International Journal of Signal Processing, Image Processing and Pattern Recognition
Vol. 8, No. 5 (2015), pp. 279-288
http://dx.doi.org/10.14257/ijsip.2015.8.5.29

A Method of Lossy Compression for RGB565 Format True Color Image

Yan Xijun and Jiang Lei

College of Computer and Information Engineering, Hohai University, Nanjing 211100, China
yan_xijun@hhu.edu.cn, joel_jiang@126.com

Abstract

In order to solve the issue of embedded image processing, a method of mixed compression for RGB565 format true color image is proposed. According to the inherent correlation of image data, the method adopts the discrete cosine transform and rearrangement to centralize the data that have higher correlation. Finally, it achieves the image compression by improving the run-length coding combined with statistical coding. Experiments show that the method can compress the true color images with RGB565 format effectively on the platform of ARM-Linux, and achieve a good compression ratio.

Key words: *RGB565 true color image, lossless compression, lossy compression, discrete cosine transform, run-length coding*

Currently, the embedded products have higher requirements for image processing. Hence, the design has focused on effective image storage and display. For image data with enormous information, it is necessary to take compression techniques to save the storage and make transmission more convenient. Now, the compression technology can be divided into two categories, lossless and lossy compression. Common methods of lossless compression include Shannon-Fano coding, Huffman coding, LZW (Lempel-ZivWelch) coding and arithmetic coding. Lossy compression includes predictive coding, transform coding, vector quantization coding, and model coding and so on. All of these methods, discrete cosine transform is often used in the realization of transform coding for the image signal.

The traditional lossless compression RLE (Run-Length encoding) encoding method has a better compression effect on the specific data with a large number of duplicate information. However, RGB565 format true color image data has been widely used in most embedded products currently. This kind of data has little continuous pixel for one color on the some line and much less for multi-line. Thus, the conventional RLE can't guarantee to get good compression effect. So a new method combined the DCT transform with RLE compression coding is proposed in this paper to achieve compression coding for color image [1-2].

1. The basic Principle of DCT and Run-length Encoding

1.1 The DCT Transform Principle

Transform coding can map image signal in time domain to frequency domain, and then deal with related coefficients generated during the coding process. The image in time domain has the features of strong data correlation and high redundancy, while in frequency domain, correlation and redundancy among data is greatly reduced. So a larger compression ratio can be got after quantization and coding. Discrete Cosine Transform (DCT) [3-7] is the most commonly used transform coding, it is considered as sub-optimal performance which is close to K-L transform.

Define a two-dimensional DCT transform with N × N dimension matrix as follows:

$$Y(\mu,\nu)=\frac{2}{\sqrt{MN}}C(\mu)C(\nu)\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}[X(m,n)\cos\frac{(2m+1)\mu\pi}{2M}\cos\frac{(2m+1)\nu\pi}{2N}] \quad (1)$$

In which, $\mu=0,1,...,M-1;\nu=0,1,...N-1$

Define a two-dimensional IDCT transform with N × N dimension matrix as follows:

$$X(m,n)=\frac{2}{\sqrt{MN}}\sum_{\mu=0}^{M-1}\sum_{\nu=0}^{N-1}[C(\mu)C(\nu)Y(\mu,\nu)\cos\frac{(2m+1)\mu\pi}{2M}\cos\frac{(2m+1)\nu\pi}{2N}] \quad (2)$$

In which, $m=0,1,...,M-1;n=0,1,...N-1$, $C(\mu),C(\nu)=\begin{cases}1/\sqrt{2},\mu,\nu=0\\1,others\end{cases}$

After a discrete cosine transform, frequency domain decomposition map the process of the human visual system. Most of the signal energy can be concentrated in the low frequency range, while high-frequency component that the human eye is not sensitive only holds very little energy, and eventually compressed into few bits.

1.2 RLE Coding Theory

The basic idea of RLE compression [8-13] is that if the data x turns up consecutively for n times in the input data stream, x will be replaced by x_n .The consecutive time is defined as the run-length n and thus the process is called the run-length coding.

The basic run-length coding is performed to finish the source data stream statistics, making the repeated string, the string length and the position of the string form a new data stream. For example:

0 1 1 2 2 2 3 3 3 3 4 4 4 4 4 5 5 5 5 5 5 6 7 8 8 9

The output sequence can be written as 0(1) 1(2) 2(3) 3(4) 4(5) 5(6) 6(1) 7(1) 8(2) 9(1) after basic RLE, the number in parentheses is the run-length. From the above example, we can conclude that if the value of continuous repetition is relative small for a colorful image, the effect of RLE will not be ideal. For example, when the number of continuous repetition is 1, the length of data stream will increase to twice its original length after the basic run-length coding. Obviously, the compression effect is not so good and the amount of data becomes larger. Basic RLE is part of lossless compression, and defines the character as its basic unit. RLE usually requires the numerical size in the range of 0 to 255, which limits the use of this algorithm.

