16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, August 25-29, 2008, copyright by EURASIP

Compressing color-indexed images by dynamically reordering their palettes

Yuk-Hee Chan, Ka-Chun Lui and P. K. Lun

Abstract – **A new lossless compression scheme for coding color-indexed images is proposed. This scheme first turns the index map of a color-indexed image into a new index map each element of which serves as an index to a pixel-dependently reordered version of the palette. This process reduces the zero-order entropy and the energy of the index map significantly, and hence a matching coding scheme can then be exploited to encode the new index map effectively. Simulation results show that the proposed method is superior to other state-of-art lossless compression algorithms for coding color-indexed images in terms of compression performance.**

I. Introduction

A color-indexed image is represented with a color index map each element of which serves as an index to select a color from a predefined set of colors called palette to represent the color of a pixel in the image[1]. Two completely different colors can be of similar index values in a palette. Hence, it is always a challenging task to compress a color-indexed image as the compression must be lossless and predictive coding techniques are generally not effective to predict an index based on the spatial correlation of the index map.

Palette reordering is a remedial process aiming at finding a permutation of the color palette to make the resulting color index map more suitable for predictive coding[2]. Various reordering methods were proposed for this purpose[3-9]. Recent contributions on this area put their focus on how to use a self-organizing neural network [10,11] to obtain a better permutation of the palette or use a post-processing step to improve the reordering performance[12]. All palette reordering methods can effectively improve the compression ratio when their outputs are encoded with JPEG-LS or lossless JPEG-2000[2].

Palette reordering is good in a way that the process is transparent to the decoder while it improves the compression performance. However, to seek for further compression performance or to achieve some extra goals, there are some other lossless algorithms which are not solely based on palette reordering. For examples, Kuroki et al.'s method [13] and Arnavut et al.'s method [14] try to improve the compression performance while Chen et al.'s method tries to support progressive coding and improve the compression performance simultaneously [15].

A new lossless compression scheme for coding color-indexed images is proposed in this paper. This scheme can be decomposed into two stages. In stage 1, it raster scans the image to be encoded and, on the fly, reassigns indices to palette colors pixel by pixel adaptively based on the predicted color of the current pixel and an updated statistical study of the processed image pixels. As a result, it is able to produce an index map of very low zero-order entropy, little energy and little spatial correlation. In stage 2, the resultant index map is further decomposed into a number of holey binary bit planes and then each of them is encoded with a context-based binary arithmetic coder. Simulation results show that the performance of the proposed compression scheme outperforms the other state-of-art lossless compression schemes in terms of output bit rate.

II. Pixel-wise Palette Reordering

This section presents the method used in stage 1 to reassign indices to palette colors pixel by pixel adaptively. Since the index assignment of the palette is pixel-dependent, the reordering method is referred to as pixel-wise palette reordering method (PPR) so as to discriminate it from the conventional palette reordering methods in which a fixed palette is designed for all pixels to share.

Let the input color-indexed image be **X** and the associated palette be $\Omega=\{\vec{c}_k \,|k=0,1\ldots N\text{-}1\}$, where N is the size of the palette and $\vec{c}_k$ is the k^{th} palette color in Ω. Without lose of generality, we assume all $\vec{c}_k$ in Ω are sorted according to their luminance and $\vec{c}_0$ (i.e. k=0) is the one of the minimum luminance. Note that this criterion can be easily satisfied through an initialization process. This sorted palette is used as a reference palette in the codec.

Based on the index map of **X**, a full-color image can be constructed with palette Ω. The image is raster scanned and processed. For each pixel, the intensity values of its three color components are individually predicted with their own corresponding color planes by using a MED predictor. MED is used in JPEG-LS and the details of its operation can be found in [16].

