Compound Image Compression for Real-Time Computer Screen Image Transmission

Tony Lin, *Member, IEEE,* and Pengwei Hao, *Member, IEEE*

***Abstract*—We present a compound image compression algorithm for real-time applications of computer screen image transmission. It is called shape primitive extraction and coding (SPEC). Real-time image transmission requires that the compression algorithm should not only achieve high compression ratio, but also have low complexity and provide excellent visual quality. SPEC first segments a compound image into text/graphics pixels and pictorial pixels, and then compresses the text/graphics pixels with a new lossless coding algorithm and the pictorial pixels with the standard lossy JPEG, respectively. The segmentation first classifies image blocks into picture and text/graphics blocks by thresholding the number of colors of each block, then extracts shape primitives of text/graphics from picture blocks. Dynamic color palette that tracks recent text/graphics colors is used to separate small shape primitives of text/graphics from pictorial pixels. Shape primitives are also extracted from text/graphics blocks. All shape primitives from both block types are losslessly compressed by using a combined shape-based and palette-based coding algorithm. Then, the losslessly coded bitstream is fed into a LZW coder. Experimental results show that the SPEC has very low complexity and provides visually lossless quality while keeping competitive compression ratios.**

***Index Terms*—Compound image compression, compound image segmentation, palette-based coding, shape-based coding, shape primitive extraction.**

I. Introduction

AS THE number of connected computers and other digital devices keeps growing, there has been a critical need for real-time computer screen image transmission technologies. Remote control software, such as AT&T virtual network computing (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 projector which provides the flexibility to site anywhere in the room without cable connecting to the presentation computer.

Manuscript received September 24, 2003; revised August 10, 2004. This work was supported in part by NFSC (60302005), in part by FANEDD (200038), and in part by NKBRPC (2004CB318005). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Charles D. "Chuck" Creusere.

T. Lin is with the National Laboratory on Machine Perception, Peking University, Beijing 100871, China (e-mail: lintong@cis.pku.edu.cn).

P. Hao is with the National Laboratory on Machine Perception, Peking University, Beijing 100871, China, and also with the Department of Computer Science, Queen Mary, University of London, London E1 4NS, U.K. (e-mail: phao@dcs.qmul.ac.uk).

Digital Object Identifier 10.1109/TIP.2005.849776

Data compression algorithms are essential for these real-time applications, since a huge amount of image data is to be transmitted in real time. One 800×600 true color screen image has a size of 1.44 MB, and 85 frames/second produce more than 100-MB data. Without data compression algorithms, it is definitely impossible to transmit such a large volume of data over the state-of-the-art bandwidth-limited networks in real time. Although the network bandwidth keeps growing, compression algorithms can achieve more efficient data transmission, especially for smart devices and wireless projectors.

Typically, there exist two ways to reduce the spatial and temporal redundancy in the screen image sequence. The first is to use image compression algorithms without any prior knowledge of the images. The second approach is to use high-level graphics languages and to exploit some prior knowledge provided by the operating system, such as updated regions, page layouts, and detailed drawing operations. Clearly, if the prior knowledge can be easily obtained, then text and graphics can be efficiently represented by original drawing operations, and only pictorial data need to be compressed. If the picture to be displayed has been in a compressed form, its bitstream can be directly transmitted. Thus, if the prior knowledge can be easily obtained and effectively exploited, the task of screen image compression can be perfectly fulfilled by drawing text strings, rendering graphics, and encoding and decoding natural pictures with traditional compression algorithms. However, there are two difficulties involved in the use of the second approach. One is the difficulty to obtain the useful prior knowledge from existing operating systems. Until today, there is no operating system having exposed its page layout information and detailed drawing operations, not to mention any unique reliable standard. The other is the difficulty to apply the prior knowledge to different client machines. With the difference of fonts and GUIs existing on different platforms, there is little confidence that the reconstructed screen image on the client resembles the original screen image on the server. Moreover, these drawing and rendering operations burden the client with a heavy computational load, while the client, such as a smart display or a wireless projector, commonly has very limited resources. In contrast, the first approach based on screen image compression is more reliable because of its platform independency. It is also inexpensive because of its low complexity and avoiding legal issues. We propose a hybrid algorithm which combines both approaches to achieve better performance. Updated regions, for instance, can be easily obtained from most platforms, and can be effectively used to remove a significant amount of temporal redundancies. This paper focuses on computer screen image

compression. The issue how to obtain and exploit the prior knowledge to facilitate compression is beyond the scope of this paper.

This paper is organized as follows. In Section II, we briefly review recent work on compound image compression. In Section III, we provide a detailed description of the SPEC algorithm, including system, segmentation, and coding. Experimental results are presented in Section IV. Finally, the paper is concluded in Section V.

II. Compound Image Compression

Computer screen images are mixed with text, graphics, and natural pictures. Only in the past three years have we seen the compression of computer generated images being studied. Li and Lei [3] developed a lossless compression algorithm, including intraplane coding and interplane coding. In [4], a modified JPEG-LS algorithm was proposed for lossless/near-lossless coding. VNC [1] developed a simple rectangle-based lossless coding, based on the assumption that GUI images are composed of filled rectangles. The image is divided into 16×16 blocks, and each block is represented by a list of rectangles. Raw data stream is recorded if the coded block data stream is longer. Obviously, these lossless algorithms are ineffective for natural pictures.

Another category of compound images is scanned document images, and its compression has been intensively studied in the past several years. In order to apply different compression algorithms to different image types, usually a scanned image is first segmented into different classes before compression. Layer-based and block-based algorithms are two main methods frequently used in the literature. Most layer-based approaches use the standard three-layer mixed raster content (MRC) representation [5], [6]. DjVu [7], [8] uses a wavelet-based coder (IW44) for the background and foreground, and JB2 for the mask. The segmentation is based on hierarchical color clustering and a variety of filters. DigiPaper [9] uses JPEG for the background, color tags for the foreground, and a token-based representation for the mask. Its segmentation is a complicated procedure, which involves connected components, shape cohesiveness, token comparison, etc. In [10], a layered coding method is presented for check image compression. An adaptive morphological filter is used for the segmentation.

Block-based approaches for scanned images are studied mainly due to its low complexity. In [11], a rate-distortion optimized segmentation was proposed by using block-thresholding. Cheng and Bouman [12] investigated two segmentation algorithms (TSMAP and RDOS) to classify 8×8 blocks into four classes (picture blocks, two-color blocks, one-color blocks, and other blocks). In [13], Cheng and Bouman extended this method for the application of the standard three-layer MRC format. JPEG-matched MRC compression [14] first decomposes each block into the standard three layers by using vector 2-means method, then uses JPEG for foreground and background layers and JBIG for mask layers. Li and Lei [15] proposed a histogram analysis to classify each 16×16 block into one of the four types: smooth block (one color), text block (two color), graphics block (four color), and image block (wavelet-based coding). GRAFIT [16] classifies 8×8 blocks into four modes and use different coding methods for each mode.

For real-time computer screen image transmission, the compression algorithm should not only achieve high compression ratios, but also have low complexity and visually lossless quality. Low complexity is very important for real-time compression, especially on smart displays and wireless projectors. On the other hand, poor image quality reduces the readability of the text and results in unfavorable user experience.

Scanned image compression algorithms can not be directly applied to the real-time compression of computer screen images, due to following significant differences between scanned images and computer screen images.

A. Source

Scanned images are captured by an electronic imaging procedure, whereas computer screen images are essentially synthetic images. Photographic image compression algorithms, such as JPEG or JPEG-2000, are still applicable to scanned images, and their performance can be improved by adopting different qualities for text/graphics and pictures. In fact, most scanned image compression algorithms use JPEG for background and foreground layers, and use JBIG2 for mask layers. Ringing artifacts caused by DCT or wavelet transform are not clearly visible around text/graphics edges, because these edges have been blurred in printing or scanning procedures. However, for computer screen images, these ringing artifacts are easily noticeable due to the sharp edges of text/graphics.

B. Spatial Resolution

Scanned images typically have higher spatial resolution than computer screen images. The minimum acceptable quality for scanned images is 300 dpi, whereas, for screen images, it is less than 100 dpi. Block-based approaches work well for scanned images, but cause severe artifacts for computer screen images. Any tiny alteration to periods, such as "i" dots and thin lines, can make the computer screen image barely readable.

C. Noise Level

Scanned images invariably contain some amount of noise, while computer screen images are free of noise. Therefore, for computer screen images, any noise introduced in compression is noticeable in text/graphics regions.

