International Journal of Signal Processing, Image Processing and Pattern Recognition
Vol.8, No.8 (2015), pp.159-168
http://dx.doi.org/10.14257/ijsip.2015.8.8.18

A Method of Oversized Image Generation Based on Stitching JPEG File Streams

Qindong Sun[1,2], Wu Hua[1,2], Minying Tian[1,2] and Yan Wang[1,2]

[1] *School of Computer* Science *and Engineering, Xi'an* University *of Technology, Xi'an* 710048, *China*
[2] *Shaanxi Key Laboratory of Network Computing and Security, Xi'an* University *of Technology, Xi'an* 710048, *China*
*sqd@*xaut.*edu.cn*

Abstract

Due to memory limitation, the displaying and processing of the oversized images have been a bottleneck for various applications, which is a hot research spot in the image processing field. The traditional methods are almost based on the BMP format to generate oversized images, of which the time and space complexity are very high. In this paper, we have deeply analyzed the encoding scheme of JPEG image and proposed an oversized image generation method based on stitching JPEG file streams. According to this method, the oversized images are block plotted and then transformed and saved as JPEG format files respectively. In the end, the block JPEG files are then combined into an integrated JPEG image. The experiment results show that, our method can effectively improve the speed of oversized image generation and reduce the memory consumption. The time and space performance of our method are better than existing methods.

Keywords: *Oversized image, JPEG file stream, Image stitching*

1. Introduction

With the wide application of high-resolution imaging devices, the size of the picture keeps increasing, and the larger wedding photo, Panorama drawing and Advertising Image has been pretty regular in daily life. The size of an image has been increased rapidly from megabytes to gigabytes. In order to improve the work efficiently, the digital printing industry usually uses typesetting software to typeset the different size photos on the fixed-width and fixed-length photo paper. And the width of photo paper is 24 inches, 30 inches or 50 inches in general. When the typesetting's finished, the result will be an output as an oversize image. Therefore, it's necessary to merge some small photos into an oversized image for printing or any other further processing.

In practical applications, how to load and process the oversized image is the mainly influential factor, which influences the processing speed. The process requires lots of memory space, that may induce the system to run slowly, or even memory overflow [1]. So, processing oversized images is becoming a research focus.

Because of the limitation of a normal computer' memory, operating systems and programming tools, it's unable to draw a large canvas. Thereupon, it is impossible to generate a complete oversized image directly. Splicing images in normal size to an oversized image is the only way.

The traditional approach to processing the oversized images is to slice them up and process them part after part. All slices of the oversized image are putted on a big BMP format canvas by utilizing GDI programming technique and then one oversized BMP image is generated. Finally, by using image compression technique, the BMP image is converted into a JPEG image with a smaller size. The above process is simple, but there is

ISSN: 2005-4254 IJSIP

a problem of excessive memory consumption, which will cause large amounts of I/O operations and consume more time.

He and Zhu proposed a method to load and display the oversized images [1-2]. The oversized image is divided into many sequential, uniform small images in appropriate size. When the images are needed to display, only several specified small images are loaded, hence the taken memory space is reduced. Du proposed A segmentation filling algorithm based on queue for the super resolution image [3]. It overcomes the problem of low efficiency, low accuracy and waste of computer resources in the process of super resolution images. Fan proposed a new algorithm to solve the problem of reconstructing a high-resolution image based on several low resolution images [4], which is easy to utilize the priori conditions as constraints to improve the convergence rate, and also can get the high resolution reconstruction image. Guo designed a software that can convert BMP images to JPEG images, and the compression rate can be adjusted by users [5].

All the above researches on oversized image have achieved a few certain results, but they are not involved with the display and output of oversized image. Li designed an improved three-dimensional output algorithm [6], which is based on scenario block output, image sequence and automatic stitching. It solved the issue of three-dimensional topographic map output effectively, but it could only be applied in a certain project.

There are already some researches about the super-resolution image with reconstruction techniques, such as a new algorithm of real-time viewing of large images based on multi-core [7], the new hybrid algorithm [8], the method using the Bayesian transform[9] and the method based on sparse signal representation [10].

