after } j \text{ and of } j \text{ in } x^t; \ n^{(j)}(j) \text{ not counting an appearance of } j \text{ at the end of this chain. Putting } j = x_{t+1}; \ldots; x_n \text{ the current state and we cut the interval according to the probabilities of } y(t)(i|j), \text{ an interval corresponding to a state } i \text{ of } \mathbb{E}. \text{ We choose as the current interval that corresponds to } x_{t+1}. \text{ Once the last processed symbol and the noted current interval } ([a; b]), \text{ there are two consecutive dyadic numbers } [-\log(b-a)] \text{ in the current range. We take for code } x^n \text{ the fractional part of the largest of these numbers.}
\]

The use of this method of compression has a defect due to the use of a table of fixed statistics translating the same problems as for the Huffman coding. That is to say that the compression of a file with atypical statistics (symbols to be poorly represented are found with a high probability of appearance) will be larger after compression. To overcome this, the most effective solution is to use an adaptive array frequencies have the same law of probability; then at each meeting of a symbol, the statistics and the intervals are adapted (see Algorithm 1 and 2).

In order to increase the chances of having a good compression ratio, we precede this coding of the transformations inspired by Run-Length Encoding, Move-To-Front and Burrows-Wheeler Transform.

The encoding inspired by Run-Length Encoding consist to replace all sequences of characters or similar bits by a number representing the number of repetitions of the character or bit followed by that character. For example, if we have a series of zero (0) in a document, 00000 will be replaced by a message of type 50. Therefore, a means is needed to differentiate the message 50 from the sequence 50 in the original data. For this purpose, we use an escape sequence that is to say a new symbol that is introduced into our alphabet. It suffices then to transmit the escape sequence before transmitting the number of times a symbol appears (see Algorithm 3 and 4) [14].

This coding is effective for compressing data where repetitions of consecutive elements are numerous (black-and-white image), which is not always the case; this is why we precede this coding of transformations inspired by Move-To-Front and Burrows-Wheeler Transform; in order to increase the chances of having more characters or repeated bits.

The encoding inspired by Move-To-Front encoding is intended to bring up zero (0) when the input word contains identical character sequences, thus producing highly compressible data (see Algorithm 5 and 6). The transformation of a chain \( L \) of length \( N \) into a vector \( R \) of length \( N \) is as follows:

- The characters of an alphabet \( A \) are placed in a list \( Y \) which contains only one and only appearance of each character.
- For each \( i \), such that \( i = 0, \ldots, N-1 \):
  - \( R[i] \) is equal to the number of characters preceding the character \( L[i] \) in the list \( Y \).
  - If the code is the same as that of the last coded character, then they are identical: we return zero (0).
  - \( L[i] \) is placed at the top of list \( Y \) (the other characters are shifted).

We precede this Burrows-Wheeler Transform-based transform coding, to increase the chances of having more consecutive characters or bits repeated.

The encoding inspired by Burrows-Wheeler's transform coding consists of rearranging a string of characters into sets of similar characters. The result contains exactly the same characters as the input. The only difference is the order in which they appear. This is useful for compression because it tends to be easy to compress a string that has a series of repeated characters. Principles of the algorithm:

- The first step is to read in a block of \( N \) symbols \( C_0 \ldots C_{N-1} \).
- The next step, we consider a block as a cyclic buffer with \( N \) strings (rotations) \( S_0 \ldots S_{N-1} \) which can be constructed so that: \( S_0 = C_0 \ldots C_{N-1} \ldots \) \( S_{N-1} = C_N \ldots C_{N-1} \).
- The third step is to sort lexicographically \( S \): \( S_0 \ldots S_{N-1} \).
The last stage of the transformation is the output of a chain L, composed of the last character of each of the rotations in their sorting order with I the number of the sorted line containing $S_0$

![Architecture of our MediCompress approach](image)

**Algorithm 1**: Component_base_ARI_coding;
1. Input: source_file;
2. Output: compressed_file;
3. BEGIN
4. low -< 0; up -< 1; range -< 1;
5. WHILE (Not_end_of_file) DO
6. symbol -< get symbol;
7. low -< low + range * lower_of_current_symbol;
8. up -< low + range * upper_of_current_symbol;
9. range -< up - low;
10. Output a code so that low≤ code < up;
11. ENDwhile
12. END

**Algorithm 2**: Component_base_ARI_decoding;
1. Input: compressed_file;
2. Output: decompressed_file;
3. BEGIN
4. low -< 0; up -< 1; range -< 1;
5. WHILE (Not_end_of_file) DO
6. value -< get code_of_symbol;
7. symbol -< symbol_suc_that_upper_symbol ≤ (value - low) / (up - low) < low_symbol;
8. range -< up - low;
9. up -< low + range * low_of_current_symbol;
10. low -< low + range * upper_of_symbol;
11. Return symbol;
12. ENDwhile
13. END