D. Computational Complexity

Real-time compression algorithms require very low complexity, whereas scanned image compression does not have such a requirement.

In this paper, we propose a low complexity and high quality compression algorithm—shape primitive extraction and coding (SPEC). SPEC accurately segments text/graphics from pictures, and provides a new lossless coding method for text/graphics. SPEC has two unique features.

1) Shape and Color: In the area of content-based image retrieval, image contents often refer to color, texture, shape, and

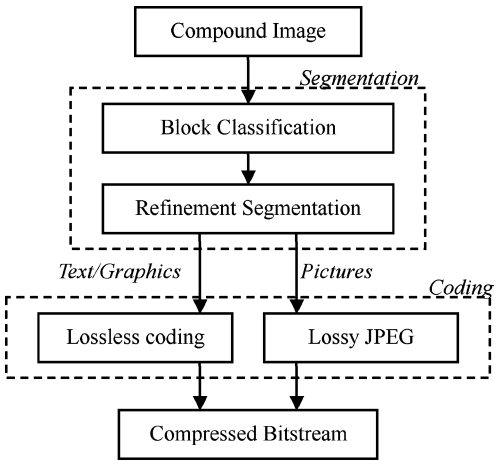


Fig. 1. Flow chart of the SPEC system.

motion (only for video). In SPEC, shape and color serve as two basic features to effectively separate text/graphics from natural pictures. Image blocks are classified into picture blocks and text/graphics blocks by thresholding the number of colors of each block, and then shape primitives of text and graphics are extracted. In addition, shape-based coding and palette-based coding are combined to encode text and graphics losslessly.

2) Segmentation and Coding: The segmentation extracts text and graphics pixels as shape primitives, and these shape primitives are exploited in the coding stage. Unlike other compound image compression algorithms, segmentation and coding are tightly integrated in the SPEC algorithm.

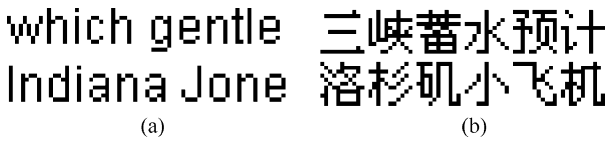
III. SPEC—The Proposed Algorithm

A. System

As shown in Fig. 1, the proposed SPEC algorithm consists of two stages: segmentation and coding. The algorithm first segments 16×16 nonoverlapping blocks of pixels into text/graphics class and picture class, and then compresses text/graphics with a new lossless coding algorithm and pictures with lossy JPEG, respectively. Finally, the lossless coded data and the JPEG data are put into one bitstream.

There are several reasons for choosing a 16×16 block size. In a 16×16 block, a pixel location (x, y) can be represented by 4-bit x and 4-bit y, totally just one byte. Similarly, the width and the height of a rectangle in such a block can be represented by 4-bit w and 4-bit h. In practice, this block size achieves a reasonable tradeoff for computer screen images. Additionally, it is easy for JPEG to compress such a block, if the chrominance subsampling is applied.

SPEC separates image into two classes of pixels: text/graphics and pictures. In block-based approaches, there are generally four types of blocks: smooth background blocks (one color), text blocks (two color), graphics blocks (four color), and picture blocks (more than four colors). In fact, the first three types can be grouped into a larger text/graphics class, which greatly simplifies the segmentation. More importantly, the combined text/graphics class can be coded by a lossless method.



(a) (b)

Fig. 2. Text samples from two webpage images.

Shape primitives refer to those elementary building units that compose text/graphics in a compound image, such as dots, lines, curves, triangles, rectangles, and others. The concept of shape primitives is inspired by the VNC algorithm [1], in which only rectangles are used to represent an image. Each rectangle can be represented by its position information (x, y, w, h) and a color tag. However, this representation is redundant for dots and lines. For simplicity, only four types of shape primitives are used in SPEC: isolated pixels, horizontal lines (one pixel high), vertical lines (one pixel wide), and rectangles (with horizontal and vertical sides). A shape primitive is defined to have the same interior color. It is possible that two shape primitives have the same shape but different colors. Straight forward, a shape primitive can be represented by a color tag and its position information, i.e., (x, y) is for an isolated pixel, (x, y, w) for a horizontal line, (x, y, h) for a vertical line, and (x, y, w, h) for a rectangle. Some English and Chinese characters in small fonts are shown in Fig. 2. These characters are mainly composed of vertical and horizontal lines. We can see that shape primitives can be used to compactly represent the textual contents.

To encode pixels of text and graphics, a simple lossless coding is designed to utilize the information of the extracted shape primitives. Shape primitives can be efficiently encoded with a shape-based coding, and other techniques like palette-based coding, color table reuse, and LZW are also integrated.

There are two reasons that we use JPEG instead of the state-of-the-art algorithm JPEG-2000 to encode pictorial pixels. On one hand, as the algorithm is designed for real-time compression, speed is the primary consideration. DCT-based JPEG is several times faster than wavelet-based JPEG-2000. On the other hand, JPEG is a block-based algorithm which is compatible with our block-based technique.

The block diagram of the detailed SPEC algorithm is shown in Fig. 3, and the details of segmentation and coding are described in the following subsections.

B. Segmentation

The segmentation is a two-step procedure, including block classification and refinement segmentation. The first step is to classify 16×16 nonoverlapping blocks into text/graphics blocks and picture blocks by thresholding the number of colors in each block. Each block is scanned to count the number of different colors. If the color number is more than a certain threshold T_1 ($T_1 = 32$ is used for SPEC), the block is classified as picture block. Otherwise, it is classified as text/graphics block. At the same time, 1-byte index data is generated for text/graphics blocks. This reduces time to encode these blocks. The underlying reason of thresholding the number of different colors is that continuous-tone pictures generally have a large number of different colors even in a small region, while text

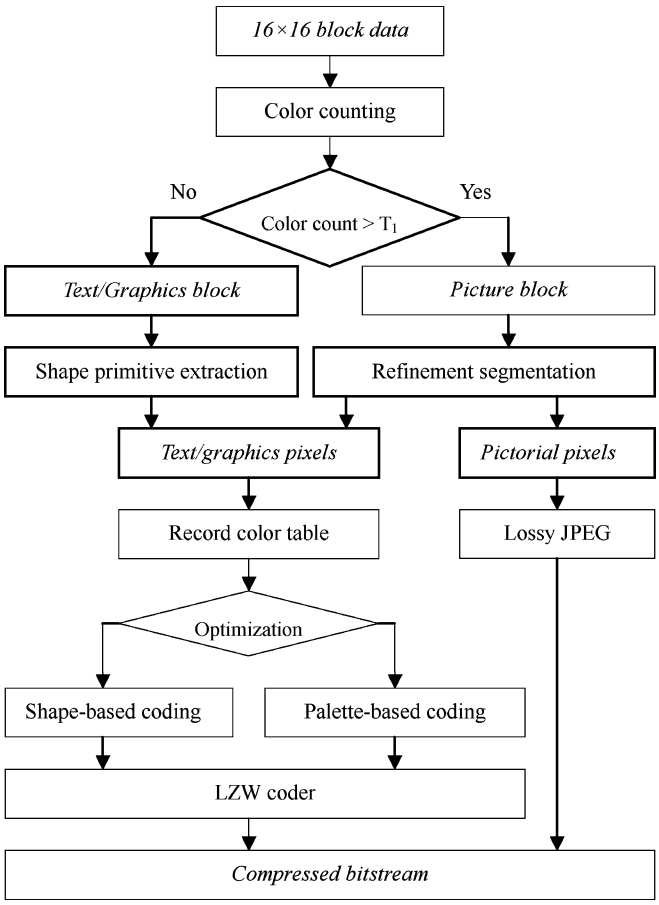


Fig. 3. Block diagram of the SPEC compression algorithm.

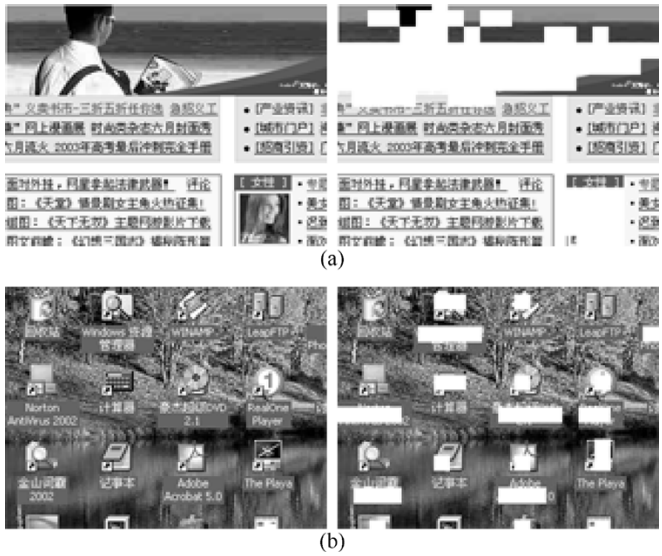


Fig. 4. Coarse block segmentation with $T_1 = 32$. (a) A portion of the webpage *web1*, and its text/graphics block image. (b) A portion of the wallpaper *wall1*, and its picture block image.

