ADAPTIVE-QUALITY IMAGE COMPRESSION ALGORITHM

Abdel Rahman Alzoubaidi[1], Tamer Al-Sous[2] and Hussein Al-Bahadili[3]

[1]Department of Computer Engineering, Al Balqa Applied University, Salt, Jordan
[2]Faculty of Information Technology, Middle-East University, Amman, Jordan
[3]Faculty of Information Technology, University of Petra, Amman, Jordan

ABSTRACT

This paper presents the description and performance evaluation of a new adaptive-quality image compression (AQIC) algorithm. The compression ratio (C) and Peak Signal to Noise Ratio (PSNR) achieved by the new algorithm are evaluated through a number of experiments, in which a number of widely-used images of large and small sizes are compressed. In all experiments, C and PNSR achieved by the new algorithm are compared against those achieved by the PNG lossless compression image formats, JPEG lossy compression image format, and ZIP and WinRAR lossless compression tools. For all experiments the new algorithm provides C≈1.6, which is higher than those achieved by PNG, ZIP, and WinRAR, and lower than JPEG; and a PNSR of more than 30 dB, which is better than that achieved by JPEG.

KEYWORDS

Image compression; lossless data compression; lossy data compression; HCDC algorithm; image quality, compression ratio; PSNR.

1. INTRODUCTION

Data compression algorithms are developed to reduce the size of data so it requires less disk space for storage and less bandwidth when transmitted over data communication channels [1]. In wireless devices, data compression reduces the amount of accumulated errors and devices power consumption due to the reduction in the amount of the exchanged data [2, 3]. There are two fundamentally different styles of data compression can be recognized depending on the fidelity of the decompressed data, these are: lossless and lossy. In lossless data compression, an exact copy of the original data is reproduced after decompression; therefore, it is used whenever it is important to have an identical copy of the original data. Examples of lossless compression applications are the popular ZIP and WinRAR, and also lossless compression is used as a post-processing component within lossy compression applications [4, 5].

On lossy data compression an approximate copy of the original data set is reproduced after decompression; therefore, it can be used whenever it is not necessary to reproduce an exact copy of the original data, such as in some image and video compression applications. Because some information is discarded, it may achieve higher data compression ratios, depending on the type of compressed data set, and the amount of variations that are allowed to be introduced in the decompressed data set. In image compression, most lossy data compression algorithms approximate the colors values regardless whether they are common colors or uncommon colors within the image color set. It is clear that any approximation to the common colors could significantly affect the decompressed image quality; therefore, in order to enhance the quality of the compressed imageit is very important to develop new algorithms that maintain highest possible image compression ratio while reserving the original image qualities [6].

This paper presents a detailed description of a new algorithm that scans the image data to identify the most common colors (MCCs) and the least common colors (LCCs) within the image, and develops a shortest possible equivalent binary code for each color that ensure exact retrieval of MCC colors and minimal approximation to LCCs. This practically introduces minimum effect on the compressed image quality, and the overall effect will depend on the image colors frequencies. Therefore, we refer to this algorithm as adaptive-quality image compression (AQIC) algorithm.

In the new algorithm, the frequencies of the 8-bit colors of the image are determined and the colors are sorted from the MCC to the LCC. The colors are then approximated by eliminating the Least-significant-Bit (LSB) if the color set has more than 128 colors, and very slightly different colors are merged together to have not more than 32 colors in the image approximated color set. The colors are then re-sorted and split into two groups. The first one includes an optimized number of colors from the MCCs, and called the most common group (MCG); while the second one includes the remaining colors, which are usually the LCCs, and called the least common group (LCG). Then, a list of color equivalent binary codes is derived to ensure that most of the colors in the MCG can be exactly retrieved, and only colors in the LCG will be slightly affected introducing minimal effect on the image quality.

The performance of the AQIC algorithm is evaluated through a number of experiments, in which the algorithm was used to compress standard images of large and small sizes. The performance of the new algorithm is compared against the performance of a number of compressed image formats (PNG and JPEG) and a number of lossless compression tools (ZIP and WinRAR).

The paper is divided into seven sections. This section provides an introduction to the main theme of the paper. The rest of the paper is organized as follows: Section 2 reviews some of the most recent and related work. A description of the AQIC algorithm is given in Section 3. Section 4 describes the decompression procedure of the algorithm. Section 5 defines the parameters that are used in evaluating the performance of the new algorithm, while Section 6 presents, compares, and discusses the experimental results. Finally, in Section 7, based on the obtained results conclusions are drawn and a number of recommendations for future work are pointed-out.

