AN EFFICIENT COLOR RE-INDEXING SCHEME FOR PALETTE-BASED COMPRESSION

Wenjun Zeng, Jin Li and Shawmin Lei*

Sharp Laboratories of America

*Microsoft Research, Beijing, China

ABSTRACT

This paper presents a fast and efficient way for color re-indexing that tends to maximize the compression performance of a palette-based compression system. The proposed scheme relates the index difference of neighboring pixels to the potential cost of bits. It optimizes the assignment of index values to colors in a one-step look-ahead greedy fashion. Experimental results suggest that the proposed re-indexing scheme can reduce the bit rate by up to 43%, when compared to a previously proposed intensity-based color-indexing scheme. Furthermore, we show that with the proposed color re-indexing scheme, the palette-based JPEG-LS and palette-based JPEG-2000 can often outperform GIF significantly.

1. INTRODUCTION

Image compression usually operates on one or multiple intensity planes of digitized images. However, for images that consist of only a small number of colors, e.g., computer graphics, it is also a common practice that each color is mapped to an index and the indexes of the pixels are then compressed, usually losslessly. Since the size of the index space is much smaller than that of the color intensity space, this color palette approach usually can achieve better compression efficiency for images with a limited number of colors. This feature has been adopted by GIF (the Graphical Interchange Format), JPEG-LS [1] (the standard for lossless and near-lossless compression of still images), and most recently, the emerging still image compression standard JPEG-2000 [2][3].

Highly compressed palettized images are needed in many applications such as game cartridges and World Wide Web (WWW) on-line services. Many images used in the WWW are stored and transmitted as GIF files, which use Lempel-Ziv compression. The Lempel-Ziv compression, however, treats the image as a one-dimensional sequence of index values, ignoring the two-dimensional nature of image data. It may not have the best possible compression performance.

Natural digital images are generally the result of the quantization of waveforms, hence many characteristics of the original signal such as smoothness are preserved. With palettized images, this smoothness may not always be present. However, it has been recognized that the index image can be re-indexed without losing any information, as long as the color palette table changes accordingly. It turns out that some indexing schemes tend to produce more compressible index image than others [4][5]. There is thus a freedom to choose an appropriate indexing scheme to maximize the compression performance.

In Section 2, we present a fast and efficient way for color re-indexing that tends to maximize the compression performance. Instead of treating the problem as a general smoothness maximization problem [5], the proposed scheme relates the index difference of neighboring pixels to the potential cost of bits. It optimizes the assignment of index values to colors in a one-step look-ahead greedy fashion. Experimental results suggest that the proposed re-indexing scheme can reduce the bit rate by up to 43%, when compared to a previously proposed color-indexing scheme [4]. It is also shown that, with the proposed color re-indexing scheme, the palette-based JPEG-LS and palette-based JPEG-2000 can often outperform GIF (by up to 60% bit-saving). The proposed scheme has been adopted in the JPEG-2000 verification model software [6].

2. AN EFFICIENT RE-INDEXING SCHEME

Essentially, we would like to address the problem of re-indexing an index image/map so that the resulting index map is easier to compress. Notice that the index image has the following properties. The number of colors is usually limited, much less than that of a natural image. Pixels with the same index tend to cluster together. Within a region of the same index, the compression efficiency is very high, and is usually independent of the index value. What consumes a lot of bits is to code the index values around transitions of different regions that have different index values. In general, the larger the difference of the neighboring index values, the more bits it takes to code.

Based on the above observations, our goal is to come up with a re-indexing scheme that tends to reduce the overall difference of index values of adjacent pixels. Fig. 1 shows a re-indexing data compression system. The original image is palettized into an index image *I* and a color palette table *T*. Table *T* shows the corresponding color of each index value. The palettization process could be lossy, and usually does not take into account the subsequent lossless compression. For some compression algorithms, such as Lempel-Ziv, the compression efficiency does not depend on the indexing scheme. However, the indexing scheme often does affect the performance of many other subsequent lossless compression algorithms. Therefore, the proposed re-indexing scheme will take the initial index image *I* as input, and output a re-indexed index image *I'*. The color palette table needs to be re-organized accordingly. The new index image *I'*, in general, tends to be smoother than the initial index image *I*, thus is more amenable to most lossless compression algorithms. At the decoder, the index image *I'* and the color palette table *T'* are retrieved from the compressed bitstream and then used to reconstruct the image to be displayed.

