Hybrid Image Coding for Real-Time Computer Screen Video Transmission

Tony Lin[a*], Pengwei Hao[ba], Chao Xu[a], and Ju-Fu Feng[a]

[a] National Laboratory on Machine Perception,
Peking University, Beijing 100871, China,
{lintong,xuchao,fjf}@cis.pku.edu.cn

[b] Department of Computer Science, Queen Mary,
University of London, London, E1 4NS, UK,
phao@dcs.qmul.ac.uk

ABSTRACT

In this paper, we present a novel hybrid image coding scheme for real-time applications of computer screen video transmission. Based on the Mixed Raster Content (MRC) multilayer imaging model, the background picture is compressed with lossy JPEG, and the foreground layer consisting of text and graphics is compressed with a block-based lossless coding algorithm, which integrates shape-based coding, palette-based coding, palette reuse, and LZW algorithm. The key technique is to extract text and graphics from background pictures accurately and with low complexity. Shape primitives, such as lines, rectangles, and isolated pixels with prominent colors, are found to be significant clues for textual and graphical contents. The shape-based coding in our lossless algorithm provides intelligence to extract the computer-generated text and graphics elegantly and easily. Experimental results demonstrate the efficiency and low complexity of our proposed hybrid image coding scheme.

Keywords: hybrid image coding, shape-based coding, palette-based coding, Mixed Raster Content, content adaptivity

1. INTRODUCTION

As the number of computers and other digital devices being connected keeps growing, there has been a critical need for real-time computer screen video transmission technologies. Remote control software, such as AT&T VNC [1], allows a person at a remote computer (the client, maybe a Linux machine) to view and interact with another computer (the server, maybe a Windows PC) across a network, as if sitting in front of the other computer. A smart display device, such as Microsoft Mira [2], acts as a portable screen with 802.11b wireless connection to a nearby desktop PC, enabling people to surf the web or browse pictures that are stored on the desktop PC. Another application is wireless projectors, providing the flexibility to site anywhere in the room without cable connections to the laptop. The challenge is that the huge amount of real-time computer screen video data should be transmitted over the cable or wireless networks. One 800×600 frame of true color screen image has a size of 1.44 MB, and 85 frames per second produces more than 100 MB data. One way to reduce the data volume is to get updated screen regions by accessing the GUI interfaces and to have the truly updated regions transmitted. History frame buffers are necessary to switching between several running windows, but cost

[*] This work was supported by NKBRSF China under Grant G1998030606 and the funding for the recipients of the National Excellent Doctoral Dissertation, China, under Grant 200038. Correspondence: Email: lintong@cis.pku.edu.cn. Telephone: (86-10)6275-5569. Fax: (86-10)6275-5569.

large memory. Due to remaining image data still being huge, an efficient image compression algorithm with low complexity and high quality representation is desired.

Computer-generated screen images are compound images, which are mixed with textual, graphical, or pictorial contents. Compound image compression is intensively studied for scanned documents, and the Mixed Raster Content (MRC) imaging model [3], [4] is frequently used to decompose a scanned compound image into two color-image layers (foreground and background) and a binary mask layer. Some published algorithms are DjVu [5], DigiPaper [6], block thresholding segmentation [7], and stripe analysis [8]. These algorithms invariably involve complicated layer-based segmentation which is computationally expensive. The alternative is block-based segmentation proposed in [9], [10], and [11], but yields poor quality for the blocks that cross the boundary of pictures and text. The compression of computer-generated compound images was investigated in JPEG variable quantization method [12], modified JPEG-LS [13], and lossless coder using intraplane and interplane coding [14]. However, totally lossy or lossless compression can not provide the content-adaptivity to compound images thus reproduces text/graphics blurred or otherwise achieves little coding gains. Although the VNC [1] provided a simple rectangle-based lossless coding algorithm for computer screen images on the assumption that most GUI images compose of filled rectangles, it is very inefficient to compress natural pictures.

For the real-time applications of computer screen video transmission, three features are essential to the expected compound image compression algorithm: very low complexity, visually lossless quality, and high compression ratio. Unfortunately none of the above mentioned algorithms meet these three requirements, which motivated the development of a new hybrid image coding scheme presented in this paper. Briefly, the new method is to compress the text and graphics as foreground losslessly and to compress the pictorial background in a lossy manner.

The rest of this paper is organized as follows. In Section 2, a block-based lossless coding algorithm is presented for lossless compression of text and graphics. The foreground extraction procedure is described in Section 3. Experiments and discussions are provided in Section 4 and 5, and the paper is concluded in Section 6.

2. LOSSLESS CODING OF TEXT AND GRAPHICS

In order to guarantee low complexity, a lossless coding algorithm by extending the VNC rectangle-based coding is proposed for text and graphics compression. In our lossless algorithm, rectangle-based coding is improved to be more efficient and shape-based, and several other techniques such as palette-based coding, run-length coding, palette reuse, and LZW algorithm are employed to form a versatile framework. Though each coding method is quite primitive by itself, a powerful lossless coding scheme is obtained by using all these methods adaptively and simultaneously.