2. The Compression Method for RGB565 Format True Color Image

In order to compress the image data with RGB565 format, RGB component should first be treated separately, then concentrate related information through DCT transform. After this, the transformed result is quantified and compression image can be obtained eventually by RLE coding. Decoding process is corresponding to the reverse process of encoding. The related flow is shown in Figure1:

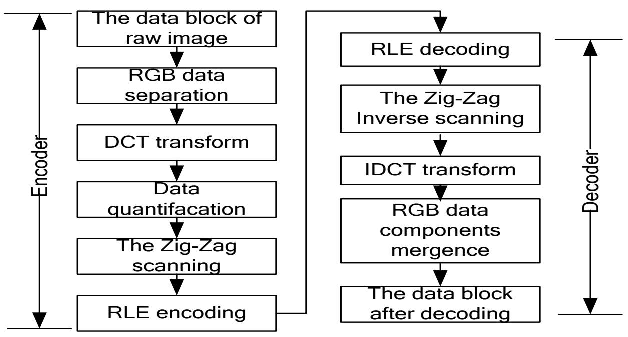

Figure 1. The Flow of Color Image Compression

2.1 DCT and IDCT Transform

This paper mainly compresses image information collected by OV9650 with 640 * 480 pixels. Each pixel occupies two bytes, and stories pixel tricolor information for RGB565 format. RGB565 format is shown in Figure 2.

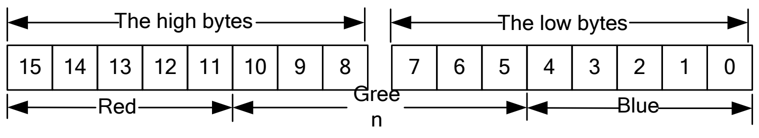

Figure 2. RGB565 Data Format

Since RLE compression is better for operation in bytes, the pixel information in two bytes should be isolated. Two separation methods are as follows:

(1)The two bytes are classified into high bytes and low bytes to perform DCT transform.

(2)The two bytes are separated into R, G, B bytes (high fill zero) to perform DCT transform.

Considered from handling approach, the two methods above are both feasible. But whether the pixels can be effectively separated is related to information relevance. The best separation method should be chose from experiments.

The image should be divided into N*N pixel blocks. When dealing with image data through DCT transform. In order to ensure orthogonality and rate of data blocks, matrix of N = 8 is generally used and 64 pixels is defined as a data block. Because of large amount information of image data, the pixels of sample between 576 and 639 will be analyzed below. The raw data of the pixels in bytes is shown in Figure 3.

$$\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 32 & 0 & 0 & 0 & 0 & 0 & 32 & 0 & 33 & 8 \\
65 & 8 & 97 & 8 & 97 & 8 & 129 & 8 & 129 & 8 & 98 & 8 & 130 & 8 & 98 & 8 \\
98 & 8 & 98 & 8 & 98 & 8 & 98 & 16 & 163 & 24 & 37 & 49 & 200 & 81 & 74 & 106 \\
107 & 114 & 171 & 114 & 171 & 114 & 171 & 114 & 170 & 114 & 170 & 106 & 105 & 98 & 41 & 90 \\
232 & 81 & 199 & 65 & 134 & 57 & 69 & 41 & 3 & 33 & 162 & 16 & 97 & 8 & 33 & 8 \\
33 & 8 & 65 & 16 & 162 & 32 & 36 & 49 & 101 & 57 & 134 & 65 & 134 & 65 & 134 & 65 \\
133 & 65 & 133 & 65 & 102 & 65 & 102 & 65 & 134 & 65 & 134 & 65 & 133 & 73 & 101 & 73
\end{pmatrix}$$

Figure 3. The Raw Data of the Pixels

Firstly, the first separation method is performed to get the low bytes and the high bytes respectively through DCT transform. Due to the large amount of data, only values of low bytes after DCT transform are shown in Figure 4.

