International Journal of Innovative Studies in Sciences and Engineering Technology (IJISSET)

ISSN 2455-4863 (Online) **www.ijisset.org** *Volume: 5 Issue: 12 | 2019*

Pixel-Position-Based Lossless Image Compression Algorithm

Eduardo L. L. Cabral; Gaianê Sabundjian; Thadeu das Neves Conti

Energy and Nuclear Research Institute - IPEN / CNEN-SP, Av. Professor Lineu Prestes 2242, Sao Paulo, SP, Brazil

Abstract. *In this paper we present a novel lossless image compression method that is very simple and fast. The method uses linear prediction followed by arithmetic coding. Different prediction functions are used to estimate the intensity of image pixels. Two variants of the prediction algorithm are presented. One variant uses two different prediction functions and the other uses three different prediction functions. The position of the pixel in the image determines which prediction function is used. The method can be applied for images of any size and of high bit-depths. Standard images available in the literature are used to test the method. The compression ratios obtained with the proposed method are compared with the compression ratios obtained with the JPEG-LS and JPEG2000 methods and the results are satisfactory.*

Keywords: *lossless image compression, predictive coding, pixel position based.*

1. INTRODUCTION

Many algorithms for image compression have been devised in the last years and new methods are introduced every year. Several surveys about lossless image compression can be found in literature presenting the different methods[1,2]. Image compression methods can be roughly classified in two categories: lossless and lossy. In lossless schemes the exact original data can be recovered, while in lossy techniques only a close approximation of the original data can be obtained.

Lossless image compression are used for images that are documents and none information can be lost, such as, medical images. In some systems, such as, medical MRI and CT scanners, a single exam generates a large set of images that must be stored in memory and many times also need to be transmitted. Here lossless image compression methods are needed and fast and efficient algorithms would improve the memory capacity and transmition rate of the data.

An image consists on a rectangular array of pixels. Each pixel of a grayscale image is a nonnegative integer interpreted as the local intensity of the image. For grayscale images the pixels intensities vary form 0 to 2^{N-1}, where N is the pixel bit-depth. Typical grayscales images are of bit-depth from 8 to 16 bits. For color images, for instance in RGB format, each pixel is formed by three 8 bit-depth nonnegative integers, each one representing the local intensities of the red (R), green (G) and blue (B) colors. Color images can be viewed as formed by three grayscale images and so the same methods used to compress grayscale images can be used to compress color images.

Lossless image compression can be achieved by several forms: coding methods, prediction methods, transform methods, and a combination of these methods. Coding methods are directly applied to the raw data treating them as a sequence of numbers. Prediction methods try to eliminate the spatial redundancy of the data. The transform domain methods exploit spatial frequency information contained in the image. Modern image compression algorithms, which are either a prediction or a transform method, after the initial operation they also apply some coding method.

Common coding methods are: Huffman, arithmetic, Golomb-Rice, LZW, and run-length. In Huffman coding each symbol of the uncompressed data is replaced by a code. The symbol codes are of variable length and symbols that occur more frequently in the data have codes of smaller lengths than symbols that occur less frequently. In arithmetic coding the idea is similar to the Huffman coding but instead of assigning one binary code for each symbol, two or more symbols can be represented by a single binary code, thus resulting into higher compression efficiency. LZW coding consists on an adaptive method that does not require all the data to be available at the start, fixed-length codes are constructed on the fly for variable-length sequences of symbols. In run-length coding the source file is decomposed into segments of identical symbols, each segment is replaced by a pair symbol-number of occurrence. Golomb-Rice coding uses a tunable parameter, M, to divide the uncompressed data into two parts: the quociente of a division by M, and the remainder. When compared to the prediction and transform methods all the coding schemes result in smaller compression ratios.

In a prediction algorithm a prediction function is used to guess the pixels intensities and then the prediction errors are calculated, i.e., the differences between actual and predicted pixels intensities. Next, the sequence of prediction errors is coded using a coding method. To calculate the prediction intensity for a given pixel the intensities of neighbor pixels already processed are used. Even using extreme simple predictors results in a better compression ratio than using a coding method without prediction. The pixel intensity of grayscale images tends to have a uniform distribution, while the prediction error distribution is close to the normal distribution. Therefore, entropy of the prediction errors is much smaller than entropy of the pixel intensities, making prediction errors easier to compress using some of the coding methods.

The most popular lossless prediction methods are JPEG-LS (LOCO)[3] and CALIC[4]. JPEG-LS and CALLIC are context based adaptive lossless image coder. In the first stage, both the algorithms use a gradient-based non-linear prediction to get a lossy image and a residual image. In the second stage the residuals are coded. JPEG-LS uses Golomb-Rice coding and CALIC can use both Huffman and arithmetic coding to encode the residuals. Both JPEG-LS and CALIC have also a mechanism to automatically trigger a binary mode which is used to code either uniform or binary subimages or both. JPEG-LS achieves a good compression ratio with a low computational time, however, CALIC is able to obtain slight better compression at the expense of additional computational time. CALIC is one of the best lossless image compression scheme reported in the literature.

The performance of the prediction algorithm depends primary on the prediction function used. Several predictors were proposed some are simple and linear, and others are more complex and non-linear. The predictors used in JPEG-LS and CALIC are non-linear and can be considered as a switching scheme based on local image gradients. Several others schemes exist such as SFALIC[5], FELICS[6], FLIC[7], APC[8], and EDP[9]. SFALIC, FELICS and FLIC have simple prediction algorithms making them advantageous in terms of the low computational time, but they still present performance in compression ratio comparable to JPEG-LS. APC and EDP have sophisticated prediction algorithms allowing them to have a gain in performance in terms of compression ratio but they are computationally slow for most practical applications.