Our lossless coding scheme is block-based. Each image is divided into 16×16-pixel blocks, with the same size as JPEG macroblocks. In contrast to region-based and layer-based segmentation, block-based segmentation is the simplest and the existing block-based coding techniques such as JPEG can be easily incorporated into. The annoying blocking artifacts, which are a well-known disadvantage of the lossy JPEG, do not exist in lossless coding algorithms. When a block contains both pictorial and textual contents, text can be extracted and encoded independently, which is described in Section 3. Therefore, our block-based lossless coding scheme offers low complexity, good compatibility, and high quality.

The shape-based coding extends the VNC rectangle-based coding by providing a set of shape primitives: isolated pixels, horizontal lines, vertical lines, and rectangles. These shape primitives are efficient to represent textual and graphical contents. A rectangle can be represented by (*x, y, w, h*) in 2 bytes, a diagonal line by (*x, y, w*) in 12 bits, a horizontal line by (*x, y, w*) in 12 bits, a vertical line by (*x, y, h*) in 12 bits, and a pixel by (*x, y*) in 1 byte. The default line width is 1 and *x, y, w* or *h* only needs 4 bits for our 16x16 blocks. Compared to VNC that all rectangles are encoded with 2 bytes, our shape-based coding creates a compact bitstream by classifying rectangles into isolated pixels, lines, and rectangles. The percentages of different shape primitives in four 800×600 computer screen images are listed in Table I, showing that most of shape primitives are isolated pixels and lines. Some English and Chinese characters are shown in Fig. 1 and Fig. 2, from which we can see that shape primitives can be used to compactly represent textual contents.

The procedure to find shape primitives in a block is similar to that in VNC, but with several modifications. First we count the colors in a block by using exact 24-bit color matches, in order to determine the appropriate coding method. If the number of colors is larger than some predefined threshold, T_l, then leave this block to lossy JPEG, because it takes much effort to compress but with little gains obtained by using the lossless algorithm. The color count beyond T_l implies that the block contains pictorial contents. In our experiments, we set $T_l = 32$ in our system. One test image *msn* is shown in Fig. 3, and the blocks having more than 32 colors are blacked out shown in Fig. 4. From Fig. 4 we can see that the threshold 32 is appropriate to pick out all the pictures and some complicated icons. To avoid useless color matches, we stop the color matching immediately after the color count is over the threshold T_l. We also take the color with most pixels as background color of the block, so only other foreground colors need to be encoded and lots of bits are saved.

TABLE I
PERCENTAGES OF SHAPE PRIMITIVES (%)

Image	Pixels	Horizontal Lines	Vertical Lines	Rectangles
msn	41	30	23	6
pku	40	27	27	5
sina	46	27	22	5
wall	59	19	20	2

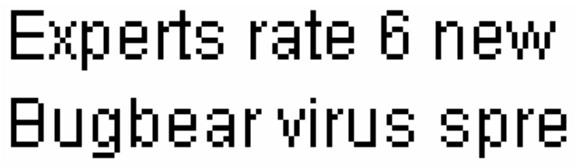
Experts rate 6 new
Bugbear virus spre

Fig. 1. Text details from webpage image *msn*.

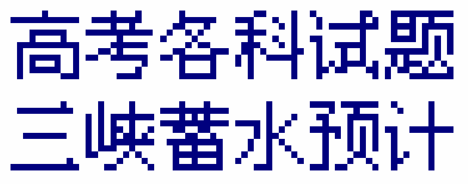
高考各科试题
三峡蓄水预计

Fig. 2. Text details from webpage image *sina*.

Shape primitives are then recognized by scanning the block pixels. Similar to VNC coder, size-first coding strategy is used to encode some bigger ones first if several rectangles of the same color are found. For example, four rectangles in Fig. 5, AEFM, ADGL, ACHK, and ABIJ, are found for some color. Apparently, rectangle ADGL has the biggest size, but it is hard to find. There are many ways to decompose one shape into multiple rectangles, and it is time-consuming to find the optimal decomposition to create the shortest representation. In our system, only horizontal rectangles and vertical rectangles are under consideration. Thus only the vertical rectangle ABIJ is encoded because it has larger size than the horizontal rectangle AEFM, and AEFM doesn't exist after ABIJ has been encoded. The numbers of shape primitives of one color (isolated pixels, horizontal lines, vertical lines, and rectangles) are represented with run-length coding because many continual zeros exist.

Although shape-based coding captures the nature of texts/graphics and usually achieves very high compression ratios, it is undesirable for blocks with complicated shapes, such as some delicate icons. In such cases, palette-based coding seems to be a good alternative to shape-based coding. A pattern of two colors needs a 1-bit mask, and a pattern of 17 or 32 colors needs at least a 5-bit mask. There are many choices to determine which colors are with palette-based or shape-based coding, and it is difficult to find the optimal decision according to the bitstream length. Therefore, we balance between complexity and performance by providing three choices: all colors by shape-based coding, or all colors by palette-based coding, or only the most complicatedly shaped color is by palette-based coding while others by shape-based coding.

