A Novel Image Compression Algorithm by Using LOG-EXP Transform*

Sheng-Chieh Huang, Liang-Gee Chen and Hao-Chieh Chang

Department of Electrical Engineering
National Taiwan University,
Taipei, Taiwan, R.O.C.

ABSTRACT

Based on the logarithmic and exponential transform, an efficient Log-Exp still image compression system is proposed. The Log-Exp compression is designed for the high quality still image, especially for the PSNR above 36. At the similar image quality (Log-Exp PSNR = *41.52* and JPEG PSNR = *41.23*), the Log-Exp compression can get the higher compression ratio than JPEG by 1.84 times for the benchmark image *lena*. Besides, the Log-Exp compression is computing in pixel-by-pixel without the block artifacts. In comparison with the JPEG compression result (bpp = *0.99*, PSNR = *26.9*), the Log-Exp compression uses less bpp (bpp = *0.87*) to get higher image quality (PSNR=*36.38*) for the benchmark image *baboon*.

1. INTRODUCTION

In recent years, the networking multimedia systems are popular in life. For example, the WWW, video conference, video phone, DVD, Set-Top Box, HDTV, Digital TV are the trend of the consumer electronic applications. In these multimedia products, the high image quality is the key requirement. So, the JPEG still image compression standard is widely used in many applications. According to the previous researches for the JPEG image transmission [1] and VLSI implementation [2], there are two key points have to be improved in JPEG. The first point is the compression ratio must be increased for the more and more large image data. Secondly, due to the JPEG is based on the DCT transform, the block artifacts make large distortion in the image quality, especially in the zoom view for the region of interest (ROI) access. To improve the above drawbacks in JPEG, a novel image compression algorithm by using LOG-EXP transform is proposed in this paper.

2. SYSTEM DESIGN

Based on the logarithmic number system (LNS) [3] properties, the compression and decompression flow of the Log-Exp image compression system are shown on *Fig. 1*. In first, the LNS fixed-point data format and pixel value difference are used to reduce pixel value bits from 8 bits to 6 bits. Secondly, the neighboring pixel value of the same object is similar or same, there are many repeat items can be removed with repeat reduction. In finial, the Huffman coding is used to remove the whole image entropy redundancy. Each function unit is described in the following sessions.

2.1 Logarithmic Transform

For the gray level image, the pixel value is usually represented in integer format. For reducing the distribution of the pixel value to locate in the small and continuous range, the pixels are processed with the logarithmic transform. As shown in *Fig. 2.*, when the difference is *12* between the *16* and *4* in the pixel values of the integer format, the difference is *2* between the *4* and *2* in the LNS format (based on log_2). The large neighboring difference between two pixels can be reduced into small range. Besides, due to the bit numbers of the pixel value are finite (i.e. finite word length effect) and the continuous distribution of the pixel value in LNS [4], the original 8 bits gray level image can be represented in 7 bits LNS format.

2.2 Neighboring Difference and Snake Scan

The neighbor pixels usually represent the same object so that the values of the pixels are similar or same. To compute the differences between the neighbor pixels, the neighboring differences will be very small and there are many same differences. According to the experiments for the benchmark image *baboon, lena, pepper*, the integer part of the neighboring differences are distributed from $10_{(2)}$ to $0_{(2)}$ (i.e. 2 to 0 in decimal). When the neighboring difference is stored in *7* bits fixed point data format (*3* bits integer part and *4* bits fraction part) in LNS, the most significant bit (MSB) is always equal to zero in the integer part. So, the *7* bits neighboring difference can be reduced to *6* bits. If the neighboring difference is negative, the additional one is appended in the finial bit of the neighboring difference.

Besides, when the pixel is processed in sequential scan and changes from this line to the next line, there will be one large neighboring difference. To avoid such kind of large neighboring difference, the snake scan is used and shown in *Fig. 3*.

2.3 Repeat Reduction and Huffman Coding

There are many repeat items of the neighboring difference can be removed with the repeat reduction unit. The repeat reduction is shown in *Fig. 4*. The left part is the neighboring difference result. In the right part, the data part is the data after removing the same value and the times part is recording how many repeat times in these data. After reducing the repeat values, the data part and times part are processed with the Huffman encoder to remove the

* This work was supported by National Science Council under Grant NSC88-2215-E-002-034.