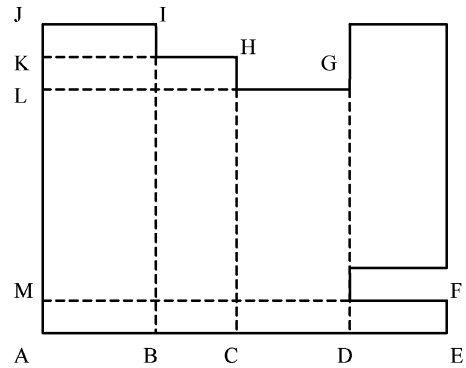


Fig. 5. Shape primitive extraction. A region in one color can be decomposed into a set of shape primitives, but there are many ways to do this decomposition. For simplicity, only ABIJ and AEFM are compared. The winner is ABIJ because it has a larger size than AEFM.

or graphics only have a small number of colors even in a large region. The block classification based on counting different colors can be extremely fast. Typical webpage images can be done within 40 ms, and wallpaper images can be done within 20 ms. Fig. 4(a) shows a portion of the webpage *web1*, and its text/graphics block image. Fig. 4(b) shows a portion of the wallpaper *wall1*, and its picture block image. Most blocks are correctly classified, except for those on the boundary of text/graphics and pictures.

The above block classification is a coarse segmentation, because classified picture blocks may contain text or graphics pixels, and text/graphics blocks also may contain pictorial pixels. Therefore, a refinement segmentation is followed to extract text and graphics pixels from picture blocks to enhance the results. Pictorial pixels in text/graphics blocks are not segmented for two reasons. First, with a proper color number threshold, the amount of pictorial pixels in text/graphics block can be relatively small; thus, these pixels can be efficiently coded with lossless methods. Second, for images with large regions of text and graphics, the coarse segmentation is computationally efficient. If the refinement segmentation is applied to all blocks, it can be very time-consuming.

The procedure of extracting shape primitives of text/graphics in a picture block is as follows. This procedure is similar to the rectangle decomposition procedure in VNC. Each picture block is scanned from left to right and from bottom to top, started from left-bottom pixel. If the current pixel has been included in a previously extracted shape primitive, the scanning procedure skips it and goes to the next pixel. Otherwise, the current pixel is set as a beginning point, and the scanning procedure then searches rightward and upward to extract shape primitives. There may be a rectangle, a horizontal line, a vertical line, or at least an isolated pixel. An irregular region is represented by a list of the four types of shape primitives, and the representation is not unique. Fig. 5 shows such a region, where four rectangles, i.e., AEFM, ADGL, ACHK, and ABIJ, are found in one color, all starting from point A. To meet the speed requirement for our real-time transmission, we only compare the widest horizontal rectangle (or horizontal line) and the highest vertical rectangle (or vertical line). A size-first strategy is used to accomplish this task. For the case of Fig. 5, only the vertical rectangle ABIJ is extracted because it has a larger size than the horizontal rectangle AEFM, and AEFM does not exist after ABIJ has been extracted. It is difficult to find the rectangle ADGL, though it has the largest size. To find the horizontal rectangle AEFM, we scan the row AE to get the width, and the rows above for the

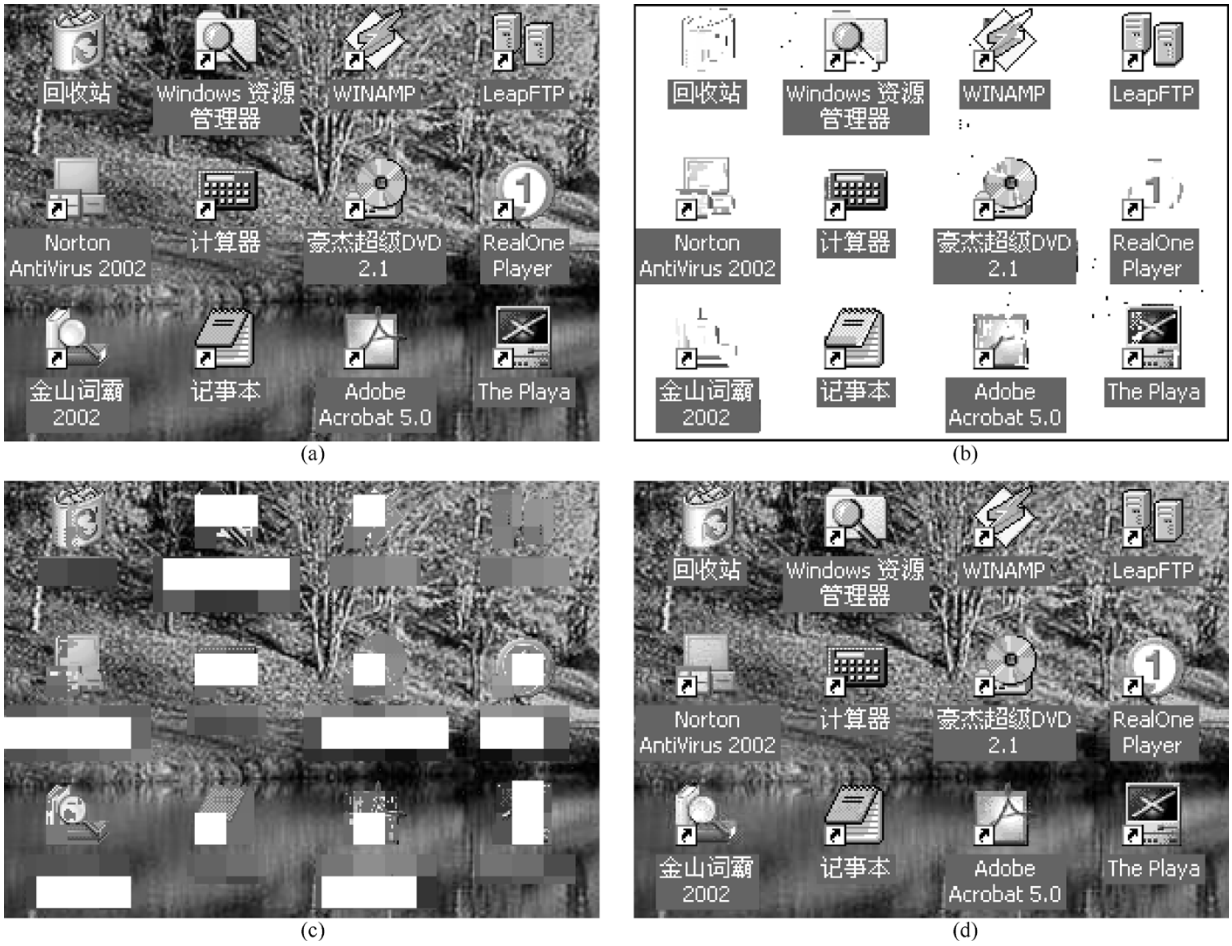


Fig. 6. Segmentation and reconstruction results of the wallpaper image *wall1*. (a) A portion of the original image. (b) Foreground layer of text/graphics pixels. (c) Background layer of pictorial pixels, with holes filled with average colors of pictorial pixels in the block. White color represents text/graphics blocks. (d) Reconstructed image (12.6:1 compression).

height. Similarly, to find the vertical rectangle ABIJ, we scan the column AJ to obtain the height, and the rows between AJ for the width.

Apparently, we must refine the shape primitives extracted in above procedure to separate text/graphics pixels from pictorial pixels. Because shape primitives include isolated pixels, every pixel in picture blocks may be misclassified into shape primitive pixels. Moreover, for monotonous regions, such as the blue sky in Fig. 10, several adjoining pixels may have the same color.