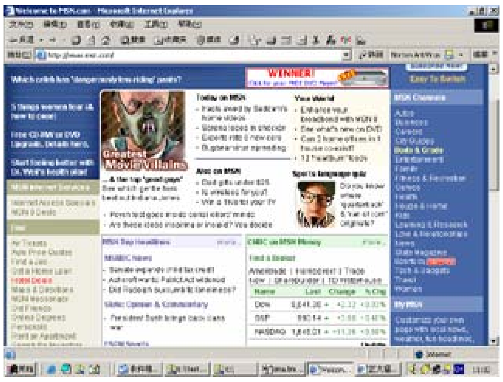

Fig. 3. Webpage *msn*.

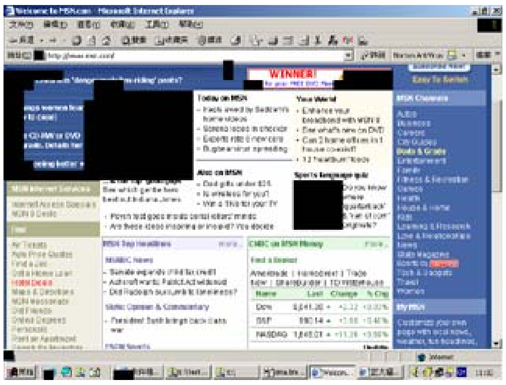

Fig. 4. Blocks having more than 32 colors are blacked out for *msn*.

We adopt color palette reuse technique to save bytes to store the RGB color table which is a fairly big overhead, and two consecutive blocks can share most of colors. For construction of an adaptive global dictionary of colors, it is difficult to maintain the dictionary and it also takes much time to do color matches. In our system, color palette of the current block is mapped into color palette of the last block, thus only the new colors are explicitly stored in bitstream for current block.

Finally, the above bitstream can be fed into an adaptive arithmetic coder, or a LZW coder for further compression. DjVu uses the ZP-coder which is a new type of adaptive binary arithmetic coder. We use zlib [15], a LZW coder, in our system.

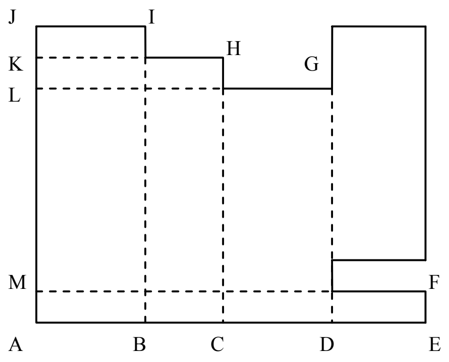

Fig. 5. Size-first strategy for shape-based coding.

3. FOREGROUND EXTRACTION

The above block-based lossless algorithm can be easily extended to a naïve hybrid coding scheme by using lossy JPEG to encode the blocks of more than T_1 colors. It is equivalent to an inter-block segmentation and coding. The disadvantage is that it makes text and graphics messy if some pictorial content exists in the same block. Examples are Fig. 9 (e) and 10 (e), as the original images are shown in Fig. 6 and Fig. 7, respectively.

A better way is to extract textual and graphical contents in each block, i.e. to employ intra-block segmentation. The key problem is how to discern text and graphics from pictures. A number of attributes of text are summarized in [6], such as high contrast, occurring repeatedly and in groups, in similar colors. Careful observation on the distinctive nature of text/graphics from pictures tells that shape primitives, such as lines and rectangles drawn with homogeneous interior colors, are significant clues for recognition of computer-generated text and graphics. Based on this observation, the task of extracting foreground can be easily performed by our shape-based coding, which serves to be not only a coding method but also a segmentation method.

There are two problems for the intra-block segmentation. One is the isolated pixels in text and graphics, such as the dot in "i" or the left-bottom pixel in "A", which can not be differentiated. If isolated pixels are coded with lossy algorithm, visual legibility for these text and graphics is degraded. Another is false alarms in pictures, e.g. there is a big possibility in blue sky region of Fig. 6 that several adjoining pixels have an exact same color. It is unwise to encode the pictorial pixels with lossless algorithms.

We propose a dynamic color palette to solve the above two problems. Firstly we assume the pure black color and pure white color are text colors, because most text are drawn in these two colors and the false alarms caused by these two colors are rare in pictures. Then, a dynamic color palette is constructed to collect all the recently appeared colors in big shape primitives. In our system, the size of the dynamic color palette is 32, and a first-in first-out strategy is used to maintain the palette size. Those colors of shape primitives with more than six pixels are put into the dynamic palette, because there is little possibility of six adjoining pixels having the exact same color in pictures. When we meet an isolated pixel, or a shape primitive with size less than four pixels, if its color is not pure black and pure white, the isolated pixel or the shape primitive is compared with the dynamic color palette. If an exact matching to a color in the dynamic palette occurs, then the isolated pixel or the shape primitive is encoded with our lossless algorithm.