The above studies have carried out a detailed analysis and elaboration for large image to display and reconstruct, which have high reference value for our research. However, the existing studies are limited to how to deal with images displayed properly on the computer, and do not involve in how to output the oversized image stably and efficiently.

In this paper, the JPEG file format, the compression/decompression process and its characteristics are analyzed. And then a method of oversized image generation based on stitching the JPEG file streams is proposed. According to the proposed method, the oversized BMP image could be divided and compressed into many blocks and then splicing them together into a complete seamless JPEG image file. The result shows that the proposed method is a stable method with less memory usage and lower time and space complexity.

2. Analysis of JPEG File Format

2.1. The JPEG File Format

JPEG is a widely used image compression standard. JPEG file would be divided into several segments for storing information, but the number and the length of segment is not ascertained [11].

JPEG files can be divided into two parts: Tag and compressed data. The tag gives all the information of image, such as width, height, Huffman tables, quantization tables, etc. Its length is two bytes and the first byte is a fixed value of 0xFF. There are many kinds of tags, but most JPEG files only contain a few. The tags frequently used are shown in Table 1.

Table 1. A Typical Electric Power Network

Symbol	Tag	Description
SOI	FFD8	Start of image
APP0	FFE0	JFIF application-specific

		data
APPn	FFE1-FFEn	application-specific data (n ,1～15)
DQT	FFDB	Define quantization table
SOF0	FFC0	Start of frame
DHT	FFC4	Define Huffman table
SOS	FFDA	Start of scan
EOI	FFD9	End of image

Among all the tags, the restart tag RST_i is a unique tag that can be interspersed in the data stream, but it only can occur at the boundaries of the MCUs (Minimum Coded Unit, MCU). In the decoding process, whenever decoder encounters a restart tag, the Huffman decoder and DC predictor will be reset and allowed to begin partial decoding from the current point. In the data stream, the restart tag marked as FFD_i (i=0,1, …, 7), shown as following, $RST_i = FFD_i$ ($_i = 0 \ldots 7$) [$RST_0=FFD_0$, …, $RST_7=FFD_7$].

2.2. JPEG Encoding and Decoding Process

JPEG mainly involves two basic compression algorithms, two data encoding methods, and four coding modes. Compression algorithms are discrete cosine transforms (DCT) which is loss-compression and predictive compression that is lossless. Data encoding methods are Huffman coding and Arithmetic coding. Coding modes are sequential mode based on DCT, progressive mode based on DCT, non-destructive mode and level mode. In practical applications, JPEG image uses DCT, Huffman coding and sequential mode, as shown in Figure 1 [12]:

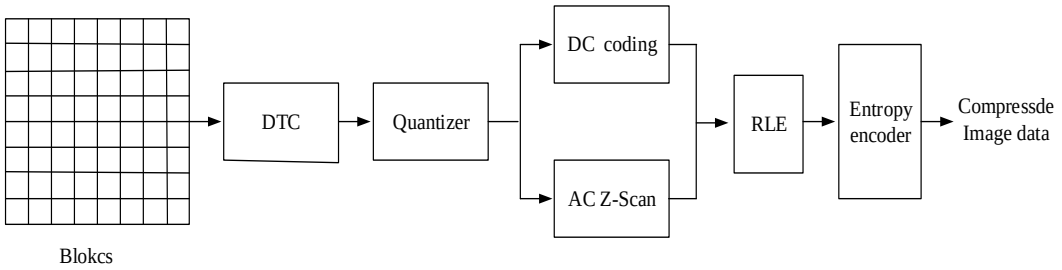


Figure 1. JPEG Encoding Process based on DCT

1) Splitting the input image into blocks of 8×8 pixels.
2) All the blocks are transformed separately by using DCT, and obtain an 8×8 matrix of DCT coefficients. The DCT transforming is lossless theoretically, but the implementation usually has a little information loss because of round-off error of using floating-point numbers and transcendental functions.
3) During the quantization process, according to dividing each of the coefficients in the DCT matrix by a weight in the quantization table, less important DCT coefficients are removed. By adjusting the weights in the quantization table, the balance between information loss and compression efficiency could be controlled.
4) During DC Coding and Zig-Zag scanning, the quantized DC coefficient is encoded as the difference from the DC in terms of the previous block in the encoding order, and all of the 64 quantized coefficients are ordered into the “zig-zag” sequence.