Given a re-indexing criterion, the optimal solution can be obtained by looking at every possible re-indexing map. Suppose there are *M* different colors, then an exhaustive search needs *M*! trials. As *M* increases, it quickly becomes impractical to do such a search. This is sometimes referred to as an *N-P* complete problem in computational optimization. The goal here is thus to identify an appropriate re-indexing criterion that can characterize the compression performance, and to come up with a greedy sub-optimal solution that is simple to implement.

Suppose in the initial index image, the index values 0,1,…, *M*-1 represent color symbols S_0, S_1, …, S_{M-1}, respectively. We would like to create a one-to-one re-indexing table that maps each symbol S_i to a new index value that also takes one integer value in the range [0, *M*-1]. We propose here a method that re-indexes one symbol at a time in a greedy fashion. Each re-assignment is optimized based on the statistics collected from the initial index image and previously executed re-assignments. The statistics gathered from the initial index image is a table that indicates the cross-counts $C(S_i, S_j)$ of two different symbols S_i and S_j. The cross-count $C(S_i, S_j)$ is defined as the number of occurrences that a pixel with symbol S_i is spatially adjacent to a pixel with symbol S_j in the initial index image. Thus, $C(S_i, S_j) = C(S_j, S_i)$. The observation here is that it is advantageous to assign close index values to symbols that are frequently located next to each other. This tends to reduce the overall index difference of adjacent pixels, resulting in a smoother index image.

The proposed re-indexing scheme is outlined here, followed by an analysis.

<u>Step 1</u>: Calculate the cross-counts $C(S_i, S_j)$ for each pair of symbol S_i and S_j based on the initial index image. Calculate the cumulative cross-counts $C_i=\sum_{j=0,j\neq i}^{M-1} C(S_i, S_j)$ for each symbol S_i.

<u>Step 2</u>: Find the symbol S_{max} that has the largest cumulative cross-counts C_i. Denote it as L_0. Put L_0 in a symbol pool *P*, i.e., $P=\{ L_0 \}$ at this point. *P* will consist of spatially ordered symbols. Denote the size of *P* as *N*, and set *N*=1. A new entry can enter *P only* from either the left end or the right end. Once a symbol enters the pool *P*, it will be indicated as assigned.

<u>Step 3</u>: A new unassigned symbol will be chosen and assigned to the left or the right end position of the pool *P*. Let us first consider the left end position. The unassigned symbol S_i that maximizes the potential function $D_i=\sum_{j=0}^{N-1} w_{(N,j)} C(S_i, L_j)$ will be chosen, where $w_{(N,j)}$ are some weights controlling the impact of $C(S_i, L_j)$ on the overall potential function D_i. In general, $w_{(N,j)}$ will depend on the physical distance between the current end position of the pool *P* and the position of L_j. The parameter *N* is used to indicate that the weight $w_{(N,j)}$, in general, may change after each iteration. A good choices of $w_{(N,j)}$ is $\log_2 (1+1/d_{(N,j)})$ where $d_{(N,j)}$ is the physical distance between the position of L_j and the end position, as analyzed in the following. Denote this chosen symbol as S_{Lmax}. Similarly, S_{Rmax} can be chosen for the right end position of *P*.

<u>Step 4</u>: One of S_{Lmax} and S_{Rmax} that has the larger potential function value D_i is assigned to the corresponding end position in the pool *P*, and is denoted as L_N. Set *N*=*N*+1. An example of the status of *P* is $P = \{ L_3 L_0 L_1 L_2 \}$ for *N*=4.

<u>Step 5</u>: If ($N < M$), go to Step 3.

