Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 07, July-2021

Seed Based Approach for Near Lossless to Lossless Image Compression

Tanuja R. Patil [1]
School of Electronics & Communication Engg
K. L. E. Technological University
Hubballi, India

Vishwanath P. Baligar [2]
School of Computer science & Engg.
K. L. E. Technological University
Hubballi, India

***Abstract*— Image compression is inevitable nowadays as there is a huge transaction of data in the form of images .In this paper, a novel approach to achieve near lossless compression and further lossless compression, is discussed. Here a seed based technique over gray scale images to achieve better quality, has been discussed. In this technique, 4 bytes of data is converted into three bytes with a transformation technique. Later a seed data is added to achieve near lossless results. Further by adding additional seed data, lossless compression can be achieved. A metric called correctness ratio is used to measure the quality of the reconstructed images. The quality of reconstructed images is near lossless i.e. very high when compared with JPEG approach at same PSNR values. Results of lossless compression are just comparable with JPEG-LS with slight compromise in compression ratio.**

Keywords—Image compression; near lossless; lossless ; comparison with JPEG ; Seed based approach.

I. INTRODUCTION

Image compression is a type of data compression applied to digital images, to reduce their cost for storage or transmission, as most of the data transmitted nowadays is in the form of images. These images can be either color or gray scale.

For image processing tasks, gray scale images can be preferably used, as much of today's display and image capture hardware can only support 8-bit images and it is more complicated and harder to process color images.

The gray scale pixel intensity value is stored as an 8-bit integer with 256 possible different shades of gray from black to white. A lot of redundancy exists in the use of 8 bit representation. This redundancy exists in 3 major forms. i.e coding redundancy, inter-pixel redundancy and pshycho-visual redundancy.

Compression can be achieved, if these redundancy can be reduced by suitable techniques. These image compression techniques will result into two types of compression i.e. lossless and lossy image compression. Lossless image compression is usually required for applications like medical field,where each & every detail is important. This is because lossy compression methods produce compression artifacts to images and sharp-edged lines become blurr when using strong compression. But lossy compression is a good choice for natural images where slight loss is acceptable to achieve smaller file size. Lossy compression is usually based on techniques that removes certain information which the human eye typically doesn't notice. Mostly used lossy compression methods are transform coding such as discrete cosine transform (DCT, used in JPEG) or wavelet transform (used in JPEG 2000). Similarly many researchers are trying with different transformation techniques.

The most widely used methods of lossless compression in images are run-length encoding (RLE), entropy coding and dictionary coders. Recently researchers are working with prediction based coding techniques to improve the compression ratio. In this paper, we discuss about a novel approach wherein 4 bytes of data is converted into three bytes by using a transformation technique by which lossy compression is achieved. A seed data is added to improve the quality, so that near lossless results have been achieved. further by adding additional seed data lossless compression can be achieved and these results are in-par with JPEG-LS methods

II. LITERATURE SURVEY

Many research works are going on to improve the compression ratio, improve the quality of reconstructed images, decrease the computation cost and reduce time complexity. We surveyed many papers and listed some highlights.

Subramanya, A, gives an overview of the major image compression techniques. The Joint Photographic Experts Group (JPEG) is a standard developed for compressing continuous-tone still images. And it has been widely accepted for still image compression throughout the industry. JPEG can be used on both gray-scale and color images. [1]

Nedhal Mohammad Al-Shereefi, discuss about the wavelet based lossy image compression. to achieve high compression ratio in images using 2D daubechies mWavelet Transform by applying global threshold for the wavelet coefficients.[2]

G. Ulacha1 and R. Stasiski, discuss about an efficient and simple context-based data modeling technique for lossless image compression. Similarly as preprocessing stage of JPEG-LS, it uses only 3 contexts, which makes it time-efficient, and does not force the message headers to be long. Enhanced, but more computationally complex versions of the method are also analyzed. Extensive experiments show that indeed, the new technique is clearly better from data compression point of view than the preprocessing stage of

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 07, July-2021

JPEG-LS, while its computational complexity is approximately the same.[3]