The transform methods use a 2-dimensional image transform such as DCT or wavelet transform. In these methods instead of the pixel intensities a matrix of transform coefficients are encoded. The transform is applied to the whole image, or to an image split into blocks. The transform methods are very good for lossy image compression but they can also be used for lossless compression. With respect to the lossless compression speed and ratio the transform methods perform worse than the predictive algorithms. For lossless compression the most popular transform schemes are the SP Transform[10] and the integer to integer wavelet transform methods.

The SP Transform requires only integer addition and bit-shift operations and yields superior compression ratio than the linear predictive coding based schemes like lossless JPEG. Several integer to integer wavelet transformations were developed for lossless image compression based on the concept of lifting and dual lifting steps[11,12,13]. These algorithms yield compression ratio results similar to the SP Transform. The JPEG200 standard of lossy and lossless image compression is a transform algorithm employing a wavelet transform[14] that is very efficient but this comes at a price of additional complexity compared to most predictive algorithms.

In this paper we introduce a simple method for lossless image compression. The method is designed to achieve high compression ratio together with high computational speed. The method is based on a fast prediction error algorithm and on the arithmetic coding of the prediction errors. Many similar algorithms using the same basic idea to calculate the prediction error as proposed in this work can be conceived. Two variants of the algorithm are presented in this work that uses two and three different predictive functions. To analyze the performance of the method we use grayscale and color natural, medical and drawing images with various bit-depths and sizes. The results obtained by the proposed method are compared with results obtained by the JPPEG-LS and JPEG2000 methods.

2. METHOD DESCRIPTION

In the proposed method the image is processed from top-left to bottom-right. First the prediction errors are calculated using different prediction functions. These functions are very simple and the position of the pixel in the image determines which prediction function is used. Then the prediction errors are coded using the arithmetic coding method. Note that decompression is also very simple and fast since it is a simple reversal of the compression process. To allow comparison with other methods a simple algorithm to detect regions with uniform intensity is introduced.

As mentioned, two variants of the predictive algorithm are presented in this work. Sections 2.1 and 2.2 present in detail the first variant of the prediction algorithm. This algorithm is named Algorithm 1. The other variant, named Algorithm 2, is presented in Sec. 2.3. The algorithm to detect regions with constant intensity is described in Sec. 2.4.

2.1 Prediction error calculation (Algorithm 1)

In Algorithm 1 to calculate the prediction error two fast linear prediction functions are employed. One function uses two neighboring pixels to guess the pixel intensity and the other uses four neighboring pixels. Each predictive function is used for half of the pixels of the image.

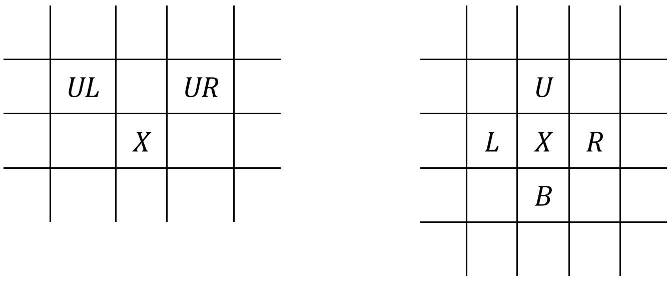
Prediction function 1 uses the pixels intensities at the upper-left neighbor (UL) and at the upper-right neighbor (UR) to predict the intensity of pixel X, as shown in Fig. 1a and given by eq. (1). Prediction function 2 uses the four pixels intensities at the upper neighbor (U), left neighbor (L), right neighbor (R) and at the bottom neighbor (B) to predict the intensity of pixel X as shown in Fig. 1b and eq. (2).

$$\text{Pred}(X) = \left\lfloor \frac{UL + UR}{2} \right\rfloor \qquad (1)$$

$$\text{Pred}(X) = \left\lfloor \frac{U + L + R + B}{4} \right\rfloor \qquad (2)$$

The symbol $\lfloor . \rfloor$ in eq. (1) and (2) denotes then floor function, i.e., downward truncation. The truncation operation seems to discard some information, however, it only removes the redundancy in the least significant bit because the sum and the difference of two integers are either both even or both odd, so that the last bit of the sum can be safely omitted since it is equal to the last bit of the difference.

The predictions for the pixels intensities calculated by functions 1 and 2 are always an average of the intensities values of neighboring pixels. This avoids having predictive values smaller than zero and grater than the maximum pixel intensity, thus keeping the predictive errors bounded inside the interval $-(2^N - 1)$ to $(2^N - 1)$ without any verification.



(a) Prediction function 1 (b) Prediction function 2

Fig. 1 Image samples showing the pixels used in the two prediction functions. (a) Prediction function 1, and (b) prediction function 2.

For the case of function 1 the pixels intensities at locations *UL* and *UR* are available for the encoder as well as for the decoder prior to processing pixel *X*. For the case of function 2, since the image is processed from top-left down to bottom-right, at first it may seem that the pixels intensities at locations *R* and *B* would not be available for the decoder prior to processing pixel *X*. Figure 2 shows the pixels of an image processed by each function, where the pixels processed using function 1 are marked with number '1' (pixels type 1), and the pixels processed using function 2 are marked with number '2' (pixels type 2). Note that for the pixels in the first line a different prediction function is used (see Sec. 2.2), so that they are not numbered in Fig. 2. Each prediction function is used for a different set of pixels in the image. Note that function 1 does not used any pixel processed using function 2 and the pixels locations used in function 2 are only the pixels processed by function 1. Thus, it is easy to observe that in the decompression stage it is possible to have all the pixels type 1 processed using function 1, before processing the pixels type 2 using function 2.