Suppose the prediction results of the red, the green and the blue color components of the current pixel (i,j) are, respectively, $r(i,j)$, $g(i,j)$ and $b(i,j)$. In vector form, the prediction result of pixel (i,j) is $\vec{v}(i,j)=(r(i,j),g(i,j),b(i,j))$. $\vec{v}(i,j)$ is then color quantized with palette Ω. In practice, the quantization result could be different from the original color of pixel (i,j). Without lose of generality, we assume that the quantization result and the original color of pixel (i,j) are, respectively, the p^{th} and the r^{th} palette colors in

Ω. For the purpose of reference, they are denoted as $\vec{c}_p$ and $\vec{c}_r$ respectively.

In the proposed scheme, statistics about the frequency of the occurrence of specific events are collected when processing the image such that the scheme can learn from the experience to improve its reordering performance adaptively. Specifically, we build five simple probability models and then merge them to form a combined probability model. Five separate tables are constructed for building these five probability models.

The first table $\{T_d(m,n)|m,n=0,1\ldots N\text{-}1\}$ is referred to as *discrepancy frequency table* (DF-Table) as its entry $T_d(m,n)$ records the occurrences of encountering a pixel whose quantized predicted color and real color are, respectively, $\vec{c}_m$ and $\vec{c}_n$ up to the moment when pixel (i,j) is processed. With this table, the conditional probability that $\vec{c}_k$ is the real color of pixel (i,j) given that $\vec{c}_p$ is the quantized predicted color of the pixel can be estimated as

$$P_p(k \mid p) = T_d(p,k) / \Sigma_{u=0}^{N-1} T_d(p,u) \quad \text{for } k=0,1\ldots N\text{-}1 \quad (1).$$

All $T_d(m,n)$ values are initialized to 1 at the very beginning and the DF-table is updated after a pixel is processed.

The other four tables, which are denoted as $\{T_N(m,n)|m,n=0,1\ldots N\text{-}1\}$, $\{T_W(m,n)|m,n=0,1\ldots N\text{-}1\}$, $\{T_{NW}(m,n)|m,n=0,1\ldots N\text{-}1\}$ and $\{T_{NE}(m,n)|m,n=0,1\ldots N\text{-}1\}$, are constructed to reflect how likely that a particular color in Ω is the real color of a pixel when the color of one of the four processed connected neighbors (i.e. the northern, the western, the northwestern and the northeastern) of the pixel is given. For example, $\{T_N(m,n)|m,n=0,1\ldots N\text{-}1\}$ records the occurrences of encountering a pixel whose northern neighbor's real color is $\vec{c}_m$ while its own real color is $\vec{c}_n$ up to the moment when pixel (i,j) is processed. Entry values of all these tables are also initialized to 1 at the very beginning and all tables are updated after a pixel is processed.

Without lose of generality, here we assume that the real colors of pixels $(i,j\text{-}1)$, $(i\text{-}1,j\text{-}1)$, $(i\text{-}1,j)$ and $(i\text{-}1,j+1)$ are, respectively, the r_1^{th}, the r_2^{th}, the r_3^{th} and the r_4^{th} palette colors in Ω. Based on the four tables, four additional context-based probability models can be derived as

$$P_{c1}(k \mid r_1) = T_W(r_1,k) / \Sigma_{u=0}^{N-1} T_W(r_1,u)$$
$$P_{c2}(k \mid r_2) = T_{NW}(r_2,k) / \Sigma_{u=0}^{N-1} T_{NW}(r_2,u)$$
$$P_{c3}(k \mid r_3) = T_N(r_3,k) / \Sigma_{u=0}^{N-1} T_N(r_3,u)$$
$$P_{c4}(k \mid r_4) = T_{NE}(r_4,k) / \Sigma_{u=0}^{N-1} T_{NE}(r_4,u)$$
$$\text{for } k=0,1\ldots N\text{-}1 \quad (2).$$

Each of them tells how likely that $\vec{c}_k$ is the real color of pixel (i,j) when the real color of a particular neighbor of pixel (i,j) is given.