Mamatha A.S, Vipula Singh discuss about a simple arithmetic calculation which uses, finding the difference between pixels , for Near Lossless Image Compression . They have used Raster, Orthogonal, Diagonal and Snake scanning methods.CR and PSNR for base value 16 is 3.628 and 35.1912 respectively.[4]

Vishwanath P. Baligar, L.M. Patnaik, G.R. Nagabhushana discuss about the design and implementation of an image coding algorithm which uses a thresholding method. Threshold is the Peak Absolute Error (PAE) allowed in the reconstructed image. It has been shown that lossless edges with near-lossless filled area give a high fidelity images. Results are compared with Set Partitioning In Hierarchical Tree (SPIHT) and proven to be better . [5]

As per Sinisa ILIC, Mile PETROVIC, Branimir JAKSIC, Petar SPALEVIC, at lower PSNR levels, there will be noise effects from the compression methodology used in JPEG. Here Contour like structures appear, which are uncomfortable for better visibility. [6]

We have discussed about some lossy compression algorithms using surrounding pixels method, pixel count method and byte shrinking method, where it is shown that at low PSNR levels, number of exact pixels in reconstructed image increases, thus reducing the contour effects that may arise in JPEG at same PSNR values.[7,12,13]

From the above survey, we got the motivation to improve the quality of images which give good correctness ratio compared to JPEG. Hence we came up with a novel seed based approach to improve the quality of the images which is described in the next section.

III. METHODOLOGY

In this method, the gray scale image $f_{[x][y]}$,is divided into blocks of four pixels .These four pixels are processed and converted into three bytes of data to produce a transformed image. For a block of 4 pixels, three byte data is generated. Later seed data is added to get near lossless and lossless results, which is explained with compression and decompression algorithms below.

A Compression algorithm with examples

1) Three Byte generation

Gray scale image f[x][y],is processed in blocks of four pixels in raster scan manner as shown in *"Fig.1"*

a	b
c	d

Fig.1 block of 4 pixels

Where, pixel

163	164
163	161

Fig.2 Sample block

$f_{[x][y]}$ is the 8 bit intensity value at row 'x and column 'y'.

a= $f_{[x][y]}$, b=$f_{[x][y+1]}$, c=$f_{[x+1][y]}$, d=$f_{[x+1][y+1]}$

For e.g. consider 4 pixel values with, a=163, b=164, c=163, d=161 shown in "*Fig.2"*

Convert all the 4 pixel values to binary form as shown below.

a=10100011 (163), b=10100100 (164),
c=10100011 (163), d=10100001 (161)

a) Byte1 generation:

'Byte1' is generated by combining 7th & 6th bits of each of the 4 pixels as shown.

Byte1= $a_{[7]}$ +$a_{[6]}$ + $b_{[7]}$ +$b_{[6]}$+ $c_{[7]}$ +$c_{[6]}$ + $d_{[7]}$ + $d_{[6]}$ =10101010

Thus generated 'Byte1' is saved in a file say'Byte1'.

b) Byte2 generation

A, 3 bit data is generated by considering 5th,4th &3rd bits of each pixel. A bit position table is created and '1' is marked in that bit position indicated by 3 bit data. In the above example, bits at bit positions 5,4,3, can be written as follows.

$a_{[5,4,3]}$ =100, $b_{[5,4,3]}$ =100, $c_{[5,4,3]}$ =100, $d_{[5,4,3]}$ =100

In all these 4 pixels, bits at 5,4,3 bit positions are 100 (3 bit data) and bit position in decimal is 4, hence '1' is entered in 4th position as shown in the Table 1.

TABLE 1. BIT POSITION TABLE FOR BYTE2

Pixels	Bit positions in decimal indicated by 5th,4th,3rd bits							
	7	*6*	*5*	*4*	*3*	*2*	*1*	*0*
a	0	0	0	1	0	0	0	0
b	0	0	0	1	0	0	0	0
c	0	0	0	1	0	0	0	0
d	0	0	0	1	0	0	0	0

Byte2 is generated by logical 'OR' of a,b,c,d $_{[5,4,3]}$ bits.