**Algorithm 3**: Component_base_RLE_coding;
1. Input: source_file;
2. Output: compressed_file;
3. BEGIN
4. meter -< 0; boolean=<false;
5. previous_symbol -< get current_symbol;
6. WHILE ((current_symbol<-next_symbol) ≥ 0) DO
7. put current_symbol in the_buffer;
8. IF (current_symbol==previous_symbol) THEN
9. count -< 0;
10. ELSE
11. WHILE ((meter < 255)and (current_symbol ≥ 0)) THEN
12. IF (current_symbol==previous_symbol) THEN
13. meter -<-meter + 1;
14. ELSE
15. boolean=<true;
16. ENDIf
17. ENDwhile
18. put meter in the buffer;
19. IF ((meter ≠ 255)and (current_symbol ≥ 0)) THEN
20. put symbol in the_buffer;
21. ENDIf
22. ENDIf
23. previous_symbol<-current_symbol;
24. ENDwhile
25. END

**Algorithm 4**: Component_base_RLE_decoding;
1. Input: compressed_file;
2. Output: decompressed_file;
3. BEGIN
4. meter -< 0; previous_symbol<-current_symbol;
5. WHILE ((current_symbol<-next_symbol) ≥ 0) DO
6. put symbol in the_buffer;
7. IF (current_symbol==previous_symbol) THEN
8. meter -<-next_symbol;
9. ENDwhile
10. put current_symbol in the_buffer;
11. ENDwhile
12. ENDIf
13. previous_symbol<-current_symbol;
14. ENDwhile
15. END

**Algorithm 5**: Component_base_MTF_coding;
1. Input: source_file;
2. Output: compressed_file;
3. BEGIN
4. Initialize the_symbol_table;
5. WHILE (Not_end_of_file) DO
6. S -< get symbol;
7. put position_S_symbol_of_the_symbol_table in the_buffer;
8. move the_symbol_S to the head_of_the_table_symbols;
9. ENDwhile
10. END

**Algorithm 6**: Component_base_MTF_decoding;
1. Input: compressed_file;
2. Output: decompressed_file;
3. BEGIN
4. Initialize the_symbol_table;
5. WHILE (Not_end_of_file) DO
6. P -< get symbol_position;
7. put symbol in position_P_of_the_symbol_table in the_buffer;
8. move the_symbol_Sto the head_of_the_table_symbols;
9. ENDwhile
10. END

**IV. IMPLEMENTATIONS AND INTERPRETATION OF RESULTS**

To find the optimal order of our components (BWT,
MTF, RLE and ARI) namely MediCompress, we compressed the x-ray medical image samples of the tooth, knee, thorax and hand (Fig. 4) [19]. Using all possible cases of the components of our approach (see Table I), although we insist on compression in the description of our experiments, we have always developed the decompress methods to make sure that our experimental results come from valid compression and decompression algorithms implementations.

The results of our experiments obtained on a laptop computer (Pentium 4, HDD: 500 GB, CPU: 2: 5 GHz (4CPUs), RAM: 4 GB and OS: Ubuntu 12.04 32 bits) are presented in Table I, Table II and Table III.

With regard to the plots of the experimental results presented in Fig. 5, we find that the optimal order of the components of our approach is MediCompress, because its red color plot has the best Compression Ratio (CR) defined as:

$$CR = 1 - \left(\frac{\text{image size after encoding}}{\text{original image size}}\right) \times 100;$$
onumber

on the four sample images compared to the other component orders (Fig. 5 and Fig. 6).

A. Comparison and discussion

To verify the effectiveness of our MediCompress approach, on database images (Fig. 4); we did a comparative study of the results with those obtained from lossless compression approaches in classical networks such as: SPIHT, Arithmetic, Huffman, Run-length encoding, Burrows-wheeler transform Move-to-front encoding, Winrar and Winzip; Table II presents the results obtained on four samples of medical images.

The quality of image reconstructed is measure using the Peak Signal to Noise Ratio (PSNR), defined as:

$$PSNR \text{ (in } dB) = 10 \log_{10}\left(\frac{P_{\text{MAX}}^2}{MSE}\right);$$

Here, $P_{\text{MAX}}$ is the maximum possible pixel value of the original image, in this case the pixels are represented using 8 bits per sample, this is 255. The Mean Squared Error (MSE) is defined as:

$$MSE = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} (x_{ij} - y_{ij})^2,$$

with N, M represented respectively the length and width of the image, and $x_{ij}$ represented respectively the value of the original image and reconstructed image.