2. LITERATURE REVIEW

This section reviews some of the most researches on image compression. A comprehensive review can be found in [7]. A novel bit-level lossless data compression algorithm based on the error correcting Hamming codes, namely, the Hamming Codes based Data Compression (HCDC) algorithm was developed by Al-Bahadili [8]. The HCDC algorithm has demonstrated an excellent performance and used by many researchers for text compression [9-10] and audio data compression applications [11-12].

Douaket. al. [13] developed a lossy image compression algorithm dedicated to color still images. They applied the Discrete Cosine Transform (DCT) followed by an iterative phase to guarantee a desired image quality. Then, to achieve the best possible compression ratio, they applied adaptive scanning providing for each (n, n) DCT block a matching (n×n) vector containing the maximum possible run of 0s at its end. Afterwards, they applied a systematic lossless encoder. Rahman et. al. [14] examined the relationship between image enhancement and data compression methods with special emphasis on image enhancement and lossy JPEG image compression. They also looked at the impact of compression on recovering original data from enhanced image.

Singh and Kumar [15] developed an Image Dependent Color Space Transform (ID-CST), exploiting the inter-channel redundancy optimally, which is very much suitable compression for large class of images. The comparative performance evaluated and a significant improvement was observed, objectively as well as subjectively over other quantifiable methods.

Telagarapu [16] analyzed the performance of a hybrid data compression scheme that uses the DCT and Wavelet transform. They concluded that selecting proper threshold method provides better PSNR. A novel Context-based Binary Wavelet Transform Coding (CBWTC) approach developed in [17], which shows that the average coding of the CBWTC is superior to that of the state-of-the-art grayscale image coders, and always outperforms the JBIG2 algorithm and other BWT-based binary coding technique for a set of test images with different characteristics and resolutions.

Ameer and Basir [18] described a simple plane fitting image compression scheme. The scheme can achieve a compression ratio of >60, while maintaining acceptable image quality. The compression ratio is further improved by optimizing its predicted model parameters to 100. The improvement in the compression ratio came at the expense of moderate to small quality degradations.

Li et. al. [19] improved the performance of the Vector Quantization (VQ) image compression and achieved a relatively high compression ratio. To obtain better reconstructed images, they developed an approach called the Transformed Vector Quantization (TVQ). A comparison of reconstructed image quality is made between TVQ, VQ, and standard JPEG approach. Hu and Chang [20] developed a lossless image-compression scheme, which is a two-stage scheme. The scheme reduces the cost for Huffman coding table while achieving high compression ratio. The scheme provides a good means for lossless image compression.

3. THE AQIC ALGORITHM

The AQIC algorithm is an adaptive-quality data compression algorithm especially develops for standstill image compression, where the quality of the compressed image depends on the range and frequency of colors within the original image. This section describes in details the compression procedure of the AQIC algorithm.

In this algorithm, the data of the original image is read one byte or character at a time; where each byte represents color value (V_i) between 0-255. Afterwards, the size of the colors set or the number of different colors in the image (N_{co}) using 8-bit color depth is found. If N_{co}>128, then the Least-Significant-Bit (LSB) of the color value is discarded and the remaining color bits are shifted one place to the right, which in turn, converts the 8-bit color to 7-bit color producing V_i between 0-127. Subsequently, the size of the colors set is changed and the new value of N_{co} must be determined using the 7-bit color depth ($N_{co}\leq 128$).

The algorithm then computes the color frequency (F_i) (i=1 to N_{co}) by dividing the counts of color i by the sum of counts of all colors (N) (i.e., $F_i=O_i/N$); where N can be calculated by:

$$N = \sum_{i=1}^{N_{co}} O_i \quad (1)$$

The colors frequencies are then sorted from the MCC to the LCC. Starting from the first MCC, the algorithm adds counts of colors that have either their 1st, 2nd, or 3rd bit differs from the first MCC. The new counts of the first MCC (O_1) can be calculated as: $O_1=O_{1i}+O_{1st}+O_{2nd}+O_{3rd}$, where O_{1i} is the initial counts for color 1; and O_{1st}, O_{2nd}, O_{3rd} are counts of colors that have either their 1st, 2nd, or 3rd bits differs from that for O_1.
For example, assume the color value of the first MCC is 35 (0100011), then the counts of color value 34 (0100010), 33 (0100001), and 39 (0100111) are added to the counts of the color value 35 and the colors 34, 33, and 39 are discarded and replaced by color value 35 during the decompression phase. So that if the initial counts of color 35 is 10, 34 is 8, 33 is 6, and 39 is 4, then the new counts for color 35 is 28. The colors 34, 33, and 39 are called the merged colors. The colors are then shifted up to fill the gap left-out by the merged color(s), and $N_{co}$is reduced by the number of merged colors (e), which is varied between 0 and 3 as it is explained in Table 1 ($N_{co}=N_{co}$-e).