Some modifications to the shape-based coding in Section 2 are needed here to adapt it to the new requirements. The regions of text and graphics pixels must be clearly defined by a binary selector layer in each block. The selector layer can be represented by a 1-bit mask, or a set of shapes, depending on which method produces the shortest bitstream.

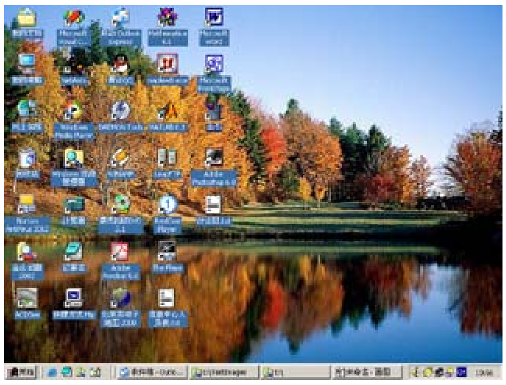
Fig. 6. An image of Windows wallpaper *wall*.

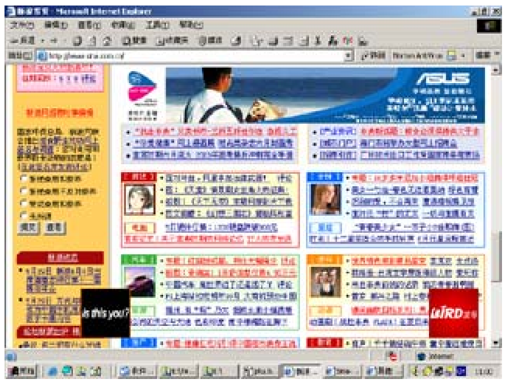
Fig. 7. Chinese webpage *sina*.

On compression side, text and graphics in one block are extracted and encoded first, and fed into a LZW coder for further compression. Then lossy JPEG is used for the remaining pixels. On decompression side, we first decode JPEG-encoded pixels as background, and then decode text and graphics as foreground to pour through the binary selector layer onto the background.

For the lossy JPEG coder, textual and graphical pixels are unused or “don’t care” pixels, because their colors can be chosen arbitrarily. A delicate way is to fill these holes with similar colors to neighbor pictorial pixels so as to reduce ringing artifacts and to improve the performance of lossy JPEG coder. In [8], average color of the previous block or the given pixels is used to fill the holes, while in [7] a multi-pass algorithm is applied to exploit the average color of neighbor pictorial pixels. For simplicity, we just fill these holes with average color of the remaining pictorial pixels in our system.

The flowchart of our compound image compression is shown in Fig. 8. As seen, our scheme provides a variety of coding methods, and new algorithms can be easily integrated into this open scheme. Some example details of experimental results are shown in Fig. 9 (f) and 10 (f), from which almost perfect visual performance is demonstrated for our hybrid compression scheme.

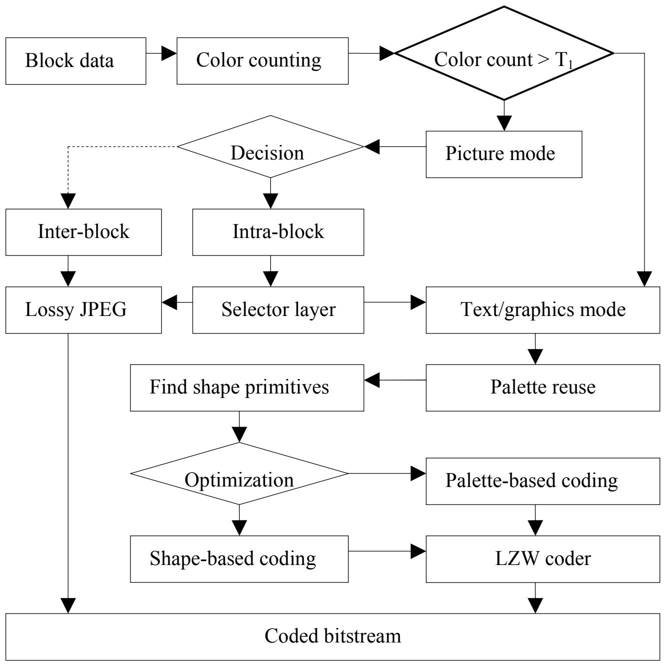

Fig. 8. Flowchart of the proposed hybrid image compression.

4. EXPERIMENTAL RESULTS

The proposed algorithms are implemented on a P-M 1.3G laptop PC, and eight 800×600 true color screen images are tested, including two Chinese (*chn1* and *chn2*) and two English document images (*eng1* and *eng2*), three webpages (*msn*, *sina*, *pku*), and one wallpaper image (*wall*). Images of *msn*, *wall* and *sina* are shown in Fig. 3, Fig. 6, and Fig. 7, respectively.