Byte2= $a_{[5,4,3]}$ OR $b_{[5,4,3]}$ OR $c_{[5,4,3]}$ OR $d_{[5,4,3]}$

Byte2=00010000 || 00010000 || 00010000 || 00010000

=00010000

Thus generated 'Byte2' is saved in a file say'Byte2'.

c) Byte3 generation

'Byte3' is generated as described. The bit positions of 2nd ,1st and 0th bits of each pixel are checked and '1' is marked in the bit position indicated by 3 bit data. In the above example, bits at bit positions 2,1,0, can be written as follows.

a $_{[2,1,0]}$ =011 (3), $b_{[2,1,0]}$ = 100 (4), c $_{[2,1,0]}$ = 011(3) $d_{[2,1,0]}$ = 001 (1) .At these positions '1's are entered to generate byte3 as shown in Table 2.

TABLE 2. BIT POSITION TABLE FOR BYTE3

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 07, July-2021

Pixels	Bit positions in decimal indicated by 2^{nd},1^{st},0^{th} bits							
	7	*6*	*5*	*4*	*3*	*2*	*1*	*0*
a	0	0	0	0	1	0	0	0
b	0	0	0	1	0	0	0	0
c	0	0	0	0	1	0	0	0
d	0	0	0	0	0	0	1	0

Byte3 is generated by logical OR of a,b,c,d [2,1,0] bit representation.

Byte3= $a_{[2,1,0]}$OR $b_{[2,1,0]}$OR $c_{[2,1,0]}$OR $d_{[2,1,0]}$

Byte3=00001000 || 00010000 || 00001000 || 0000000 =00011010,

Thus generated 'Byte3' is saved in a file say'Byte3'

2) Seed Generation

Now, seed generation is done by counting the number of '1's in byte2 & byte3.The count of number of '1's in byte2 and byte3, will be either 1,2,3, or 4. If the number of '1's is one, reconstruction will be perfect and seed data is not required, but if the number of '1's is 2,3, or 4,we need to add extra bits called as seeds so that perfect or lossless reconstruction is possible. If the seeds for count 2 &3 are added, it will result in near lossless image i.e. very high quality image with good compression ratio and if seeds for count=4 are added, it will result in lossless image compression.

a) Seed generation for count of number of '1's in byte2 or byte3 =2

Consider another block of 4 pixels, where a count of '2'is expected in byte2 or byte3.for e.g. a=163, b=157,c=163, d=157. Their binary representation can be given as a=10 100 011,b=10 011 101,c=10 100 011,d=10 011 101.Here, bits at bit positions 5,4,3, can be written as follows'

a[5,4,3] =100, b[5,4,3] =011, c[5,4,3] =100, d [5,4,3] =011

Byte2 is generated by logical 'OR' of a,b,c,d $_{[5,4,3]}$ bits as described in Table 2.

Byte2= $a_{[5,4,3]}$ OR $b_{[5,4,3]}$ OR $c_{[5,4,3]}$ OR $d_{[5,4,3]}$

Byte2=00010000 || 00001000 || 00010000 || 00001000

=00 011 000

Now ,to generate the seeds, the bit positions are encoded as follows i.e lower bit position(which is 3) is given the code'0', and higher bit position (which is 4) is given the code as '1'.

For pixel a, bit position is 4 , code is'1'.For pixel b, bit position is 3, code is '0' .For c, bit position is 4,code is '1' and for d, bit position is 3, code is '0'. Hence a 4 bit seed named 'seed1' is generated by combining the code for each of them as,

Seed1=1 0 1 0

This is how a 4 bit seed is generated for one block and stored in a separate file say 'seed1' file.

b) Seed generation for a count of number of '1's in byte2 / byte3 =3

Consider another example of 4 pixels, where a count =3 is expected for e.g. a=163, b=157,c=143, d=157

their binary representation can be given as

a=10 100 011,b=10 011 101,c=10 001 111,d=10 011 101

In the above example, bits at bit positions 5,4,3, can be written as follows'

$a_{[5,4,3]}$=100, $b_{[5,4,3]}$ =011, $c_{[5,4,3]}$ =001, $d_{[5,4,3]}$ =011,and'1' is marked in bit position table

Byte2 is generated by logical OR of a, b, c, d [5,4,3] bit representation.