In our approach, when assigning a new color index to a palette color, probability models $P_p(k \mid p)$, $P_{c1}(k \mid r_1)$, $P_{c2}(k \mid r_2)$, $P_{c3}(k \mid r_3)$ and $P_{c4}(k \mid r_4)$ are merged to form a single probability model to determine the likeliness that the concerned palette color is the real color of the pixel currently processed. In particular, the combined probability model is given as

$$P(k) = L(k) / \Sigma_{u=0}^{N-1} L(u) \quad \text{for } k=0,1\ldots N\text{-}1 \quad (3),$$

where

$$L(k) = w_0 T_d(p,k) + w_1 T_W(r_1,k) + w_2 T_{NW}(r_2,k) + w_3 T_N(r_3,k) + w_4 T_{NE}(r_4,k)$$
$$\text{for } k=0,1\ldots N\text{-}1 \quad (4).$$

The larger the value of $L(k)$, the more likely that $\vec{c}_k$ is the real color of pixel (i,j).

Weighting factors w_0, w_1, w_2, w_3 and w_4 are used to weight the contribution of the five probability models according to their significance in the merging. They can be determined with a training process with a set of training images. We found that $\{w_0, w_1, w_2, w_3, w_4\}=\{4,2,1,2,1\}$ was a good choice in our simulation study.

It is possible that some neighbors of pixel (i,j) are out of the image boundary. In that case, corresponding items are removed from eqn. (4) when computing $L(k)$. For example, when one processes a pixel on the left boundary of an image, the western and the northwestern neighbors are missing and hence eqn. (4) becomes $L(k) = w_3 T_N(r_3,k) + w_4 T_{NE}(r_4,k) + w_0 T_d(p,k)$.

Once $L(k)$ is determined for pixel (i,j), the colors in palette Ω are adaptively reordered based on $L(k)$. In particular, $\vec{c}_k$'s are sorted according to the values of $L(k)$ for $k=0,1\ldots N\text{-}1$ in descending order. If there exist two different colors $\vec{c}_l$ and $\vec{c}_d$ such that $L(l)=L(d)$, $\vec{c}_l$ and $\vec{c}_d$ will be sorted according to their Euclidean distances to $\vec{c}_p$. The closer one is put in front of the other. If they are still not distinguishable, their order will be determined by their ranking in reference palette Ω.

The position of $\vec{c}_r$ in the newly reordered queue can be used as an index to the queue and is used to represent the pixel in the output of the reordering method. Note the queue forms a transient version of palette Ω. After processing this pixel, $T_d(p,r)$, $T_W(r_1,r)$, $T_{NW}(r_2,r)$, $T_N(r_3,r)$ and $T_{NE}(r_4,r)$ are incremented by 1 to update the frequency count of corresponding events. The algorithm proceeds to process next pixel in the scanning path by repeating the same procedures until all pixels are processed.

In the decoder, to decode a pixel, the same process is carried out to determine the same transient version of palette Ω. As soon as the index for the pixel is received, it can be used to fetch the corresponding color in the transient version of palette Ω to reconstruct (i) a color index map all elements of which share a single palette of fixed assignment of color indices such as Ω, or (ii) the full-color image directly.

The order of the algorithm's complexity is $O(NS_i)$, where S_i is the image size in terms of number of pixels and N is the number of colors in the palette.

III. Properties of Reindexed Index Maps

Simulations were carried out to evaluate the performance of the reordering methods proposed in Section II. Twenty-three 8-bit color-quantized Kodak images downloaded from

ftp://ftp.ieeta.pt/~ap/images/kodak/768x512/256/ were used as the inputs to the proposed palette reordering method. Figure 1 shows the effect of the proposed palette reordering method.

Figure 1a shows image *Kodak256-05*, the fifth testing image in the set, and Figure 1b shows a typical color index map of Figure 1a. This index map is associated with a palette whose colors are sorted by luminance (i.e. palette Ω). Figure 1c is the processing result of the proposed pixel-wise palette reordering method. One can see that very few indices are of large values in the index map of the result.

Figure 2 shows the histograms of Figures 1b and 1c. One can see that the sharp peak of the histogram of the output of PPR. As a matter of fact, it appears as a monotonic decreasing function and drops sharply from the order of 10^5 to the order of 10^3 in 33 indices. Consequently, the zero-order entropy of our results is significantly reduced as compared with that of the reference. In particular, the zero-order entropy values of Figures 1b and 1c are, respectively, 7.190 and 3.980 bpp (bits per pixel).

