Research Journal of Applied Sciences, Engineering and Technology 12(6): 680-685, 2016
DOI:10.19026/rjaset.12.2716
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: August 5, 2015 Accepted: September 7, 2015 Published: March 15, 2016

Research Article

Lossy Image Compression by Rounding the Intensity Followed by Dividing (RIFD)

Dr. M.A. Otair and F. Shehadeh
Faculty of Computer Sciences and Informatics, Amman Arab University, 11953 Amman, Jordan

Abstract: Several millions of digital images are transmitted every minute via mobile applications. The main feature of these images is their huge sizes. However, most of their details are not important such as natural images. Continuous efforts are achieved to utilize the wireless bandwidth and capacity for mobile users. One of the most significant efforts is the image compression. The aim of this study is to introduce a new lossy technique called RFID for compressing images, in order to overcome these problems by achieving high compression ratio. The proposed technique depends on increasing the redundancy and similarity among the neighboring pixels of images by rounding the pixels' intensities followed by the dividing process, which makes compression feasible. It can be applied alone or followed by any lossless compression algorithm. Experimental results show a great performance when RIFD followed by Huffman algorithm.

Keywords: Compression ratio, huffman algorithm, lossy compression, mobile applications

INTRODUCTION

Image compression is the method of decreasing the image size without affecting the image quality or it will be affected at an acceptable level. Thus, images transmission over networks can be obtained by the compressed size which saves the memory space, as well (Mathur *et al*., 2012). However, complexities of the compression algorithms should be low enough for a given system in addition to enhance the transmission throughput.

Lossless and lossy are the two main categories of compression. In lossless compression, the original image file can be recovered from the compressed one. In this category of compression only a few mechanisms are followed for the reduction of data and then the data reduction is very restricted (White Paper, 2008). Lossy compression are particularly convenient for specific applications such natural images where slight (or not perceived) loss of accuracy is not noticeable and passable to accomplish a significant decreasing in bit rate.

Lossy compression techniques are relinquishing a specific loss of fidelity in transmission for fully incremented compression. These techniques behave effective when implemented on the digitized voices or digital natural images. By their nature, the analog representations of these digital forms are not exemplary to start with; consequently it will be acceptable if the input and output are not exactly matching (Nelson, 1992).

After a famous algorithm called Huffman (1952) coding was introduced in 1952 many though that no more research was needed in this area because Huffman (1952) was performing optimally. A few decades later several new algorithms were starting to develop which fixed some of the problems that had been discovered with the Huffman algorithm.

Evolution of lossy compression techniques is motivated by comprehensive researches in 1980s, which concentrate on the human eye limitations. So, all of lossy algorithms depends on the fact that little adjustments and loss of information via coding/encoding processes overwhelmingly do not destroy the image quality as realized by the human eyes. Smaller files and shorter encoding time are the target to be developed by many commercial companies and different researchers for many years.

In a particular space of memory, reduced image size by compression algorithm is assisting in store additional images. It also decreases the required time to transmit images over the Internet (Verma *et al*., 2012). So, the main objective of this study is to develop and propose a new lossy image compression technique where a slight information loss is accepted. However, the proposed compression technique has a very significant feature which is considerable compression efficiency.

IMAGE COMPRESSION BACKGROUND

Image compression techniques encode the original images with less number of bits by reducing the

Corresponding Author: Dr. M.A. Otair, Faculty of Computer Sciences and Informatics, Amman Arab University, 11953 Amman, Jordan

redundancies of pixel values into an image and to transmit or store them in an effective shape (Wei, 2008).

Accessing the multimedia through Internet and the telecommunications network makes the demand for communication of multimedia is increasing dramatically. With the explosive growing of digital cameras and its use, then transferring, manipulating and storing of digital images become very important subjects to be handled especially that images sizes are very large and takes most of the disk spaces. The main partition of the communication bandwidth is occupied by the multimedia data which comprises the image data. Consequently developing of robust image compression techniques is totally mandatory (Ames, 2002). Digital images have a common feature where there is high correlation between the neighboring pixels, which causes highly information redundancy. The main goal of image compression techniques is to represent the images with less correlated pixels. Generally in image compression, irrelevancy and redundancy are used as two primary principles. In Irrelevancy, not noticeable pixel values are neglected and the redundancy is removed from the original image in the Redundancy (Gautam, 2010).