0-7803-5471-0/99/$10.00©1999 IEEE

large amount of similar neighboring differences redundancy.

On the other side, the decompression phase reconstructs image from the compressed data by the Huffman decoding, restoring the repeat value, adding the neighboring difference with the previous pixel and then making the exponential function computation.

3. IMPLEMENTATION and EVALUATION

There are four stages: logarithmic transform, neighboring difference, repeat reduction, and Huffman encoding in compression phase. The logarithmic transform and the neighboring difference remove the pixel representation redundancy. And then, the repeat reduction deletes the repeat of the neighboring difference to remove the same object redundancy. In finial, the Huffman encoder removes the whole image entropy redundancy. The total compression ratio is calculated with the product of the value repeat reduction compression ratio and the Huffman coding compression ratio. In the following table, the compression ratios in each stage are presented.

	Lena	**Pepper**	**Baboon**
Value Repeat Compression Ratio	*1.55*	*1.47*	*1.41*
Huffman Coding Compression Ratio	*5.42*	*5.38*	*5.22*
Total Compression Ratio	*8.40*	*7.91*	*7.36*
PSNR	*41.52*	*41.51*	*41.31*

Table. 1. Compression Ratio for Gray Level Benchmark.

The high PSNR is necessary for high quality image. The following table shows the comparison when the PSNR is above *36*.

. Benchmark Image baboon.

Log-Exp PSNR	*41.31*	*36.38*
Log-Exp Compression Ratio	*7.36*	*9.19*
JPEG PSNR	*37.23*	*34.53*
JPEG Compression Ratio	*2.39*	*2.98*

. Benchmark Image pepper:

Log-Exp PSNR	*41.51*	*36.62*
Log-Exp Compression Ratio	*7.91*	*10.06*
JPEG PSNR	*39.23*	*36.45*
JPEG Compression Ratio	*4.13*	*9.14*

. Benchmark Image lena:

Log-Exp PSNR	*41.52*	*36.62*
Log-Exp Compression Ratio	*8.40*	*11.05*
JPEG PSNR	*41.23*	*36.91*
JPEG Compression Ratio	*4.57*	*11.13*

Table. 2. The Compression Ratio vs. PSNR Comparison when the PSNR above 36.

Based on the similar PSNR, the comparison of the compression ratio is shown on the following table.

	Lena	**Pepper**	**Baboon**
Log-Exp PSNR	*41.52*	*41.51*	*41.31*
Log-Exp Compression Ratio	*8.40*	*7.91*	*7.36*
JPEG PSNR	*41.23*	*39.23*	*42.42*
JPEG Compression Ratio	*4.57*	*4.13*	*1.74*

Table. 3. The Compression Ratio Comparison on the Similar PSNR.

In the benchmark image *lena*, when the PSNR is similar (Log-Exp PSNR = *41.52* and JPEG PSNR = *41.23*), the Log-Exp compression effect is superior to JPEG by *1.84* times.

In table 4, the PSNR comparison is shown when the compression ratio is similar.

Baboon	***Log-Exp***	***JPEG***
PSNR.	*36.38*	*26.7*
Compression Ratio.	*9.19*	*8.25*

Pepper	***Log-Exp***	***JPEG***
PSNR.	*41.51*	*37.30*
Compression Ratio.	*7.91*	*6.91*

Lena	***Log-Exp***	***JPEG***
PSNR.	*41.52*	*38.29*
Compression Ratio.	*8.40*	*8.25*

Table. 4. The PSNR Comparison on the Similar Compression Ratio.

The subject view comparison is shown in *Fig. 5.* and *Fig. 6.* The *Fig. 5.* shows the benchmark image *baboon*'s left eye part. Though the the bit per pixel (bpp) of the JPEG compression is more than the Log-Exp compression, the image quality of the Log-Exp is superior to the JPEG. Besides, the block artifacts of the JPEG image are very obvious in this figure. The Log-Exp compression is processing pixel by pixel, there is no block artifacts.