The proposed palette reordering method also reduces the energy carried in the original index map significantly. The root mean square values of the indices (energy) in the index planes shown in Figures 1b and 1c are, respectively, 114.1 and 14.3.

As making the reordering transparent to the decoder is not the concern in our case, the proposed reordering methods do not exploit a bijective mapping as in conventional reordering methods [2-12] to reindex the palette colors, which allows it to change the sorted index distribution and reduce the zero-order entropy remarkably. Figure 3 shows the reduction in zero-order entropy that the proposed method can achieve when processing the testing images. We note here that the reference index maps are associated with the palettes whose colors are sorted by their luminance.

IV. Coding of Reindexed Indexed Maps

Since its zero-order entropy is significantly lowered, the new index map can be easily compressed with well-developed lossless compression algorithms such as JPEG-LS and JPEG-2000.

However, to compress the index map further, we propose to use a dedicated bit-plane coding scheme instead. In this scheme, a reindexed index map is decomposed into N-1 bit planes as follows.

$$B_k(i,j)=\begin{cases}1 & \text{if } I(i,j)>k\\ 0 & \text{if } I(i,j)=k\\ \text{don't care} & \text{if } I(i,j)<k\end{cases} \quad \text{for } k=0,1,\ldots N\text{-}2 \qquad (5)$$

where $B_k(i,j)$ is the $(i,j)^{\text{th}}$ element of the k^{th} bit plane and $I(i,j)$ is the new index value of pixel (i,j). For reference, this approach of bit-plane separation is referred to as *value-based bit-plane separation* (VBS).

Starting from k=0, bit planes are gradually constructed as k increases. Once a bit plane is defined, its bits are raster scanned and encoded with context-based entropy coding. In other words, bit planes of lower k values are encoded first. If $B_k(i,j)$ is a don't care bit, there must be $B_l(i,j)$=0 for a particular $l<k$. Accordingly, the coding of a don't care $B_k(i,j)$ bit can be skipped as bit plane l was encoded and the fact of $I(i,j)=l$ is known to the decoder.

As most indices in the output index maps are of values identical or close to zero and the distribution of the index values can be modeled with an exponential function, the number of don't care bits found in a particular bit plane increases significantly as k increases.

Each holey binary bit-plane B_k is encoded with a context-based binary arithmetic coder. Figure 4 shows the general form of the context templates used in the proposed coding scheme. It is possible that the binary context of $B_k(i,j)$ contains some don't care pixel bits when $B_k(i,j)$ is encoded with context-based entropy coding. In that case, the don't care bits can be filled with either 0 or 1. In our realization, it is filled with 0 for simplicity.

As the value of k gets larger, there are more don't care bits in bit plane B_k and, accordingly, the context template covers more don't care bits when it moves around. To achieve better performance and avoid context dilution problem, a context template of smaller size is used to encode a bit plane of larger k. As shown in Figure 4, the pixel positions of the context template are numbered. Instead of using all template locations, only first L positions are used, where L is a function defined as

$$L=\lceil 9-\log_2(k+1)\rceil \qquad \text{for } k=0,1,\ldots N\text{-}2 \qquad (6).$$

In the suggested binary arithmetic coder, the forgetting factor α and the biasing constant Δ for updating the conditional probability for having bit '1', $P(1|context)$, are, respectively, 0.985 and 0.006. In particular, the probability for having bit '1' when the context is binary pattern i again is updated by

$$P(1\,|\,context=binary\ pattern\ i)=(t_i+\Delta)/(s_i+2\Delta) \quad \text{for } i=0,1\ldots 4095 \qquad (7)$$

where t_i and s_i are updated whenever a context of binary pattern i is encountered using

$$t_i:=\begin{cases}\alpha\, t_i+1 & \text{if the current pixel value is 1}\\ \alpha\, t_i & \text{else}\end{cases} \qquad (8)$$

and

$$s_i:=1+\alpha\, s_i \qquad (9)$$

The initial values of t_i and s_i are, respectively, 1 and 2 for all context patterns i. The suggested values of parameters α and Δ were selected based on the work of Reavy and Boncelet [17] on binary image compression.