5) The sequence of each blocks produced by the zigzag scanning is encoded with RLE(run-length encoding) technique to reduce the number of zeroes in the sequence, which could improve the compression rate.
6) Finally, the data are Huffman encoding, and output the data in units of block.

The image data stream is divided into numerous smallest coding units, named MCU, which is a square matrix of pixels data. According to the tag SOF0, the sampling factors of different color component, the horizontal sampling factors and vertical sampling factors of *Y*, C_r, C_b, could be acquired. Currently most of the image sampling factor is 4:1:1 or 1:1:1. The 4:1:1 represents (2×2):(1×1):(1×1) and the 1:1:1 represents (1×1):(1×1):(1×1). We define $H_{\max}$ as the maximum horizontal sampling factor and $V_{\max}$ as the maximum vertical sampling factor. Then the width of an MCU is $H_{\max} \times 8$, and the height is $V_{\max} \times 8$.

The MCU sequence of 32px×36px image is shown in Figure 2, where solid line represents the boundary between each MCU and dotted line indicates the boundaries of the MCU's internal data units. It assumes that the sampling factor of this image is 4:1:1, hence $H_{\max} = 2$ and $V_{\max} = 2$. The width of an MCU is 16 pixels and the height is 16 pixels. Therefore, the width of an image is exactly double of an MCU's width and the height is slightly more than the double of an MCU's height, it needs extra triplex rows filled with MCUs. At last, in the data stream, an MCU is divided into four 8*8 data units and its sequence is MCU1, MCU2, MCU3, MCU4, MCU5, MCU6.

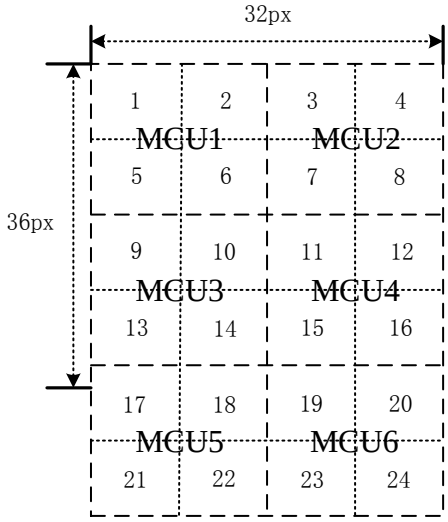


Figure 2. MCU Sequence of the Image

The decoding process of JPEG file is beginning with the read image size and the sampling coefficient information from the file header, which is used to calculate the size of MCUs. And then, the number of MCUs contained in the image is calculated. Finally, MCU is decoded one by one until the EOI tag is detected.

After the analysis of JPEG, we know that JPEG files are divided into MCUs for encoding and decoding, and each MCU contains numbers of 8*8 image blocks. Thus, we can make use of this characteristic of JPEG files to generate an oversized image.

We can divide a bitmap into several blocks with specific regulation and each block contains numbers of MCUs, and then convert each blocks to a JPEG file streams and store them in memory and stitch all the JPEG file streams together by inserting restart tags FFDi (i = 0 ... 7) among each streams. Finally, modify the header information of the

image file and output the stitched JPEG file to disk. By this way, an oversized image could be stitched and generated efficiently.

3. The Method based on Stitching JPEG File Stream

According to the above analysis, an oversized image is firstly divided into blocks, which are drawn and converted to small JPEG files separately. Lastly, all the small file streams are stitched to a seamless JPEG as an oversized image. The procedure mainly includes the follow steps:

1) Obtaining the width and the height of the oversized image as W and H, W and H is in units of pixels.
2) Setting the sampling style of the JPEG image compression, the maximum horizontal and vertical sampling factors $H_{\max}$ and $V_{\max}$, which are both in units of pixels. So the width and height of MCU is $H_{\max}\times 8$ and $V_{\max}\times 8$.
3) Generating one canvas $C_i(i+1,2,....)$ whose width is $W_i(i=1,2,...)$, and height is $H_i(i=1,2,...)$.The canvases refilled with the small block of pictures that generated by the original oversized image. Wherein, the width of the canvas C_i is the same as the width of the oversized image W_i in step 1, i.e.:

$$Wi = W \tag{1}$$

The number of MCUs in each row of the canvas R_i is calculated as follows:

$$R_i = \begin{cases} W_i / (H_{\max}\times 8), & W_i \% (H_{\max}\times 8) = 0 \\ W_i / (H_{\max}\times 8)+1, & W_i \% (H_{\max}\times 8) \neq 0 \end{cases} \tag{2}$$

