Efficient Adaptive Arithmetic Coding Based on Updated Probability Distribution for Lossless Image Compression

William Puech, Mohamed Selim Bouhlel, Atef Masmoudi

Journal of Electronic Imaging 19(2), 023014 (Apr–Jun 2010)

Efficient adaptive arithmetic coding based on updated probability distribution for lossless image compression

Atef Masmoudi
Higher Institute of Biotechnology
Sciences and Technologies of Images and Telecommunications
Sfax, 3038 Tunisia

William Puech
University of Montpellier II
Laboratory LIRMM
UMR 5506
161, rue Ada
34392 Montpellier Cedex 05
France
E-mail: william.puech@lirmm.fr

Mohamed Salim Bouhlel
Higher Institute of Biotechnology
Sciences and Technologies of Images and Telecommunications
Sfax, 3038 Tunisia

Abstract. *We propose an efficient lossless compression scheme for still images based on arithmetic coding. The scheme presents a novel adaptive arithmetic coding that updates the probabilities of pixels only after detecting the last occurrence of each pixel and then removes the redundancy from the original image effectively. The proposed approach has interestingly low computational complexity. In addition, unlike other statistical coding techniques, arithmetic coding in the proposed scheme is not solely dependent on the pixel probability distribution but also on the image block sorting. The proposed method is compared to both static and adaptive order-0 models while taking into account compression ratios and processing time. Experimental results, based on a set of 100 gray-level images, demonstrate that the proposed scheme gives mean compression ratios that are 5.5% higher than those by the conventional arithmetic encoders as well as significantly faster than the order-0 adaptive arithmetic coding.* © *2010 SPIE and IS&T.* [DOI: 10.1117/1.3435341]

1 Introduction

Recently, many lossless schemes for image compression[1–4] based on arithmetic coding (AC)[5,6] have been proposed. The goal of these schemes is to represent an image with the smallest possible number of bits without loss of any information in order to speed up transmission and minimize storage requirements.[7] AC is a statistical coding technique that uses probability estimates of pixel intensities to assign a single bitstream for the data set and achieve a high compression ratio. The AC needs to work with a modeler that estimates the probability of each pixel at each iteration in the encoding and decoding processes. The modeler needs to find the probability distribution of those symbols that maximizes the compression efficiency and send this information to the decoder as header information to ensure decodability. The models can be static or adaptive. On the one hand, the static model uses fixed probabilities for all pixels, and when AC works with a static model, it is called static arithmetic coding (SAC).[8] On the other hand, an adaptive model dynamically estimates the probability of each pixel based on the previously encoded pixels, and when AC works with an adaptive model, it is usually referred to as adaptive arithmetic coding (AAC). When the adaptive model considers the probability of just the symbol, without any other additional information, it is called adaptive arithmetic coding order-0 (AAC-0).[9,10] Note that static models are less efficient than adaptive models because the probability table must be saved as header information.

Recently, many AAC-oriented coding schemes have been proposed in the literature. Carpentieri,[5] while presenting his lossless image compression algorithm, proposes to select dynamically, for each pixel position, one of the large number of possible probability distributions and encode the current pixel prediction error by using the selected distribution as the model for the arithmetic encoder. His scheme is slow, especially for large images. Golchin and Paliwal[6] propose to combine a context classification scheme with adaptive prediction and entropy coding to produce an adaptive lossless image encoder. They maximize the benefits of adaptivity using both adaptive prediction and entropy coding. Matsuda *et al.*[11] propose a lossless coding scheme using a block-adaptive prediction technique to remove redun-

Paper 09202RR received Oct. 16, 2009; revised manuscript received Apr. 12, 2010; accepted for publication Apr. 16, 2010; published online Jun. 1, 2010.

1017-9909/2010/19(2)/023014/6/$25.00 © 2010 SPIE and IS&T.

dancy. The image prediction errors are encoded using a kind of context-adaptive AC. They propose a model that approaches the probability density of errors by a generalized Gaussian function. The encoding algorithm is also very slow—e.g., when the size of the images is 512×512, the coder takes between 10 and 20 min for the encoding process. Recently, Kuroki *et al.*[12] have presented an AAC with prediction errors in lossless image compression. They propose a model that estimates the probability density of each error pixel by Laplacian distribution with zero mean. The compression ratios in Ref. 12 are at an average 5% higher than those by the conventional arithmetic encoders. From the description of these schemes, we can notice that AAC requires a large amount of computations. In this paper, we propose an AAC for lossless image compression that is quite efficient in terms of compression ratio and computational time. The idea behind our approach is to use a dynamic probability table that is updated only after encoding the last occurrence of each pixel.