Table 1. Associate colors.

e	**Explanation**
0	No merged color is found
1	Only one merged color is found (33 or 34 or 39)
2	Two merged colors are found (33 & 34 or 33 & 39 or 34 & 39)
3	Three merged colors are found (33 & 34 & 39)

The merging procedure continues until all possible colors are merged. By the end of this procedure, it can be approved that the value of N_{co} is always less than or equal to 32 ($N_{co}\leq 32$). The new colors list is then sorted in descending according to their new frequencies, and the new N_{co} and the sorted color list should be stored in the compressed image header.

The next step is the main core and contribution of the AQIC algorithm, which is called the encoding procedure, in which each 8-bit or 7-bit color value is replaced with the shortest possible binary representation to ensure maximum compression ratio. Although, there are many algorithms that have been developed and used to derive optimum colors equivalent codes (binary representation), such as Huffman, adaptive Huffman, Shannon-Fano, HCDC], etc. [7-10. Here, we develop a different, adaptive and very efficient coding procedure to ensure maximum possible compression ratio.

In AQIC encoding procedure, the sorted colors are divided into two groups. The first group comprises a carefully selected and optimized number of colors from the MCCs and it is called the most common group (MCG); while the second group comprises the remaining colors, which are usually the LCCs, and called the least common group (LCG). In order to maximize the compression ratio, the number of colors in the MCG (G_M) should be equal to 2^{m-1}, where m is the number of bits required to represent the colors in MCG. The number of colors in LCG (G_L) is calculated as $G_L=N_{co}-G_M$, and the number of bits required to represent the colors in LCG (n) is calculated as $n=1+[ln(N_{co}-G_M)/ln(2)]$. It can also be easily realized that for a maximum compression m should always be less than or equal to n ($m\leq n$).

The colors in each group is encoded, i.e., converted to *m*- or *n*-bit binary sequence equivalent to its sequence number in the group starting from 0 to G_M-1 or G_L-1 for MCG and LCG, respectively. This is similar to the adaptive coding used in [9] but for each group separately. In order to distinguish the MCG colors from the LCG colors during the decompression process, the MCG colors are preceded by 0, while the LCG colors are preceded by 1, except when G_M=1, where the color is replaced by 1-bit only ("0") or when G_L=1, where the color is replaced by 1-bit only ("1").

Based on the above discussion and to simplify the encoding procedure, we calculate all possible combinations of G_M and G_L and consequently *m* and *n* for each N_{co}, and the results obtained are summarized in Table 2.

Table 2. Associate colors.