How an image is stored? A digital image can be represented by a two (grayscale image) or three (colored image) dimensional array of pixels. When the image is a grayscale, then the pixels will have a positive integer numbers as the intensities values. The typical bit depth of grayscale images could be 8 or 16 bits, then the range of pixel values of the image will be between 0 to 2^N-1 (Where N is bit depth). For example, if the bit-depth is 8 then the range of intensity values will be between 0 and 255. White is 255, Black is 0 and the gray levels will be represented by the other values 1 to 254). These values are encoded using binary code representation, 11111111 is the code for 255 and 10000111 is code for 135 as the following:

128	64	32	16	8	4	2	1
1	0	0	0	0	1	1	1

Equivalent to 135 in decimal coding system, the summation of all values above every one: 128+4+2+1.

The compression algorithms of colored images can be derived from the compression algorithms of grayscale images (Starosolski, 2007). In the colored images, each pixel is represented by three values for: red, green and blue known as RGB (each color occupies 8 bits and the total of 24 bits). In an RGB image, an additive color model is used to represent the color by merging the intensities for the three components RGB of the pixel (Pane and Joe, 2005). For example, black is represented by setting the zeros for red, blue and green. To represent white, the maximum value 255 will be set for RGB for the pixel. The other colors are represented by merging the three colors with different intensity values results a new color as single value for the pixel (Pane and Joe, 2005).

Compression performance measures: Measuring the compression algorithm performance depends on the type of the application and based on the different criteria that used. However, the capacity efficiency is the main performance criterion (in other cases, time efficiency can be considered). Generally, measuring the image compression algorithm performance is sophisticated and it is not an easy task because the manner of compression depends on bit-stream redundancy in the original image. The performance could depend on the structure and the type of the original image or it is based on the compression algorithm category: lossless or lossy. If a lossless image compression algorithm is measured, then the time and capacity efficiencies will be less examined when the measuring will be for the lossy compression ones.

The Performance of compression algorithms can be measured or evaluated using one or more of the following (Kodituwakku and Amarasinghe, 2007):

$$compressio\ n\_ratio = \frac{size\_before\_compressio\ n}{size\_after\_compressio\ n} \quad (1)$$

$$saving\_percentage = \frac{size\_before\_compr - size\_after\_compr}{size\_before\_compr}\% \quad (2)$$

All the above evaluation measures use the image size in order to examine the efficiency of the algorithm. However, other performance evaluation measures can be used such: probability distribution, computational complexity and compression time.

Specifically, the quality measures of lossy algorithms can be classified into: Subjective and Objective measures. In the subjective, the compressed image will be evaluated by human observer who judges the quality of the algorithm based on the quality rating in Table 1 (Frendendall and Behrend, 1960):

Table 1: Rating of compressed image quality

Value	Rating	Description
1	Excellent	An image of extremely high quality, as good as you could desire.
2	Fine	An image of high quality, providing enjoyable viewing. Interference is not objectionable.
3	Passable	An image of acceptable quality. Interefence is not objectionable.
4	Marginal	An image of poor quality; you wish you could improve it. Interefence is somewhat objecetionable.
5	Inferior	A very poor imag, but you could watch it. Objectionable interefence is definitely present.
6	Unusable	An image so bad that you could not watch it.

RIFD (THE PROPOSED ALGORITHM)

Most the digital images have the same feature which is an information redundancy. These redundancies could be removed as much as possible by minimizing the bit-depth of the pixels' values into minimal number of bits in order to accomplish high compression ratio. The proposed algorithm in this study tries to remove or minimize the information redundancy included into the images while keeping its recognized visual quality and comparable to the original one. For instance; if the bit-depth for each pixel in the original image is 8 bits and the compression algorithm reduces the required bits to 5, then the image will be compressed. The only drawback of this algorithm is there will be some precision loss and some slight distortions could not be negated. However, as one of lossy compression this algorithm produces an acceptable distortion in many imaging applications because a slight loss in the image is not noticed by the human eye.

The main idea of the proposed algorithm is based on two facts:

- The neighboring pixels are correlated or very similar
- The human eye can perceive a very limited number of intensities. So; if the intensity values of the adjacent pixels are rounding to the same value, then the redundancy will increase and the updated intensity values will not noticeable by human eyes. Increasing the information redundancy assists the image to be more compressed. Therefore, finding a less correlated representation of image is one of the most important tasks.

One of the basic concepts in compression is the reduction of redundancy and irrelevancy. This can be done by removing duplication from the image. Sometime, Human Visual System (HVS) cannot notice some parts of the signal, i.e., omitting these parts will not be noticed by the receiver. This is called as Irrelevancy.