The ideal scheme, but not possible, to obtain high compression ratio would be to use only the prediction function 2 for all the pixels of the image. However such scheme would not be lossless because some pixel intensities would not be available at the right moment in the decompression stage. Thus, the proposed scheme using eq. (1) and (2) exploit the local redundancies of the image in a very efficient way.

Using the prediction function 1 (eq. 1) the prediction error for pixel (i, j) is calculated as shown by eq. (3).

$$e(i,j) = p(i,j) - \left\lfloor \frac{p(i-1, j-1) + p(i-1, j+1)}{2} \right\rfloor \qquad (3)$$

where *i* and *j* represents respectively the row and column locations of the pixel, $e(i,j)$ is the prediction error, $p(i,j)$ is the pixel intensity representing pixel *X* in Fig. 1a, $p(i-1,j-1)$ is the pixel intensity representing pixel *A* in Fig. 1a, and $p(i-1,j+1)$ is the pixel intensity representing pixel *B* in Fig. 1a.

-	-	-	-	-	-	-	-
1	2	1	2	1	2	1	2
2	1	2	1	2	1	2	1
1	2	1	2	1	2	1	2
2	1	2	1	2	1	2	1
1	2	1	2	1	2	1	2
2	1	2	1	2	1	2	1
1	2	1	2	1	2	1	2

Fig. 2 Image example showing the scheme used to apply the prediction functions. Note that for the pixels in the first line a different prediction function is used, so that they are not numbered.

Using the prediction function 2 (eq. 2) the prediction error for pixel (i, j) is calculated as given by eq. (4).

$$e(i,j) = p(i,j) - \left\lfloor \frac{p(i-1, j) + p(i, j-1) + p(i, j+1) + p(i+1, j)}{4} \right\rfloor \qquad (4)$$

where $p(i,j)$ is the pixel intensity representing pixel *X* in Fig. 1b, $p(i-1,j)$ is the pixel intensity representing pixel *C* in Fig. 1b, $p(i,j-1)$ is the pixel intensity representing pixel *D* in Fig. 1b, $p(i,j+1)$ is the pixel intensity representing pixel *E* in Fig. 1b, and $p(i+1,j)$ is the pixel intensity representing pixel *F* in Fig. 1b.

The division by 2 and floor operations can be very efficiently computed via a single bit-shift operation, and a division by 4 can be efficiently calculated by two bit-shift operations. Thus the calculation of the prediction errors using eq. (3) and (4) involves only integer summations and differences, and bit-shift operations making the algorithm extremely fast.

Before calculating the prediction error it is necessary to verify if the pixel is type 1 or 2. The easiest way to perform this verification is by calculating the module 2 of the sum of the row and column indexes of the pixel. Thus, an option would be: if $(i + j)$ module 2 equals to 1 then pixel (i, j) is type 1, otherwise if $(i + j)$ module 2 equals 0 then pixel (i, j) is type 2. Therefore, each pixel of the image is associated with one prediction function depending only on its row and column positions.

2.2 Border pixels (Algorithm 1)

Several exceptions exist for applying eq. (3) and (4) at the borders of the image, thus these equations have to be modified accordingly. Since applying function 1 for the pixels in the first line is not possible, then for the first line the prediction errors are calculated as in eq. (5).

$$e(1, j) = \begin{cases} p(1,1), \text{ for } j = 1 \\ p(i, j) - p(i, j-1), \text{ for } j > 1 \end{cases} \tag{5}$$

Thus the prediction error for pixel (1,1) is the value of the intensity of this pixel in the uncompressed image. Note that eq. (5) states that the predicted pixel intensity at any position in the first line is simply the pixel intensity of the left neighbor pixel.

Figure 3 presents image samples with the pixels used in the two prediction functions for the pixels located in the first and last column and in the last line of the image. For the pixels in the first column the prediction errors are calculated according to eq. (6) and (7) respectively for prediction functions 1 and 2.

$$e(i, j) = p(i, j) - p(i-1, j+1) \tag{6}$$

$$e(i, j) = p(i, j) - \left\lfloor \frac{p(i-1, j) + p(i, j+1) + p(i+1, j)}{3} \right\rfloor \tag{7}$$

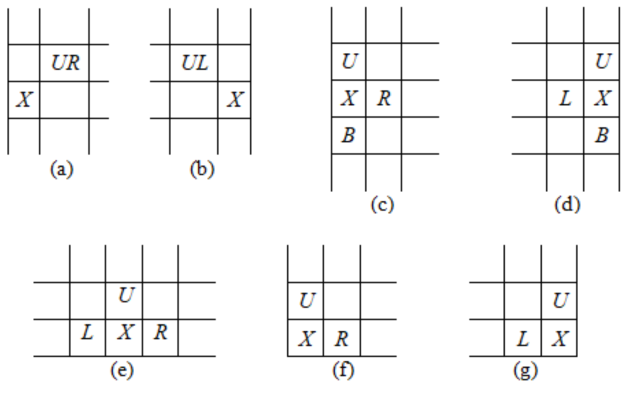


Fig. 3 Image samples showing the pixels used in the two prediction functions for the pixels at the borders: (a) and (b) prediction function 1 for first and last columns, (c) and (d) prediction function 2 for first and last columns, (e) prediction function 2 for the bottom row, and (f) and (g) prediction function 2 for bottom corners.