![Fig. 4.Overview of sample images](image)

![Fig. 5.Overview of sample images](image)

### Table I: Results of experiments (size in bytes)

<table>
<thead>
<tr>
<th>Compressor</th>
<th>Teeth</th>
<th>Knee</th>
<th>Thorax</th>
<th>Hand</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original file size</td>
<td>405054</td>
<td>256390</td>
<td>379369</td>
<td>391554</td>
</tr>
<tr>
<td>ARI-MTF-BWT-RLE</td>
<td>74098</td>
<td>108030</td>
<td>155949</td>
<td>119133</td>
</tr>
<tr>
<td>ARI-MTF-RLE-BWT</td>
<td>74113</td>
<td>108048</td>
<td>155979</td>
<td>119107</td>
</tr>
<tr>
<td>ARI-MTF-BWT-MTF</td>
<td>74099</td>
<td>108038</td>
<td>155988</td>
<td>119112</td>
</tr>
<tr>
<td>ARI-MTF-ARI-RLE</td>
<td>74099</td>
<td>108037</td>
<td>155988</td>
<td>119111</td>
</tr>
<tr>
<td>BW-MTF-ARI-RLE</td>
<td>30196</td>
<td>21834</td>
<td>35904</td>
<td>25303</td>
</tr>
<tr>
<td>MediCompress</td>
<td>28609</td>
<td>21268</td>
<td>33089</td>
<td>23649</td>
</tr>
<tr>
<td>BWT-ARI-MTF</td>
<td>34536</td>
<td>25531</td>
<td>38367</td>
<td>28120</td>
</tr>
<tr>
<td>BWT-ARI-MTF-ARI</td>
<td>34028</td>
<td>24761</td>
<td>37738</td>
<td>27452</td>
</tr>
<tr>
<td>BWT-ARI-RI-RI</td>
<td>70720</td>
<td>94016</td>
<td>134610</td>
<td>110682</td>
</tr>
<tr>
<td>BWT-ARI-MTF</td>
<td>66387</td>
<td>68382</td>
<td>113373</td>
<td>81264</td>
</tr>
<tr>
<td>MTF-ARI-BWT</td>
<td>75936</td>
<td>103705</td>
<td>162111</td>
<td>122351</td>
</tr>
<tr>
<td>MTF-ARI-MTF</td>
<td>74276</td>
<td>100375</td>
<td>151523</td>
<td>118582</td>
</tr>
<tr>
<td>RLE-ARI-MTF</td>
<td>78316</td>
<td>111959</td>
<td>162420</td>
<td>123552</td>
</tr>
<tr>
<td>RLE-ARI-MTF-ARI</td>
<td>33680</td>
<td>25215</td>
<td>42599</td>
<td>29855</td>
</tr>
<tr>
<td>RLE-ARI-MTF</td>
<td>89871</td>
<td>127460</td>
<td>191967</td>
<td>146649</td>
</tr>
<tr>
<td>RLE-ARI-MTF</td>
<td>86618</td>
<td>119482</td>
<td>178546</td>
<td>138232</td>
</tr>
</tbody>
</table>

### Table II: Results of experiments 2 (size in bytes)

<table>
<thead>
<tr>
<th>Compressor</th>
<th>Teeth</th>
<th>Knee</th>
<th>Thorax</th>
<th>Hand</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original file size</td>
<td>405054</td>
<td>256390</td>
<td>379369</td>
<td>391554</td>
</tr>
<tr>
<td>MediCompress</td>
<td>28609</td>
<td>21268</td>
<td>33089</td>
<td>23649</td>
</tr>
<tr>
<td>SPIHT</td>
<td>28341</td>
<td>21268</td>
<td>31762</td>
<td>23529</td>
</tr>
</tbody>
</table>
Based on the plots of the experimental results presented in Fig. 7 and Fig. 8, we find that our approach (MediCompress) whose curve is in red color has a good Compression Ratio compared to the curves of approaches: SPIHT, Arithmetic, Huffman, Run-length encoding, Burrows-wheeler transform Move-to-front encoding, Winrar and Winzip. Table III, Fig. 9 and Fig. 10 show that the MediCompress PSNR (dB) and MSE results have been satisfactory compared to the JPEG2000 standard implemented in OpenJPEG library, and the compressor without loss of the algorithm SPIHT implemented in QccPackSPIHT4 library.

![Comparative histogram Compression Ratio (%) of MediCompress with other compressions approaches](image)

![Comparative Compression Ratio (%) plots of our approach with other compressions approaches](image)

V. CONCLUSION

In this paper, we have proposed MediCompress: a lossless data compression approach using entropic coding, applied to incorruptible data (like those from the medical field, astronomy field, etc.). This approach is based on classical compressions techniques namely Arithmetic, Run-length encoding, Burrows-wheeler transform and Move-to-front encoding. In view of the experimental results (CR, MSE and PSNR), we can say that the results presented throughout this work are satisfactory since MediCompress approach effectively reduced the amount of data contained in medical images, and the reconstructed images without loss of information because of security and legislation purpose for the image.

ACKNOWLEDGMENT

This research was supported in part by the Directorate of Central Africa and Great Lakes Region (DCAGLR) of Francophonie University Association (AUF).

REFERENCES