The rest of this paper is organized as follows: Section 2 presents the AC principle. In Sec. 3, we detail our proposed method and present the mathematical proof. In Sec. 4, we provide experimental results to verify the performance of the proposed approach. Conclusions are drawn in Sec. 5.

2 Overview of Arithmetic Coding

In this section, we will present the basic notions of AC,[10,13–15] which is one of the efficient techniques for lossless data compression. Given an alphabet $\mathcal{A}$ composed of n mutually distinct symbols $\mathcal{A}=\{\alpha_1,\ldots,\alpha_n\}$ and the probability distributions $P=\{p_1,\ldots,p_n\}$ of the symbols where p_k is the probability of occurrence of α_k, for $k\in\{1,\ldots,n\}$. Shannon[16] proved that the number of bits needed to encode data cannot be less than the entropy of P, defined by:

$$H(P)=\sum_{k=1}^{n}-p_k\log_2(p_k). \tag{1}$$

Since AC produces a rate that approaches the entropy of the encoded data,[16,17] it is widely used in modern image and video compression algorithms such as JBIG, JBIG2, JPEG2000, H263, and H.264/AVC. The main idea of AC is to represent data by the interval [0, 1]. Thus, each binary value on this interval can uniquely identify the encoded data. Initially, the AC determines the symbols to be encoded, as well as the probabilities of these symbols. To estimate the probabilities of symbols, the AC works with a modeler that can be static or adaptive. Note that the best and the most popular approaches are based on the choice of a good model to obtain maximum compression of the data. Last, the encoder performs the encoding algorithm and generates the bitstream.

The encoding algorithm used in SAC works conceptually as follows: Let X be a message to compress that is composed of m events—i.e., $X=x_1,\ldots,x_m$. Each event x_i, with $i\in\{1,\ldots,m\}$, takes a value from an alphabet $\mathcal{A}$ composed of n symbols $\mathcal{A}=\{\alpha_1,\ldots,\alpha_n\}$. We denote the probability distributions P of the symbols of the alphabet $\mathcal{A}$ as $P=\{p_1,\ldots,p_n\}$, where p_k is the probability of α_k, for $k\in\{1,\ldots,n\}$. AC begins by subdividing the interval [0, 1) into n nonoverlapping intervals.[7] Let $[L\alpha_k,H\alpha_k)$ be the interval corresponding to the symbol α_k, with $H\alpha_k-L\alpha_k=p_k$. Let L be the lower interval limit and H be the higher interval limit. The output of AC is an interval $[L,H)$ in the range [0, 1), and the bitstream is the binary representation of any value in the interval $[L,H)$. In the SAC technique, the first step consists of calculating the true frequency table for all symbols. However, in the AAC-0 (Ref. 9), the modeler updates the frequency of each symbol by incrementing its count just after it has been encoded. No additional information is required to be sent to the decoder that mirrors the encoder's operations.

It should be noticed that AC generates a file consisting of a header followed by the compressed bitstream of the input message. The header information contains the frequency/probability table of all the symbols. After having encoded a message X of m events with AC, we can decode this sequence by using the probability table and the corresponding bitstream.

Let S be the size of the current interval $S=H-L$. Then, the encoding algorithm will assign a message X to this interval. In AC, the length of the bitstream is equal to the product of the probabilities of the individual symbols, and it decreases at each iteration:

$$S=p_1\times\cdots\times p_m=\prod_{i=1}^{m}p_i. \tag{2}$$

If we let $p(X)$ denote the probability of the message X, then $p(X)=S$. Therefore, we can bound the number of bits required to represent the message X by $-\lfloor\log_2[p(X)]\rfloor+2$ bits (Ref. 13), where $\lfloor.\rfloor$ is the floor function.

Arithmetic coders are therefore most useful when there are large probabilities in the probability distribution. In the next section, we show how to update the probabilities to reduce the data size.