For the pixels in the last column the prediction errors are calculated according to eq. (8) and (9) respectively for prediction functions 1 and 2.

$$e(i, j) = p(i, j) - p(i-1, j-1) \tag{8}$$

$$e(i, j) = p(i, j) - \left\lfloor \frac{p(i-1, j) + p(i, j-1) + p(i+1, j)}{3} \right\rfloor \tag{9}$$

For the pixels type 2 presented in the last row the prediction errors are calculated according to eq. (10).

$$e(i, j) = p(i, j) - \left\lfloor \frac{p(i-1, j) + p(i, j-1) + p(i, j+1)}{3} \right\rfloor \tag{10}$$

For the pixels type 2 in the two bottom corners the prediction errors are calculated by modifying eq. (10) to exclude the right or the left neighbor pixel, depending if the pixel is in the right or in the left corner.

2.3 Prediction error calculation (Algorithm 2)

Several alternative algorithms using the same philosophy of the predictive algorithm described in the previous sections are possible. Schemes using two, three or more predictive functions can be conceived. Another algorithm, named Algorithm 2, would use three prediction functions instead of two functions as in Algorithm 1. In this algorithm the prediction function 1 uses the pixel intensity only at the left neighbor (*L*) to predict the intensity of pixel *X*, as shown in Fig. 4a and given by eq. (11). Prediction function 2 uses the four pixels intensities at the upper-left neighbor (*UL*), upper-right neighbor (*UR*), bottom left-neighbor (*BL*) and at the bottom-left neighbor (*BR*) to predict the intensity of pixel *X* as shown in Fig. 4b and eq. (12). Prediction function 3 uses the four neighbor pixels intensities at the upper neighbor (*U*), left neighbor (*L*), bottom left neighbor (*B*) and at the right-neighbor (*R*) to predict the intensity of pixel *X* as shown in Fig. 4c and eq. (13).

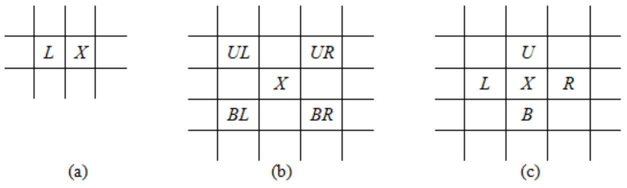


Fig. 4 Image samples showing the pixels used in the three prediction functions of the algorithm 2. (a) Prediction function 1, (b) prediction function 2, and (c) prediction function 3.

$$\text{Pred}(X) = L \tag{11}$$

$$\text{Pred}(X) = \left\lfloor \frac{UL + UR + BL + BR}{6} \right\rfloor \quad (12)$$

$$\text{Pred}(X) = \left\lfloor \frac{U + L + R + B}{8} \right\rfloor \quad (13)$$

As in the case of the Algorithm 1 each prediction function is used for a different set of the pixels in the image to guarantee that all the necessary neighboring pixels intensities are available for the decoder prior to processing pixel *X*. Figure 5 shows the pixels of an image processed by each one of the prediction functions. The pixels processed using function 1 are marked with number '1', the pixels processed using function 2 are marked with number '2', and the pixels processed using function 3 are marked with number '3'. As can be seen in Fig.5, each pixel of the image is associated with one prediction function depending only on its row and column positions.

Note that Algorithm 2, similarly as Algorithm 1, can be efficiently computed using only integer summations and differences, and bit-shift operations making the algorithm extremely fast.

-	1	1	1	1	1	1	1
2	3	2	3	2	3	2	3
1	1	1	1	1	1	1	1
2	3	2	3	2	3	2	3
1	1	1	1	1	1	1	1
2	3	2	3	2	3	2	3
1	1	1	1	1	1	1	1
2	3	2	3	2	3	2	3

Fig. 5 Image example showing the scheme used to apply the prediction functions for algorithm 2. Note that the prediction error of pixel (1,1) is the value of the pixel intensity of the uncompressed image.

2.4 Detection of regions with uniform intensity

For further compression and to allow comparison on the same basis with other lossless image compression methods a simple algorithm to detect regions with uniform intensity is included in the method. Two independent searches are performed, one by columns and the other by rows. Thus, this algorithm only detects columns or rows with constant intensity. Only groups larger than eight pixels of equal intensity are considered in the search. Comparing the results from the search by columns and the search by rows the one that provides the largest number of pixels is used.

Each region with constant intensity is coded using four integer numbers: the pixel intensity, the row and column of the first pixel of the region, and the number of pixels. These four values for all regions are placed in a vector and this vector is coded using the arithmetic coding method.

Note that the matrix with the prediction errors is transformed into a vector and the pixels that belong to the regions with uniform intensity are eliminated from this vector. Thus, the size of the vector with the prediction errors decreases according to the number of pixels belonging to the regions with uniform intensity.

3. RESULTS

The two algorithms used to calculate the prediction errors following by the arithmetic coding are implemented. The effectiveness of these algorithms is measured using the compression ratio (CR), which is the ratio of the size of the original image to the size of the compressed data stream. Tests are performed with and without detection of regions with pixels of uniform intensity.

For the evaluation three common test images used in the literature and some images of the ITU-T T.24 image set[15] are used. The three common test images are natural color images and the ITU-T T.24 image set is composed by several types of images varying from medical, natural, drawing, text, color and grayscale. Figures 6, 7, 8, 9, 10 and 11 show grayscales miniatures versions of the images used for testing. Table 1 presents the main characteristics of these images and the compression ratio results obtained with the three proposed algorithms. For the color images processing is done for each color plane independently and then the resulting three compressed images are placed together in a single file. The compression ratios obtained with the JPEG-LS and JPEG200 methods are also presented in Table 1 for comparison.