N_{co}	$G_M(m)$	$G_L(n)$	$G_M(m)$	$G_L(n)$	$G_M(m)$	$G_L(n)$	$G_M(m)$	$G_L(n)$	$G_M(m)$	$G_L(n)$
1	1 (1)	0(0)								
2	1 (1)	1 (1)								
3	1 (1)	2(2)								
4	1 (1)	3(3)	2(2)	2(2)						
5	1 (1)	4(3)	2(2)	3(3)						
6	1 (1)	5(4)	2(2)	4(3)						
7	1 (1)	6(4)	2(2)	5(4)	4(3)	3(3)				
8	1 (1)	7(4)	2(2)	6(4)	4(3)	4(3)				
9	1 (1)	8(4)	2(2)	7(4)	4(3)	5(4)				
10	1 (1)	9(5)	2(2)	8(4)	4(3)	6(4)				
11	1 (1)	10(5)	2(2)	9(5)	4(3)	7(4)				
12	1 (1)	11(5)	2(2)	10(5)	4(3)	8(4)				
13	1 (1)	12(5)	2(2)	11(5)	4(3)	9(5)	8(4)	5(4)		
14	1 (1)	13(5)	2(2)	12(5)	4(3)	10(5)	8(4)	6(4)		
15	1 (1)	14(5)	2(2)	13(5)	4(3)	11(5)	8(4)	7(4		
16	1 (1)	15(5)	2(2)	14(5)	4(3)	12(5)	8(4)	8(4)		
17	1 (1)	16(5)	2(2)	15(5)	4(3)	13(5)	8(4)	9(5)		
18	1 (1)	17(6)	2(2)	16(5)	4(3)	14(5)	8(4)	10 (5)		
19	1 (1)	18(6)	2(2)	17(6)	4(3)	15(5)	8(4)	11(5)		
20	1 (1)	19(6)	2(2)	18(6)	4(3)	16(5)	8(4)	12(5)		
21	1 (1)	20(6)	2(2)	19(6)	4(3)	17(6)	8(4)	13(5)		
22	1 (1)	21(6)	2(2)	20(6)	4(3)	18(6)	8(4)	14(5)		
23	1 (1)	22(6)	2(2)	21(6)	4(3)	19(6)	8(4)	15(5)		
24	1 (1)	23(6)	2(2)	22(6)	4(3)	20(6)	8(4)	16(5)		
25	1 (1)	24(6)	2(2)	23(6)	4(3)	21(6)	8(4)	17(6)		
26	1 (1)	25(6)	2(2)	24(6)	4(3)	22(6)	8(4)	18(6)		
27	1 (1)	26(6)	2(2)	25(6)	4(3)	23(6)	8(4)	19(6)		
28	1 (1)	27(6)	2(2)	26(6)	4(3)	24(6)	8(4)	20(6)		
29	1 (1)	28(6)	2(2)	27(6)	4(3)	25(6)	8(4)	21(6)		
30	1 (1)	29(6)	2(2)	28(6)	4(3)	26(6)	8(4)	22(6)		
31	1 (1)	30(6)	2(2)	29(6)	4(3)	27(6)	8(4)	23(6)		
32	1 (1)	31(6)	2(2)	30(6)	4(3)	28(6)	8(4)	24(6)	16(5)	16(5)

It can be seen from Table 2 that when N_{co}>3 there are more than one possible combination for G_M and G_L and subsequently *m* and *n*. For example, for N_{co}=12, the possible combination for G_M and G_L are 1 and 11, 2 and 10, and 4 and 8. Since, we have the values of *m*, *n*, G_M, G_L, and all values of O_i (*i*=1 to N_{co}); the length of the compressed binary sequence (S_b) for each combination can be calculated using the following general equation:

$$S_b = m\sum_{i=1}^{G_M} O_i + n\sum_{i=1}^{G_L} O_{G_M+i} \tag{2}$$

The combination of G_M and G_L (m and n) that will be used to encode the colors is the one that provides minimum S_b, which subsequently provides the maximum compression ratio. The optimum combination will depends on the individual color count (O_i) (i=1 to N_{co}). In this way, we can be sure that the optimum coding rate is always selected.

After selecting the optimum combination for G_M and G_L (m and n), the algorithm starts constructing the colors equivalent binary codes depending on the selected m and n. For example, Table 3 presents the equivalent binary codes of the sorted color list of 12 colors (N_{co}=12) for the three possible combinations for m and n mentioned above.

The algorithm, then start construction the compressed binary sequence by marching through the image starting from the first color value up to the last one and replacing each color with its equivalent binary code.

Table 3. Sample of colors equivalent binary codes.

i	***Color Equivalent Binary Code***		
	***m*=1, *n*=11**	***m*=2, *n*=10**	***m*=4, *n*=8**
1	**0**	**00**	**000**
2	10000	**01**	**001**
3	10001	10000	**010**
4	10010	10001	**011**
5	10011	10010	1000
6	10100	10011	1001
7	10101	10100	1010
8	10110	10101	1011
9	10111	10110	1100
10	11000	10111	1101
11	11001	11000	1110
12	11010	11001	1111

