A Low Complexity Image Compression Algorithm for IoT Multimedia Applications

Giuseppe Campobello, Antonino Segreto
Department of Engineering - University of Messina (Messina, Italy)

***Abstract*—This paper focuses on a novel lossless image compression algorithm which can be efficiently used for Internet of Things (IoT) multimedia applications. The proposed algorithm has low memory requirements and relies on a simple and efficient encoding scheme. Thus it can be easily implemented even in low-cost microcontrollers as those commonly used in several IoT platforms. Despite its simplicity, comparison results on different image datasets show that the proposed algorithm achieves compression ratios comparable with other more complex state-of-the-art solutions.**

I. Introduction

The Internet of Things (IoT) paradigm has been recently extended to multimedia applications such as, for instance, traffic monitoring [1], unmanned aerial vehicle (UAV) [2], wireless endoscopies [3] and smart glasses [4]. In this context, image compression techniques can be used with the aim of reducing storage resources and increasing lifetime of battery-powered devices [5]. Usually it is assumed that IoT devices have enough storage and processing resources to run complex and powerful algorithms [6]. However, with the aim to achieve low-cost systems, IoT platforms rely often on resource constrained microprocessors with only a few kilobytes of memory [7]–[9]. Therefore low-complexity and low-memory compression algorithms are usually preferred if not mandatory [10], [11].

High performance image compression standards exist, e.g., JPEG2000 [12], [13], H.264 [14] and HEVC [15]. However memory requirements of these compression standards are not compatible with memory resources of several IoT devices.

Compression algorithms can be broadly classified in lossless and lossy compression algorithms [16], [17]. In both cases the first metric considered to evaluate their performance is the Compression Ratio (CR), here defined as the ratio between the number of bits before and after compression. As general rule, lossy compression algorithms allow to achieve much higher compression ratios, however lossless algorithms are preferred in some applications, for instance in biomedical instruments where lossless compression ensure that image details are not lost causing errors in medical diagnosis [18]–[20].

Although several lossless image compression algorithms exist, most of them are not suitable when only limited storage and computational resources are available [21]. Therefore, in the case of resource constrained devices, the trade-off between computational complexity and the achievable compression ratio must be further considered.

In this paper we present a novel lossless image compression algorithm. The proposed algorithm is based on simple arithmetic operations and an efficient encoding scheme, therefore it can be easily implemented even in resource constrained microcontrollers as those commonly used in several IoT platforms.

In this paper we show that, despite its simplicity, the proposed algorithm achieves compression ratios comparable with other state-of-the-art lossless compression algorithms based on more complex predictors and encoding schemes.

The rest of this paper is organized as follows: in Sec. II related works are discussed; in Sec. III the proposed algorithm is introduced; in Sec. IV performance and complexity of the proposed algorithm and other state-of-the-art lossless compression schemes are compared. Finally, conclusions and future works are drawn in Sec. V.

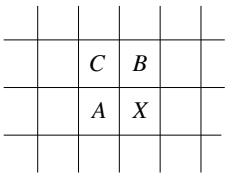
II. Related works

The basic idea behind several lossless image compression algorithms is to exploit spatial correlation to reduce image redundancy. Accordingly, many lossless compression algorithms operate in two phases:

- in the first phase, a predictor is used; basically, in this phase, immediate neighbours pixel are used to obtain an estimate $\hat{x}_{ij}$ of the color of the actual pixel x_{ij}. An example is given by the Median Edge Detection (MED) predictor used in JPEG-LS [22] standard where

$$\begin{aligned}\hat{x}_{ij} &= max\{x_{i-1,j}, x_{i,j-1}, x_{i-1,j-1}\} + \\ &\quad min\{x_{i-1,j}, x_{i,j-1}, x_{i-1,j-1}\} + \\ &\quad -x_{i-1,j-1} \end{aligned} \tag{1}$$

  Other commonly used predictors are reported in Fig. 1 but several techniques exist ranging from nonlinear prediction schemes to more computationally intensive solutions based on fractals and neural networks [23].



#	$\hat{X}$
0	0
1	A
2	B
3	C
4	$A+B-C$
5	$A+(B-C)/2$
6	$B+(A-C)/2$
7	$(A+B)/2$
MED	$max\{A,B,C\}+$ $+min\{A,B,C\}-C$

Fig. 1. Example of predictors used for image compression.