Whether a shape primitive is classified into a text/graphics class depends on its size and color. If its size is larger than some threshold T_2 ($T_2 = 3$ is used for SPEC), the shape primitive is extracted as text/graphics pixels. Otherwise, the color of this shape primitive is compared to a dynamic palette of recent text/graphics colors. If an exact color match is found, the shape primitive is taken as text/graphics. If a shape primitive has a size larger than a threshold T_3 ($T_3 = 5$ for SPEC), its color is put into the dynamic color palette. The dynamic color palette is implemented with a first-in first-out buffer. Because color matching is frequently applied, we maintain the dynamic palette with a small size for computational efficiency. In SPEC, there are eight entries in the dynamic palette, and pure black and pure white are considered as default text/graphics colors

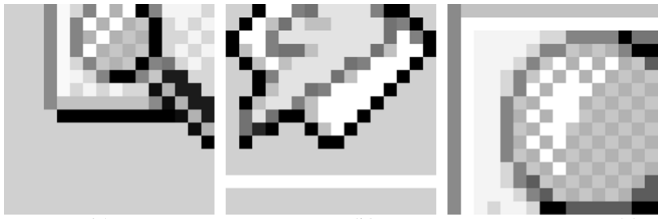
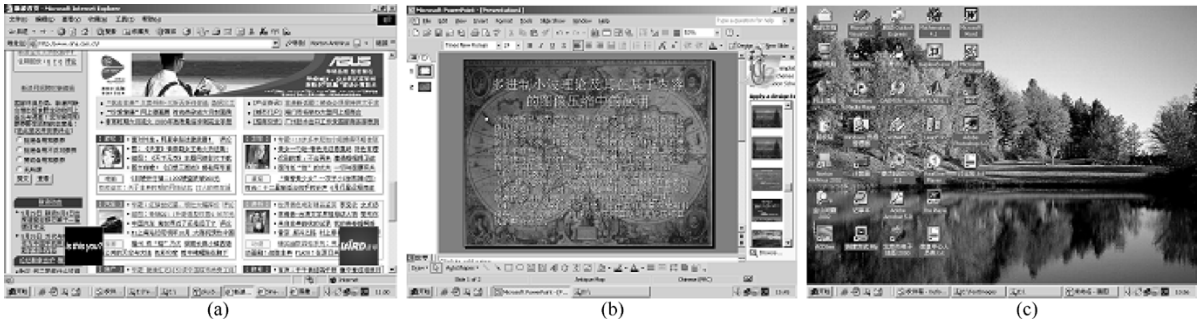


Fig. 7. Lossless text/graphics coding. Three methods are compared to generate minimal coding length: 1) all colors are shape-based coded, 2) all colors are palette-based coded, 3) the most shape-complicated color is palette-based coded, while other colors are shape-based coded. (a) 1) 89 bytes; 2) 128 bytes; 3) 104 bytes. (b) 1) 97 bytes; 2) 96 bytes; 3) 105 bytes. (c) 1) 139 bytes; 2) 128 bytes; 3) 122 bytes, where the cyan color is 1-bit palette-based coded.

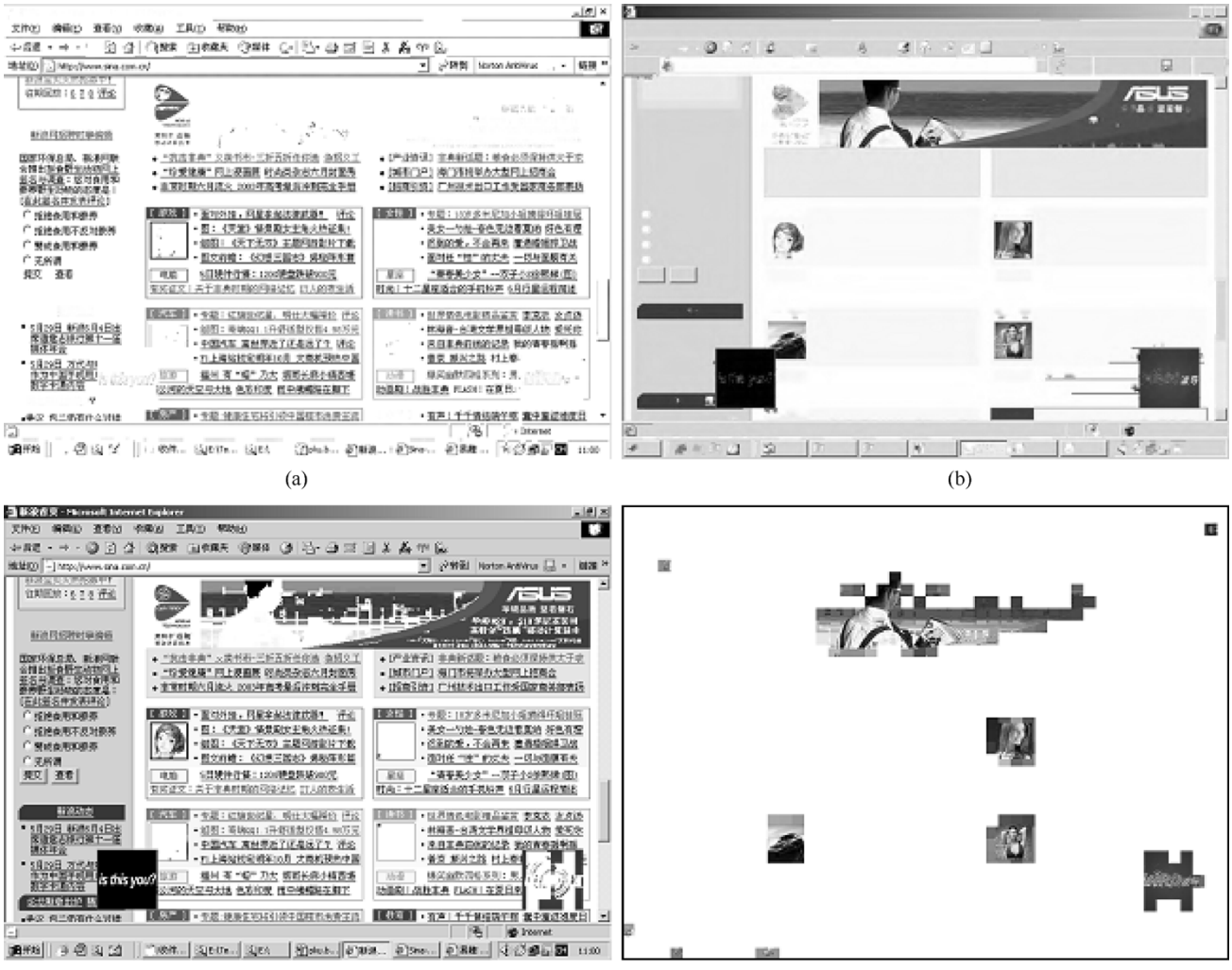
for computer screen images. In other words, if a shape primitive has pure black or pure white color, it is directly classified into text/graphics.

There are several reasons for designing such a procedure to detect shape primitives of text and graphics. First, most pictorial pixels are found to be isolated pixels, because there is little possibility that several neighboring pictorial pixels have exactly the same color. Even if this happens, the size of these pictorial pixels is usually small. Moreover, if the neighboring pictorial pixels have a large size, they are classified into text/graphics



(a) (b) (c)

Fig. 8. Test images. (a) *web1*. (b) *ppt1*. (c) *wall1*.



(a) (b) (c) (d)

Fig. 9. Segmentation results of *web1*. (a) Foreground layer separated by DjVu. (b) Background layer separated by DjVu. (c) Foreground layer separated by SPEC. (d) Background layer separated by SPEC.

and can be still efficiently coded by lossless coding. Second, for shape primitives of small sizes, we can make decision based on most recent colors of text/graphics pixels. In document images or computer-generated images, the color of textual and graphical content generally has some coherence, and it is unusual that text/graphics colors change frequently. Finally, this procedure is computationally efficient.

This two step segmentation successfully segment the image into two parts—text/graphics pixels and pictorial pixels, where text/graphics pixels include all pixels of text/graphics blocks and shape primitive pixels, and pictorial pixels are the remaining pixels in picture blocks. Fig. 6(b) shows segmented text/graphics pixels for Fig. 6(a), a portion of the wallpaper *wall1*. The remaining pictorial pixels are shown in Fig. 6(c). Most of the icon pixels are correctly classified into text/graphics. It is difficult to detect the small portion of misclassified icon pixels, because their complicated color patterns are very similar to ground-truth pictorial pixels.