3 Proposed Approach

In this section, we present an efficient method that allows us to update probabilities in the AC process in order to reduce the data size. We propose a new AAC that dynamically calculates the probability of the current pixel based on all pixels that precede it. The proposed AAC requires two passes. In the first pass, the true probability table is calculated and inserted in the compressed header file, whereas in the second pass, the AAC encodes the input image and dynamically updates probabilities when finding the last occurrence of the current pixel.

3.1 *Overview of the Proposed Method*

Let X be an image of m pixels. Instead of coding X with a fixed probability distribution, a dynamic probability table is used. Therefore, while encoding the image pixels, we verify whether we have encoded the last occurrence of the current pixel. If this has been the case, we must subtract the frequency of the corresponding pixel from the image size, and hence the probability table must be recomputed to encode the remained pixels. In the end, we provide a file decomposed into a header followed by the compressed bitstream of the original image. Note that the initial frequencies need to be transmitted to the decoder as header information. Furthermore, if we suppose that we have an alphabet of n

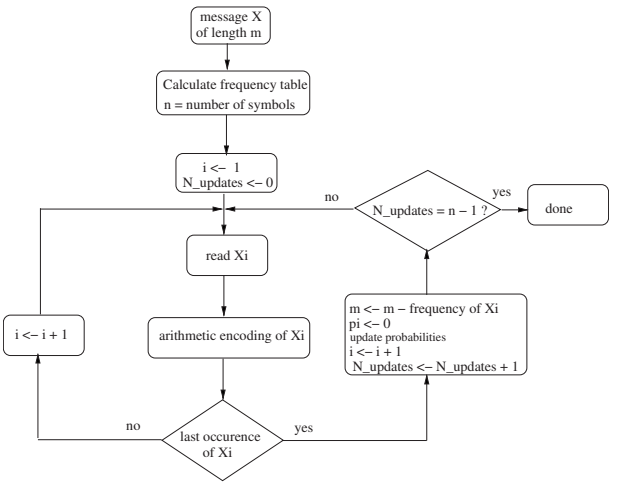

Fig. 1 Block diagram of the proposed encoding scheme.

symbols, we have at the most updated the probability table $n-1$ times, whatever the size of the input sequence. However, the traditional AAC-0 (Ref. 9) updates probabilities $n/2$ times, on average, for each pixel. So, if we have an image with m pixels, we must update the probability table $m\times(n/2)$ times for the traditional AAC-0, which requires a large amount of computation with our approach. Our approach is elaborated in the form of flowcharts in Fig. 1 and Fig. 2. Thus, Fig. 1 describes how the proposed encoding algorithm works, while Fig. 2 describes the decoding one.

3.2 *Mathematical Proof*

In this section, we proceed to develop a mathematical proof in order to prove that our approach is better than the SAC and always offers higher compression rate. Let X be a sequence of m events $X=x_1,\ldots,x_m$, taking values from an alphabet of n symbols $\mathcal{A}=\{\alpha_1,\ldots,\alpha_n\}$. We denote the probability distributions P of each symbol of the alphabet $\mathcal{A}$ as $P=\{p_1,\ldots,p_n\}$, where $p_k=f_k/m$, and f_k represents the frequency of the corresponding symbol α_k from the total length of the sequence X.

Note that the number of bits required representing the sequence X, after applying SAC, is calculated as:

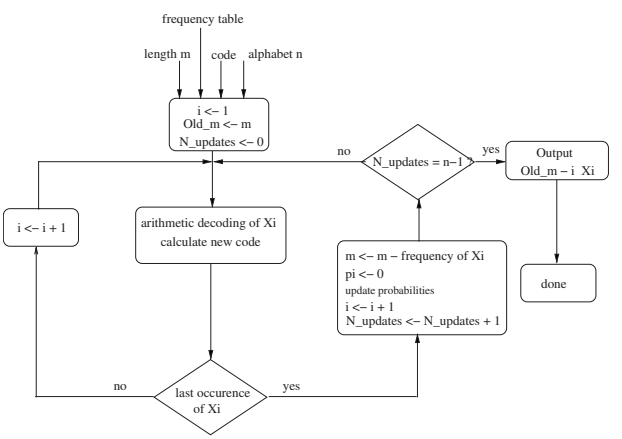

Fig. 2 Block diagram of the proposed decoding scheme.

$$N_{SAC}=\left\lfloor -\log_2\left(\prod_{i=1}^{n}p_i^{f_i}\right)\right\rfloor+2=\left\lfloor -\log_2\left[\prod_{i=1}^{n}\left(\frac{f_i}{m}\right)^{f_i}\right]\right\rfloor+2. \quad (3)$$

However, to calculate the size of the compressed file, according to our approach, we use N_{AAC} bits, which is given by:

$$N_{AAC}=\left\lfloor -\log_2\left[\prod_{i=1}^{n}p(\alpha_i|\alpha_1,\ldots,\alpha_{i-1})^{f_i}\right]\right\rfloor+2. \quad (4)$$

Proposition.