978-9-0827-9703-9/19/$31.00 ©2019 IEEE

TABLE I
A FEW LOSSLESS IMAGE COMPRESSION ALGORITHMS.

Algorithm	GIF [30]	PNG [31]	JPEG-LL [32]	JPEG-LS [22]	CALIC [33]	SFALIC [34]	BTPC [35]	Proposed
Predictor	—	set of 5 linear predictors	set of 8 linear predictors	MED + context modelling	GAP + context modelling	set of 9 linear predictors	set of 7 linear predictors	MED
Encoder	LZW	DEFLATE	Huffman or AC	AC	Huffman or AC	GR	Huffman	RAKE

- In the second phase, the difference $r_{ij} = x_{ij} - \hat{x}_{ij}$, henceforward named residue, is encoded using a codebook specifically optimized to achieve a mean codeword length near the Shannon's entropy [24]. Huffman Coding [25], Arithmetic Coding (AC) [26], Golomb-Rice (GR) [27] [28] and Lempel-Ziv-Welch (LZW) [29] are example of encoding techniques commonly used in this phase.

Compression algorithms based on the previous approach are referred as prediction encoding algorithms. Examples of such kind of compression algorithms are GIF [30], PNG [31], JPEG-LL [32], JPEG-LS [22], CALIC [33], SFALIC [34] and BTPC [35]. We summarize their characteristics in Tab. I.

Comparison results show that, among the above mentioned algorithms, CALIC and JPEG-LS achieves in most cases higher compression ratios [36] at the cost of more complex prediction schemes based on context modelling. Context modelling is able to exploit high-order structures such as texture patterns however requires a considerable amount of memory.

Another class of lossless compression algorithms is based on transforms. A well known and powerful compression algorithm belonging to the this class is SPIHT [37]. SPIHT is mainly based on Discrete Wavelet Transform (DWT) and an efficient sorting technique. SPIHT relies on three steps that cause increase in complexity and need huge memory. Some efforts have been carried out in order to adapt SPIHT to memory-constrained devices [38] [39]. However obtaining a low-memory implementation of SPIHT is still an open research problem [40].

Comparison results reported in [20], [41] show that, when lossless image compression is considered, prediction encoding methods outperform transform based solutions. Therefore in this paper we consider prediction encoding algorithms for comparison purpose.

III. PROPOSED ALGORITHM

The proposed algorithm is based on an efficient encoding scheme for binary sequences recently proposed in [42] and named RAKE (the name derives from the homonymous tool used in agriculture). In [42] the authors showed that in the case of sparse sequences, i.e. sequence with a few non-zero bits, the RAKE algorithm is able to outperform other encoding techniques such as the Run-Length Encoding (RLE) [17].

For the sake of readability, we report a short description of the RAKE algorithm in the next subsection.

A. RAKE algorithm

Basically, RAKE encodes positions of non-zero bits in an efficient manner. We can explain the RAKE algorithm by considering a sliding window of length T that moves forward over the original (uncompressed) binary sequence (see Figure 2). The window operates as a hand-rake able to catch T bits at a time. In particular, every time that T bits are caught, an output codeword is generated according to the following two possible cases:

1) There are no set bits within the window (i.e. all T bits are zeros). In this case a single zero bit codeword is used and the sliding window is moved forward by T positions;
2) At least a set bit is found. In this case a codeword of $1 + \lceil \log_2 T \rceil$ bits is generated where the first bit is set to 1 and the other $\lceil \log_2 T \rceil$ bits are used to encode the position $p \in [0, ..., T-1]$ of the first non-zero bit within the window. Then the window moves forward by $p+1$ positions (i.e. immediately after the set bit that has been already encoded).

The above operations are repeated until the sliding window reaches the end of the sequence to be compressed. It is worth noting that only counting operations are needed for implementing the above procedure.

In Figure 2 a simple example is reported showing how the sequence $S_{in} = [010000001010000]$ of $n = 15$ bits is compressed by the RAKE algorithm to produce a compressed sequence $C_{out} = RAKE(S_{in}) = [10101101010]$ of 11 bits.