The *Fig. 6.* is the subject view of the benchmark image *lena*'s hat. For the texture of the *lena*'s hat, there are the very serious block artifact in whole image and the blur distortion on the left upper part in JPEG compressed image. At the similar bpp (Log-Exp = *0.95*, JPEG = *0.97*) requirement, the subject view of the Log-Exp image is more clear in the texture of the *lena*'s hat and without the block artifact distortion.

4. CONCLUSION

The LOG-EXP image compression can get high compression ratio for the complex texture image (e.g. benchmark image *baboon*) and the high quality image, especially the PSNR above 36. The pixel-by-pixel processing style is useful to avoid the block artifact for the high quality image compression [5].

REFERENCES

[1] Y. H. Han, J. J. Leou, "Detection and Correction of Transmission Errors in JPEG Images," IEEE Trans. on Circuit and System for Video Technology, Vol. 8, No. 2, Apr. 1998.

[2] M. Kovac, N. Ranganathan, "JAUGUAR: A Fully Pipelined VLSI Architecture for JPEG Image Compression Standard." Proceeding of the IEEE, Vol. 83, No. 2, Feb. 1995.

[3] S.C. Huang, L.G. Chen, T.H. Chen, "The Chip Design of A 32-b Logarithmic Number System," IEEE Sym. on Circuit and System, vol. 4, pp. 167-170, 1994.

[4] S.C. Huang, L.G. Chen, T.H. Chen, "A 32-bit Logarithmic Number System Processor," Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology, Vol. 14, No. 3, pp. 311-319, Dec. 1996.

[5] S. C. Huang, L.G. Chen, "LOG-EXP Compression System Design and Implementation," IEEE International Sym. on Consumer Electronics, 1998.

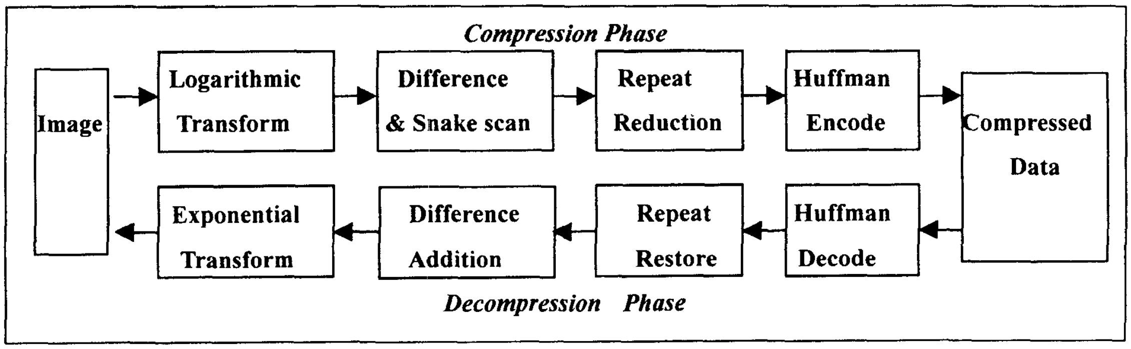


Fig. 1. Log-Exp Image Compression System.

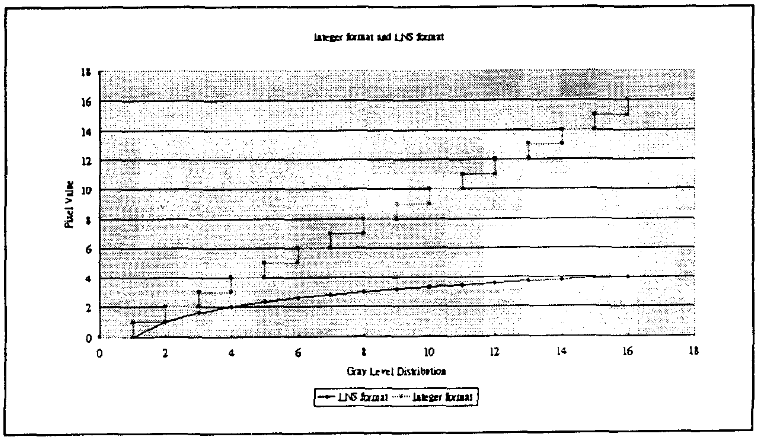


Fig. 2. The Integer and LNS Format for Pixel Value Range.