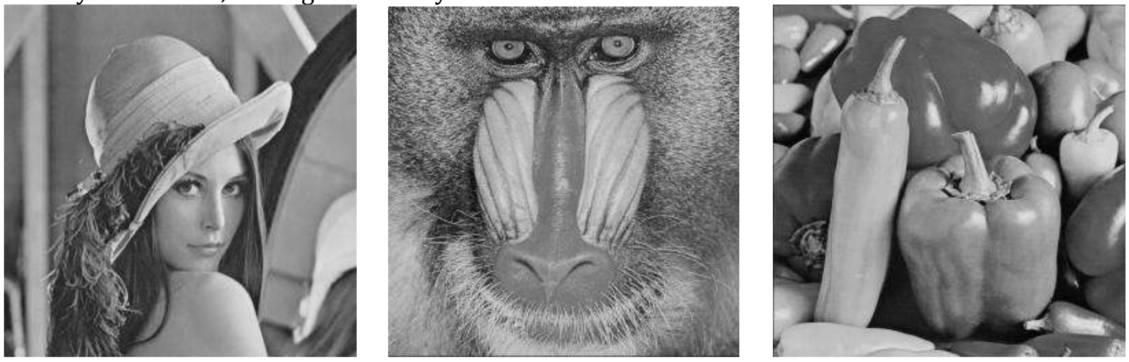
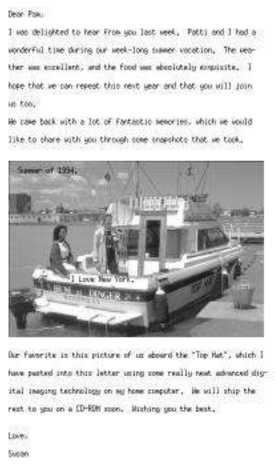

a) LENNA b) MANDRILL c) PEPPERS

Fig. 6 Miniature grayscale versions of the common images used for testing the algorithms (images are originally in color).

a) CAFE b) CATS c) BIKE

a) BIKE3 b) TOOLS c) WOMAN

Fig. 7 Miniature grayscale versions of the natural color images of the ITU-T T.24 set used for testing the algorithms (images are originally in color).



Dear Pam:

I was delighted to hear from you last week. Patti and I had a wonderful time during our week-long summer vacation. The weather was excellent, and the food was absolutely exquisite. I hope that we can repeat this next year and that you will join us too.

We came back with a lot of fantastic memories, which we would like to share with you through some snapshots that we took.

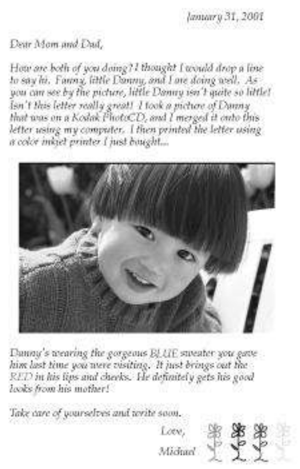
Summer of 1994.

I Love New York.

Our favorite is this picture of us aboard the "Top Hat", which I have pasted into this letter using some really neat advanced digital imaging technology on my home computer. We will ship the rest to you on a CD-ROM soon. Wishing you the best,

Love,

Susan



January 31, 2001

Dear Mom and Dad,

How are both of you doing? I thought I would drop a line to say hi. Fanny, little Danny, and I are doing well. As you can see by the picture, little Danny isn't quite so little! Isn't this letter really great! I took a picture of Danny that was on a Kodak PhotoCD, and I merged it onto this letter using my computer. I then printed the letter using a color inkjet printer I just bought...

Danny's wearing the gorgeous BLUE sweater you gave him last time you were visiting. It just brings out the RED in his lips and cheeks. He definitely gets his good looks from his mother!

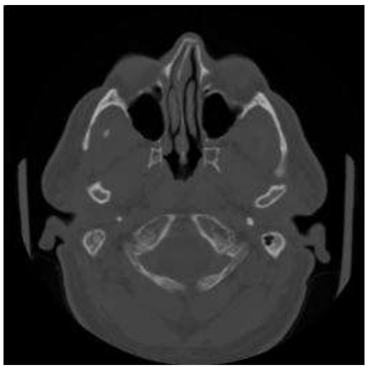
Take care of yourselves and write soon.

Love,

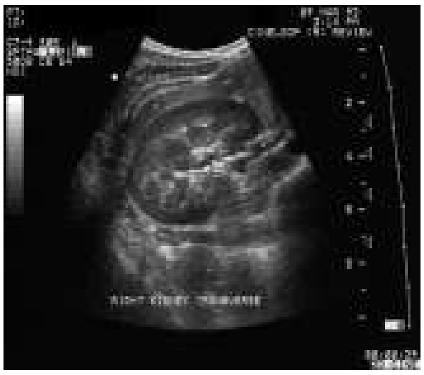
Michael

a) Compound 1 (CMPND1) b) Compound 2 (CMPND2)

Fig. 8 Miniature grayscale versions of the compound images of the ITU-T T.24 set used for testing the algorithms (images are originally in color).


a) Computer Tomography (CT)

b) Finger print (FINGER)


c) Ultrasound (US)