According to [42], the following expression provides the optimal value of T for a binary sequence of length n with k non-zero bits:

$$T = \left(\frac{n}{k} - 1\right) \cdot \ln(2) \tag{2}$$

The value of T must be known in order to reconstruct the original sequence, thus a few bits must be added at the beginning of the compressed sequence. As suggested in [42], T can be represented with only $\mathcal{O}(\log_2(\log_2(T)))$ bits by constraining the value of T to be a power of two. In this case the RAKE algorithm has a negligible overhead.

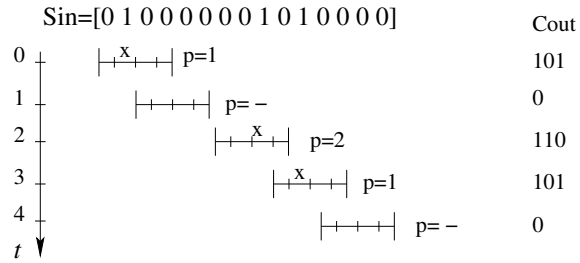


Fig. 2. Example of RAKE compression algorithm ($T = 4$).

B. Using the RAKE algorithm for image compression

The RAKE algorithm can be used in combination with every transformation or pre-processing technique able to obtain a sparse binary sequence from the original data set. For instance, in [42] the authors combined RAKE and one-hot encoding for compression of environmental signals, i.e. temperature and humidity, and in [43] RAKE has been used for compression of ECG signals. In this paper we show how RAKE can be profitably used for image compression.

The basic idea is to apply the RAKE algorithm to the bit planes obtained after prediction. More precisely, we consider an image as a set of blocks of $B \times B$ pixels each and we process each block $[x_{ij}]$, where $i, j \in [1, ..., B]$, in three steps:

1) *MED prediction*: the MED predictor is applied to the original image by obtaining, for each block, a set of residues $[r_{ij}]$; moreover, in this phase, the minimum number of bits w needed for residues representation is evaluated;
2) *Zig-zag remapping*: residues are remapped on the basis of the following equation:

$$r'_{ij} = 2 \cdot |r_{ij}| - (r_{ij} < 0) \tag{3}$$

   As shown in [43] remapping increases sparsity and therefore it is able to improve the RAKE efficiency. Henceforward we referred to remapped values $[r'_{ij}]$ as zig-zag encoded residues.
3) *RAKE encoding*: in this step the RAKE algorithm is applied to zig-zag encoded residues. More precisely, zig-zag encoded residues $[r'_{ij}]$ are logically decomposed into $w+1$ binary matrices, i.e. $A^{(1)}, ..., A^{(w+1)}$, where $A^{(k)}$ represents the k-th bit plane of each block; the RAKE algorithm is applied to each binary matrix $A^{(k)}$ to obtain a set of $w+1$ compressed strings, $RAKE(A^{(k)})$ with $k \in [1, ..., w+1]$.

Finally, compressed strings representing bit planes of each block are concatenated to obtain the compressed block

$$C_{out} = [w, RAKE(A^{(1)}), ..., RAKE(A^{(w+1)})].$$

Note that, from a computational point of view, only counting and simple arithmetic operations are needed for implementing the proposed algorithm.

Moreover the same memory locations used to store the original image can be used to store encoded residues $[r'_{ij}]$. Finally, no additional memory is needed to store the matrix $A^{(k)}$. In fact the element $A^{(k)}_{ij}$ of $A^{(k)}$ coincides with the k-th bit of r'_{ij} so that $[A^{(1)}, ..., A^{(w+1)}]$ is only a logical representation of the residues $[r'_{ij}]$.

C. Considerations on the Block Size

Henceforward we consider blocks of 16×16 pixels each (i.e. $B = 16$). In this case only 256 words at a time have to be stored by IoT devices and this number of words can be easily handled. Moreover, considering a compression factor of two, compressed blocks of 8-bit grayscale images can be accommodated into a payload of less than 128 bytes. This length is compatible with the maximum payload length specified by several wireless communication protocols commonly used for IoT (i.e. IEEE802.15.4, ZigBee and BLE, to name just a few). So that, as usual in edge computing applications, a block at a time can be processed by IoT devices and sent to a second remote device that acts as a gateway and which can offer more storage and computational resources to reconstruct and further process the original image.

IV. Comparison results

In this section we compare the proposed compression algorithm with other state-of-the-art lossless compression techniques in terms of the following metrics:

- *Compression Ratio (CR)*, here defined as the ratio between the number of bits before and after compression;
- *Bits Per Pixel (BPP)*, i.e. the number of bits per pixel required by the compressed image.