<u>Step 6</u>: Assign integers 0, 1, …, *M*-1 to the spatially ordered symbols in the pool *P* in left-to-right or right-to-left order. A re-indexed index image is generated by replacing the initial index value *i* with the new index value assigned to S_i.

Analysis:

Some critical points for the proposed scheme are analyzed here. The re-indexing starts with the symbol that is most frequently located adjacent to other symbols. Notice that it is the difference of two neighboring pixel index values, rather than the index values themselves, that matters. Therefore, we use the symbol S_{max} as a reference point, and assign the symbol that is most frequently located adjacent to S_{max} right next to it, i.e., the new index values of these two symbols are next to each other.

Given a set of already assigned index values (which span a continuous range of integer values), the next step is to determine which remaining symbol should be assigned the upper-bound index value or the lower-bound index value. This is realized in Step 3 of the above outline. The scheme is greedy in the sense that the new symbol can only be allocated to the left or right end position of the pool *P*. Within this constraint, we seek to optimize each new allocation. How to choose an appropriate allocation criterion is a critical issue. The measure D_i used in Step 3, in some sense, measures how often pixels marked with the candidate symbol are located adjacent to pixels marked with already assigned symbols. One particular choice of the weight $w_{(N,\,j)}$ may be better for a specific subsequent lossless coding scheme than for others. It will be shown that our particular choice of $\log_2 (1+1/\, d_{(N,j)})$ for $w_{(N,\,j)}$ is a justifiable, perhaps near optimal choice if LOCO-I/JPEG-LS [1][7] is to be used to code the index image losslessly.

LOCO-I follows a traditional predictor-modeler-coder structure. The value of the current pixel is predicted from one of its immediate neighboring pixel values. It is shown [7] that, approximately, the number of bits it takes to code a residue error have a $\log_2$ relationship with the magnitude of that residue error. In each iteration, one of the remaining unassigned symbols will be chosen to fill in an end position of the pool *P*. If S_i is to be assigned, the total bits needed to code those transition pixels between S_i and each of the already assigned symbols in the pool is in the order of $\sum_{j=0}^{N-1} \log_2 d_{(N,j)}\ C(S_i, L_j)$. If, instead, S_i is not to be assigned at this iteration, the total bits needed to code those transition pixels between S_i and each of the already assigned symbols will increase in general. Assuming S_i will be assigned in the next iteration, then the extra amount of bits needed ΔB_i is

$$\begin{aligned}\Delta B_i &= \textstyle\sum_{j=0}^{N-1} \log_2 (d_{(N,j)} +1)\ C(S_i, L_j) - \\ &\quad \textstyle\sum_{j=0}^{N-1} \log_2 d_{(N,j)}\ C(S_i, L_j) \\ &= \textstyle\sum_{j=0}^{N-1} \log_2 (1+1/\, d_{(N,j)})\ C(S_i, L_j) \\ &\approx 1/\ln 2 \textstyle\sum_{j=0}^{N-1} 1/\, d_{(N,j)}\ C(S_i, L_j) \text{ when } d_{(N,j)} >>1.\end{aligned}$$

Therefore, with the weight $w_{(N,\,j)}$ chosen to be $\log_2 (1+1/\, d_{(N,j)})$, Step 3 in the re-indexing procedure tends to assign the symbol that will result in the largest saving of coding bits to the end position. This is a one-step look-ahead optimization process. This also suggests that it is reasonable to impose the constraint that a new symbol will only be allocated to the end positions of the pool.

It is possible to implement the proposed re-indexing scheme using $O(M^2)$ operations. Typically, it takes a fraction of second to a few seconds, depending on *M*. For example, for the 218x74 "af29" image with *M*=71, it takes about 250 ms on a Sun Ultra 2 workstation.

3. EXPERIMENTAL RESULTS

We first tested on a set of icon-like graphics images with sizes ranging from 218x74 to 550x550. These images have only a limited number of colors. Each image is first palettized, resulting in a color palette table and an index image. The initial indices are generated using a luminance-intensity-based approach [4]. Specifically, the indices 0, …, *M*-1 will be assigned to the colors in the descending order of their luminance intensity. This is a reasonably good indexing scheme. It assigns close index values to colors with close luminance intensity values.