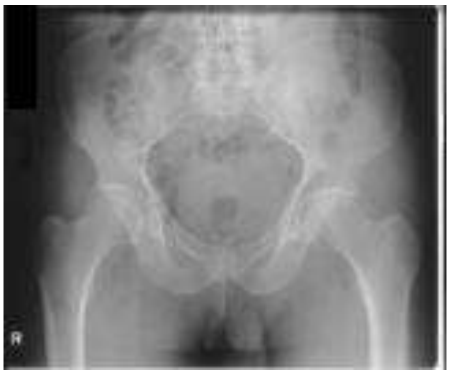
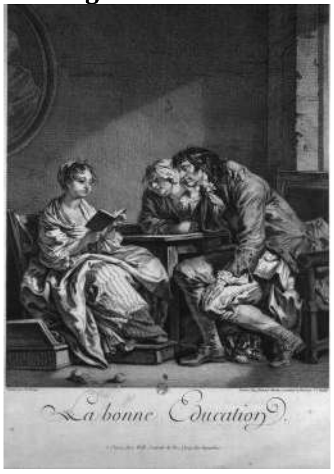
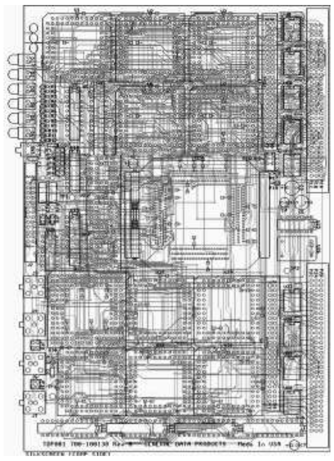

d) X-RAY

Fig. 9 Miniature of medical images of the ITU-T T.24 set used for testing the algorithms.



a) Educational (EDUC)


b) Printed circuit (PC)

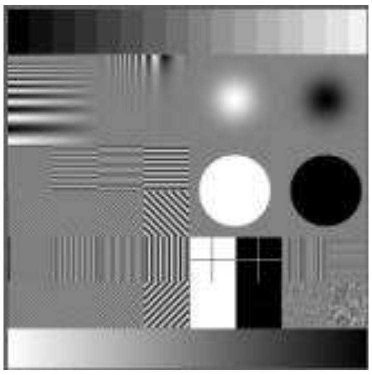
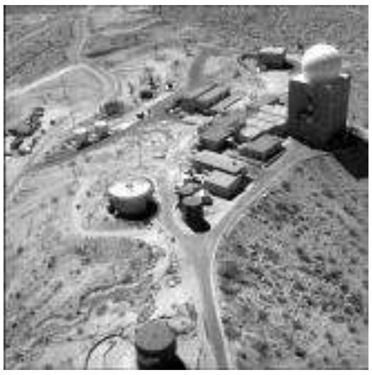
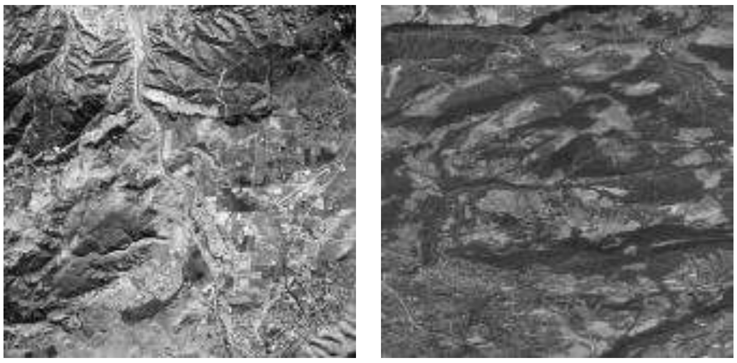

c) TARGET

Fig. 10 Miniature grayscale versions of drawing images of the ITU-T T.24 set used for testing the algorithms (the PC image is originally in color).


a) Aerial Photo (AERIAL2)

b) Mountains (MAT)

c) Seismic data (SEISMIC)



d) Texture 1 (TXSTUR1) e) Texture 2 (TXTUR2)

Fig. 11 Miniature of natural grayscale images of the ITU-T T.24 set used for testing the algorithms.

In Table 1 the best achieved results for the compression ratio obtained with the two proposed algorithms for each image are marked with bold-italic. Algorithm 1 yields better results for ten images, while Algorithm 2 is better for thirteen images. The average compression ratio obtained with Algorithm 1 is 2.228 and with Algorithm 2 is 2.286, thus Algorithm 2 is a little more efficient. However, the differences between the compression ratios obtained by the two algorithms are small and therefore we can even say they have almost the same performance.

From the results shown in Table 1 it we can see that in the average the proposed method is slightly worse than the JPEG-LS and JPEG2000 methods. The proposed method is better than the JPEG-LS and JPEG2000 methods only for three images, TXTUR1, PC and TARGET. The average compression ratio achieved using Algorithm 2 is approximately 7% lower than average ratio obtained with the JPEG2000 method and about 11% lower than that obtained with the JPEG-LS method. Certainly better results could be achieved with the proposed method if we used an improved algorithm for detection of uniform regions.

Table 1. Characteristics of the images used to test the proposed method and compression ratio results obtained using the proposed two algorithms and using the JPEG-LS and JPEG200 methods.