Obviously the above metrics are related by $BPP \cdot CR = b$ where b is the number of bits per pixel of the original/uncompressed image.

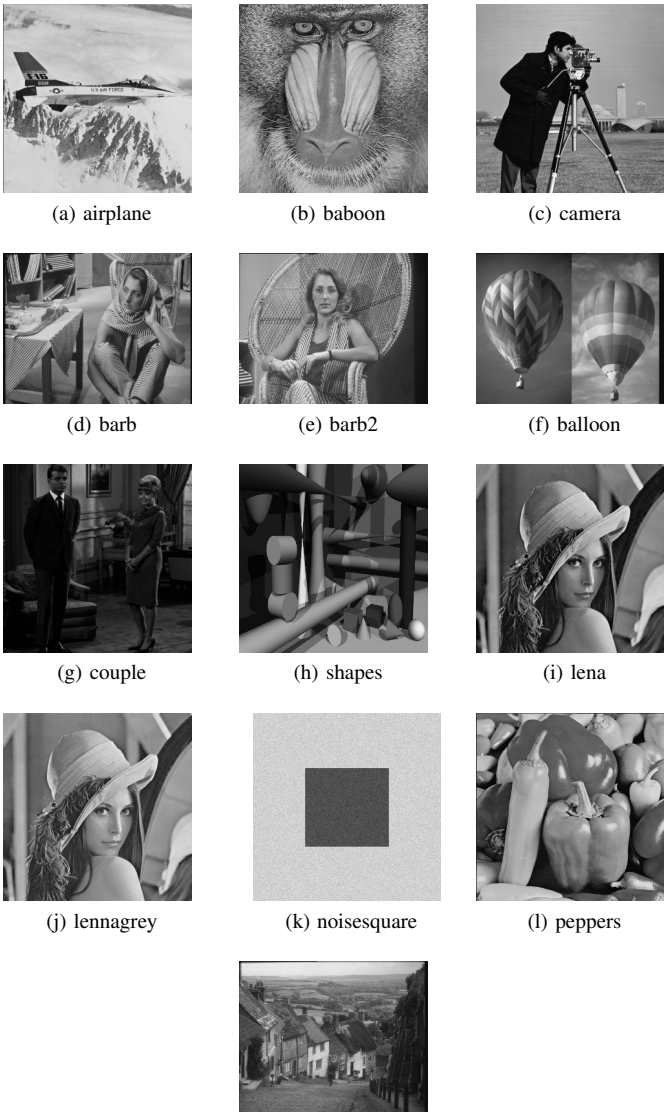
The test images used in this paper are available in [44] and [45]. In particular, test images in [44] contains Lena and other well-known 8-bit grayscale images shown in Fig. 3. Test images in [45] is a set of 1400 monochromatic images with a single closed contour (for sake of space only a few of them are shown in Fig. 4).

At first we proved that RAKE can be more efficient than arithmetic codes (AC). For this purpose we applied both RAKE and AC to the same set of remapped residues obtained with a MED predictor and zig-zag remapping. As it is possible to observe in Tab. II, RAKE achieves better results for almost all images with an average improvement of 0.35 BPP and a maximum improvement of 1.2 BPP (obtained on camera image).

TABLE II
Bits Per Pixel (BPP) achieved with RAKE and AC when applied to the same set of remapped residues.

Image name	Image size (WxH)	AC	RAKE
airplane	(512x512)	4.352	4.047
baboon	(512x512)	6.436	6.194
balloon	(576x720)	3.217	3.123
barb	(576x720)	4.543	4.895
barb2	(576x720)	5.209	5.010
camera	(256x256)	5.713	4.545
couple	(256x256)	4.994	3.894
goldhill	(576x720)	4.805	4.678
lena	(512x512)	5.070	4.874
lennagrey	(512x512)	4.716	4.489
noisesquare	(256x256)	6.733	5.842
peppers	(512x512)	4.977	4.907
shapes	(512x512)	1.687	1.402
	Average BPP	**4.804**	**4.454**

In Tab. III we reported BPP achieved with RAKE and other lossless compression algorithms when applied to the set of grayscale images shown in Fig. 3. As it is possible to observe, despite its simplicity, RAKE outperforms GIF and PNG and achieves compression results similar to those



(a) airplane (b) baboon (c) camera

(d) barb (e) barb2 (f) balloon

(g) couple (h) shapes (i) lena

(j) lennagrey (k) noisesquare (l) peppers

(m) goldhill

Fig. 3. Grayscale images used for tests.