Our lossless algorithm without LZW followed (OURS-) and with LZW followed (OURS) are compared with VNC and LZW (zlib 1.1.4 with default level 6). JBIG2 and JPEG-LS are not tested, because JBIG2 is only for binary documents and JPEG-LS was reported inferior to LZW in [14]. Compression ratios and computational complexities for four document images are listed in Table II and Table III, indicating that our algorithm with LZW followed (OURS) achieves the highest compression ratios and satisfactory encoding time. Specifically, our algorithm with LZW (OURS) greatly outperforms the LZW by offering higher compression ratios and only requiring less than seventy milliseconds of encoding time. Also our algorithm without LZW followed (OURS-) beats the VNC algorithm by doubling the compression ratios in a comparable encoding time. In summary, our lossless algorithm encodes four document images with compression ratios from 20 to 35 if without LZW, or from 30 to 60 if with LZW followed. The amazing high compression ratios can be explained that our lossless algorithm essentially captures the nature of text. Table IV lists details for output bytes and coding gains of different techniques employed in our compression scheme, indicating that our shape-based coding, palette reuse technique and LZW make significant contributions while palette-based coding achieves not much coding gains.

Three webpages and the wallpaper image are tested by our hybrid coding algorithm, two lossless algorithms (VNC and LZW), and three lossy algorithms (JPEG, JPEG-2000, and DjVu). JPEG-2000 is the state-of-the-art lossy image compression standard, and in our experiments Jasper codec is used. DjVu is the benchmark scheme of compound image compression, and we use Any2DjVu [16] web service to create DjVu files. We set the quality factor of JPEG and the compression ratio of JPEG-2000 to meet our algorithm, but we cannot control the compression ratio of DjVu. Table V and Table VI are details of compression ratios and computational complexities, respectively. We don't know the coding time of DjVu. Some details of image *wall* and *sina* coded with lossy algorithms are shown in Fig. 9 and Fig. 10.

TABLE II
COMPRESSION RATIOS FOR DOCUMENT IMAGES
BY FOUR LOSSLESS ALGORITHMS

Image	chn1	chn2	eng1	eng2
VNC	12.1	17.8	13.3	14.2
LZW	33.3	43.5	44.8	44.9
OURS-	21.8	35.6	28.8	30.7
OURS	**34.2**	**59.3**	**47.9**	**50.4**

TABLE III
COMPUTATIONAL COMPLEXITY FOR DOCUMENT IMAGES
BY FOUR LOSSLESS ALGORITHMS

Image	Encoding Time (msec.)				Decoding Time (msec.)			
	chn1	chn2	eng1	eng2	chn1	chn2	eng1	eng2
VNC	30	31	31	31	13	10	12	11
LZW	196	194	197	195	23	22	22	22
OURS-	42	34	39	37	14	13	14	13
OURS	63	46	54	52	16	14	14	14

TABLE IV
OUTPUT BYTES AND CODING GAINS OF DIFFERENT TECHNIQUES EMPLOYED
IN OUR HYBRID IMAGE COMPRESSION SCHEME

Image	Shape	Palette	Palette Reuse	Run-Length	LZW
msn	30K (-34%)	1K (-52%)	12K (-43%)	22K (-24%)	43K (-38%)
pku	24K (-34%)	3K (-28%)	9K (-42%)	17K (-16%)	34K (-41%)
sina	41K (-35%)	8K (-24%)	15K (-41%)	26K (-22%)	56K (-37%)
wall	58K (-39%)	2K (-9%)	3K (-19%)	3K (-37%)	10K (-26%)
chn1	25K (-40%)	1K (-11%)	11K (-46%)	17K (-37%)	39K (-35%)
chn2	18K (-34%)	2K (-11%)	3K (-66%)	9K (-2%)	22K (-42%)
eng1	25K (-34%)	4K (-11%)	3K (-62%)	11K (-2%)	28K (-41%)
eng2	23K (-34%)	3K (-8%)	3K (-61%)	10K (-2%)	26K (-41%)

Coding gains of shape-based coding are computed in comparison with VNC rectangle-based coding.

In comparison, VNC lossless algorithm usually requires the shortest encoding time, but its encoding performance is the worst; LZW requires more encoding time than our algorithm, but its compression ratios are inferior to our algorithm, especially worse for natural pictures. JPEG produces poorest visual quality, if at the same compression ratio as ours. JPEG-2000 is not suitable to compress textual contents, though its performance on natural pictures is very good. Usually JPEG-2000 requires about one second encoding an 800×600 image. The visual quality of DjVu is acceptable, but some ringing artifacts around text regions are noticeable. The compression ratios of DjVu are greatly inferior to our algorithm, and it was reported in [10] that DjVu spends over 10 seconds to encode one 768×928 scanned document image. To trade-off between complexity, coding efficiency, and image quality, the best choice is our hybrid algorithm, which produces perfect visual quality, and achieves high compression ratios and low computational complexities simultaneously, for examples in Fig. 9(f) and 10(f). In an 802.11b wireless network with a realistic throughput of 2.5-4 Mbps, approximately two full-screen 800×600 true color pictures can be transmitted in one second by using our hybrid algorithm.