After the construction of the compressed binary sequence, the algorithm enters the last stage, which includes: conversion of the compressed binary sequence to 8-bit character string, construction of the compressed image header (which should include all information necessary for the decompression process), appending the compressed string to the image header, creation of the compressed image file, and save the combined image header and compressed string into the compressed image file. In this case, the size of the compressed image file is calculated as: S_c=H+$\lceil S_b/8\rceil$, where S_c is the size of the compressed image in Bytes, H is the length of the compressed image header, and S_b is the length of the compressed binary sequence. Figure 1 outlines the compression procedure of the AQIC algorithm.

```
Read image data
Calculate the number of colors ($N_{co}$)
If ($N_{co}$>128)
    Convert 8-bit color to 7-bit color
    Re-calculate the number of colors ($N_{co}$)
End If
If ($N_{co}$=1)
    Set $m$=1 and $n$=0
    Construct the colors equivalent binary codes by setting the color equivalent binary code to “0”
Else If ($N_{co}$=2)
    Calculate the occurrence ($O_i$) or counts of each color
    Calculate the sum of counts of all colors ($N$)
    Calculate the colors frequencies ($F_i$=$O_i$/$N$)
    Sort the colors according to $F_i$ from the MCC to the LCC
    Set $m$=1 and $n$=1
    Construct the colors equivalent binary codes by setting the MCC to “0” and the LCC to “1”
Else
    Calculate the occurrence ($O_i$) or counts of each color
    Calculate the sum of counts of all colors ($N$)
    Calculate the colors frequencies ($F_i$=$O_i$/$N$)
    Sort the colors according to $F_i$ from the MCC to the LCC
    Perform merging procedure
    Re-calculate new number of colors ($N_{co}$)
    Re-calculate the new colors frequencies ($F_i$=$O_i$/$N$)
    Re-sort the colors according to the $F_i$ from the MCC to the LCC
    Calculate the optimum $G_M$ and $G_L$ ($m$ and $n$)
    Construct the colors equivalent binary codes
End If
Read image data and replace each color value with its equivalent $m$ or $n$ bit binary code.
Convert the compressed binary sequence to 8-bit character string.
Construct the compressed image header containing all information required during decompression.
Append the compressed image string to the image header.
Create a compressed image file.
Save the combined image header and the compressed image string into the compressed image file.
```

Figure 1.The compression procedure of the AQIC algorithm.

4. Decompression Procedure of the AQIC Algorithm

The decompression procedure of the AQIC algorithm is very simple and straightforward and it is accomplished much faster than the compression process, therefore, the algorithm behaves as an asymmetric compression algorithm due to the difference between the compression and the decompression processing times.

The decompression algorithm of the AQIC algorithm can be divided into two main phases. In the first phase, the algorithm reads in the header data and prepares the list of the colors values. In particular, it reads the values of G_M and G_L and computes m, n, and N_{co}. Then it reads the sorted colors values from V_i (i=1 to N_{co}).

Finally, in this phase, the algorithm constructs the list of the colors equivalent binary codes. In the second phase, which is the core of the decompression procedure, the algorithm reads in the compressed image data and converts every compressed color binary code to its equivalent uncompressed color value to re-construct the decompressed image. Figure 2 outlines the procedure for the decompression procedure of the AQIC algorithm.

```
Read-in the compressed image data
Extract the compressed image header and get values of $G_M$ and $G_L$.
Calculate $m$, $n$ and $N_{co}$=$G_M$+$G_L$.
Read-in the colors values ($V_i$ for $i$=1 to $N_{co}$)
Extract the compressed image string.
Convert the compressed image string to binary sequence.
If ($N_{co}$=1) Then
    Construct colors equivalent binary codes (MCC=“0”).
    Do
        Read-in 1-bit at a time.
        Find the associate color from the above list and append it to the uncompressed image data.
    Loop until end of image compressed binary sequence.
Else If ($N_{co}$=2)
    Construct the colors equivalent binary codes (MCC="0" and LCC="1").
    Do
        Read-in 1-bit at a time.
        Find the associate color from the above list and append it to the uncompressed image data.
    Loop until end of image compressed binary sequence.
Else
    Construct the colors equivalent binary codes.
    Do
        Read in 1-bit ($B$)
        If (B="0") Then
            Read in ($m$-1)-bit.
            Find the associate color from the above list and append it to the uncompressed image data.
        Else
            Reads in ($n$-1)-bit.
            Find the associate color from the above list and append it to the uncompressed image data.
        End If
    Loop until end of image compressed binary sequence.
End If
```

Figure 2.Procedure for the decompression procedure of the AQIC algorithm.

5. PERFORMANCE MEASURES

The performance of the AQIC algorithms is evaluated in terms of two parameters, namely: compression ratio (C) and Peak Signal to Noise Ratio ($PSNR$). C represents the ratio between the sizes of the original image file (S_o) and the size of the compressed image file (S_c). It can be calculated as [1]:

$$C = S_o/S_c \tag{3}$$

The MSE is the cumulative squared error between the compressed and the original images, and it can be calculated by [1, 7]:

$$MSE = \frac{1}{X\cdot Y}\sum_{y=1}^{Y}\sum_{x=1}^{X}[I(x,y) - K(x,y)]^2 \tag{4}$$

Where $I(x,y)$ is the original image, $K(x,y)$ is the approximated version (which is actually the decompressed image) and X and Y are the dimensions of the images. A lower value for MSE means lesser error.