The basic idea of the proposed algorithm is to check the whole pixels and transform each pixel into a number divisible by 10 according to the following conditions. Here, a new formula is proposed to transform any number in the range 0-255 into a number that when divided by 10 the result is always lying between 0-9. Therefore, the pixels 200, 201 and 202 are the same for the human eye. Hence, a novel algorithm have been proposed to transform each pixel in the range 0-255 into the following numbers 0, 10, 20, 30, 40,…, 200, 210,..., 250, (i.e., multiples of 10). After that, it narrows the range of pixel values which can be represented by 5 bits rather than 8 bits by dividing the new values (0, 10, 20,… 250). This algorithm called RIFD because it is Rounding the Intensity Followed by Division. It consists of two main steps: rounding and dividing.

221	232	231	242	246	247	251	250
220	227	231	236	242	241	250	251
221	215	221	232	240	247	251	251
217	216	216	225	237	241	245	247
216	221	217	222	231	235	242	247
220	216	222	215	227	231	242	247
216	216	211	216	222	227	237	247
217	216	211	216	217	222	237	235

Fig. 1: 8×8 block as a sample

220	230	230	240	250	250	250	250
220	230	230	240	240	240	250	250
220	210	220	230	240	250	250	250
220	220	220	220	240	240	240	250
210	220	220	220	230	230	240	250
220	220	220	210	230	230	240	250
210	210	210	210	220	230	240	250
220	210	210	210	220	220	240	240

Fig. 2: 8×8 Block AfterRounding Process

22	23	23	24	25	25	25	25
22	23	23	24	25	24	25	25
22	21	22	23	24	25	25	25
22	22	22	22	24	24	24	25
21	22	22	22	23	23	24	25
22	22	22	21	23	23	24	25
21	21	21	21	22	23	24	25
22	21	21	21	22	22	24	24

Fig. 3**:** 8×8 Block after Dividing Process

Additionally, it can be seen that the new pixels are always having zero remainder when divided by 10. Consequently, the resulting numbers are multiples of 10 between 0-250, which are 26 values: 0, 10, 20, 30, … and 250. Hence, if we divide these numbers by 10 again we will get remainder range from 0-25. Now, let us take a practical example, Fig. 1 is an exactly 8×8 block have been taken from any arrays in any image.

After the rounding step of RIFD is applied (i.e., transform each pixel into multiple of 10), the new 8×8 block will be as in Fig. 2.

After the dividing step of RIFD is applied (i.e., decrease the bit-depth of each pixel), the new 8×8 block will be as in Fig. 3.

Table 2: Binary representation for pixel values from 0 to 25

Pixel value	0	1	2	3	4	5	6	7	8	9	10	….	25
Binary value	0	1	10	11	100	101	110	111	1000	1001	1010	….	11001

It is well known that every decimal number 0-255 is represented by 8-bits (1 byte: ASCII coding representation). After transformation into by ten and making each number between 0-25 (255/10 = 25). So, every new intensity value will be represented by 5 bits only as shown in Table 2.

The main benefit of the RIFD is to increase redundancy or the probability of each pixel (it can notice that from Fig. 2). Thus, this will prepare the image to be much compressed efficiently using any lossless technique such as Huffman coding.

How RIFD does work?

Encode phase (Assume that the bit-depth is 8 bits):

- Read an intensity value f(x, y).
- The intensity value in each image pixel is rounded to the nearest tenth. For example, if the intensity value is 157, then it will be rounded to 160.
- Divide the rounded intensity by 10 and store it as new intensity value g(x, y). For example, 160 is divided by 10 and then 16 will store as new intensity value for g(x, y).
- To optimize RIFD, Huffman technique could be implemented in order to increase the compression performance.

Important note: If the encode phase is implemented on the colored images, then the pervious steps will be applied 3 times on the RGB for each pixel (Matlab code to process colored images in Appendix-A). If the bit-depth is 16, then the intensity will be rounded to the nearest thousand and then divided by 1000. For example, if the intensity is 64,235 it will be rounded to 64,000 and then divided by 1000 to produce new intensity 64. So, all new intensity values will be between 0 and 66 (i.e., every new intensity value will be represented by 6 bits).

Decode phase:

- Decode the code generated by Huffman algorithm.
- Multiply each intensity value by 10. For example, 16 multiply by 10 to retrieve the rounded intensity value 160.