TABLE V
COMPRESSION RATIOS FOR THREE WEBPAGES AND WALLPAPER IMAGE

Image	msn	pku	sina	wall
VNC	5.7	8.9	5.4	1.1
LZW	10.9	18.9	12.9	1.5
JPEG	23.6	31.4	19.0	10.7
JASPER	23	31	19	11
DjVu	8.2	7.5	5.4	9.3
OURS	23.3	31.6	19.3	11.7

TABLE VI
COMPUTATIONAL COMPLEXITY FOR THREE WEBPAGES AND WALLPAPER IMAGE

Image	Encoding Time (msec.)				Decoding Time (msec.)			
	msn	pku	sina	wall	msn	pku	sina	wall
VNC	38	35	43	80	16	14	20	13
LZW	213	198	223	338	24	23	23	58
JPEG	102	106	107	123	68	63	63	76
JASPER	870	891	981	991	410	380	430	560
DjVu								
OURS	99	76	110	220	29	22	29	126

5. DISCUSSIONS

1. Lossless coding for colored text and graphics. The forthcoming JBIG2 standard is only for binary image compression, and much effort is needed to extend the JBIG2 for colored text and graphics. Token-based compression, or token dictionaries, is the heart of JBIG2. Although token dictionaries are useful to represent repeatedly occurring English characters, they offer little coding gains for Asian characters such as Chinese characters. Moreover, token-based compression requires connected component analysis and token matching, which demands heavy computations. Ideal compact representation of text and graphics is to decompose a document into a list of drawing operations, such as “fill a rectangle”, “draw a line”, or “draw a line of text”. However, this decomposition procedure is difficult because it is the reverse of drawing a compound document. Multilayer document representation and our proposed shape-based coding appear to approximate the above ideal document representation.

2. Wavelet vs. DCT for lossy picture coding. Wavelet-base coding, like JPEG 2000, is reported to offer many functionalities like ROI, and provide better quality than DCT-based JPEG. On the other hand, wavelet-based transform requires several times of computational complexity than DCT, which obstructs wavelet-based coding from real-time applications.
3. Extension to JPEG and MPEG-1/2/4. Our block-based lossless coding algorithm can be easily integrated into existing JPEG and MPEG-1/2/4 coders. With this extension, compound image such as commercial posters, cartoon movies, and commercial videos can be compressed with text and graphics delivering visually lossless quality.
4. Extension to scanned document compression. The difference between computer-generated compound images and scanned document images is that scanned document images invariably contain visually noticeable noise. Our hybrid image compression scheme can be applied to color quantized scanned documents, but its performance needs experimental verifications.

6. CONCLUSION

We have presented an efficient hybrid image coding scheme with very low complexity for computer screen images. Two significant contributions are the shape-based segmentation to extract text and graphics, and a block-based lossless coding algorithm which employs several techniques. Experimental results demonstrate the advantages of our hybrid image coding scheme with low complexity, high compression ratio, and visual lossless image quality. We also discussed some useful extensions and other applicable prospects. Our future work is to improve the coding efficiency of our lossless algorithm and to extend to JPEG, MPEG, and scanned document compression applications.

REFERENCES

1. http://www.uk.research.att.com/vnc/index.html
2. http://www.microsoft.com/windows/smartdisplay/
3. Draft Recommendation T.44, Mixed Raster Content (MRC), ITU-T Study Group 8, Question 5, May 1997.
4. R. de Queiroz, R. Buckley and M. Xu, "Mixed raster content (MRC) model for compound image compression," *Proc. EI'99, VCIP*, SPIE Vol. 3653, pp. 1106-1117, Feb. 1999.
5. L. Bottou, P. Haffner, P. G. Howard, P. Simard, Y. Bengio, and Y. LeCun, "High quality document image compression with 'DjVu'," *Journal of Electronic Imaging*, vol. 7, no. 3, pp. 410-425, July 1998.
6. D. huttenlocher, P. Felzenszwalb, and W. Rucklidge, "DigiPaper: A versatile color document image representation," in *Proc. ICIP*, vol. I, pp. 219-223, Oct. 1999.
7. R. de Queiroz, Z. Fan, and T. D. Tran, "Optimizing block-thresholding segmentation for multilayer compression of compound images," *IEEE Trans. Image Processing*, vol. 9, pp. 1461-1471, Sep. 2000.
8. D. Mukherjee, N. Memon, and A. Said, "JPEG-matched MRC compression of compound documents," in *ICIP'01*, pp. 434-437, 2001.
9. H. Cheng and C. A. Bouman, "Multilayer document compression algorithm," in *ICIP'99*, pp. 244-247.
10. X. Li and S. Lei, "Block-based segmentation and adaptive coding for visually lossless compression of scanned documents," in *Proc. ICIP*, vol. III, pp. 450-453, 2001.
11. D. Mukherjee, C. Chrysafis, and A. Said, "Low complexity guaranteed fit compound document compression," in *Proc. ICIP*, vol. I, pp. 225-228, 2002.
12. K. Konstantinides and D. Tretter, "A JPEG variable quantization method for compound document," *IEEE Trans. Image Processing*, vol. 9, pp. 1282-1287, no. 7, July 2000.
13. F. Ono, I. Ueno, T. Takahashi, and T. Semasa, "Efficient coding of computer generated images with acceptable picture quality," in *ICIP*, vol. 2, pp. 653-656, 2002.
14. X. Li and S. Lei, "On the study of lossless compression of computer generated compound images," in *ICIP'01*, vol. 3, pp. 446-449, 2001.
15. http://www.gzip.org/zlib/.
16. http://Any2DjVu.djvuzone.org/.