V. Simulation Results

Simulations were carried out to evaluate the performance of the proposed coding scheme. To have a fair comparison with the performance of other state-of-art lossless coding algorithms, all the testing images used in the simulations were obtained from the ftp site ftp://ftp.ieeta.pt/~ap/images. They are organized into 13 groups: (1) a set of 18 synthetic images referred to as *Synthetic*, (2)-(7) 6-, 7- and 8-bit color-

quantized versions of image set *Natural1* which contains 23 natural images in the "Kodak" database, and (8)-(13) 6-, 7- and 8-bit color-quantized versions of image set *Natural2* which contains 12 popular natural images. In Groups (5)-(7) and (11)-(13), Floyd-Steinberg dithering is applied during color quantization.

Table 1 lists the compression performance achieved by various lossless compression algorithms for coding color-indexed images in terms of bits per pixel (bpp). Unless it is specified, each figure in the table shows the average value of the bpp values (one for each testing image) of all testing images in a group. The data provided in columns 1-4([5-8]), column 5([14]) and column 6([11]) are, respectively, copied from [2], [14] and [11]. The simulation results presented in column 7 are obtained with the codec provided by the authors of [15].

Among these evaluated algorithms, [5], [6], [7], [8] and [11] are based on conventional palette reordering. Their outputs can be encoded with JPEG-LS or JPEG-2000 as suggested by the authors. Table 1 shows the results using JPEG-LS as JPEG-LS generally provides a better compression performance than JPEG-2000.

The last column shows the simulation results of the proposed coding scheme. The bit overhead for coding all header information including the palette was taken into account in the calculation of the figures.

VI. Conclusions

A pixel-wise palette reordering method is proposed in this paper to reshape the statistical properties of the index map of a color-indexed image. This method scans the index map pixel by pixel. At each pixel, it adaptively reorders the colors in the image palette and then replaces the pixel's original index in the index map with the position of the pixel's color in the reordered palette as the new index. Eventually, it generates a new index map each element of which serves as an index to a pixel-dependently reordered version of the palette. The resultant index map contains very low zero-order entropy and energy. Besides, most of the spatial correlation among pixels can be removed in the new index map.

A dedicated coding scheme is also proposed to encode the outputs of the proposed pixel-wise palette reordering method. Both bit-plane coding and entropy coding techniques are exploited to make a good use of the statistical properties of the reindexed index maps. Accordingly, a good compression ratio can be easily achieved. Simulation results reveal that the coding performance of the proposed approach is better than other state-of-art lossless image compression methods when coding color-indexed images.

Acknowledgements

This work was supported by Centre for Signal Processing, The Hong Kong Polytechnic University (Grants A-PA3U and 1-BB9T). The authors would also like to thank the authors of [15] for providing their codec to obtain the simulation results of their paper.

References

[1] M.T. Orchard and C.A. Bouman, "Color quantization of images," IEEE Trans. On signal Processing, Dec 1991, Vol.39, No.12, pp.2677-2690.

[2] A.J. Pinho and A. J. R. Neves, "A survey on palette reordering methods for improving the compression of color-indexed images," IEEE Trans. Image Processing, Vol.13, No.11, Nov 2004, pp.1411-1418.

[3] A. Zaccarin and B. Liu, "A novel approach for coding color quantized images," IEEE Trans. Image Processing, vol. 2, Oct. 1993, pp. 442-453.

[4] A. Spira and D. Malah, "Improved lossless compression of color-mapped images by an approximate solution of the traveling salesman problem," Proc. IEEE ICASSP'01, Vol.III, May 2001, pp.1797-1800.