It can be seen from Figure 4 that the data transformed by the DCT is not concentrated in the upper left portion with low-frequency energy as expected. It is scattered, and not suitable for compression coding.

$$\begin{pmatrix}
-30.62500 & 52.418166 & 107.50643 & -112.01707 & 76.37498 & 28.542884 & 38.407676 & -52.243427 \\
140.51229 & -1.796596 & 80.042211 & 85.856968 & -150.34781 & -58.592876 & 103.567799 & -14.419893 \\
-85.00502 & -14.54113 & -177.55326 & -86.08171 & 33.295644 & -41.115667 & -2.632536 & 21.497936 \\
-48.12083 & -98.32239 & -18.91928 & 164.06689 & 81.860107 & 96.369266 & -238.63017 & 91.430724 \\
-1.624996 & -24.39975 & 33.404731 & -106.12402 & 11.874980 & 99.706565 & 68.560406 & -9.810974 \\
52.441597 & 179.71382 & -67.646280 & 46.276072 & 7.939724 & -113.16310 & -23.298750 & -64.431154 \\
-20.66824 & -83.29064 & 53.367443 & -83.381203 & -35.574666 & 0.387452 & 79.303239 & -36.949909 \\
-90.66767 & 46.986914 & 96.286625 & 99.201563 & -23.403668 & -64.787613 & 31.914988 & 61.892784
\end{pmatrix}$$

Figure 4. Values of Low Bytes after DCT Transform

Then the second method is used to get R, G, B transform values respectively. Because of the large amount of data, only R component values after DCT transform are shown in Figure 5.

$$\begin{pmatrix}
37.500000 & -2.918433 & 1.115200 & -0.463095 & 0.499999 & 0.079308 & 0.079255 & 0.109524 \\
-23.08939 & -0.575934 & 1.176355 & -0.262170 & -0.014951 & -0.269330 & 0.038718 & 0.235576 \\
-12.64332 & 0.377895 & -1.758883 & 0.072677 & -0.230969 & 0.028819 & -0.021446 & -0.022378 \\
3.760004 & 7.852684 & -1.886756 & 0.865561 & 0.214061 & -0.119827 & 0.300738 & 0.286339 \\
14.749999 & -3.755585 & 1.712459 & -0.341330 & -0.250001 & 0.160668 & -0.438727 & -0.056996 \\
-10.20319 & -9.559312 & 1.110354 & -0.589497 & -0.095371 & 0.268323 & 0.016087 & -0.316733 \\
-5.045696 & 9.051835 & 0.728554 & 0.479459 & 0.095671 & 0.042579 & 0.008884 & 0.427957 \\
9.500628 & 1.294756 & -4.383262 & 0.938747 & -0.565561 & 0.010802 & -0.135150 & -0.057950
\end{pmatrix}$$

Figure 5. Red Component Values after DCT Transform

As can be seen from the Figure above, the data transformed by the DCT is concentrated in the upper left portion with low-frequency energy as expected. And the high frequency part tends to zero. Values of G, B components after DCT transform also satisfy the trend in favor of the latter part of RLE compression.

The result can be seen from comparison of Figures 4 and 5, since R, G, B tricolor with the same pixel are not uniform distribution, the data can be better to separated through the second method.

2.2 Data Quantification and the Zig-Zag Scanning

As can be seen from above, the DCT transform changes the energy distribution of the image signal, most of the energy about the image information concentrate in low frequency part, which is in the upper left corner of the matrix block. Most of the DCT coefficients in the high frequency portion are very close to zero. In fact, DCT transform itself can’t save memory space by compressing the data, but if we compress the data after giving up the value in high-frequency part, the memory space can be saved a lot. During the process, the data is required to be quantified. Two quantitative methods are widely used currently:

(1) Mild quantification is that the data should be rounded up and the fraction part should be removed.

(2) Given a quantitative threshold value, if the high frequency component is greater than the threshold value, the coefficient is set to 0; otherwise the integer part is reserved.

In order to obtain a higher compression ratio, method of the quantitative threshold is adopted in this paper. When the quantization threshold value is set to 5.0, the data of R component in Figure 5 is shown in Figure 6 after quantification.

$$\begin{pmatrix} 37 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & -9 & 0 & 0 & 0 & 0 & 0 & 0 \\ -5 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 6. The Data of R Component after Quantification

After the data block quantification, it still needs compression coding. The RLE compression method used in this paper requires the ability to scan the data block and arrange the coefficients in order. Two scanning methods are mainly used now:

(1) The progressive scanning as is shown in Figure 7 (a). It scans from the beginning of one line to the end of this line.

(2) The Zig-Zag scanning which arranges the 64 elements as Figure 7 (b) shows.