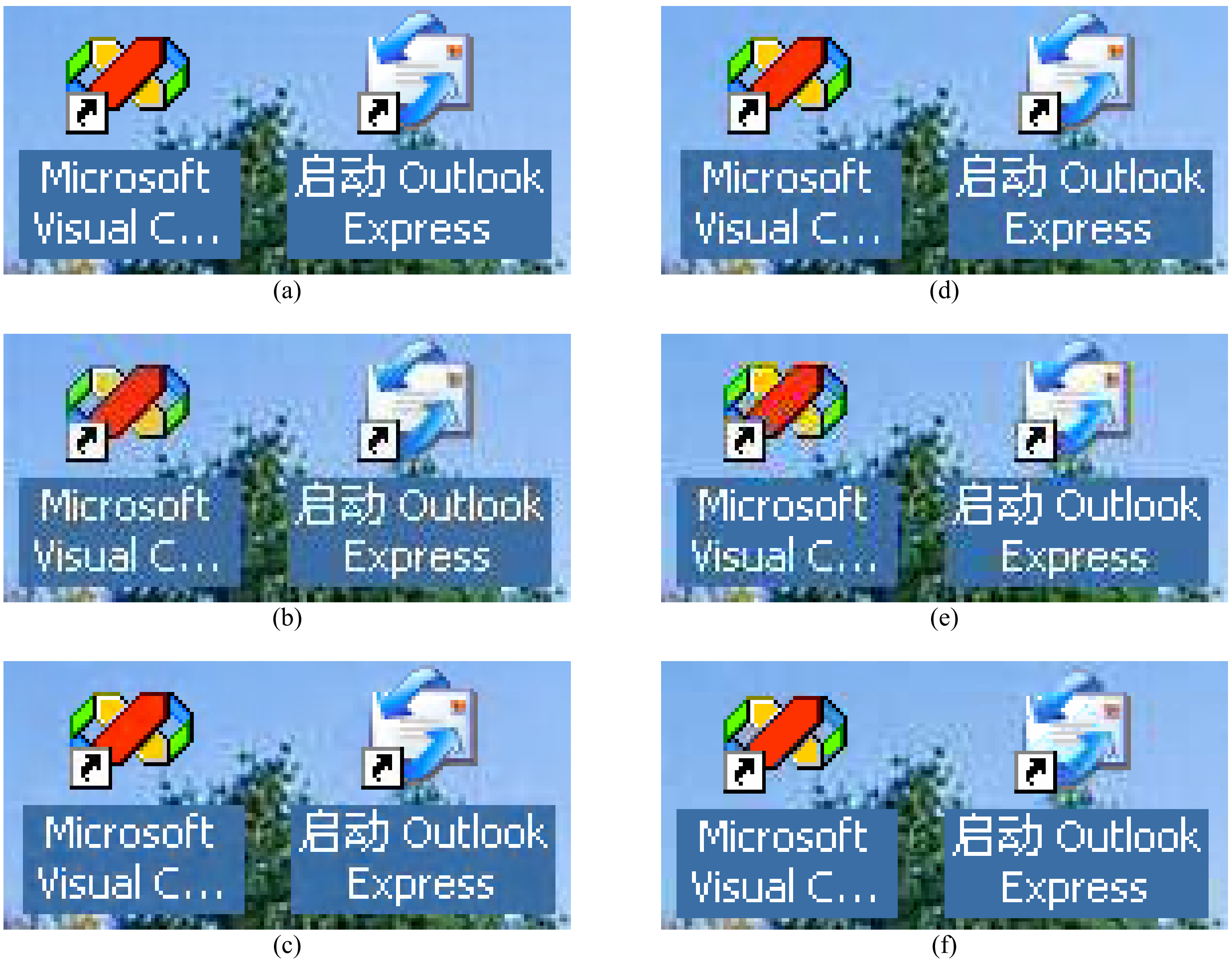

Fig. 9. Details of the *wall* image. (a) Original data, 1407KB. (b) JPEG, quality factor 75, 131KB, encoding 123ms, decoding 76ms. (c) JPEG 2000, compression ratio 11, 128KB, encoding 991ms, decoding 560ms. (d) DjVu, 151KB, run time unknown. (e) Our hybrid coding without intra-block segmentation, 97KB, encoding 143ms, decoding 66ms. (f) Our hybrid coding (with intra-block segmentation), 122KB, encoding 220ms, decoding 126ms.