Image					Compression method			
Name	Type	Bits	Pixels	Size (Kbytes)	Algor. 1	Algor. 2	JPG-LS	JPG2000
LENNA	Natural, color	24	512x512	768	***1.707***	1.639	1.761	1.765
MANDRIL	Natural, color	24	512x512	768	1.240	***1.244***	1.295	1.328
PEPPERS	Natural, color	24	512x512	768	***1.626***	1.586	1.681	1.620
CT	Medical, gray	12	512x512	384	***2.636***	2.518	3.000	3.072
FINGER	Medical, gray	8	512x512	256	***1.360***	1.340	1.407	1.407
US	Medical, gray	8	512x488	224	2.712	***2.794***	3.027	2.605
X-RAY	Medical, gray	12	2048x1680	5,040	***1.967***	1.915	1.827	1.981
AERIAL2	Natural, gray	8	2048x2048	4,096	1.476	***1.481***	1.513	1.470
MAT	Natural, gray	8	1528x1146	1,710	2.282	***2.285***	2.790	2.564
SEISMIC	Natural, gray	8	512x512	256	1.903	***2.431***	2.783	2.753
TXTUR1	Natural, gray	8	1024x1024	1,024	1.221	***1.251***	1.240	1.170
TXTUR2	Natural, gray	8	1024x1024	1,024	1.411	***1.436***	1.488	1.420
CATS	Natural, color	24	768x512	1,152	***2.848***	2.811	2.954	4.535
CAFÉ	Natural, color	24	2048x2560	15,360	1.373	***1.379***	1.569	1.707
BIKE	Natural, color	24	2048x2560	15,360	***1.631***	1.624	1.843	2.051
BIKE3	Natural, color	24	781x919	2,103	***1.710***	1.708	1.819	1.547
TOOLS	Natural, color	24	1524x1200	5,358	1.324	***1.332***	1.423	1.387
WOMAN	Natural, color	24	2048x2560	15,360	***1.635***	1.590	1.791	2.099
CMPND1	Photo+text, color	24	512x768	1,152	4.958	***5.132***	6.330	5.565
CMPND2	Photo+text, color	24	1024x1400	4,200	***5.063***	4.908	6.060	6.140
EDUC	Drawing, gray	8	2850x4096	11,400	1.498	***1.504***	1.751	1.761
PC	Drawing, color	24	1575x2185	10,082	4.989	***5.650***	5.329	2.414
TARGET	Drawing, gray	8	512x512	256	2.648	***2.865***	3.657	3.710
Mean compression ratio					2.228	2.286	2.536	2.437

In order to present a more thorough analysis of the algorithm Fig. 12 presents the histograms of the grayscale WOMAM image and of the prediction error calculated with Algorithm 1. As we can see the histogram of the original image is wide spread over the entirely pixel intensity range [0, 256], while the histogram of the prediction error is very narrow around the zero mean, looking like a normal distribution with standard deviation equals to 12.2.

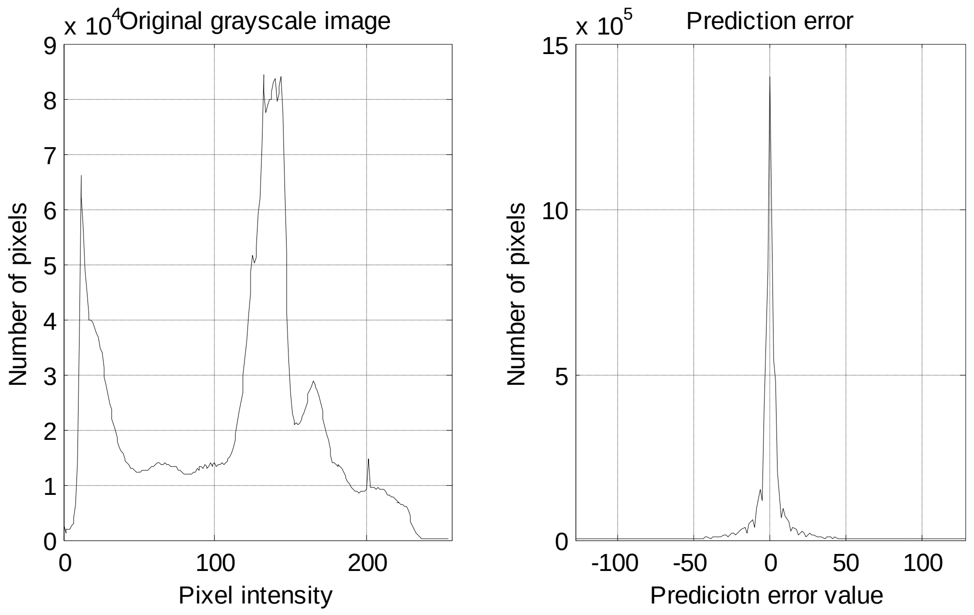


Fig. 12. (a) Histogram of the grayscale version of the original WOMEN. (b) Histogram of the prediction error calculated with algorithm 1.

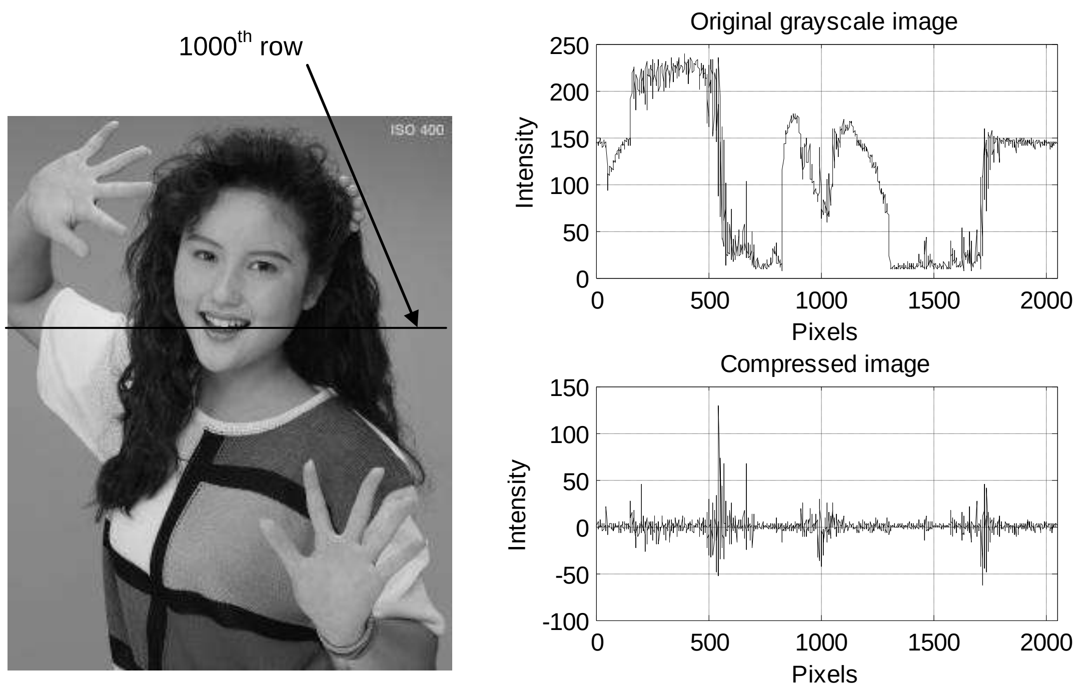


Fig. 13 (a) Miniature of the grayscale version of the original WOMEN image. (b) Pixel intensities of the original grayscale image along the 1000th row. (c) Prediction error calculated using algorithm 1 along the 1000th row.