The compression ratio with our AAC approach is better than with SAC because

$$\left\lfloor -\log_2\left[\prod_{i=1}^{n}p(\alpha_i|\alpha_1,\ldots,\alpha_{i-1})^{f_i}\right]\right\rfloor+2<\left\lfloor -\log_2\left[\prod_{i=1}^{n}\left(\frac{f_i}{m}\right)^{f_i}\right]\right\rfloor+2. \quad (5)$$

Proof.

So, we must prove that

$$\prod_{i=1}^{n}p(\alpha_i|\alpha_1,\ldots,\alpha_{i-1})^{f_i}>\prod_{i=1}^{n}\left(\frac{f_i}{m}\right)^{f_i}. \quad (6)$$

We proceed to prove by induction with n, which is the number of symbols in the message X. We start with $n=2$. Suppose that the sequence X is composed only of events from an alphabet of two symbols $\mathcal{A}=\{\alpha_1,\alpha_2\}$ with respective probabilities f_1/m, f_2/m. Without loss of generality, suppose that the message X is terminated by the event α_2.

By using SAC to encode the sequence X, the number of bits needed can be calculated using the product of the probabilities:

$$PoP_{SAC}=\left(\frac{f_1}{m}\right)^{f_1}\left(\frac{f_2}{m}\right)^{f_2}. \quad (7)$$

On the other side, by using our approach to encode the sequence X, the number of bits needed is also obtained by the product of the probabilities:

$$PoP_{AAC}=\left(\frac{f_1}{m}\right)^{f_1}\left(\frac{f_2}{m}\right)^{T_1-f_1}, \quad (8)$$

where $T_1=inf\{k\geq 1\,|\,\text{encoding the last occurrence of }\alpha_1\}$.

We must then compare PoP_{SAC} and PoP_{AAC}, as described in Eqs. (7) and (8), respectively. So, we have to compare $(f_2/m)^{f_2}$ with $(f_2/m)^{T_1-f_1}$. The objective is then to show that $T_1-f_1<f_2$.

We know that $T_1<m$; thus, $T_1-m<0$ and $T_1-m+f_2<f_2$. Since we know that $m=f_1+f_2$, thus $T_1-f_1=T_1-m+f_2$, and then we have:

$$T_1-f_1<f_2. \quad (9)$$

From Eq. (9), we can conclude that $N_{AAC}<N_{SAC}$, where N_{AAC} and N_{SAC} are respectively the number of bits needed to encode data using AAC and SAC.

Table 1 Comparison results between our approach and an SAC and an AAC-0.

	SAC	AAC-0	Proposed AAC		
	Compression ratio	Compression ratio	Compression ratio	Gain (SAC) (%)	Gain (AAC0) (%)
Average	1.140	1.144	**1.146**	0.511	0.203
Min	1.046	1.047	**1.048**	0.067	0.051
Max	1.998	2.001	**2.003**	3.974	3.398

Now, with $m>2$, we assume that X is composed of n events from an alphabet $\mathcal{A}=\{\alpha_1,\ldots,\alpha_n\}$ with the probabilities f_i/m. Without loss of generality, we suppose that X is terminated by α_n.

The total number of bits needed to encode the sequence X with static arithmetic coding is related to Eq. (3). With $T_i=inf\{k\geqslant 1 \,|\, \text{encoding the last occurrence of } \alpha_i\}$, for $i\in\{1,\ldots,n-1\}$, we may now assume that $T_i<T_j$ if $i<j$ for $i,j\in\{1,\ldots,n-1\}$.