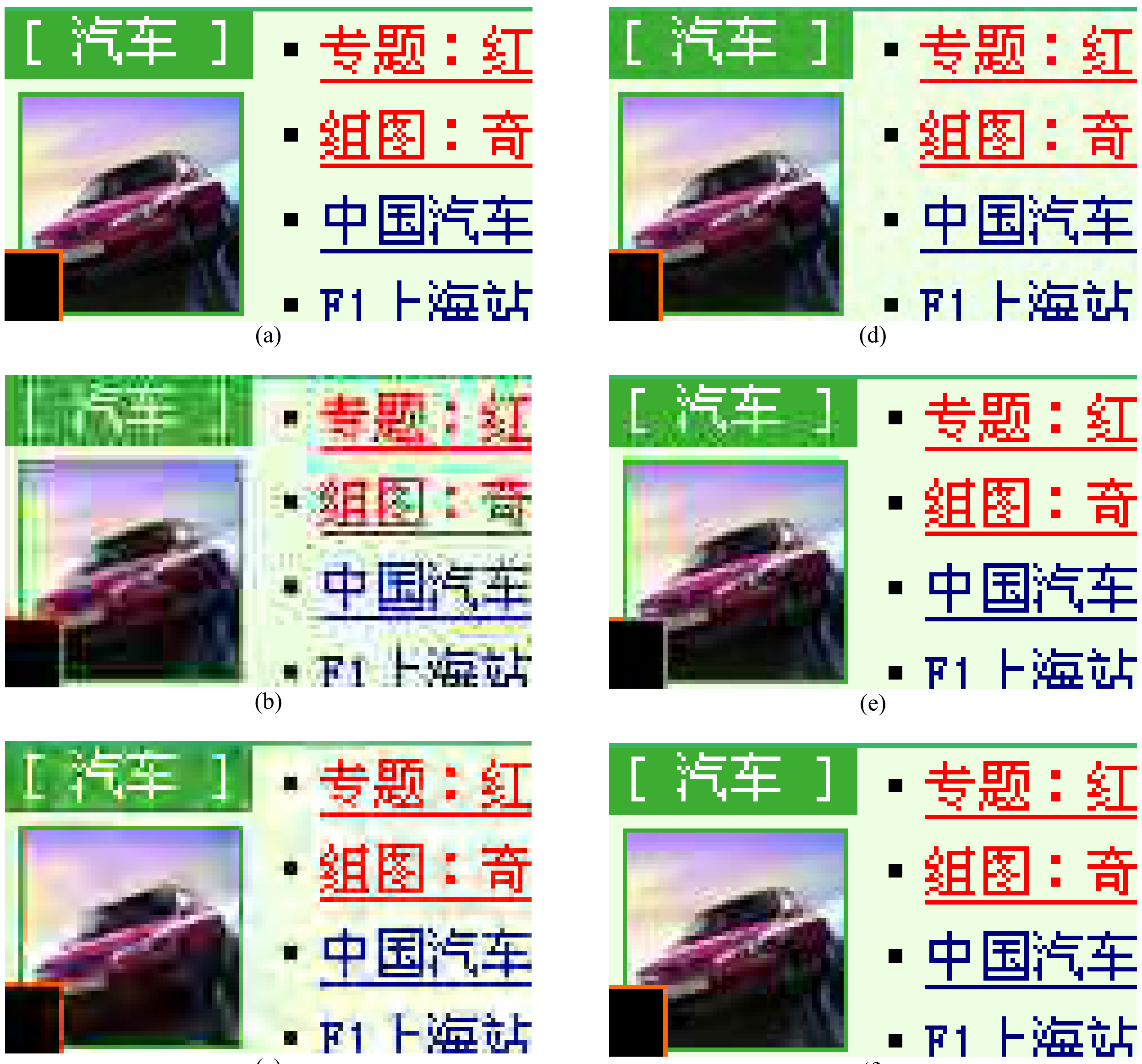

Fig. 10. Details of the *sina* image. (a) Original data, 1407KB. (b) JPEG, quality factor 17, 74KB, encoding 107ms, decoding 63ms. (c) JPEG 2000, compression ratio 19, 74KB, encoding 981ms, decoding 430ms. (d) DjVu, 260KB, run time unknown. (e) Our hybrid coding without intra-block segmentation, 66KB, encoding 104ms, decoding 24ms. (f) Our hybrid coding (with intra-block segmentation), 74KB, encoding 110ms, decoding 29ms.