The $PSNR$ is a measure of the peak error, which is most commonly used as a measure of quality of reconstruction of lossy compression images, and it is usually expressed in terms of the logarithmic decibel (dB) scale as follows [1, 7]:

$$PSNR = 20\ \cdot log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \tag{5}$$

Where, MAX_I is the maximum possible pixel value of the image. When the pixels are represented using 8-bit per pixel, this is 255. More generally, when samples are represented using linear Pulse Code Modulation (PCM) with p-bit per pixel, MAX_I is 2^p-1. For 24-bit image, the definition of $PSNR$ is the same except MSE is the sum over all squared value differences divided by image size and by 3. Alternately, for color images the image is converted to a different color space and *PSNR* is reported against each channel of that color space.

A higher value of *PSNR* is good because it means that the ratio of signal-to-noise is higher. Here, the signal is the original image, and the noise is the error in reconstruction. So, if a compression scheme having a lower *MSE* (higher *PSNR*), it can be recognized as a better one. Typical values for the *PSNR* in lossy image and video compression are between 30 and 50 dB, where higher is better. Acceptable values for wireless transmission quality loss are considered to be about 20 dB to 25 dB. In lossless compression, the two images are identical, and thus *MSE* is zero. In this case *PSNR* is undefined [1].

6. Experimental Results and Discussions

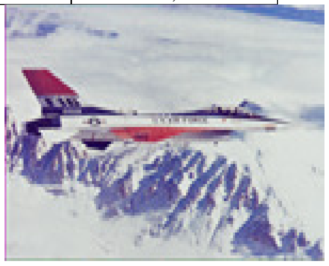
A number of experiments are performed to evaluate the performance of the AQIC algorithm. These experiments use the algorithm to compress a set of widely-used test images of large and small sizes. In particular, we selected five images, the dimensions and sizes of these images are given in Table 4 and the images are shown in Figure 3.

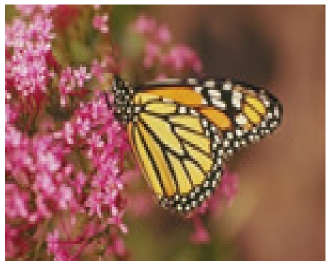
Table 4. Dimensions and Sizes of test images.

#	Image	Large images		Small images	
		Dimensions (Pixel)	**Size (Byte)**	**Dimensions (Pixel)**	**Size (Byte)**
1	Flowers	500x362	543,054	239x240	172,854
2	CornField	512x480	737,334	256x240	184,374
3	AirPlane	512x512	786,486	320x213	204,534
4	Monarch	768x512	1,179,702	300x240	216,054
5	Girl	720x576	1,244,214	320x231	221,814

Flowers.BMP

CornField.BMP


AirPlane.bmp


Monarch.BMP

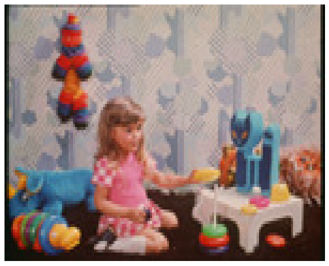

Girl.BMP

Figure 3. Test images.

The *C* and *PNSR* of AQIC and a number of standard lossless and lossy compressed image formats, and lossless compression tools are listed in Tables 5 and 6. It can be seen from Table 5 that AQIC achieves a *C* of ≈1.6. This is because the compressed colors span over two groups, each of 16 colors because the images selected have a continuous color distribution; so that each uncompressed 8-bit color is expressed with only 5-bit (1-bit identifying the group number and 4-bit identifying the sequence of the color within the group). The variation from 1.6 is due to the effect of the adding the compressed file header to the compressed image.

AQIC achieves higher *C* than that achieved by the lossless PNG and less than that achieved by the lossy JPEG. This is at the cost of some reduction/improvement in the image quality, where AQIC provides better image quality than that produced by the JPEG format and of course less image quality in comparison with PNG. It is also clear from Table 5 that AQIC compression ratio is higher than ZIP and competitive to WinRAR.

Table 5. Comparing *C* for various images.