Let $f_k^{(i)}$ be the occurrence of α_k within $]T_{i-1},T_i]$, with $f_k=\Sigma_{i=1}^{k} f_k^{(i)}$ for $i\in\{1,\ldots,n-1\}$ and $T_0=1$. Thus, we may prove that:

$$\prod_{k=1}^{n-1}\prod_{i=1}^{k}\left(\frac{f_k}{m-f_{i-1}}\right)^{f_k^{(i)}} > \prod_{k=1}^{n-1}\left(\frac{f_k}{m}\right)^{f_k}. \tag{10}$$

We know that

$$\prod_{i=1}^{k}\left(\frac{f_k}{m-f_{i-1}}\right)^{f_k^{(i)}} > \prod_{i=1}^{k}\left(\frac{f_k}{m}\right)^{f_k^{(i)}}, \tag{11}$$

with

$$\prod_{i=1}^{k}\left(\frac{f_k}{m}\right)^{f_k^{(i)}} = \left(\frac{f_k}{m}\right)^{\left(\Sigma_{i=1}^{k} f_k^{(i)}\right)} = \left(\frac{f_k}{m}\right)^{f_k}. \tag{12}$$

Therefore, Eq. (12) is true. So, we can conclude that the conventional SAC is less efficient than our AAC. Thus, we prove that thanks to our approach, we will able to further reduce the size of the compressed data.

4 Experimental Results

We have applied our proposed method to more than 100 gray-level images of size 512×512 with 8 bits per pixel. In order to evaluate the compression efficiency of our approach, we have used the compression ratio factor C_r:

$$C_r=\frac{\text{Total size in bits of the input image}}{\text{Total size in bits of compressed bitstream}}, \tag{13}$$

which compares the size of the original image with the size of the bitstream.

Throughout this paper, we propose to compare our method with the conventional approaches applied directly on the pixels. The average, min, and max compression ratios for all tested images are listed in Table 1. We can see that the compression ratios for the proposed AAC are between 0.067% and 3.974% higher than those obtained by SAC, and it outperforms AAC-0 by a factor of 0.203% on average.

The proposed AAC yields a better compression if the image blocks are sorted for clustering the input data. Thus, we propose to partition the image into square blocks of pixels and then sort the image blocks according to chosen characteristics. For the experimental results presented in Table 2, we propose to sort the image blocks at the decreasing order of the mean value M_b using Eq. (14):

Table 2 Comparison results between our approach and an SAC and an AAC-0 by sorting the image blocks.

	SAC	AAC-0	Proposed AAC		
	Compression ratio	Compression ratio	Compression ratio	Gain (SAC) (%)	Gain (AAC0) (%)
Average	1.140	1.144	**1.207**	5.811	5.201
Min	1.046	1.047	**1.094**	0.602	0.497
Max	1.998	2.001	**2.102**	11.310	7.087

(a) (b)

Fig. 3 (a) Original image of size 512×512. (b) Original image partitioned into blocks of size 32×32 and sorted by the M_b value.

$$M_b = \frac{1}{N_b \times N_b} \sum_{i=0}^{N_b-1} \sum_{j=0}^{N_b-1} P(i,j), \tag{14}$$

where $N_b \times N_b$ corresponds to the size of the image blocks, and $P(i,j)$ represents the pixel P in the i'th row and j'th column. Figure 3 shows an image and its blocks sorted from lightness to darkness. Thus, we suggest calculating the mean value using Eq. (14) of each block of size 32 ×32 and then sorting all blocks according to the calculated values. Tables 1 and 2 show that the compression ratios obtained by SAC and AAC-0 are the same whether sorting the image blocks or not. This is due to the fact that the SAC and the AAC-0 efficiency depend only on the probability density of the encoded symbols and not on the scan order of the input data. From Table 2, the obtained results highlight that we increase the average compression ratios of 5.5%. Note that these results are obtained after adding the initial blocks' positions to the header of compressed file.

Table 3 lists the compression ratios of some well-known gray-level images that are available from the University of Waterloo Greyset2 collection.[18] We have used these in order to further evaluate the performance of our AAC and to compare the obtained results with both SAC and AAC-0.

We deduce from the experiments that the updating technique of probabilities and the image block-sorting step improve the compression efficiency of SAC and AAC-0 by a factor that is on the order of 5.5% on average. Note that for some images, our AAC works much better than SAC and AAC-0, and we have an improvement of about 13% in the compression ratios.