Byte2=$a_{[5,4,3]}$ OR $b_{[5,4,3]}$ OR $c_{[5,4,3]}$ OR $d_{[5,4,3]}$
Byte2= 00011010 (Thus here, number of '1's are three)
Now , the bit positions are encoded as follows i.e lower bit position(here,1) is given the code'00', middle bit position(here,3) is given the code'01'higher bit position(here.4) is given the code'10'
Now, the bit positions for a,b,c,d are checked and assigned the code as shown.
For pixels a,b,c,d, bit positions are 4,3,1,3 resply. Hence a 8 bit seed is generated by combining the code for each of them as,
seed2=10 01 00 01
This is how the seed is generated for number of '1's =3, and stored in a separate file say 'seed2' file.

c) Seed generation for number of '1's =4

Consider another example for e.g. a=163, b=157,c=178, d=143.Their binary representation can be given as

a=10 100 011, b=10 011 101,c=10 110 010,d=10 001 111

In above example, bit positions for the bits 5,4,3,can be written as follows'

$a_{[5,4,3]}$ =100, $b_{[5,4,3}$ =]011, $c_{[5,4,3]}$ =110, $d_{[5,4,3]}$ =001

Byte2 is generated by logical OR of a,b,c,d [5,4,3] bit representation.

Byte2=$a_{[5,4,3]}$ OR $b_{[5,4,3]}$ OR $c_{[5,4,3]}$ OR $d_{[5,4,3]}$
byte2= 01 011 010
The bit positions are arranged in ascending order and later the codes are assigned. Bit position1(i.e.1)='00', bit position2(i.e.3) = '01' bit position3(i.e.4)='10', bit position4(i.e.6) ='11'.
Now, the bit positions for a, b, c, d are checked and assigned the code as shown.
Bit positions of pixels a,b,c,d are 4,3,6,1 resply. Hence 8 bit seed is generated by combining the code for each of them as,
seed3=10 01 11 01

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 07, July-2021

This is how the seed is generated for number of '1's =4, and stored in a separate file say 'seed3' file.

3) Byte and seed generation for bits at 2,1,0 positions

Similarly for the bits[2,1,0] , above steps as mentioned in III.A.1 & III.A.2 are repeated to generate the seeds & saved in 'seed4','seed5','seed6' files.

All the above transformed files are further Huffman compressed to achieve better compression.

B. Decompression algorithm

If the image is reconstructed with the transformed three bytes without the seeds, it will result into lossy image with very low PSNR. But near lossless image with high PSNR can be achieved by reading seeds file generated for count=2 and 3.It will result into high quality image.

1) Decompression algorithm to achieve Near Losssless compression

Read and Huffman decode the following files, byte1','byte2',byte3','seed1','seed2', 'seed4','seed5' to reconstruct all 8 bits of 4 pixels.

a) *Reconstruction of 7^{th} & 6^{th} bits*

Bits at bit position 7 & 6 for all 4 pixels is obtained from byte1 file.

b) Reconstruction of bits 5,4,3

Bits at bit position 5,4,3 have to be reconstructed from the byte, read from 'byte2' ,seed1 & seed2 files.

Read the byte from byte2 file, Get the count of number of '1's. Arrange the bit positions in ascending order. i.e. lower bit position (L) & higher bit position(H).

If the count is '2', read the 'seed1' file to get the seed value. From the seed value, get the code to assign the bit positions. for e.g.if seed value is (1 01 0),then higher bit (H)is assigned to pixel $a_{[5,4,3]}$ & $c_{[5,4,3]}$, (L) is assigned to $b_{[5,4,3]}$ & $d_{[5,4,3]}$. Thus we can reconstruct bits 5,4,3 of all 4 pixels.