Matlab code of RIFD: The Matlab Code of RIFD as follow:

```
for x = 1:M % M is the number of image rows
for y = 1:N % N is the number of image rows
        if mod( f(x, y), 10) ==9
            f(x, y) = g(x, y) + 1;
    else if mod( f(x, y), 10) == 8
        f(x, y) = g(x, y) + 2;
    else if mod( f(x, y), 10) == 7
        f(x, y) = g(x, y) + 3;
    else if mod( f(x, y), 10) == 6
        f(x, y) = g(x, y) + 4;
    else if mod( f(x, y), 10)==5
        f(x, y) = g(x, y) - 5;
    else if mod( f(x, y), 10) == 4
        f(x, y) = g(x, y) - 4;
    else if mod( f(x, y), 10)==3
        f(x, y) = g(x, y) - 3;
    else if mod( f(x, y), 10) == 2
        f(x, y) = g(x, y) - 2;
    else if mod( f(x, y), 10) == 1
        f(x, y) = g(x, y) - 1;
    else if mod( f(x, y), 10) == 0
         f(x, y) = g(x, y) ;
  % end; 10 times here for 10 if statements
  g(x, y) = g(x,y) /10;
 end; % end for x
end; % end for y
```

RESULTS AND DISCUSSION

This section presents the experiment accomplished in order to evaluate the performance of RIFD. Matlab R2010a is used in all experiments and it shows that the proposed algorithm is very significant improvement in the compression ratios values.

A new test set of several natural continuous colored and grayscale images was designed to examine the performance of RIFD algorithm. The major cause of designing this set was that there is no overtly obtainable standard set of test images including huge images with high quality, which were primarily obtained with different sizes and bit-depths 8 and 16 bits of grayscale and colored images. The set include natural continuous colored and grayscale images of different bit depths (8 and 16 bits), different sizes (up to 181,737 bytes) and format (png and jpg).

Table 3: Comparative between Huffman and RIFD based on CR

Image name	Image type	Original image size	Image size by Huffman	CR by Huffman	Image size by RIFD	CR by RIFD
Cameraman (png)	Grayscale 8 bit	38267	33697	1.13	8708	4.39
Lena(png)	Grayscale 8 bit	38936	36344	1.07	9225	4.22
House(png)	Grayscale 8 bit	34985	28549	1.22	6089	5.74
Peppers (png)	Grayscale 8 bit	40181	37978	1.05	10074	3.98
Lena (png)	Colored RGB	181737	169546	1.07	42600	4.27
Bird (jpg)	Colored RGB	32629	31340	1.04	3124	10.44
Lisa(jpg)	Colored RGB	33618	31142	1.07	2250	14.9
Lena (png)	Grayscale 16 bit	76156	63463	1.20	22435	3.39

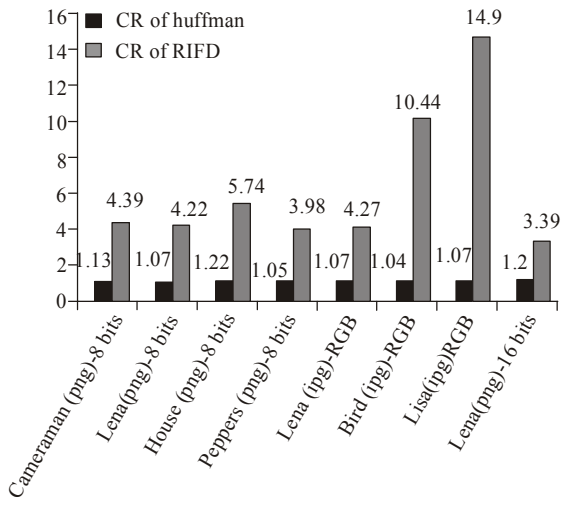

Fig. 5: Comparison of compression ratio between huffman and RIFD

The tested images in the designed test set and the images after decompression using RIFD are presented in Appendix-B. As mentioned before that the losses causes by the proposed algorithm is not perceived by the human observers.

In order to test the performance of RIFD, the compression ratio was used as the main compression measure because it is the mostly used measure in the literature. However, the performance of RIFD must be compared with the performance of a well-known compression algorithm (Huffman algorithm is used). Table 3 shows the result of all experiments over 8 images with different bit-depth, size, format and type. The columns of this table ass follow: image name, image type, original image size, compressed image size by Huffman, compression ration using Huffman, compressed image size by RIFD and compression ration using RIFD respectively. Moreover, all the compression ratios (in the last column) resulted from RIFD depicted its superiority over the compression ratios that obtained from classical Huffman algorithm (in the 5th column).