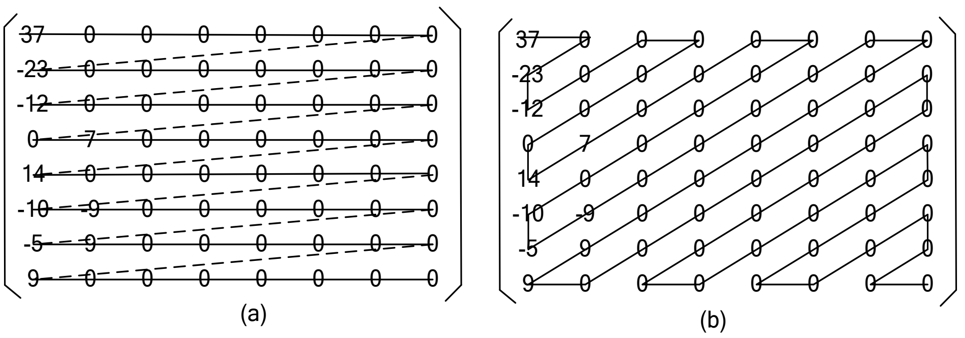

Figure 7. Two Scanning Methods of R Component

If the progressive scanning is used, R component sequence after RLE compression is 37(1) 0(7)-23(1) 0(7)-12(1) 0(8) 7(1) 0(6) 14(1) 0(7) 10(1)-9(1) 0(6) 5(1) 9(1) 0(6) 9(1) 0(7), which requires 36 bytes to encode. If the Zig-Zag scanning is adopted, R component sequence after RLE compression is 37(1) 0(1) -23(1)-12(1) 0(6) 14(1) 7(1) 0(8)-10(1)-5(1)-9(1) 0(11) 9(2) 0(28), which requires 28 bytes to encode. It can be seen that the Zig-Zag scanning make the low-frequency component occurs before the high-frequency component. It increases the number of continuous "0" in the sequence.

The Zig-Zag scanning can be achieved by the index array n in the C programming language, as is shown in Table 1. Values in Z_MAP[i] can be seen as the quantification data block after index access, and the Zig-Zag data sequence can be got after arranging the data in order. The inverse process of the Zig-Zag scanning follows the same way through Z_UNMAP [i].

Table 1. The Zig-Zag Scanning Mapping and Inverse Mapping

The array of the Zig-Zag scanning mapping	The array of the Zig-Zag inverse scanning mapping
char Z_MAP [64]= { 0, 1, 8, 16, 9, 2, 3, 10, 17, 24, 32, 25, 18, 11, 4, 5, 12, 19, 26, 33, 40, 41, 34, 27, 20, 13, 6, 7, 14, 21, 28, 35, 42, 49, 56, 57, 50, 43, 36, 29, 22, 15, 23, 30, 37, 44, 51, 58, 59, 52, 45, 38, 31, 39, 46, 53, 60, 61, 54, 47, 55, 62, 63};	char Z_UNMAP [64]= { 0, 1, 5, 6, 14, 15, 27, 28, 2, 4, 7, 13, 16, 26, 29, 42, 3, 8, 12, 17, 25, 30, 41, 43, 9, 11, 18, 24, 31, 40, 44, 53, 10, 19, 23, 32, 39, 45, 52, 54 20, 22, 33, 38, 46, 51, 55, 60, 21, 34, 37, 47, 50, 56, 59, 61, 35, 36, 48, 49, 57, 58, 62, 63};

2.3 The Improved RLE Compression Method

After obtaining the ideal data sequence, the data sequence still needs to be compression coding to get the compression image information. RLE compression algorithm is widely used because it is simple and efficient. The original data currently used by RLE compression is generally smaller than one byte to facilitate statistical counting of coding. Thus, before RLE coding, R, G, B components should be handled. The data transformed by DCT is converted into two parts, as Figure6 shows. F(0,0) = 37 is the DC component of the data block, and other coefficient are AC components.

(1) component is represented in 5 bits. Thus, its value range is 0 to 31. The value of th e DC component ranges from 0 to 255, which can be expressed by unsigned char data typ e in C programming language. The AC component ranges from 128 to 127 which can be e xpressed by signed char data type in C language.

(2) G component is represented in 6 bits. Thus, its value range is 0 to 63. The value of the DC component ranges from 0 to 511. The data requires to be divided by 2 firstly if expressed by unsigned char data type in C programming language. The AC component ranges from -128 to 127 and its data should do the same operation if expressed by signed char data type in C language.