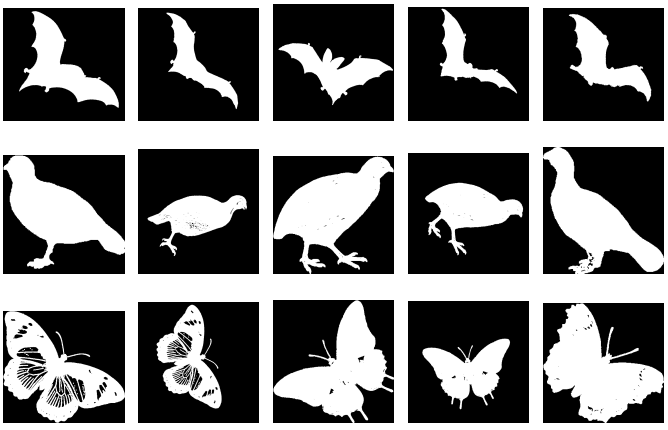


Fig. 4. Some images of the monochromatic dataset used for tests.

TABLE III
BITS PER PIXEL (BPP) ACHIEVED WITH RAKE AND OTHER LOSSLESS COMPRESSION ALGORITHMS ON GRAY-SCALE IMAGES

Image	CALIC	JPEG-LS	GIF	PNG	RAKE
airplane	3.703	3.818	6.547	5.750	4.047
baboon	5.897	6.037	9.360	7.281	6.194
balloon	2.788	2.904	6.315	5.424	3.123
barb	4.339	4.692	8.740	7.011	4.895
barb2	4.480	4.687	8.631	6.871	5.010
camera	4.172	4.316	6.769	5.971	4.545
couple	3.567	3.701	6.599	5.681	3.894
goldhill	4.370	4.477	7.690	6.563	4.678
lena	4.463	4.608	8.502	7.174	4.874
lennagrey	4.087	4.239	8.072	6.828	4.489
noisesquare	5.408	5.685	6.810	5.901	5.842
peppers	4.398	4.513	8.236	7.110	4.907
shapes	0.914	1.214	2.481	1.430	1.402
Average BPP	**4.045**	**4.222**	**7.288**	**6.076**	**4.454**

obtained with CALIC and JPEG-LS. Despite CALIC and JPEG-LS achieve better results, it should be considered that proposed algorithm does not rely to context modelling. As already observed in Sec. II context modelling is used to exploit high-order structures, such as texture patterns, but requires a considerable amount of memory. In particular up to 367 and 576 conditioning contexts are needed respectively by JPEG-LS and CALIC. Therefore, when applied to grayscale images, RAKE provides a highly memory-efficient compression algorithm with about 5% performance penalty compared to JPEG-LS and about 10% compared to CALIC.

Finally, in Tab. IV we reported BPP and CR achieved with RAKE and other lossless compression algorithms applied to the set of 1400 monochromatic images reported in [45]. In the case of monochromatic images we used a slightly modified version of the MED predictor obtained by replacing arithmetic operations in eq.(1) with simple logical XOR operations. As a consequence zig-zag remapping can be avoided. As it is possible to observe, in the case of monochromatic images RAKE outperforms both CALIC and JPEG-LS. In particular, in comparison to JPEG-LS, the RAKE improves the compression ratio by 79%. Even in comparison to CALIC a substantial improvement of 28% is achieved on the compression ratio.

TABLE IV
AVERAGE BPP AND CR OBTAINED WITH RAKE AND OTHER LOSSLESS ALGORITHMS ON A SET OF 1400 MONOCHROMATIC IMAGES

	CALIC	JPEG-LS	GIF	PNG	RAKE
Average BPP	0.067	0.089	0.128	0.107	0.054
Average CR	23.3	16.7	10.2	14.5	29.9

V. CONCLUSION AND FUTURE WORKS

In this paper we have presented a simple and effective algorithm for lossless image compression. Using only arithmetic and counting operations, the proposed algorithm achieves compression ratios comparable with other state-of-the-art lossless compression algorithms based on more complex predictors and encoding schemes. Considering its inherent low complexity and memory requirement, the algorithm is

well suited for resource constrained IoT devices. As future work, the possibility to apply the RAKE algorithm for lossy compression and the trade-off between time and memory complexity will be investigated.