Figure 5 is a graphical representation for columns: 1, 5 and 8 respectively from the Table 3. By reviewing the column charts, it is noticeable that RIFD gives better compression ratio for all images in test set. Moreover, the performance of RIFD is better than the classical Huffman ten or more times with the colored images and especially the .jpg images (Bird.jpg and Lisa.jpg). The average of compression ratio for all images in the test set using Huffman was 1.11, while it was 6.42 using RIFD.

CONCLUSION

This study presents a novel lossy image compression algorithm called RIFD. The mechanism of RIFD works by rounding and dividing the intensity of each pixel in the image. This will decrease the range of the intensities and then increase the redundancy of these intensities which helps in better compression performance. Its results are especially good for all natural images of high bit depths and for colored images.

Appendix-A: RIFD Matlab Code to Compress the Colored Images

```
A = imread('lena.png'); % or any other image
 for f = 1:255
  for h = 1:255
    for c = 1:3
      if mod(A(x, y, z),10) == 9
           r(x, y, z) = A(x, y, z)+1;
       else if mod(A(x, y, z),10) == 8
           r(x, y, z) = A(x, y, z)+2;
       else if mod(A(x, y, z),10) == 7
           r(x, y, z)=A(x, y, z)+3;
else if mod(A(x, y, z), 10) == 6
           r(x, y, z) = A(x, y, z)+4;
else if mod(A(x, y, z),10) == 5
           r(x, y, z) = A(x, y, z)+5;
else if mod(A(x, y, z),10) == 4
           r(x, y ,z) = A(x, y, z)-4;
else if mod(A(x, y, z),10) == 3
           r(x, y, z) = A(x, y, z)-3;
else if mod(A(x, y, z),10) == 2
           r(x, y, z) = A(x, y, z)-2;
else if mod(A(x, y, z),10) == 1
           r(x, y, z) = A(x, y, z)-1;
else if mod(A(x ,y, z),10) == 0
           r(x, y, z) = A(x, y, z);
       end; end; end; end; end; end; end; end; end; end;
    end; % end for x
 end; % end for y
end; % end for z
r = r/10;
B = rgb2gray(A);
% Apply Huffman coding of the resulted image – code found at Matworks website
```

Appendix-B: Sample of test set images

Image name	Original image	Image after RIFD decompression
Camerman.png		
Lena.png Grayscale 8 bits		
Lena.png Colored RGB		
Bird.jpg Colored RGB		
Lisa.jpg Colored RGB		

REFERENCES

Ames, G., 2002. Image Compression. Retrieved form: www.cis.upenn.edu/~eas205.

Frendendall, G.L. and W.L. Behrend, 1960. Picture quality-procedures for evaluating subjective effects of interference. P. IRE, 48: 1030-1034.

Gautam, B., 2010. Image compression using discrete cosine transform & discrete wavelet transform. B.Sc. Thesis, National Institute of Technology.

Huffman, D.A., 1952. A method for the construction of minimum-redundancy codes. P. IRE, 40(9): 1098-1101.

Kodituwakku, S.R. and U.S. Amarasinghe, 2007. Comparison of lossless data compression algorithms for text data. Indian J. Comput. Sci. Eng., 1(4): 416-425.

Mathur, M.K., S. Loonker and D. Saxena, 2012. Lossless Huffman coding technique for image compression and reconstruction using binary trees. Int. J. Comp. Tech. Appl., 3(1): 76-79.

Nelson, M., 1992. The Data Compression Book. M&T Books, New York.

Pane, J.F. and L. Joe, 2005. Making better use of bandwidth data compression and network management technologies. Prepared for the United States Army, The RAND Corporation.

Starosolski, R., 2007. Simple fast and adaptive lossless image compression algorithm. Software Pract. Exper., 37(1): 65-91.

Verma, P., P. Verma, A. Sahu, S. Sahu and N. Sahu, 2012. Comparison between different compression and decompression techniques on MRI scan images. Int. J. Adv. Res. Comput. Eng. Technol. (IJARCET), 1(7): 109-113.

Wei, W., 2008. An Introduction to Image Compression. National Taiwan University, Taipei, Taiwan, ROC, 2008.

White Paper, 2008. An Explanation of Video Compression Techniques. Axis Communications.