Assuming the height of Canvas C_i is H_i and contains n MCUs, then

$$H_i = n \times (V_{\max}\times 8) \tag{3}$$

Where *n* is a positive integer, and its values should satisfy the following conditions:

$$(n \times R_i) < 65536 \tag{4}$$

In formula (3) and (4), the total number of MCUs contained in each canvas should be an integer multiple of $_r Ri$, and it should be less than 65535.

Because the total size of all the canvas is equal of the size of the oversized image in step 1, the number of blocks k can be calculated as follow:

$$k = \begin{cases} H / H_i, & H \% H_i = 0 \\ H / H_i + 1, & H \% H_i \neq 0 \end{cases} \tag{5}$$

In formula (5), if the value of $H\%H_i$ is zero, the height of canvas H_i equals to the height in formula (5), an integer multiple of $V_{\max}\times 8$. If the value is not zero, the height of the last canvas is the difference between the height of the oversized image in step 1 and the sum of the height of any other canvases, thus:

$$H_k = H - \sum_{i=1}^{k-1} H_i \tag{6}$$

4) Drawing each piece of the image on the specified canvas obtained in step 3.

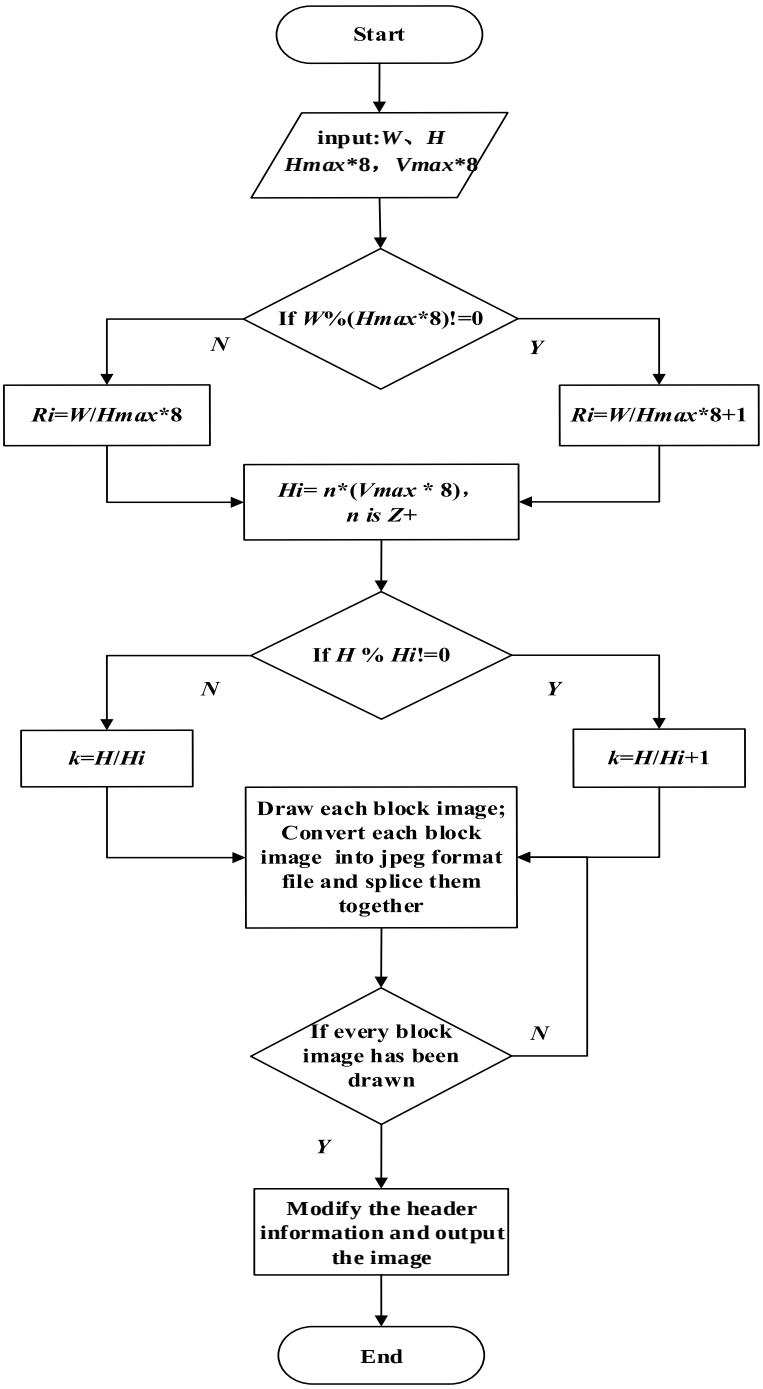


Figure 3. Stitching Method based on the JPEG File Stream

5) Using the parameters gained in Step 2 to compress the first image block which has been successfully drawn in Step 4 into JPEG streams and store it in memory. Then, compressing the other blocks and appending the stream whose head information has been removed to previous one. Finally, modifying the end tag FFD_9 of previous stream to restart tag FFD_i between the two adjacent streams. (The n is circulated from 0 to 7 and there's $i = n\%8$, n is the current numerical order of image block).
6) Modifying the head information of JPEG file stream generated in Step 5. The width and height within the header information are revised as the total width and

height of the canvas. Finally the JPEG stream is outputted to an oversized image file.

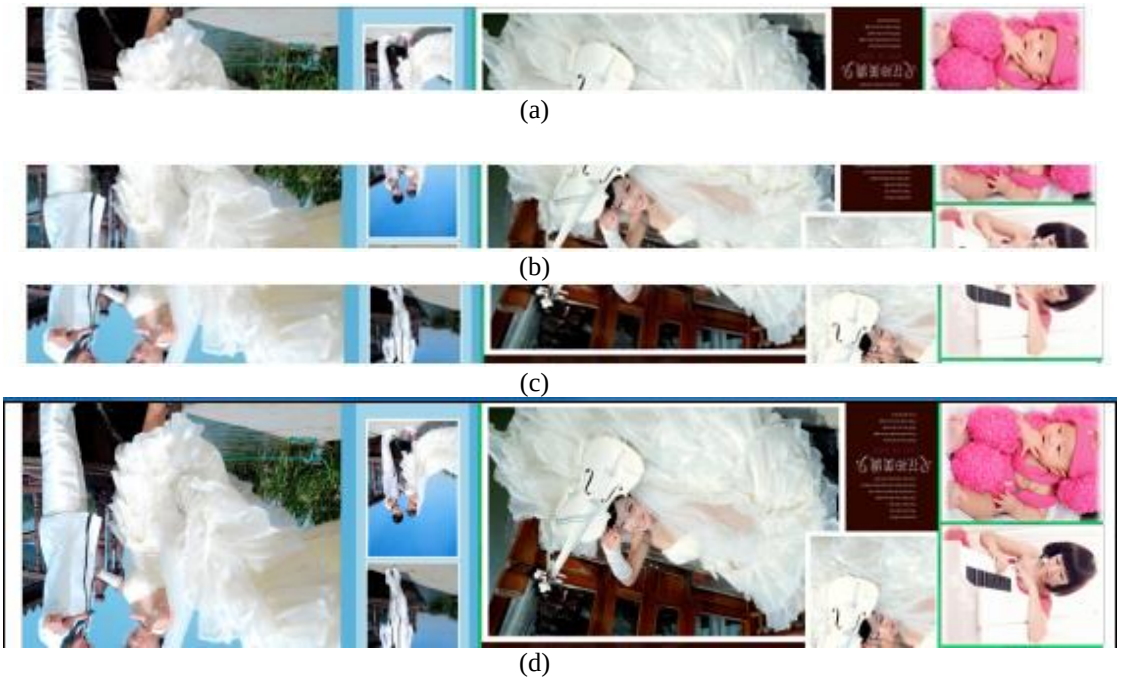
In step 5, all the canvases are successfully converted to JPEG stream, wiped header information when stitching them together except for the first block image stream.