REFERENCES

[1] A. Celesti, A. Galletta, L. Carnevale, M. Fazio, A. Ĺay-Ekuakille, and M. Villari, "An IoT Cloud System for Traffic Monitoring and Vehicular Accidents Prevention Based on Mobile Sensor Data Processing," *IEEE Sensors Journal*, vol. 18, no. 12, pp. 4795–4802, June 2018.

[2] N. Hossein Motlagh, T. Taleb, and O. Arouk, "Low-Altitude Unmanned Aerial Vehicles-Based Internet of Things Services: Comprehensive Survey and Future Perspectives," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 899–922, Dec 2016.

[3] R. Hamza, K. Muhammad, Z. Lv, and F. Titouna, "Secure video summarization framework for personalized wireless capsule endoscopy," *Pervasive and Mobile Computing*, vol. 41, pp. 436 – 450, 2017.

[4] E. D. Buyser, E. D. Coninck, B. Dhoedt, and P. Simoens, "Exploring the Potential of Combining Smart Glasses and Consumer-grade EEG/EMG Headsets for Controlling IoT Appliances in the Smart Home," *IET Conference Proceedings*, January 2016.

[5] O. Ghorbel, W. Ayedi, M. W. Jmal, and M. Abid, "Images compression in WSN: Performance analysis," in *14th IEEE International Conference on Communication Technology*, Nov 2012, pp. 1363–1368.

[6] S. Nastic, H. Truong, and S. Dustdar, "Data and control points: A programming model for resource-constrained iot cloud edge devices," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2017, pp. 3535–3540.

[7] G. Campobello, S. Serrano, A. Leonardi, and S. Palazzo, "Trade-Offs between Energy Saving and Reliability in Low Duty Cycle Wireless Sensor Networks Using a Packet Splitting Forwarding Technique," *EURASIP Journal on Wireless Communications and Networking*, vol. 2010, pp. 8:1–8:11, January 2010.

[8] G. Campobello, O. Giordano, A. Segreto, and S. Serrano, "Comparison of local lossless compression algorithms for Wireless Sensor Networks," *J. Network and Computer Applications*, vol. 47, pp. 23–31, 2015.

[9] A. Kassie and M. Mohammed, "Privacy Preservation of End-to-End Communication for Resource-Constrained Devices in IoT," in *2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, May 2018, pp. 1–9.

[10] S.-W. Lee and H.-Y. Kim, "An energy-efficient low-memory image compression system for multimedia IoT products," *EURASIP Journal on Image and Video Processing*, vol. 87, pp. 1–15, 2018.

[11] G. Campobello, A. Segreto, and S. Serrano, "Data Gathering Techniques for Wireless Sensor Networks," *Int. J. Distributed Sensor Networks*, vol. 2016, pp. 1–17, Mar 2016.

[12] *Information technology – JPEG 2000 image coding system: Core coding system*, ISO/IEC Std. 15 444–1, 2016.

[13] M. Boliek, J. S. Houchin, and G. Wu, "JPEG 2000 next generation image compression system features and syntax," in *Proc. of International Conference on Image Processing*, vol. 2, Sep 2000, pp. 45–48.

[14] *ISO/IEC 14496–10, ITU-T H.264 – H.264 : Advanced video coding for generic audiovisual services*, ISO/IEC, ITU-T Std. 14 496–10, 2008.

[15] *HEVC – High Efficiency Video Coding*, ISO/IEC Std. 23 008–2, 2016.

[16] D. Salomon and G. Motta, *Handbook of Data Compression (5.ed.)*. Springer, 2010.

[17] K. Sayood, *Introduction to data compression (4.ed.)*. The Morgan Kaufmann series in multimedia information and systems, 2012.

[18] J. Kivijärvi, T. Ojala, T. Kaukoranta, A. Kuba, L. Nyl, and O. Nevalainen, "A comparison of lossless compression methods for medical images," *Computerized Medical Imaging and Graphics*, vol. 22, no. 4, pp. 323–339, 1998.

[19] M. Ahmadi, A. Emami, M. Hajabdollahi, S. M. R. Soroushmehr, N. Karimi, S. Samavi, and K. Najarian, "Lossless Compression of Angiogram Foreground with Visual Quality Preservation of Background," in *40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, July 2018, pp. 5158–5161.