[5] N. D. Memon and A. Venkateswaran, "On ordering color maps for lossless predictive coding," IEEE Trans. Image Processing, vol. 5, Nov. 1996, pp. 1522-1527.

[6] W. Zeng, J. Li, and S. Lei, "An efficient color re-indexing scheme for palette-based compression," Proc. IEEE ICIP'00, Vancouve, BC, Canada, Vol. III, Sept. 2000, pp. 476-479.

[7] A. J. Pinho and A. J. R. Neves, "A note on Zeng's technique for color reindexing of palette-based images," IEEE Signal Processing Letter, vol. 11, Feb. 2004, pp. 232-234.

[8] S. Battiato, G. Gallo, G. Impoco, and F. Stanco, "An efficient Re-indexing algorithm for color-mapped images," IEEE Trans. on Image Processing, Vol.13, Nov. 2004, pp.1419-1423.

[9] N. D. Memon and R. Rodila, "Transcoding GIF images to JPEG-LS," IEEE Trans. on Consumer Electronics, Vol.43, No.3, Aug 1997, pp.423-429.

[10] S.C. Pei, Y.T. Chuang and W.H. Chuang, "Effective palette indexing for image compression using self-organization of Kohonen feature map," IEEE Transactions on Image Processing, Vol.15, No.9, Sept. 2006, pp.2493-2498.

[11] S. Battiato, F. Rundo and F. Stanco, "Self organizing motor maps for color-mapped image re-indexing," IEEE Trans. on Image Processing, Vol.16, No.12, Dec 2007, pp.2905-2915.

[12] J. Z. C. Lai and Y. C. Liaw, "A novel approach of reordering color palette for indexed image compression," IEEE Signal Processing Letters, Vol.14, No.2, Feb. 2007, pp.117-120

[13] N. Kuroki, T. Yamane and M. Numa, "Lossless coding of color quantized images based on Pseudo distance," Proceedings, 47th IEEE International Midwest Symposium on Circuits and Systems, 2004, Vol.I, pp.245-247.

[14] Z. Arnavut and F. Sahin, "Lossless compression of color palette images with one-dimensional techniques," Journal of Electronic Imaging, Vol.15, No.2, 023014, Apr-Jun 2006

[15] X. Chen, S. Kwong, and J. Feng, "A New Compression Scheme for Color-Quantized Images", IEEE Trans. on Ciruits and Systems for video technology, Vol. 12, No 10, Oct 2002, pp.904-908.

[16] ISO/IEC JTC 1/SC 29/WG 1 JPEG LS image coding system, ISO Working Document ISO/IEC JTC1/SC29/WG1 N399 – WD14495, July 1996.

[17] M. D. Reavy and C. G. Boncelet, "An algorithm for compression of bilevel images," IEEE Trans. on Image Processing, Vol.10, No.5, May 2001, pp.669-676.

11	8	6	9	12
7	3	2	4	10
5	1	X		

X : the pixel of interest

Fig 4. The context template used in VBS.