In above steps, the width and height of canvases, as well as MCUs, are in units of pixels. Figure 3 is a flow chart of the proposed method.

4. Experimental Results

The experiments are carried out on a computer configured with Pentium (R) Dual-Core CPU E6600 processor and 4.0GB memory. We stitched five different size of the oversized images and compared the algorithm complexity of time and space with traditional methods.

Figure 4 shows the result of the proposed stitching method, in which (a), (b), (c) are images to be stitched, (d) is the oversized image stitched with (a), (b), and (c) by our proposed method.



(a)

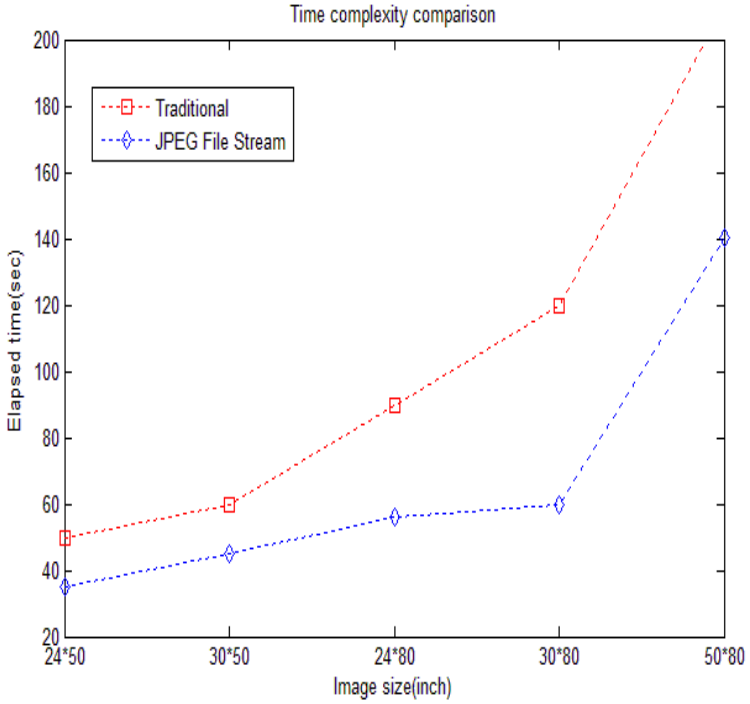
(b)

(c)

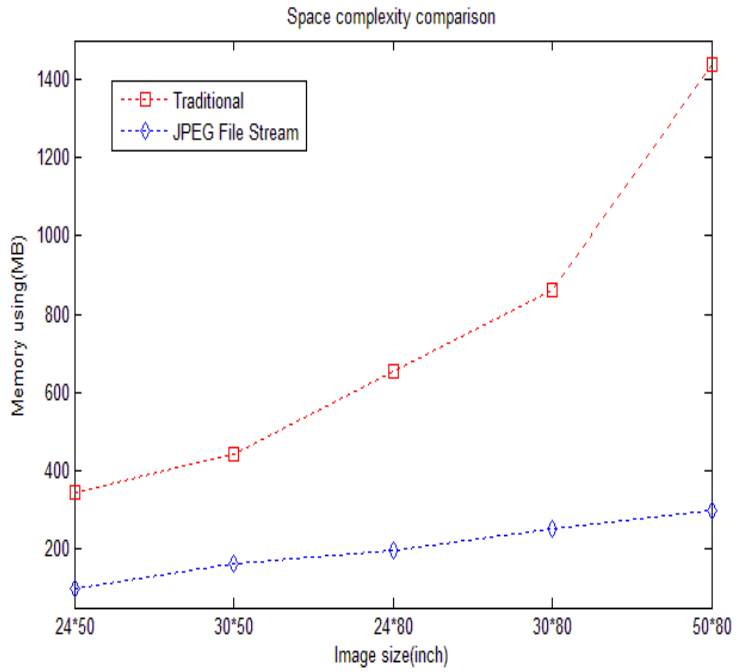
(d)

Figure 4. Generating an Oversized Image with JPEG File Stream Method

Figure 5 is experiment result of time and space complexity contrasted in the stitching process between our method and traditional method. From figure (a), we can find that with the increase of image size the time required for stitching increase obviously, that is mainly due to the increase of area of pictures. However, there is only little change in memory usage with the increase of image size in our method. Because the memory mainly store the current processed image and will not change greatly with the increase of image size, our method is superior to traditional methods in these two aspect.