[20] K. C. Pathak and J. N. Sarvaiya, "Lossless medical image compression using transform domain adaptive prediction for telemedicine," in *International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, March 2017, pp. 1026–1031.

[21] M. Yang and N. Bourbakis, "An overview of lossless digital image compression techniques," in *48th Midwest Symposium on Circuits and Systems*, vol. 2, Aug 2005, pp. 1099–1102.

[22] *Information technology – Lossless and near-lossless compression of continuous-tone still images: Extensions*, ISO/IEC Std. 14 495–2, 2003.

[23] G. V. M. Lakshmi, "Implementation of image compression using Fractal Image Compression and neural networks for MRI images," in *International Conference on Information Science (ICIS)*, Aug 2016, pp. 60–64.

[24] C.E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.

[25] D.A. Huffman, "A method for construction of minimum redundancy codes," *Proc. IRE*, vol. 40, pp. 1098–1101, 1952.

[26] J. Rissanen and G. G. Langdon, "Arithmetic Coding," *IBM Journal of Research and Development*, vol. 23, no. 2, pp. 149–162, March 1979.

[27] S. Golomb, "Run-length encodings (Corresp.)," *IEEE Transactions on Information Theory*, vol. 12, no. 3, pp. 399–401, July 1966.

[28] R. F. Rice, "Some practical universal noiseless coding techniques," *JPL Publication*, April 1979.

[29] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, May 1977.

[30] "Graphics Interchange Format, Version 89a," W3C, 1990.

[31] *Information technology – Computer graphics and image processing – Portable Network Graphics (PNG): Functional specification*, ISO/IEC Std. 15 948, 2004.

[32] *Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines*, ISO/IEC Std. 10 918–1, 1994.

[33] X. Wu and N. Memon, "CALIC-a context based adaptive lossless image codec," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 4, May 1996, pp. 1890–1893.

[34] R. Starosolski, "Simple fast and adaptive lossless image compression algorithm," *Software: Practice and Experience*, vol. 37, no. 1, pp. 65–91, 2007.

[35] J. A. Robinson, "Exploiting local colour dependencies in binary tree predictive coding," in *International Conference on Visual Information Engineering (VIE)*, July 2003, pp. 37–40.

[36] M. Ali, M. Murshed, S. Shahriyar, and M. Paul, "Lossless image coding using binary tree decomposition of prediction residuals," in *Picture Coding Symposium (PCS)*, May 2015, pp. 194–198.

[37] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243–250, June 1996.

[38] T. W. Fry and S. A. Hauck, "SPIHT image compression on FPGAs," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 9, pp. 1138–1147, Sep 2005.

[39] K. Liu, E. Belyaev, and J. Guo, "VLSI Architecture of Arithmetic Coder Used in SPIHT," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 4, pp. 697–710, April 2012.

[40] H. ZainEldin, M. A. Elhosseini, and H. A. Ali, "Image compression algorithms in wireless multimedia sensor networks: A survey," *Ain Shams Engineering Journal*, vol. 6, no. 2, pp. 481 – 490, 2015.

[41] J. Shukla, M. Alwani, and A. K. Tiwari, "A survey on lossless image compression methods," in *2nd International Conference on Computer Engineering and Technology*, vol. 6, April 2010, pp. V6–136–V6–141.

[42] G. Campobello, A. Segreto, S. Zanafi, and S. Serrano, "RAKE: A simple and efficient lossless compression algorithm for the Internet of Things," in *25th European Signal Processing Conference (EUSIPCO)*, Aug 2017, pp. 2581–2585.

[43] G. Campobello, A. Segreto, S. Zanafi, and S. Serrano, "An Efficient Lossless Compression Algorithm for Electrocardiogram Signals," in *26th European Signal Processing Conference (EUSIPCO)*, Sep. 2018, pp. 777–781.

[44] "8-bit garyscale test images." [Online]. Available: http://www02.smt.ufrj.br/~eduardo/MMP/test_set.html

[45] "Shape descriptors for non-rigid shapes with a single closed contour." [Online]. Available: http://www.imageprocessingplace.com/downloads_V3/root_downloads/image_databases/MPEG7_CE-Shape-1_Part_B.zip