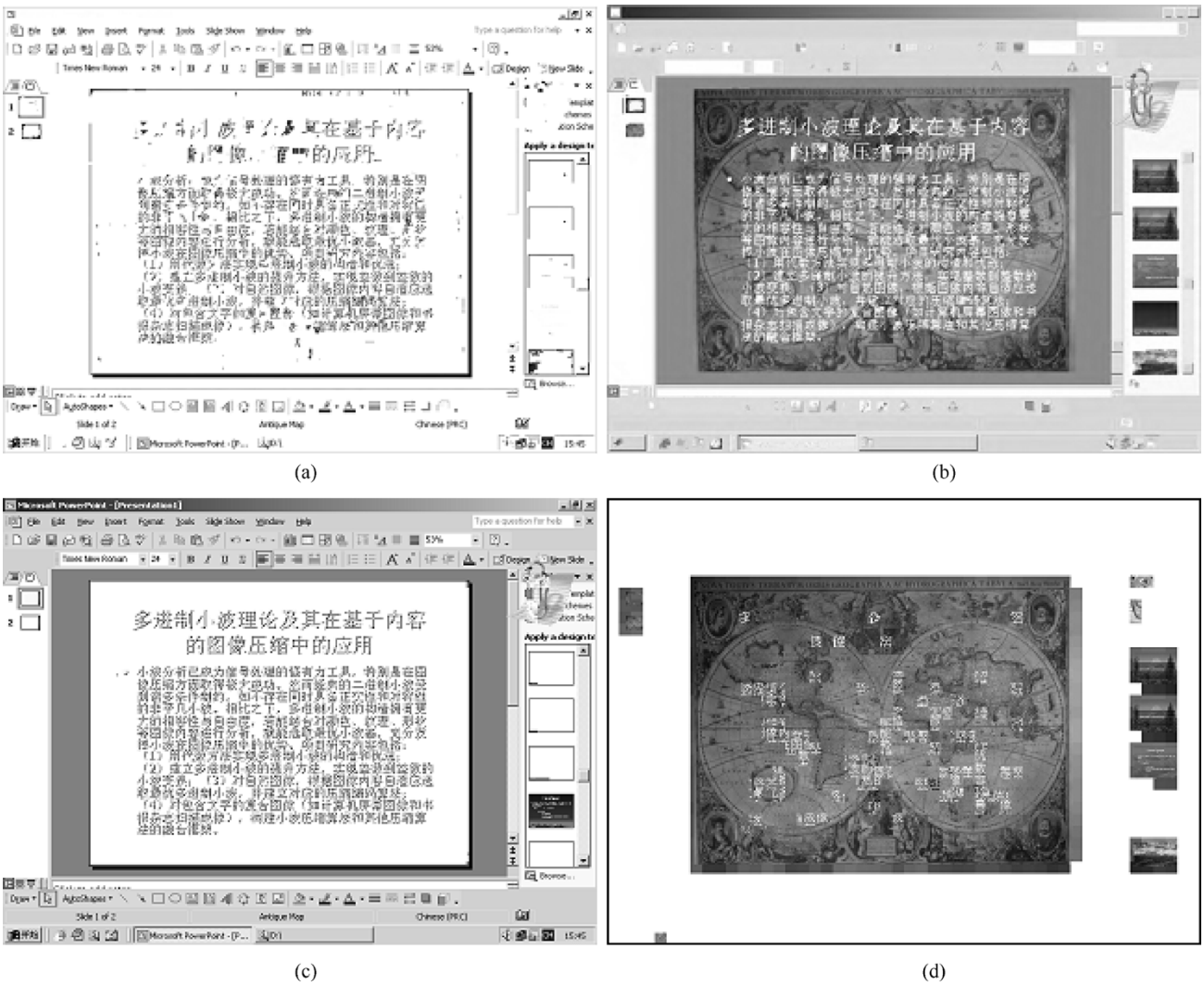


Fig. 10. Segmentation results of *ppt1*. (a) Foreground layer separated by DjVu. (b) Background layer separated by DjVu. (c) Foreground layer separated by SPEC. (d) Background layer separated by SPEC.

C. Lossless Coding of Text/Graphics Pixels

Our lossless coding of text/graphics pixels is mainly based on shape primitives, which creates a compact representation of shape primitives of text and graphics. To compress text/graphics blocks, shape primitives are extracted firstly. The extraction procedure is similar to that of picture blocks. In addition, in block classification we can find the color with the largest size, and this color is recorded as background color. For a text/graphics block, the background color is usually segmented into interior text regions and graphics regions. The background color is recorded, and, thus, the coding of those background color pixels can be skipped. The shape primitives in other colors are extracted from text/graphics blocks, and all shape primitives extracted from text/graphics blocks and picture blocks are losslessly coded.

A simple shape-based coding is used to represent shape primitives. In a 16×16 block, 8-bit (x, y), 12-bit (x, y, w), 12-bit (x, y, h), and 16-bit (x, y, w, h) are used to represent isolated pixels, horizontal lines, vertical lines, and rectangles, respectively. For each color, we use a run-length encoding scheme to represent the counts of four types of shape primitives.

Sometimes there are too many small shape primitives in a complicated block. This makes shape-based coding inefficient. Therefore, palette-based coding will be a good alternative. Consider a two-color 16×16 block with color white as background and 100 isolated black pixels, shape-based coding needs 100 bytes to represent these isolated pixels, while palette-based coding only needs 32 bytes to represent a 1-bit mask. For multiple colors, palette-based coding uses a multiple-bit mask. We choose one from the following three cases to achieve the minimal code length: 1) all colors are shape-based coded, 2) all colors are palette-based coded, 3) only the most shape-complicated color is palette-based coded, while other colors are shape-based coded. The most shape-complicated color is the color that generates the maximum coding length when all colors are shape-based coded.

Fig. 7 illustrates examples for the three cases. The block in Fig. 7(a) is shape-based coded, and the block in Fig. 7(b) is palette-based coded. The block in Fig. 7(c) is coded by a combination of palette-based coding and shape-based coding. The cyan color is the most shape-complicated color with 49 isolated pixels. Using 1-bit palette-based coding, the cyan color is coded