(3)B component is represented in 5 bits. Thus, its value range is 0 to 31. The value of t he DC component ranges from 0 to 255, which can be expressed by unsigned char data ty pe in C programming language. The AC component ranges from 128 to 127 which can be expressed by signed char data type in C language.

Through the analysis of large amounts of data after DCT transform, we can conclude that most energy of data block stored in DC part, which can be saved directly without compression coding. The total bytes of R, G, B DC component are 14400. And non-zero data duplication of AC component is less, thus in order to prevent the data swelling phenomenon in RLE algorithm, the data is saved directly without encoding, mainly compress the AC coefficient "0". The encoding format is shown in Figure 8.

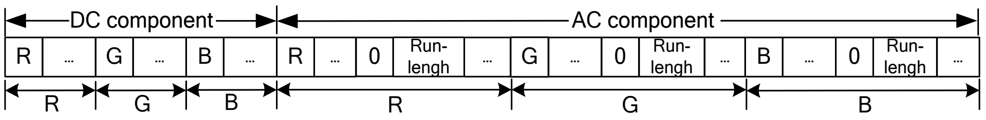

Figure 8. The Improved RLE Compression Encoding

In the improved RLE encoding, the size of run-length is likely to be more than one byte due to large number of data "0". The upper limit of run-length can be determined as maximum value represented by three bytes because the image size is 640 * 480 pixels. Two high level bits of the first bytes indicates the length of the run-length itself, and 0, 1, 2 are corresponding to the run-length byte serial number respectively. The encoding algorithm is shown in Table 2.

Table 2. The Improved RLE Encoding Algorithm

Run-length value	Bytes number		
	0	1	2
Val < 64	0x40 | Val		
64 <= Val < 16384	0x80 | (Val >> 8)	Val & 0xFF	
Val >= 16384	0xC0 | (Val >> 16)	(Val >> 8) & 0xFF	Val & 0xFF

3. The Improved Image Compression Algorithm Experiment

In this paper, the basic image information collected by OV9650 is 640 * 480 pixels, and each pixel has 2 bytes, so the file size of original image is 614400 bytes. The main idea of the algorithm is to transform the high frequency component to "0", and then discard it. Therefore, with the increase of quantization value, the more the high frequency information is lost, the lower the compression ratio we can get. Table 3 is given the DC component of the R, G, B tricolor, the corresponding AC component and the compression ratio, when the quantification value is 5.0, 7.0, 10.0, 12.0, and 15.0 respectively.

Table 3. Experiment Results

Quantification value	DC component	AC component (R)	AC component (G)	AC component (B)	compression ratio
5.0	14400	29544	69485	23786	0.223332
7.0	14400	18867	50726	14722	0.160669
10.0	14400	10903	33578	8630	0.109925
12.0	14400	8144	26188	6516	0.089922
15.0	14400	5584	19247	4389	0.070996

The corresponding images after compression coding are (b) ~ (f) in Figure9, and (a) is the original image. As can be seen from the Figure, the edge information of the image which indicates the high-frequency component becomes very blurred when the compression ratio is lower, but the overall message of the image has not been greatly affected.