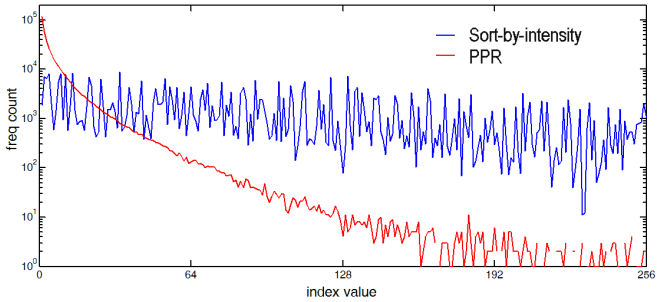


Fig 2. Histograms of Figs. 1b and 1c

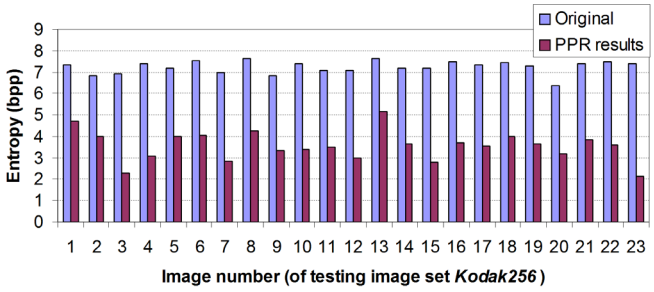
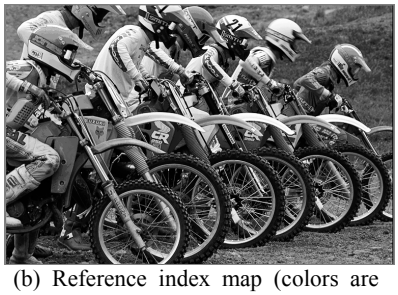
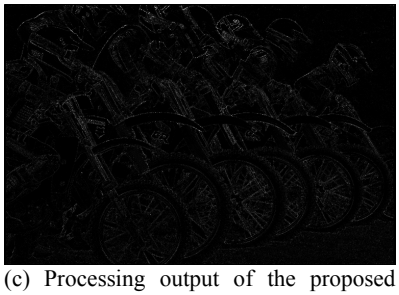


Fig. 3 Reduction in zero-order entropy when the proposed palette reordering method is applied (testing image set: *Kodak256*)

(a) Color-quantized image



(b) Reference index map (colors are sorted by luminance)



(c) Processing output of the proposed PPR

Fig. 1 Processing results of the proposed pixel-wise reordering methods.

		Lossless coding algorithms for coding color-indexed images								
Testing image Group	Number of colors	Memon et al.[5]	Zeng et al.[6]	Pinho et al.[7]	Battiato et al.[8]	Arnavut et al.[14]	Battiato et al.[11]	Chen et al.[15]	Kuroki et al.[13]	Ours
Synthetic										
(1)	variable	2.243	2.452	2.333	2.650	1.740	2.306	1.728	2.535	1.615
Natural1 (Kodak) - without dithering										
(2)	256	4.203	4.907	4.475	5.530	3.549	4.440	3.633	4.275	3.351
(3)	128	3.441	3.844	3.552	4.287	2.954	3.526	2.915	3.553	2.771
(4)	64	2.641	2.804	2.709	3.144	2.327	2.710	2.247	2.862	2.200
average of groups (2)-(4)		**3.428**	**3.852**	**3.579**	**4.320**	**2.944**	**3.558**	**2.932**	**3.563**	**2.774**
Natural1 (Kodak) – with dithering										
(5)	256	4.870	5.595	5.120	6.232	4.124	---	4.128	4.716	3.892
(6)	128	4.112	4.537	4.225	5.146	3.363	---	3.375	4.028	3.254
(7)	64	3.341	3.590	3.420	3.915	2.327	---	2.709	3.382	2.680
average of groups (5)-(7)		**4.108**	**4. 574**	**4.255**	**5.098**	**3.272**	**---**	**3.404**	**4.042**	**3.275**
Natural 2 - without dithering										
(8)	256	4.607	5.491	4.881	5.983	4.023	4.779	3.936	4.549	3.786
(9)	128	3.771	4.339	3.979	4.729	3.220	3.785	3.087	3.751	3.018
(10)	64	2.793	3.060	2.912	3.281	2.497	2.921	2.386	3.043	2.408
average of groups (8)-(10)		**3.724**	**4.297**	**3.924**	**4.664**	**3.247**	**3.828**	**3.136**	**3.781**	**3.071**
Natural 2 – with dithering										
(11)	256	5.063	6.006	5.322	5.455	4.345	---	4.271	4.858	4.115
(12)	128	4.205	4.766	4.423	5.314	3.557	---	3.510	4.173	3.442
(13)	64	3.417	3.712	3.548	4.127	2.842	---	2.626	3.506	2.806
average of groups (11)-(13)		**4.228**	**4.828**	**4.431**	**4.965**	**3.582**	**---**	**3.469**	**4.179**	**3.455**

Table 1. Performance of different lossless image compression methods for coding color-indexed images in terms of bpp.