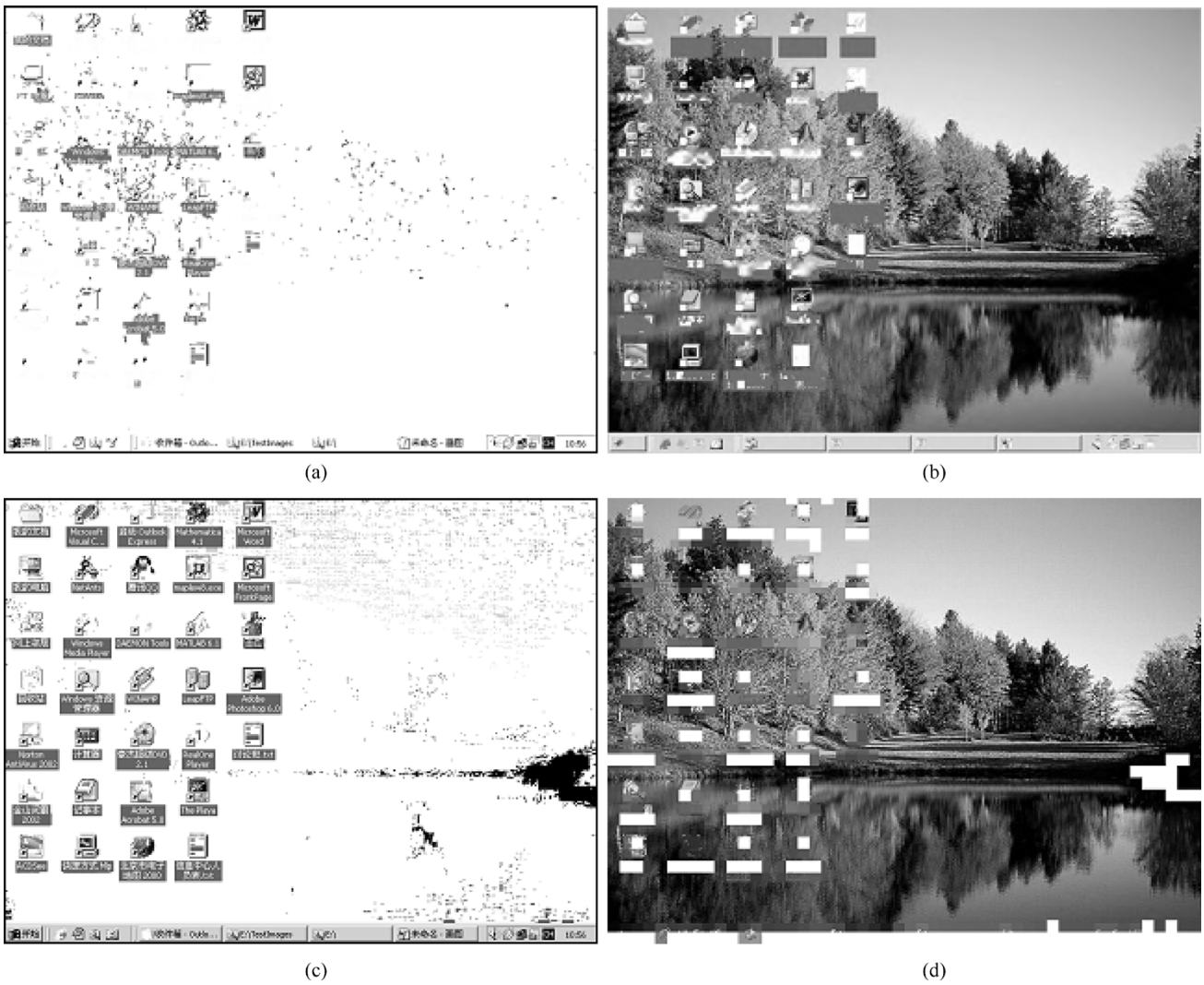


Fig. 11. Segmentation results of *wall1*. (a) Foreground layer separated by DjVu. (b) Background layer separated by DjVu. (c) Foreground layer separated by SPEC. (d) Background layer separated by SPEC.

in 32 bytes, while shape-based coding needs 49 bytes to encode the same information.

We apply a color table reuse technique to represent the colors of shape primitives. Generally, most colors are the same in the color tables of two consecutive blocks. If a color of the current block is found to match the color table of the previous block, it is represented by a 1-byte index. Otherwise, it is represented by 3-byte (R, G, B) format. Though we can construct a global dictionary of colors, it is difficult to maintain the global dictionary. Color matching is time-consuming in a large global dictionary and, therefore, is not suitable for real time applications.

For each block, the lossless coding stream is organized as following. First, the color number and the color table are recorded. Second, the encoding method is specified by the number of colors being shape coded, the number of colors being palette-based coded. If there is a background color, we record its index in the block color table, and if there is a color being coded by 1-bit palette, we record its index, too. Then, shape primitives in each color are represented by a combined shape-based and palette-based coding algorithm. Finally, the above coded stream is fed into a LZW coder, zlib [17], for further compression.

D. JPEG Coding of Pictorial Pixels

SPEC compresses pictorial pixels in picture blocks using a simple JPEG coder [18]. In order to reduce ringing artifacts and to achieve higher compression ratio, text/graphics pixels in the picture block are removed before the JPEG coding. These pixels are coded by the lossless algorithm. Actually, their values can be arbitrarily chosen, but it would be better if these values are similar to the neighbor pictorial pixels. This produces a smooth picture block. We, therefore, fill in these holes with the average color of pictorial pixels in the block.

IV. Experimental Results

For our experiments, ten 800×600 true color computer screen images are tested, including four webpages (*web1*, *web2*, *web3*, and *web4*), four PowerPoint images (*ppt1*, *ppt2*, *ppt3*, and *ppt4*), and two wallpaper images (*wall1* and *wall2*). There are a large number of Chinese characters in *web1*, *web2*, *ppt1*, and *ppt2*. The PowerPoint and the wallpaper images are very challenging to compress because English or Chinese characters are directly drawn on background pictures. Three test images (*web1*, *ppt1*, and *wall1*) are shown in Fig. 8.

TABLE I
LOSSLESSLY COMPRESSED FILE SIZES (KILOBYTES)/ENCODING TIMES IN MILLISECONDS FOR SEGMENTED FOREGROUND IMAGES

Image	SPEC	LZW	JPEG-LS
web1	62/108	61/220	269/16
web2	64/111	65/221	263/16
web3	63/104	57/187	202/15
web4	83/130	65/196	274/18
ppt1	46/86	50/219	158/13
ppt2	75/125	84/232	272/17
ppt3	81/120	86/231	228/15
ppt4	49/91	48/188	150/12
wall1	39/70	47/190	137/11
wall2	26/54	32/175	106/10

The original file size is 1407 KB.

TABLE II
LOSSLESSLY COMPRESSED FILE SIZES (KILOBYTES) OF EIGHT BI-LEVEL DOCUMENT IMAGES

Image	SPEC	LZW	PNG	MH	MMR	JBIG	LDF	DjVu*
chn1	20	16	16	26	21	12	11	7
chn2	16	14	14	22	16	9	9	6
chn3	17	14	14	23	17	10	9	6
chn4	19	14	15	25	19	10	10	6
eng1	19	14	14	23	17	9	9	7
eng2	26	19	19	32	26	12	12	8
eng3	24	17	17	29	24	10	10	6
eng4	20	14	15	27	20	9	9	6

The uncompressed 1-bit file size is 59 KB. *DjVu may be lossy.

TABLE III
LOSSLESSLY COMPRESSED FILE SIZES (KILOBYTES) OF NINE CCITT BI-LEVEL IMAGES

Image	SPEC	LZW	PNG	MH	MMR	JBIG	LDF	DjVu
1	29	28	27	44	20	13	13	10
2	22	27	24	43	13	8	8	9
3	44	42	39	78	29	20	20	17
4	110	89	89	121	69	48	48	29
5	52	50	47	79	33	23	23	18
6	27	32	28	73	19	11	12	13
7	101	106	103	140	74	51	51	38
8	36	41	39	72	21	13	13	15
10	107	79	77	145	137	62	62	58

The uncompressed 1-bit file size is 496 KB.

TABLE IV
LOSSLESSLY ENCODING/DECODING TIMES IN MILLISECONDS FOR NINE CCITT BI-LEVEL IMAGES

Image	SPEC	JBIG	JBIG2 (MQ)
1	118/14	770/710	841/1092
2	123/13	1260/1210	981/1092
3	134/25	1210/1100	1021/1101
4	190/52	1260/1210	911/1132
5	141/27	1270/1210	1012/1112
6	122/18	1260/1150	1002/1081
7	140/44	1430/1320	932/1142
8	148/29	1320/1210	832/1102
10	245/49	1090/940	931/1142

The last two columns of data from [20] are tested on a 500 MHz P3 machine.

TABLE V
LOSSLESSLY ENCODINGLOSSLESSLY COMPRESSED FILE SIZES (KILOBYTES) OF TEN COMPOUND IMAGES

Image	SPEC*	VNC	LZW
web1	100	264	111
web2	116	285	110
web3	93	217	91
web4	167	459	212
ppt1	285	711	509
ppt2	205	566	286
ppt3	187	558	203
ppt4	210	556	385
wall1	471	1303	984
wall2	424	1285	1073

The original file size is 1407 KB. *The lossless SPEC uses integer reversible DCT and a quantization table of all 1's.

A. Segmentation

The three images in Fig. 8 are segmented by SPEC and DjVu (DjVu Shop 2.0) for comparison. The segmentation results are shown in Figs. 9–11. From Fig. 9, we notice that both algorithms correctly classify text into foreground layers. The difference is that DjVu tends to classify most of the graphical windows and the background colors into the background layer, whereas SPEC only classify the complicated pictorial regions as the background. It implies that SPEC can achieve higher fidelity in text/graphics regions than DjVu. The segmented image in Fig. 9 is Fig. 8(a), which has a simple layout. For complex images such as Fig. 8(b) and (c), SPEC clearly out-performs DjVu by classifying most of the texts and graphical contents correctly. Compared with DjVu, this better segmentation directly attributes to higher quality of SPEC compressed images. It can be seen from Fig. 11(c) that there are some monotonous pictorial regions in the wallpaper misclassified as foreground by SPEC. However, these misclassified regions can still be efficiently coded by our lossless coding method.

B. Lossless Coding

First, we compare our lossless coding with LZW and JPEG-LS for segmented foreground layers of the ten test images. Table I shows the compressed file sizes and encoding times. We can see that JPEG-LS spends the least encoding times but achieves very poor compression ratios. SPEC and LZW provide comparable compressions, but SPEC only spends half of the encoding time that LZW needs. In comparison, SPEC achieves a better trade-off between compression ratio and encoding time.

Second, we test the lossless coding performance for bi-level images. The SPEC algorithm is adapted to bi-level image inputs, and it is compared with LZW, PNG, MH (CCITT G3), MMR (CCITT G4), JBIG, and JBIG2 (LuraDocument LDF format, and DjVu JB2). Two sets of bi-level images are tested. The first set contains eight 800×600 computer screen images (text only). The second set includes nine CCITT scanned document images (available from [20]).