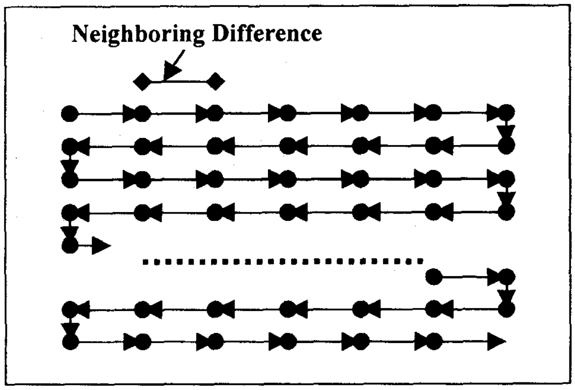
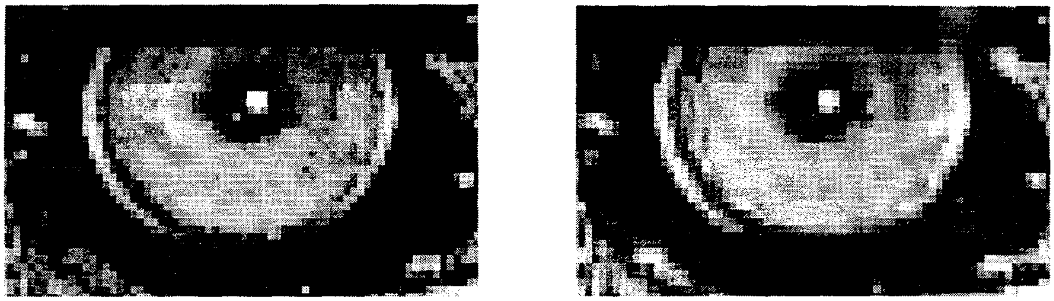


Fig. 3. Neighboring Difference and Snake Scan.

Difference	Data	Times
$00.0010_{(2)}$	$00.0010_{(2)}$	
$00.0111_{(2)}$		
$00.0111_{(2)}$		
$00.0111_{(2)}$	→ $00.0111_{(2)}$	$111_{(2)}$
$00.0001_{(2)}$	$00.0001_{(2)}$	
$00.0111_{(2)}$		
$00.0111_{(2)}$	→ $00.0111_{(2)}$	$11_{(2)}$

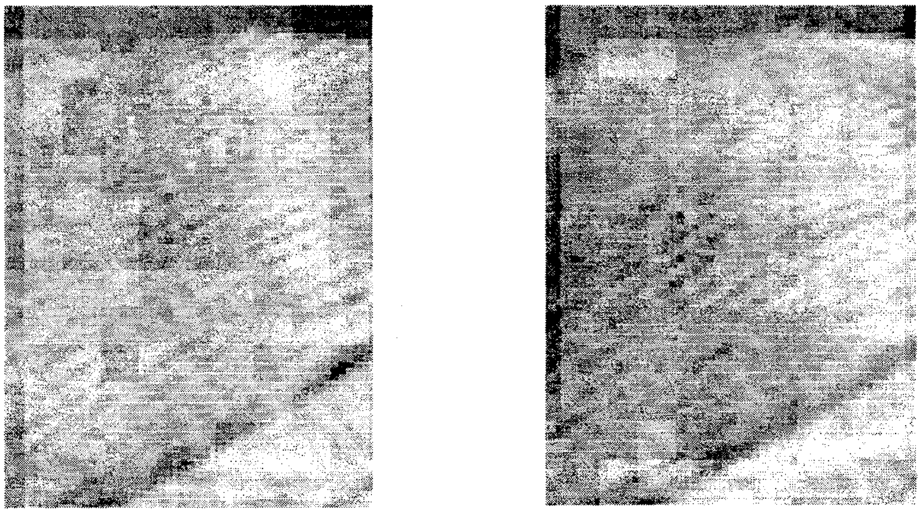
Fig. 4. Repeat Reduction.



Log-Exp: bpp = 0.87, PSNR = 36.38

JPEG: bpp = 0.99, PSNR = 26.9

Fig. 5. The Left Eye of the Benchmark Image *baboon*.



Log-Exp: bpp = 0.95, PSNR = 41.52

JPEG: bpp = 0.97, PSNR = 38.29

Fig. 6. The Hat of the Benchmark Image *lena*.