If the count is '3', read the 'seed2' file to get the seed value. From the seed value, get the code to assign the bit positions. for e.g. if seed value is (01 10 01 00),then middle bit (M)is assigned to pixel a[5,4,3] ,(H) is assigned to b[5,4,3],(M) is assigned to c[5,4,3] and (L) is assigned to d[5,4,3].Thus we can reconstruct bits 5,4,3 of all pixels.

c) Reconstruction of bits 2,1,0.

Similarly bits [2,1,0] can be reconstructed by the same procedure as described above in III.B.1 section, by reading 'byte3','seed4' and 'seed5' files.

For a count of number of '1's =4, the bit positions are assigned randomly to each of the four pixels. By this, we may get slight loss in the reconstruction , but, energy of four pixels is retained and it results in near lossless compression which is described in section IV.

But by adding seeds for count=4 we get lossless compression which is explained in section IV.

2) Decompression algorithm to achieve lossless image compression

- Repeat Steps explained in section III.B.1
- If the count of number of '1's in byte2 or 'byte3'=4, then read the file'seed3' to reconstruct 5^{th},4^{th},3^{rd} bits of all 4 pixels. and by reading 'seed6' file ,reconstruct 2^{nd}.1^{st}.0^{th} bits of all 4 pixels. By this reconstruction procedure all the bits of 4 pixels can be reconstructed and lossless compression can be achieved. And results are discussed in section IV.

IV. DISCUSSION OF RESULTS

A Results for near lossless compression

Above algorithm is implemented on standard test images and reconstructed images are shown in figures Fig3-Fig.8.Left side are the original images and right side are the reconstructed images. We can see the quality of reconstructed images, which is very high .They almost resemble the original image . The quality of reconstructed image is measured by another metric called as correctness ratio to measure how many pixels are same as original in the reconstructed image and are tabulated in Table 3 & 4. The Compression ratio achieved by our approach is in the range of 1.3-1.5. Results show that , using this approach reconstructed image is near lossless with high PSNR. The correctness ratio is also higher i.e .the number of correct pixels as that of original using our approach are far higher than JPEG approach measured at same PSNR values.

Table 3 Comparison of correct pixels with JPEG

Images	PSNR	Correct pixels obtained by JPEG approach	Correct pixels by our approach
Lena	41.14	50,484	179631
Barbara	38.58	40486	166163
Baboon	36.66	30182	148482
Boat	39.68	49586	171708
Aya-matsura	39.67	52478	180864
pepper	38.7	39,867	1,75,324

Table 4 Comparison of correctness ratio with JPEG

Images	PSNR	Correctness ratio by JPEG approach	Correctness ratio by our approach
Lena	41.14	0.19	0.68
Barbara	38.58	0.15	0.63
Baboon	36.66	0.11	0.56
Boat	39.68	0.18	0.65
Aya-matsura	39.67	0.2	0.68
pepper	38.7	0.15	0.67

B. Results on Images

Fig 3 Lena

Fig.4 Barbara

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 07, July-2021

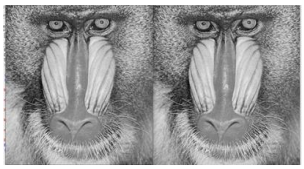

Fig. 5 Baboon

Fig, 6 Boat

Fig. 7 Aya Matsura

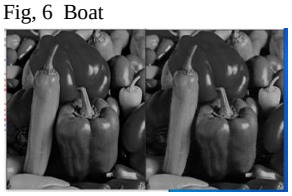

Fig. 8 Pepper

C. Results for lossless compression

We have implemented lossless algorithm for the above listed images and obtained fully perfect images without any loss. Table 5 shows the bits per pixel (bpp) required for our algorithm as compared to JPEG-LS and these results show that bpp is slightly higher , but this approach gives a simple and innovative approach for lossless compression.