Compressed file sizes with the first test set are given in Table II. SPEC and MMR achieve similar compression ratios. LZW and PNG perform better than MH, MMR, and SPEC. DjVu JB2 achieves the maximum compression ratios, but it cannot guarantee lossless coding (even under the lossless option). In comparison with other lossless algorithms, JBIG and LDF perform best. The average encoding and decoding times of SPEC are 24 and 7 ms, respectively. The coding times of other algorithms are not available.

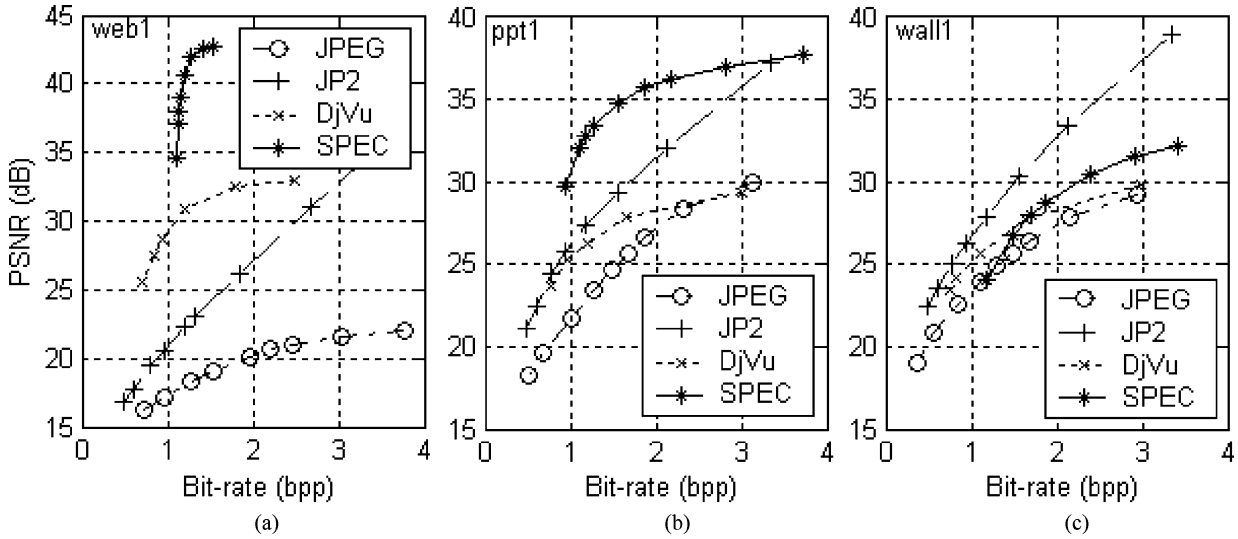


Fig. 12. PSNR plots for JPEG, JPEG 2000, DjVu, and SPEC. (a) *web1*. (b) *ppt1*. (c) *wall1*.

Fig. 13. Compression results of *web2*. (a) A portion of the original image. (b) JPEG, 127 KB (11.3:1 compression). (c) JPEG-2000, 73 KB (19.4:1 compression). (d) DjVu, 84 KB (16.8:1 compression). (e) SPEC, 73 KB (19.4:1 compression).

Compression file sizes of nine CCITT images are given in Table III. (Some results can be found on the websites of [20] and [21] with slight difference). SPEC, LZW, and PNG achieve similar compression ratios, which are far better than MH. MMR performs better than SPEC, LZW, and PNG. In comparison, JBIG, LDF, and DjVu give expected best results. Table IV lists the en-

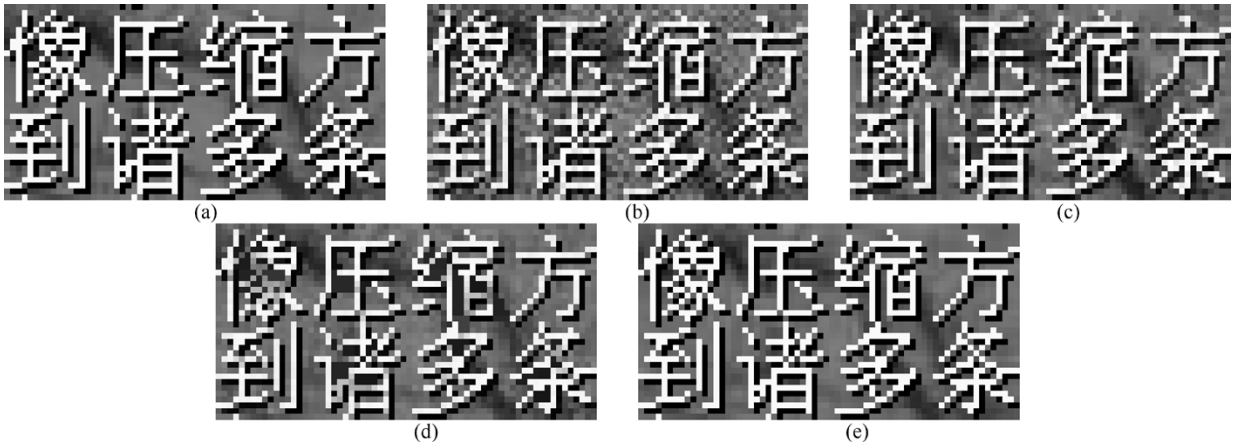


Fig. 14. Compression results of *ppt1*. (a) A portion of the original image. (b) JPEG, 100 KB (14.1:1 compression). (c) JPEG-2000, 77 KB (18.3:1 compression). (d) DjVu, 99 KB (14.2:1 compression). (e) SPEC, 77 KB (18.3:1 compression).

Fig. 15. Compression results of *wall1*. (a) A portion of the original image. (b) JPEG, 90 KB (15.9:1 compression). (c) JPEG-2000, 112 KB (12.6:1 compression). (d) DjVu, 97 KB (14.5:1 compression). (e) SPEC, 112 KB (12.6:1 compression).

coding and decoding times of SPEC, JBIG, and JBIG2, showing that SPEC needs the least encoding and the least decoding time.

C. Computer Screen Image Compression

The proposed SPEC algorithm is compared with JPEG (IJG JPEG 6b), JPEG 2000 (Jasper 1.6 [19]), DjVu (DjVu Shop 2.0), and two lossless algorithms (VNC [1] and LZW zlib [17]) on a 1.3-GHz PM machine. In the settings of DjVu shop, we set the document type as "color document." The lossless mask and no character thickening are chosen for higher quality.

Compressed file sizes of VNC, LZW, and lossless SPEC (using integer reversible DCT and a quantization table of all 1s) are given in Table V. VNC achieves very poor compression ratios. The lossless SPEC clearly out-performs LZW, especially for wallpaper and PowerPoint images.

PSNR plots for JPEG, JPEG 2000, DjVu, and SPEC are shown in Fig. 12. For the image *web1*, SPEC and DjVu clearly out-perform traditional algorithms such as JPEG and JPEG 2000. SPEC provides 5- to 10-dB improvement over DjVu and more than 20 dB over JPEG. Notice that the curve for SPEC only varies from 1 to 1.5 dB, since most of the image

TABLE VI
ENCODING/DECODING TIMES IN MILLISECONDS FOR TEN COMPOUND IMAGES

Image	SPEC	JPEG	JP2	VNC	LZW
web1	124/29	124/71	1042/150	42/20	223/23
web2	128/31	122/70	1002/150	44/21	225/24
web3	114/25	111/63	841/150	37/17	190/24
web4	162/42	111/64	921/150	49/26	224/28
ppt1	172/75	117/67	1021/150	55/18	282/39
ppt2	204/61	110/63	921/150	56/24	274/35
ppt3	182/55	108/61	891/150	52/18	274/29
ppt4	160/61	109/62	871/150	49/16	262/35
wall1	266/129	116/66	1032/150	79/14	336/58
wall2	247/126	115/67	1031/150	76/15	334/60

TABLE VII
ORIGINAL/COMPRESSED DATA SIZES (KILOBYTES) FOR THE CODING METHODS IN SPEC

Image	Shape	Palette	Color	RLE	LZW	JPEG
web1	1255/65	43/5	29/19	37/28	98/63	112/6
web2	1229/64	60/8	30/19	37/29	100/65	131/8
web3	1140/68	12/1	32/20	43/32	98/64	65/5
web4	1009/82	101/13	43/27	55/35	131/83	210/15
ppt1	717/47	59/7	16/10	21/17	73/45	685/32
ppt2	778/69	74/9	30/24	39/32	110/74	493/15
ppt3	769/67	86/15	34/24	37/29	114/81	414/12
ppt4	765/52	25/3	23/16	30/24	79/49	513/23
wall1	276/39	16/2	14/12	19/16	61/42	1350/70
wall2	271/28	22/2	9/7	12/10	45/30	1349/59

Shape: shape-based coding; Palette: palette-based coding; Color: color table reuse; RLE: run-length encoding.

is losslessly coded. For the image *ppt1*, SPEC still serves as an upper bound. At high bit rates, the performance of JPEG 2000 is very close to that of SPEC. DjVu is better than JPEG, but inferior to JPEG 2000. For the image *wall1*, JPEG 2000 is the best. At low bit rates, DjVu is better than SPEC. But SPEC surpasses DjVu at high bit rates. Both DjVu and SPEC achieve 1 to 2 dB better than JPEG.