There is one other important issue in an image compression scheme—the processing time. The analysis has been done using the same computer (Intel Core 2 Duo 2.93-GHz CPU with 2-GB RAM). Table 4 shows the comparisons between the average processing time of the proposed AAC and AAC-0 (Ref. 9) algorithms by using 100 images of different sizes. It is clear that the proposed AAC algorithm is significantly faster than the AAC-0 algorithm. Note that this improvement in the processing time is due to the reduction in the number of probability updates.

Table 4 Comparison of the processing time between the proposed AAC and AAC-0 algorithms.

Image size in pixels	Total elapsed time (s) using the AAC-0 algorithm	Total elapsed time (s) using the proposed AAC algorithm
256×256	0.06	0.03
512×512	0.25	0.09
1024×1024	1.06	0.36

5 Conclusion

In this paper, we have presented a novel method to improve the compression efficiency of AC. This improvement is

Table 3 Comparison results between our approach and an SAC and an AAC-0 by sorting the image blocks for some well-known gray-level images.

	SAC	AAC-0	Proposed AAC		
	Compression ratio	Compression ratio	Compression ratio	Gain (SAC) (%)	Gain (AAC0) (%)
Barbara	1.068	1.070	**1.096**	2.55	2.37
Boat	1.103	1.122	**1.133**	2.64	0.97
France	1.229	1.234	**1.423**	13.63	13.28
Goldhill	1.066	1.069	**1.102**	3.26	2.99
Lena	1.071	1.073	**1.117**	4.11	3.93
Peppers	1.053	1.056	**1.079**	2.40	2.13
Zelda	1.085	1.100	**1.135**	4.40	3.08
Average	**1.097**	**1.104**	**1.155**	**5.07**	**4.46**

achieved by updating dynamically the probability table only after encoding the last occurrence of each symbol and coding sorted image blocks, which provides a more appropriate probability model. The proposed coding scheme depends on the pixels' occurrence and the image block sorting. Experimental results for a good sample of images demonstrate that our coding scheme outperforms the conventional arithmetic encoders in terms of compression ratios by a factor that is on the order of 5.5% on average. From a viewpoint of complexity, the proposed scheme reduces the amount of computation when updating the probability table. The obtained compressed ratios are very important, especially in lossless compression.

References

1. B. Carpentieri, M. J. Weinberger, and G. Seroussi, "Lossless compression of continuous-tone images," *Proc. IEEE* **88**(11), 1797–1809 (2000).
2. T. J. Chuang and J. C. Lin, "A new algorithm for lossless still image compression," *Pattern Recogn.* **31**(9), 1343–1352 (1998).
3. S. Sudharsanan and P. Sriram, "Block-based adaptive lossless image coder," in *Proc. IEEE Int. Conf. on Image Processing*, vol., 1, pp. 120–123, IEEE, Vancouver (2000).
4. W. Xiaolin, "An algorithmic study on lossless image compression," in *Data Compression Conference*, pp. 150–159, IEEE Computer Society Press, Snowbird, UT (1996).
5. B. Carpentieri, "A new lossless image compression algorithm based on arithmetic coding," in *Proc. 9th Int. Conf. on Image Analysis and Processing*, vol. II, pp. 54–61, Springer, Florence, Italy (1997).
6. F. Golchin and K. K. Paliwal, "A lossless image coder with context classification, adaptive prediction and adaptive entropy coding," in *Proc. IEEE International Conference on Acoustics Speech and Signal Processing*, pp. 2545–2548, Seattle, WA (1998).
7. L. J. Karam, "Lossless Image Coding," in *Handbook of Image & Video Processing*, A. Bovik, Ed., Chap. 5.1, pp. 461–474, Academic, New York (2000).
8. F. Rubin, "Arithmetic stream coding using fixed precision registers," *IEEE Trans. Inf. Theory* **25**(6), 672–675 (1979).
9. D. Salomon, *Data Compression: The Complete Reference*, 4th ed., Springer-Verlag, London (2007).
10. I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Commun. ACM* **30**(6), 520–540 (1987).
11. I. Matsuda, N. Shirai, and S. Itoh, "Lossless coding using predictors and arithmetic code optimized for each image," in *Proc. Int. Workshop Visual Content Processing and Representation*, vol. 2849, pp. 199–207, Springer, Heidelberg (2003).
12. N. Kuroki, T. Manabe, and M. Numa, "Adaptive arithmetic coding for image prediction errors," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS 04)*, vol. III, pp. 961–964, Vancouver, Canada (2004).
13. P. G. Howard and J. S. Vitter, "Arithmetic coding for data compression," *Proc. IEEE* **82**(6), 857–865 (1994).
14. G. G. Langdon, "An introduction to arithmetic coding," *IBM J. Res. Dev.* **28**(2), 135–149 (1984).
15. A. Moffat, R. M. Neal, and I. H. Witten, "Arithmetic coding revisited," *ACM Trans. Inf. Syst.* **16**(3), 256–294 (1998).
16. C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.* **27**, 379–423 (1948).
17. N. Abramson, *Information Theory and Coding*, McGraw-Hill, New York (1963).
18. Available at http://links.uwaterloo.ca/repository.html.