The re-indexing scheme is then applied to the initial index images. The output index images are then subject to lossless compression by JPEG-2000 [6] and JPEG-LS [7]. These two cases will be referred to as palette-based JPEG-2000 and palette-based JPEG-LS, respectively.

Fig. 2 shows the original "Party8" image (luminance only), the intensity-based index image, and the re-indexed index image. The displayed index images have been subject to Gamma correction with the same gamma factor of 6 for display purpose. The re-indexed index image appears to be much smoother than the initial intensity-based index image. This is expected to greatly facilitate the subsequent lossless coding.

Table 1 shows the experimental results. For palette-based JPEG-2000, wavelet transform usually hurts the compression efficiency. All the results reported here for palette-based JPEG-2000, except for "af29" and "bod7", are obtained without wavelet transform, i.e., the entropy coder is directly applied to the spatial index image. For "af29" and "bod7", the results presented in Table 1 are obtained using a 2-level wavelet transform with the lifting filter 9 (the Haar filter) which performs best among the set of lifting filters provided in JPEG-2000 VM software [6]. For these two cases, wavelet transform helps to further reduce the file size by about 4-5%. The index values are down-shifted by their center value 2^{D-1}, where *D* is the bit depth of the index values, for more efficient bit-plane entropy coding. The file size includes the size of the uncompressed color table. It is observed that, for palette-based JPEG-2000 and palette-based JPEG-LS, the proposed re-indexing scheme, on the average, reduces the bit rate by 23.3% and 20.5% respectively, when compared to the intensity-based indexing scheme. With the proposed re-indexing scheme, palette-based approaches also outperform GIF by a bit rate saving of up to 60%.

Table 1 also shows the results for two 2048x2560 natural color images "woman" and "bike" (palettized using 256 colors). Similar improvements are observed.

4. REFERENCES

[1] "FCD 14495, Lossless and near-lossless coding of continuous tone still images (JPEG-LS)," ISO/IEC JTC1/SC29 WG1 (JPEG/JBIG), July 1997.

[2] "Information Technology – JPEG 2000 Image Coding System," ISO/IEC FCD15444-1: 2000 (V1.0, Mar. 2000).

[3] W. Zeng and S. Lei, "Option of JPEG2000 for coding palettized images - a proposal," JTC1/SC29/WG1 N1453, Maui, Hawaii, Dec. 1999.

[4] A. Zaccarin and B. Liu, "A novel approach for coding color quantized images, " *IEEE Tran. Image Proc*., vol. 2, no. 4, pp. 442-453, 1993.

[5] N. Memon and A. Venkateswaran, "On ordering color maps for lossless predictive coding," *IEEE Tran. Image Proc*., vol. 5, no. 11, pp. 1522-1527, Nov. 1996.

[6] "JPEG 2000 Verification Model 7.0 Software", ISO/IEC JTC1/SC29/WG1 N1685, April 2000.

[7] M. Weinberger, G. Seroussi and G. Sapiro, “LOCO-I: A low complexity, context-based, lossless image compression algorithm,” *Proc. IEEE Data Compression Conference*, March, 1996.