Figs. 13–15 compare the quality of the reconstructed images compressed by SPEC with those by JPEG, JPEG 2000, and DjVu at similar compression ratios. In terms of reconstructed image quality, DjVu and SPEC perform much better than JPEG and JPEG 2000, because JPEG and JPEG 2000 usually result in annoying ringing artifacts around the edges of text characters. Sometimes DjVu gives an impression that the text is blurred, as characters filled with background picture colors. There are also some missing and false pixels in DjVu coded characters due to misclassification. For the two icons in Fig. 15, both DjVu and SPEC achieve acceptable image quality, but some segmentation errors are noticeable. Artifacts around the icons are less visible in the DjVu coded image, because DjVu uses wavelet-based coding which has better smoothing effects.

Table VI lists the encoding and decoding times of the tested algorithms. Typically, SPEC and JPEG spend 100 or 200 ms to encode each test image, but JPEG 2000 needs around one second. We cannot obtain the exact encoding/decoding time of DjVu. Empirically, DjVu takes about 10 s to encode each image.

Table VII lists the detailed compression results for the coding methods adopted in SPEC. Shape-based coding can achieve a compression ratio from 10:1 to 20:1 for original pixel data. Palette-based coding plays a secondary role by offering a small coding gain. A savings between 30% and 40% is achieved by LZW. Generally, JPEG achieves 10:1 to 30:1 compression ratios. For typical 800×600 computer screen images, the SPEC compressed files with acceptable quality have sizes less than 100 KB.

We conclude that: 1) for compound image compression, SPEC outperforms traditional compression algorithms such as JPEG and JPEG-2000; 2) SPEC achieves lower complexity and higher quality than DjVu; and 3) the coding efficiency of SPEC can be significantly improved if more sophisticated lossless coding methods are employed.

V. CONCLUSION

We presented a compound image compression algorithm, SPEC, for real-time computer screen image transmission. Two main contributions of this work are: 1) an accurate segmentation algorithm is developed to separate text/graphics from pictures and 2) a lossless coding method is designed for text/graphics compression. Experimental results demonstrate that SPEC is an algorithm of low complexity. It also provides excellent visual quality and competitive compression ratio. Our future work is to improve the accuracy of the segmentation and efficiency of the lossless coding. It is also possible to modify SPEC for compression of scanned document images.

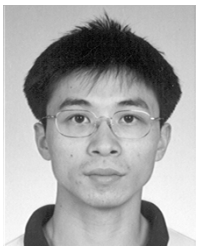
ACKNOWLEDGMENT

The authors would like to thank X. Tang for many stimulating discussions and for proofreading the paper, as well as the anonymous reviewers for their constructive remarks.

REFERENCES

[1] [Online]. Available: http://www.uk.research.att.com/vnc/index.html
[2] [Online]. Available: http://www.microsoft.com/windows/smartdisplay/
[3] X. Li and S. Lei, "On the study of lossless compression of computer generated compound images," in *Proc. Int. Conf. Image Processing*, vol. 3, 2001, pp. 446–449.
[4] F. Ono, I. Ueno, T. Takahashi, and T. Semasa, "Efficient coding of computer generated images with acceptable picture quality," in *Proc. Int. Conf. Image Processing*, vol. 2, 2002, pp. 653–656.
[5] "Mixed raster content (MRC)," ITU-T Study Group 8, Draft Recommendation T.44, 1997.
[6] R. de Queiroz, R. Buckley, and M. Xu, "Mixed raster content (MRC) model for compound image compression," *Proc. SPIE*, vol. 3653, pp. 1106–1117, 1999.
[7] L. Bottou, P. Haffner, P. G. Howard, P. Simard, Y. Bengio, and Y. LeCun, "High quality document image compression with DjVu," *J. Electron Imag.*, vol. 7, no. 3, pp. 410–425, Jul. 1998.
[8] P. Haffner, L. Bottou, P. G. Howard, and Y. LeCun, "DjVu: Analyzing and compressing scanned documents for Internet distribution," presented at the Int. Conf. Document Analysis and Recognition, Sep. 1999.
[9] D. huttenlocher, P. Felzenszwalb, and W. Rucklidge, "DigiPaper: A versatile color document image representation," in *Proc. Int. Conf. Image Processing*, vol. I, Oct. 1999, pp. 219–223.
[10] J. Huang, Y. Wang, and E. K. Wong, "Check image compression using a layered coding method," *J. Electron. Imag.*, vol. 7, no. 3, pp. 426–442, Jul. 1998.
[11] R. de Queiroz, Z. Fan, and T. D. Tran, "Optimizing block-thresholding segmentation for multilayer compression of compound images," *IEEE Trans. Image Process.*, vol. 9, pp. 1461–1471, Sep. 2000.
[12] H. Cheng and C. A. Bouman, "Document compression using rate-distortion optimized segmentation," *J. Electron. Imag.*, vol. 10, no. 2, pp. 460–474, Apr. 2001.
[13] H. Cheng, G. Feng, and C. A. Bouman, "Rate-distortion based segmentation for MRC compression," in *Proc. SPIE Color Imaging: Device-Independent Color, Color Hardcopy, and Applications*, vol. 4663, San Jose, CA, Jan. 21–23, 2002.

[14] D. Mukherjee, N. Memon, and A. Said, "JPEG-matched MRC compression of compound documents," in *Proc. Int. Conf. Image Processing*, 2001, pp. 434–437.
[15] X. Li and S. Lei, "Block-based segmentation and adaptive coding for visually lossless compression of scanned documents," in *Proc. Int. Conf. Image Processing*, vol. III, 2001, pp. 450–453.
[16] D. Mukherjee, C. Chrysafis, and A. Said, "Low complexity guaranteed fit compound document compression," in *Proc. Int. Conf. Image Processing*, vol. I, 2002, pp. 225–228.
[17] [Online]. Available: http://www.gzip.org/zlib/
[18] [Online]. Available: http://www.codeproject.com/bitmap/tonyjpeglib.asp
[19] [Online]. Available: http://www.ece.uvic.ca/~mdadams/jasper/
[20] [Online]. Available: http://www.imagepower.com/compress/
[21] [Online]. Available: http://www.nkp.cz/start/knihcin/digit/vav/bi-level/Compression_Bi-level_Images.html



Tony Lin (M'04) was born in Sichuan, China, in 1974. He received the B.Sc. degree in mathematics from Chang-Chun Normal College, China, in 1996 and the Ph.D. degree in applied mathematics from Peking University, Beijing, China, in 2001.

He was a Visiting Student with Microsoft Research Asi, Beijing, in 1999 and 2000. In 2002, he joined National Laboratory on Machine Perception at Peking University, where he is currently an Assistant Professor. His research interests include wavelet-based image coding, content-based image retrieval, computer vision, and machine learning.

Pengwei Hao (M'98) was born in the north of Shaanxi Province, China, in 1966. He received the B.Sc. degree in computer science and the M.Sc. degree in computer graphics from Northwestern Polytechnical University, Xi'an, China, in 1988 and 1994, respectively, and the Ph.D. degree in image processing from the Institute of Remote Sensing Applications, Chinese Academy of Sciences, Beijing, in 1997.

From 1997 to 1999, he was a Lecturer at the Center for Information Science, Peking University, Beijing. In 2000, he was a Visiting Scientist for three months with the Centre for Vision, Speech, and Signal Processing, University of Surrey, Surrey, U.K. In 2002, he was appointed to Lecturer at Queen Mary, University of London, London, U.K. He is also currently an Associate Professor with the Center for Information Science, Peking University, Beijing. His research interests include data and image compression, data hiding, signal sampling and reconstruction, and computer graphics.