Image	**PNG**	**JPEG**	**ZIP**	**WinRAR**	**AQIC**
Large images					
Flowers	1.089	5.949	1.226	1.603	1.600
Corn Field	1.126	7.139	1.268	1.698	1.600
Air Plane	1.229	8.748	1.400	1.889	1.600
Monarch	1.247	9.063	1.451	2.229	1.600
Girl	1.195	8.603	1.351	2.310	1.600
Small images					
Flowers	1.247	7.011	1.372	1.846	1.599
Corn Field	1.118	5.838	1.246	1.523	1.599
Air Plane	1.064	6.701	1.186	1.774	1.599
Monarch	1.066	6.425	1.180	1.783	1.599
Girl	1.071	5.256	1.191	1.564	1.599

Table 6.Comparing *PNSR* for various images.

Large images					
Image	**PNG**	**JPEG**	**ZIP**	**WinRAR**	**AQIC**
Flowers	∞	28.03	∞	∞	33.26
Corn Field	∞	30.14	∞	∞	31.90
Air Plane	∞	29.96	∞	∞	32.76
Monarch	∞	35.50	∞	∞	32.60
Girl	∞	33.13	∞	∞	33.06
Small images					
Flowers	∞	30.41	∞	∞	32.80
Corn Field	∞	29.12	∞	∞	32.58
Air Plane	∞	26.74	∞	∞	33.04
Monarch	∞	30.67	∞	∞	32.79
Girl	∞	31.26	∞	∞	32.30

For many images, AQIC provides standards compressed image quality and also achieves higher image quality than JPEG, where *PSNR* is always above 30 dB, where higher is better; while JPEG for some cases has <30 dB *PNSR* values. The PNG, ZIP and WinRAR are lossless compression;

therefore, they present undefined *PNSR* (∞). AQIC almost provides the same performance in terms of *C* and *PNSR* for both large-size and small-size images compression, while all other compression formats or tools provide variable and unpredicted performance

7. Conclusions

The main conclusions of this paper are: for images of continuous color distribution, the AQIC algorithm can achieve a compression ratio of ≈1.6 as explained above. The compression ratio achieved by AQIC is higher than that achieved by the lossless PNG and less than that achieved by the lossy JPEG. This is at the cost of some reduction/improvement in the decompressed image quality. In particular, the new algorithm provides higher *C* than ZIP and competitive performance to WinRAR for almost all standard test images. In terms of quality, the new algorithm provides better image quality than that produced by JPEG and less image quality in comparison with PNG.

The main recommendations for future work are to perform further investigations to cover a wider range of image sizes and of various colors frequencies, and evaluate the compression ratio after applying lossless compression to images compressed with the new algorithm.

References

[1] Sayood, K. (2012). Introduction to data compression (4th Ed.). Morgan Kaufmann.

[2] Kolo, J. G., Shanmugam, S. A., Lim, D. W. G., Ang, L. M., and Seng, K. P. (2012). An adaptive lossless data compression scheme for wireless sensor networks. Journal of Sensors, Vol. 2012, Article ID 539638, 20 pages. doi:10.1155/2012/539638.

[3] Xu, R., Li, Z., Wang, C., & Ni, P. (2003). Impact of data compression on energy consumption of wireless-networked handheld devices. Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS '03), 302-311.

[4] Kung, W.-Y., Kim, C.-S., &. Kuo C.-C. J. (2005). Packet video transmission over wireless channels with adaptive channel rate allocation. Journal of Visual Communication and Image Representation, Vol. 16, Issue 4-5, 475-498.

[5] Rueda, L. G., &Oommen, B. J. (2006). A Fast and Efficient Nearly-Optimal Adaptive Fano Coding Scheme.

[6] Brittain, N. J., & El-Sakka, M. R. (2007). Grayscale true two-dimensional dictionary-based image compression. Journal of Visual Communication and Image Representation, 35–44.

[7] Alsous, T. (2013). Developing a high-performance adjustable-quality data compression scheme for multimedia messaging. M.Sc Thesis. Middle East University, Faculty of Information Technology, Amman-Jordan.

[8] Al-Bahadili, H. (2008). A novel lossless data compression scheme based on the error correcting Hamming codes. Computers & Mathematics with Applications, Vol. 56, Issue 1, 143–150.

[9] Al-Bahadili, H., & Rababa'a, A. (2010). A bit-level text compression scheme based on the HCDC algorithm. International Journal of Computers and Applications (IJCA), Vol. 32, Issue 3.

[10] Al-Bahadili, H., & Al-Saab, S. (2011). Development of a novel compressed index-query Web search engine model. International Journal of Information Technology and Web Engineering (IJITWE), Vol. 6, No. 3, 39-56.