Figure 13a shows a miniature of the original WOMAN image in grayscale format with it's 1000th row marked, and Fig. 13b and 13c show respectively the pixel intensities of the grayscale image and the prediction error calculated using Algorithm 1 along the 1000th image row. From Fig. 13c we observe that the predictive errors are all very close to zero, except for those pixels in regions of high contrast. Finally, the results shown in Fig. 13 and 14 demonstrate the great capability of the algorithm to reduce the dynamic range of the pixel intensities of the image.

4. CONCLUSIONS

In this paper we present a novel lossless image compression algorithm that is very simple and fast. The algorithm uses linear prediction followed by arithmetic coding. Two variants of the algorithm are present. Algorithm 1 uses two different prediction functions and Algorithm 2 uses three predictive functions.

The most significant characteristics of these algorithms are: (1) it uses very simple prediction functions; (2) it does not use the context to determine which prediction function to use for each pixel; (3) the prediction function used for each pixel depends only on the pixel position in the image, so that each pixel of the image is associated with one prediction function depending only on its row and column positions; (4) the prediction is always an average of pixel intensities of the neighboring pixels; (5) the predictive functions 2 and also function 3 of Algorithm 2 use four or more neighbors to predict the pixel intensity, thus exploiting better and cleverer the local redundancies of the image; and (6) the algorithms can be efficiently implemented using only integer summations and differences, and bit-shift operations. These characteristics make the algorithms simple, fast and efficient.

We test the two variants of the algorithm with standard test images available in the literature and the compression ratios obtained are satisfactory. Algorithm 2 shows slightly better results than Algorithm 1 but the performance of the two algorithms are very similar.

A simple method for uniform region detection is implemented together with the proposed algorithms to allow better comparison with existing methods. Although the results of compression ratio obtained with the proposed method are satisfactory they are slightly worse than the results obtained with the JPEG-LS and JPEG2000 methods. Certainly better results could be achieved with the proposed method if we use an improved algorithm for uniform region detection.

REFERENCES

[1] B. C. Vemuri, S. Sahni, F. Chen, C. Kapoor, C. Leonard, and J. Fitzsimmons, "Lossless image compression", Available at http://citeseer.nj.nec.com/559352.html.

[2] J. Shukla, M. Alwani and A. K. Tiwari, "A Survey On Lossless Image Compression Methods, available at http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5486344.

[3] M.Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," Image Processing, IEEE Transactions on, vol. 9, no. 8, pp. 1309–1324, Aug. 2000.

[4] X. Wu and N. Memon, "Context-based, adaptive, lossless image coding," Communications, IEEE Transactions on, vol. 45, no. 4, pp. 437–444, Apr. 1997.

[5] R. Starosolski, "Simple fast and adaptive lossless image compression algorithm," Softw. Pract. Exper., vol. 37, pp. 65–91, Jan. 2007.

[6] P. Howard and J. Vitter, "Fast and efficient lossless image compression," in Data Compression Conference, 1993. DCC '93., Snowbird, Utah, Mar. 1993, pp. 351–360.

[7] Z. Wang, M. Klaiber, Y. Gera, S. Simon, and T. Richter, "Fast Lossless Image Compression With 2d Golomb Parameter Adaptation Based on JPEG-LS", 20th European Signal Processing Conference (EUSIPCO 2012) Bucharest, Romania, August 27-31, 2012.

[8] Deng G., Ye H.: Lossless image compression using adaptive predictor combination symbol mapping and context filtering. Proceedings of the IEEE International Conference on Image Processing, Kobe, Japan, Oct. 1999, Vol. 4, pp. 63-67.

[9] Li X., Orchard M. T.: Edge-Directed Prediction for Lossless Compression of Natural Images. IEEE Transactions on Image Processing, June 2001, Vol. 10(6), pp. 813-17.

[10] A. Said and W. A. pearçman, "Reversible image compression via multiresolution representation and predictive coding", IEEE Trans. Image Processing, Vol. 2094, pp. 664-674, Nov. 1993.

[11] A. R Calderbank, I. Daubechies, W. M. Swedens, and B.-L. Yeo, "Wavelet transforms that map integer to integers', Applied and Computational Harmonic Analysis 5, 332–369 (1998).

[12] A. Zani, M. Boliek, E. L. Schwarts, and M. J. Gornish, "CREW lossless/lossy medical image compression", Technical Report CRC-TR-9526, RICOH California Research Center, 1995.

[13] S. Dewitte and J. Cornelis, "Lossless integer wavelet transform', Technical Report IRIS-TR-0041, Royal Meteorological Institute Belgium, 1996.

[14] C. Christopoulos, A. Skodras, T. Ebrahimi, "The JPEG2000 Still Image Coding System an Overview", IEEE Transactions on Consumer Electronics, 46(4):1103-27, 2000.

[15] International Telecommunication Union (ITU), Telecommunication Standardization Sector, "ITU-T Recommendation T.24, Series T: Terminals for Telematic Services - Standardized digitized image set", June, 1998.