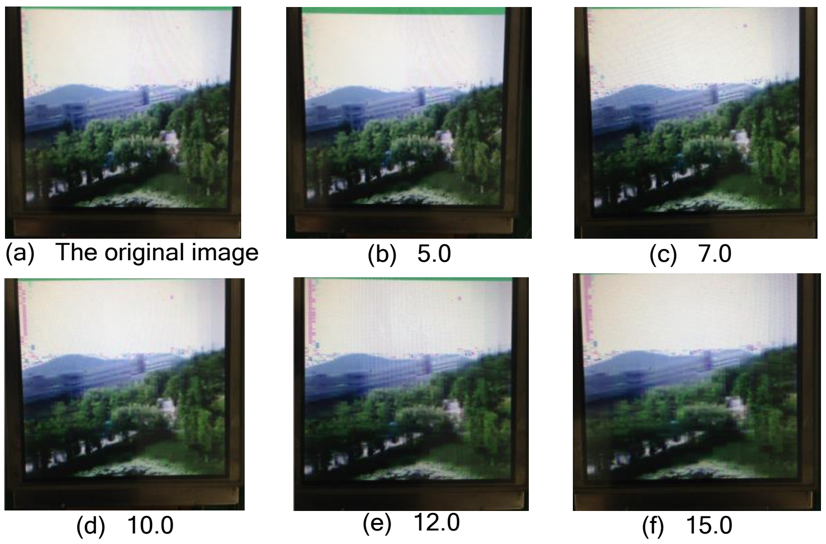

Figure 9. The Image after Decoding with Different Compression Ratios

When the quantification value is 5.0, the compression ratio is 0.27771, Figure 9 (b) have a better image fidelity. When the quantization value is 15, the compression ratio of 0.099512, Figure 9 (f) appears obvious distortion and the edge information becomes extremely blurred. The quantization value can be adjusted to achieve the purpose of good image compression according to the system requirements. Thus the method is suitable for RGB565 format color image compression. It has the characters of simple, fast, high compression efficiency, precision and so on.

4. Conclusion

According to the characteristics of RGB565 image data format in embedded systems, the compression method in this paper first perform DCT transform of the original image, separate related information, and then quantify, rearrange, and remove high-frequency information which human eye is not sensitive. Finally, the improved RLE is used to achieve the image information with less data amount. The image decoding is corresponding to the reverse of the process above. Experimental results show that the method can achieve better compression ratio within the acceptable range of lossy compression.

References

[1] J. Tang and Z. M. Hu, “Application of the DCT transform in lossy color image RLE compression”, Computer Knowledge and Technology, **(2007)**, pp. 346-347.
[2] P. Y. Yu, “The gray image compression method based on DCT and RLE”, The journal of Hunan university of science and technology, no. 2, **(2010)**, pp. 89-92.
[3] F. Yun and S. W. Run, “An image retrieval method using DCT features”, Journal of Computer Science and Technology, no. 6, **(2001)**, pp. 865-873.
[4] T. Darwish and M. Bayoumi, “Coefficient Elimination Algorithm for Low Energy Distributed Arithmetic DCT Architectures”, The Journal of VLSI Signal Processing - Systems for Signal, Image, and Video Technology, no. 3, **(2003)**, pp. 355-369.
[5] T. Edue, R. Grisel, H. Cherifm, *et al.*, “The distribution of the DCT coefficients”, Proc IEEE ICA SSP.A delaide, Australia, **(1994)**, pp. 365-368.
[6] C.-S. Park and Kim, “Blind Measurement of Blocking Artifacts in both Pixel and DCT Domains”, Journal of Mathematical Imaging and Vision, no. 3, **(2007)**, pp. 279-284.
[7] S. Zhao and J. Liu, “Image retrieval in DCT compressed domains”, Journal of Beijing University of Posts and Telecommunications, no. 5, pp. 5- 9.
[8] A. Banerjee and A. Halder, “An Efficient Dynamic Image Compression Algorithm based on Block Optimization, Byte Compression and Run-Length Encoding along Y-axis”, 2010 3rd IEEE International

Conference on Computer Science and Information Technology (ICCSIT 2010), no. 7, **(2010)**, pp. 356-360.
[9] A. Sreedevi, D. S. Jangamshetti, H. Aithal and A. kumar, "Lossless Compression of Microarray Images by Run Length Coding", International Conference on Bio-inspired Systems and Signal Processing (ICBSSP 2010), no. 10, **(2010)**, pp. 69-72.
[10] AI-Wahaib and M. Safa, "A lossless image compression algorithm using duplication free run-length coding", Proceedings - 2nd International Conference on Network Applications, Protocols and Services, (NETAPPS 2010), no. 9, **(2010)**, pp. 245-250.
[11] Y. Y. Lu and W. L. Ding, "A universal compression algorithm improved by run-length encoding", Journal of Zhejiang University of Technology, no. 1, **(2007)**, pp. 60-64.
[12] X. Z. Lin and J. B. Wan, "The lossy RLE compression of true color image", Microelectronics and Computer, no. 11, **(2004)**, pp. 81-84.
[13] K. Y. He, "An improved RLE compression method", Journal of WuHan University of traffic science and technology, no. 4, **(1999)**, pp. 412-414.
[14] L. F. Zhu, "The hybrid lossy compression of DCT and RLE based on MATLAB gray image", Computer Knowledge and Technology, no. 21, **(2009)**, pp. 5763-5765.

Authors

Yan Xijun, Received his Ph.D degrees from HoHai University, he is an associate professor of electrical and computer engineering at HoHai University, his research interests include multi-sensor system and information fusion, information processing system and its applications and information processing for wireless sensor networks.

Jiang Lei, Received his bachelor's degrees from HoHai University, he is a master of electrical and computer engineering at HoHai University, his research interest is information processing for wireless sensor networks.