[11] Al-Zboun, F., Al-Bahadili, H., Abu Zitar, R., &Amro, I. (2011). Hamming correction code based compression for speech linear prediction reflection coefficients. International Journal of Mobile &Adhoc Network, Vol. 1, Issue 2, pp. 228-233.

[12] Amro, I., Abu Zitar, R., & Al-Bahadili, H. (2011). Speech compression exploiting linear prediction coefficients codebook and hamming correction code algorithm. International Journal of Speech Technology, Vol. 14, No. 2, 65-76.

[13] Douak, F., Benzid, R., &Benoudjit, N. (2011). Color image compression algorithm based on the DCT transform combined to an adaptive block scanning. AEU - International Journal of Electronics and Communications, Vol. 65, Issue 1, 16–26.

[14] Rahman, Z., Jobson, D. J., &Woodell, G. A. (2011). Investigating the relationship between image enhancement and Image compression in the context of the multi-scale retinex. Journal of Visual Communication and Image Representation, Vol. 22, Issue 3, 237–250.

[15] Singh, S. K., & Kumar, S. (2011). Novel adaptive color space transform and application to image compression. Journal of Signal Processing: Image Communication, Vol. 26, Issue 10, 662–672.

[16] Telagarapu, P., Naveen, V. J., Prasanthi, A. L., &Santhi, G. V. (2011). Image Compression Using DCT and Wavelet Transformations. International Journal of Signal Processing, Image Processing and Pattern Recognition, Vol. 4, No. 3, 61-74.

[17] Pan, H., Jin, L. Z., Yuan, X. H., Xia, S. Y., & Xia, L. Z. (2010). Context-based embedded image compression using binary wavelet transform. Journal of Image and Vision Computing, Vol. 28, Issue 6, 991–1002.

[18] Ameer, S., &Basir, O. (2009). Image compression using plane fitting with inter-block prediction. Journal of Image and Vision Computing, Vol. 27, Issue 4, 385–390.

[19] Li, R. Y., Kim, J., & Al-Shamakhi, N. (2002). Image compression using transformed vector quantization. Journal of Image and Vision Computing, Vol. 20, Issue 1, 37–45.

[20] Hu, Y. C., & Chang, C. C. (2000). A new lossless compression scheme based on Huffman coding scheme for image compression. Journal of Signal Processing: Image Communication, Vol. 16, Issue 4, 367-372.

AUTHORS

Abdel Rahman Alzoubaidi is Associate Professor at Department of Computer Systems Engineering and Computer Center Director at Al-Balqa Applied University.He studied Automations and Computer Engineering at the Technical University of Iasi and then left for Mu'tah University where he worked as teaching assistant.He obtained his MPhil and DPhil in data Communications and Computer Networks in 1996.He subsequently joined the Department of Computer Engineering at Mu'tah University as faculty where he became Associate Professor in 2004 and served as Director of Computer Center from 1996 to 2007. He was appointed President's Assistant for ICT.His research interests center on improving computing systems, Cloud Computing, eLearning, Cyber Security, Wireless and Ad-hoc networks.He worked as ICT consultant for the Ministry of Education. He designed and implemented many innovative ICT projects in Jordan. He has given numerous invited talks and tutorials, and is a consultant to companies involved in Internet technologies.

Tamer Sous (tamer_sous@hotmail.com) holds a B.Sc degree in Computer Science from Zaytoonah Private University, Amman, Jordan in 2006. He received his M.Sc in Computer Science for the Middle East University, Amman, Jordan in 2013. His research interests include image processing and compression, multimedia applications, computer architecture, wire and wireless computer networks, distributed and cloud computing, software development, and Web and mobile applications programming.

Hussein Al-Bahadili (hbahadili@uop.edu.jo) is an associate professor at University of Petra. He received his PhD and M.Sc degrees from University of London (Queen Mary College) in 1991 and 1988. He has published many papers in different fields of science and engineering in numerous leading scholarly and practitioner journals, and presented at leading world-level scholarly conferences. He published four novel algorithms in Computer Networks, Data Compression, Network Security, and Web Search Engine. He supervised more than thirty PhD and M.Sc theses. He edits a book titled Simulation in Computer Network Design and Modeling: Use and Analysis, which is published by IGI-Global. He has published more than ten chapters in prestigious books in Information and Communication Technology. He is also a reviewer for a number of books. His research interests include computer networks design and architecture, routing protocols optimizations, parallel and distributed computing, cryptography and network security, data compression, software and Web engineering.