(a) Comparison of time complexity



(b) Comparison of space complexity

Figure 5. Comparison of Time and Space Complexity

According to the above experiment, it's not difficult to find that the time and memory usage has been improved significantly compared with the old method. With the increase of image size, there is only a bit change in memory usage. Our new method has solved the problem of memory overflow effectively. At present, the method has been used in automatic typesetting system for stitching the layout as an oversized photo.

5. Conclusion

In this paper, we proposed a new method of oversized image generation based on stitching the JPEG file streams. The method could reduce the memory usage and improve the time performance of the algorithm by reducing the I/O operations with external storage device. Through the contrast in the experiment, the results show that the proposed method has particular advantages in space and time than traditional method.

Acknowledgments

The authors would like to thank the anonymous reviewer and editors for their helpful comments and suggestions. The research presented in this paper is supported partly by the International Cooperation Project of Shaanxi Province (No.: 2013KW01-01), The Key Laboratories Development Program of Shaanxi Province Education Department (No.: 13JS085), and the National Natural Science Foundation of China (No.: 61172124).

References

[1] D. He, "The analysis and method of loading for oversized images [J]", *Science & Technology Information*, vol. 28, no. 10, (**2010**), pp. 251-251.
[2] Y. Zhu and W. Liu, "Research of Blocking Display Large Pixels Pictures Algorithm Based on Windows CE [J]", Science Technology and Engineering, vol. 15, no. 8, (**2008)**, pp. 4170-4173.
[3] J. Du, C. Song, X. Jiang and Y. Chen, "Segmentation Filling Algorithm for Super Resolution Image [J]", Computer Engineering, vol. 37, no. 6, (**2011**), pp. 203-205.
[4] C. Fan, N. Sun and X. Xia, "Super-resolution Reconstruction Based on Image Sequences [J]", Infrared Technology, vol. 32, no. 5, (**2010)**, pp. 279-282.
[5] W. Guo, T. Feng, W. Luo and J. Liu, "BMP images converted to JPEG images design [J]", FUJIAN COMPUTER, no. 8, (**2006**), pp. 8-8.
[6] F. Li, G. Wan, X. Cao, Y. Zhang and Z. Zhang, "Researches on 3D Electronic Map Printing Out [J]", Journal of Geomatics Science and Technology, vol. 27, no. 4, (**2010**), pp. 310-312.
[7] X. Yang, D. Xu and L. Zhao, "Real time viewing of large images based on multi-core [J]", Journal of Image and Graphics, vol. 16, no. 2, (**2011)**, pp. 152-160.
[8] D. Glasner, S. Bagon and M. Irani, "Super-Resolution from a Single Image [C]", IEEE International Conference on Computer Vision, (**2009**), pp. 349-356.
[9] M. O. Camponez, E. Ottoni, T. Salles and M. Sarcinelli-Filho, "Super-Resolution Image Reconstruction Using Nonparametric Bayesian INLA Approximation [J]", IEEE transactions on image processing, vol. 21, no. 8, (**2012**), pp. 3491-3501.
[10] J. Yang, J. Wright, T. Huang and Y. Ma, "Image Super-Resolution via Sparse Representation [J]", IEEE transactions on image processing, vol. 19, no. 11, (**2010**), pp. 2861-2873.
[11] A. Anand and H. Aggarwal, "Bmp to jpeg - the conversion process [J]", Journal of Global Research in Computer Science, no. 2, (**2011**), pp. 186-198.
[12] M. C. Stamm, S. K. Tjoa, W. S. Lin, and K. J. Ray Liu, "Anti-forensics of jpeg compression [C]", 2010 IEEE International Conference on Acoustics Speech and Signal Processing, (**2010**), pp. 1694-1697.