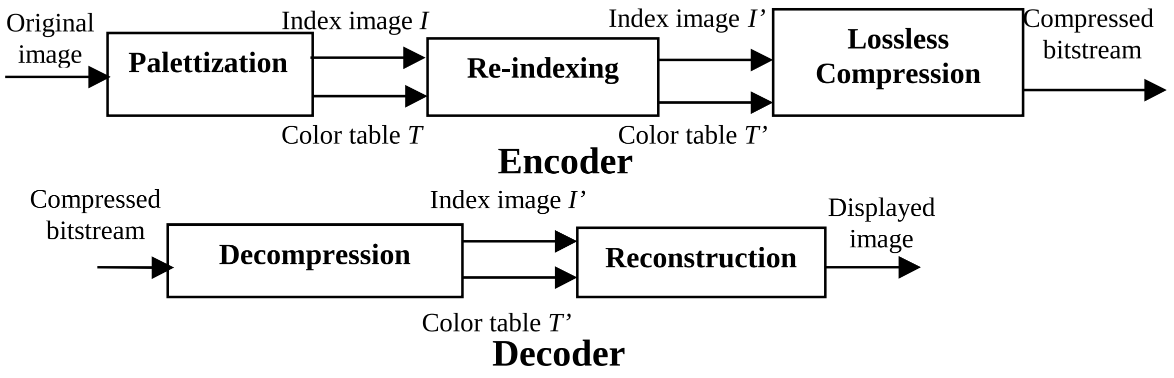

Fig. 1: The general architecture of the proposed re-indexing compression system

Graphics images	# of colors	Losslessly compressed bitstream size (bytes)						
		Palette-based JPEG-LS			Palette-based JPEG 2000			GIF
		Intensity-based	Re-indexing	Saved bits(%)	Intensity-based	Re-indexing	Saved bits(%)	
Af29	71	5554	4566	17.8	6540	5476	16.3	5600
Andrene	8	2318	1580	31.8	2306	1462	36.6	2034
Bod7	46	5102	4545	10.9	6435	5369	16.6	5642
Party8	12	7274	5983	17.8	8273	7423	10.3	17650
Pizza	7	11163	8294	25.7	9073	5915	34.8	10142
Rob	5	3332	2435	26.9	2721	2571	5.5	3595
Sam	8	1841	1613	12.4	2090	1192	43.0	2071
Woman	256	3527774	3544633	-0.5	3352470	2988274	10.9	3692257
Bike	256	3574296	3288853	8.0	3515853	2980061	15.2	3367021

Table 1: Comparisons between different indexing schemes and lossless compression schemes.

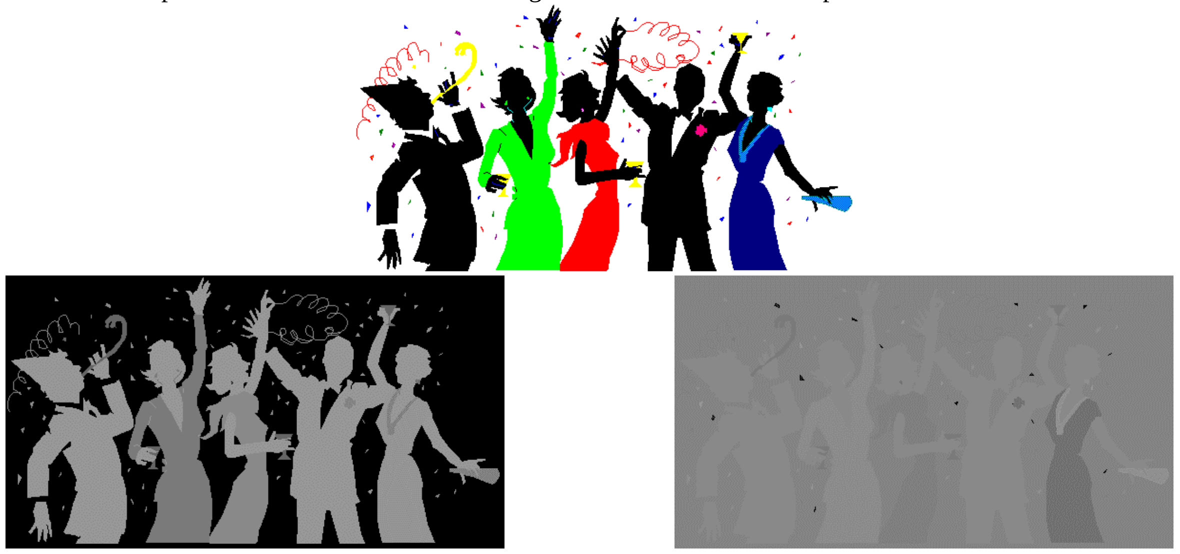

Fig. 2: Original image (Top) and index images of “Party8” (BottomLeft: intensity-based; BottomRight: proposed scheme).