Atef Masmoudi received his engineering diploma in informatics from the Ecole Nationale des Sciences de l'Informatique in 2003 and a master's in automatic and signal processing from the National Engineering School of Tunisia in 2005. Since September 2005, he has been working as assistant professor of computer science in Tunisia. He is currently preparing his PhD thesis at the National Engineering School of Sfax in Tunisia and Montpellier II in France. He is a member of the Research Unit (R.U.): Sciences and Technologies of Image and Telecommunications, Tunisia, and a member of the Montpellier Laboratory of Informatics, Robotics, and Microelectronics, France. He was a member of the organization committee of the IEEE International Conference: Sciences of Electronic, Technologies of Information and Telecommunication in 2005, 2007, and 2009. His main research area is the combination of compression and encryption by using arithmetic coding and chaos theory.

William Puech received a diploma of electrical engineering from the University of Montpellier, France, in 1991, and a PhD Degree in signal-image-speech from the Polytechnic National Institute of Grenoble, France, in 1997. He started his research activities in image processing and computer vision. He served as a visiting research associate at the University of Thessaloniki, Greece. From 1997 to 2000, he was an assistant professor at the University of Toulon, France, with research interests including methods of active contours applied to medical images sequences. Between 2000 and 2008, he was an associate professor, and since 2009, he has been a full professor in image processing at the University of Montpellier, France. He works in the LIRMM Laboratory (Laboratory of Computer Science, Robotics, and Microelectronics of Montpellier). His current interests are in the areas of protection of visual data (image, video, and 3-D objects) for safe transfer by combining watermarking, data hiding, compression, and cryptography. He has applications in medical images, cultural heritage, and video surveillance. He is the head of the ICAR team (Image & Interaction), and he has published more than 10 journal papers, 4 book chapters, and more than 60 conference papers. Puech is a reviewer for more than 15 journals (*IEEE Trans. Image Processing, IEEE Trans. Multimedia, Signal Processing, J. Applied Signal Processing, J. Electronic Imaging*, etc.) and for more than 10 conferences (IEEE ICIP, EUSIPCO, WIAMIS, IWDW, etc.). He is an IEEE member and SPIE member. Since 2005, he has been in the TPC of EUSIPCO, and since 2009, he has been the area chair for "Image and Multidimensional Signal Processing" of EUSIPCO.

Mohamed Salim Bouhlel received an engineering diploma from Sfax National School of Engineering (ENIS) in 1981 and DEA and doctor's degree in automatic and software engineering from the National Institute of Applied Sciences at Lyon, France, in 1981 and 1983, respectively. He is currently the director of Sfax High Institute of Electronics and Communication (ISECS). In 1999, he received the gold medal with highest honors of the jury in the first International Meeting of Invention, Innovation, and Technology (Dubai). He was the vice president of the Tunisian Association of Specialists in Electronics. He is the editor in chief of the *International Journal of Electronics, Technology of Information, and Telecommunications*, chair of the International Conference Sciences of Electronic, Technologies of Information and Telecommunication (2003 to 2007 and 2009) and E-MEDISYS (2007, 2008, and 2010), and a member of the program committee of many international conferences. In addition, he is an associate professor at the Department of Image and Information Technology in the Higher National School of Telecommunication ENST-Bretagne (France). He is the director of a research unit in the domain of image processing and telecommunication. He was a member of TEMPUS program CD JEP-31069-2003 titled "Elaboration d'un Programme d'Enseignement Théorique et Pratique on Biotechnologie." He has participated in many collaborations between Sfax University and other institutions in Europe and North America.