Table 5 Compression ratio and bpp for lossless compression

Images	C.R.	Bpp with Jpeg-LS	Bpp with our algorithm
Lena	1.35	4.24	5.54
Barbara	1.28	5.0	5.9
Baboon	1.27	5.2	6.0
Boat	1.34	4.25	5.6
Aya-matsuura	1.36	4.1	5.41
pepper	1.30	4.71	5.57

V. CONCLUSION AND FUTURE SCOPE

Using seed based approach, reconstructed image is near lossless with high PSNR. The correctness ratio is also higher i.e .the number of correct pixels as that of original using our approach are far higher than JPEG approach measured at same PSNR values. Lossless image also can be obtained with a slight compromise with compression ratio. This approach is less computation intensive, with time complexity $O(n^2)$. It is simpler approach for compression as well as decompression. Further compression ratio can be improved by optimizing the encoding method of seeds.

ACKNOWLEDGMENT

I wholeheartedly thank our guide Dr. Vishwanath P. Baligar for his constant guidance and motivation throughout the work. I Thank our Vice Chancellor Dr.Ashok Shettar and Principal, Dr.P.G.Tewari for providing all the support and facilities required to carry out the work. I thank our H.O.D. Dr. Nalini C.Iyer for her constant support and encouragement. I thank all those who has directly or indirectly supported to carry out the research work.

REFERENCES

[1] Subramanya, A. (2001). Image compression technique. IEEE Potentials, 20(1), 19–23. doi:10.1109/45.913206

[2] Nedhal Mohammad Al-ShereefiImage Compression Using Wavelet Transform Journal of Babylon University/Pure and Applied Sciences/ No.(4)/ Vol.(21): 2013

[3] G. Ulacha1 and R. Stasiski2 new simple context-based technique for lossless image compression.

[4] Mamatha A.S, Vipula Singh, “Near Lossless Image System”Asia Pacific Conference on Postgraduate Researchin Microelectronics & Electronics (PRIMEASIA),Dec 2012

[5] Vishwanath P. Baligar, L.M. Patnaik, G.R. Nagabhushana ,“Low complexity, and high fidelity image compression using fixed threshold method”www.Elsevier.com ,Information Sciences 176 (2006) 664–675

[6] Sinisa ILIC, Mile PETROVIC, Branimir JAKSIC, Petar SPALEVIC, Ljubomir LAZIC, Mirko MILOSEVIC, “Experimental analysis of picture quality after compression by different methods”, PRZEGLĄD ELEKTROTECHNICZNY, ISSN 0033-2097, R. 89 NR 11/2013 9.

[7] T. R. Patil, V. P. Baligar and R. P. Huilgol, "Low PSNR High Fidelity Image Compression Using Surrounding Pixels," 2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET), Kottayam, India, 2018, pp. 1-6, doi: 10.1109/ICCSDET.2018.8821082.

[8] Fabian Mentzer, Eirikur Agustsson↤ Michael Tschannen Radu Timofte Luc Van Goo Conditional Probability Models for Deep Image Compression *https://openaccess.thecvf.com/content_cvpr_2018/papers/Mentzer*.

[9] V.P.Baligar L.M.Patnaik, G.R.Nagabhushan,” High compression and low order linear predictor for lossless coding of grayscale images” www.elsevier.com,Image& vision computing21(2003) 543-550 5.

[10] M. J. Weinberger, G. Seroussi, and G. Sapiro, “LOCO-I: A low complexity, context-based, lossless image compression algorithm,” in Proc. 1996 Data Compression Conference, Snowbird, UT, Mar. 1996, pp. 140–149 6.

[11] A. M. Raid, W. M. Khedr, M. A. El-dosuky and Wesam Ahmed, “Jpeg image compression using discrete cosine transform - A survey”, International Journal of Computer Science & Engineering Survey (IJCSES) Vol.5, No.2, April 2014 7.

[12] Tanuja R.Patil ,Vishwanath P.Baligar, “A pixel count approach for lossy image compression” In book: ICT Analysis and Applications (pp.369-377) DOI:10.1007/978-981-15-8354-4_37.

[13] Tanuja R.Patil ,Vishwanath P.Baligar, “Byte shrinking approach for lossy image compression” ICTCS 2020,First online 6-7-21,DOI: 10.1007/978-981-16-0882-7_13