The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

Compressing color-indexed images with an adaptive palette reordering method

Lui Ka-Chun

A thesis submitted in partial fulfillment of the requirements for

the Degree of Master of Philosophy

September 2007

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

_____ (Signed)

Lui Ka Chun (Name of Student)

To my parents

Abstract

Color-indexed images are widely found in various image applications nowadays. An efficient compression algorithm for coding color-indexed images can help to reduce their data size for saving both transmission bandwidth and data storage requirement.

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. 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. Conventional palette reordering methods generally reorder palette colors to form a static palette whose index assignment is common to all pixels for making the reordering transparent to the decoder.

In this thesis, an adaptive palette reordering method is proposed. Unlike those conventional palette reordering methods, this method adaptively reorders the palette to make the index assignment pixel-dependent. By so doing, the reordering is no longer transparent to the decoder. However, the resultant index map of the original color-indexed image can be of much lower zero-order entropy, smaller index variance and less spatial correlation, which makes the index map much easier to be encoded efficiently with a typical lossless codec such as JPEG-LS.

Various lossless coding algorithms for color-indexed images can then be developed based on the proposed adaptive palette reordering algorithm. Simulation results show that their coding performance is better than state-of-art lossless compression algorithms including those are not based on palette reordering technique. In particular, when an index map was separated into binary bit planes with our suggested approach and then encoded with a context-based binary arithmetic coding scheme, an average compression ratio of 2.44:1 could be achieved.

Index prediction, color reordering and DF-Table merging are some of the key functional components carried out in the proposed adaptive palette reordering algorithms. In practice, each one of them can be realized in different ways. Some of their realizations were evaluated and the result is also reported in this thesis.

Author's Publications

(List of publications of the author on which this thesis is based)

International Journal Paper

1. Ka-Chun Lui, Yuk-Hee Chan and P.K. Lun, "Compressing color-indexed images with an adaptive palette reordering method," submitted for possible publication in IEEE transactions on Image Processing

International Conference Papers

1. Ka-Chun Lui and Yuk-Hee Chan, "An adaptive palette reordering method for compressing color-indexed image", Proceedings, IEEE TENCON'06, Hong Kong, 14-17 Nov 2006, pp.1-4
2. Ka-Chun Lui and Yuk-Hee Chan, "A lossless compression algorithm for color-indexed images using adaptive palette reordering," Proceedings, 15th European Signal Processing Conference (EUSIPCO 2007), Sep 3-7, 2007, Poznan, Poland, pp.812-815

Acknowledgements

First of all, I would like to take this opportunity to express my honest gratitude to my supervisor, Dr. Chris Yuk-Hee Chan, for his guidance and tutorial throughout the course of this research effort. I believe that this knowledge will be much helpful to my future career.

I also wish to thank my friends and colleagues for their encouragement. Their expert knowledge has helped me to resolve a lot of difficult problems in my study. Thanks are also extended to Mr. W. H. Wong who have worked with me.

I should also not be miserly to send my acknowledgements to all classmates. The helpful discussions with them have proved to be beneficial and inspiring. I would not grow faster without their experience sharing.

Last, but not least, I must express my feeling of thankfulness and appreciation to my parents, my brothers, my wife, my sons and my daughter for their never-ending support. Without their backing, this study would not have the chance to be completed.

Table of Contents

List of Figures

List of Tables

Statement of Originality

The following contributions reported in this thesis are claimed to be original.

1. An adaptive palette reordering method which scans pixels of a color-indexed image and, on the fly, reassigns indices to palette colors pixel by pixel adaptively based on an updated statistical study of the processed image pixels and the predicted color of the current pixel.
2. A detailed study on how the proposed adaptive palette reordering method reduces the zero-order entropy, the variance and the spatial correlation of the index map of a color-indexed image to make the index map easier to be encoded.
3. A framework of how to realize lossless compression of color-indexed images based on adaptive palette reordering.
4. A detailed study on how the actual realization of different functional components such as color reordering, index prediction and DF-Table merging affect the overall performance of adaptive palette reordering and hence the coding performance.
5. Various lossless coding algorithms based on the proposed adaptive palette reordering algorithm.
    - Encoding the index map with JPEG-LS/Lossless JPEG-2000
    - Encoding the index map with significance-based bit-plane coding as presented in Section 7.3
    - Encoding the index map with value-based bit-plane coding as presented in Section 7.4

Chapter 1 – Introduction

1.1 The addressed problems

Color-quantized images [Orchard91] are widely used in various applications especially Internet applications nowadays to reduce communication bandwidth and storage requirement. A color-quantized image is generally represented with a color index map each element of which serves as an index to select a color from a predefined set of colors to represent the color of a pixel in the image. The predefined set of colors is called a palette. Color-quantize image can be generated with common color quantization algorithms such as median-cut [Heckbert82], center-cut [Gervautz90], octree [Joy93] and 3D frequency diffusion [Lo03]. They are also referred to as color-indexed images.

To reduce the size of a color-indexed image further, lossless compression techniques are generally used because the index used to pick a particular palette color must be exact in decoding. A minor difference between two index values may result in a serious color shift.

Predictive coding technique is widely used in lossless compression. In fact, most lossless coding algorithms such as CALIC [Wu97] and JPEG-LS [Weinberger00] are based on predictive coding and entropy coding techniques. CALIC and JPEG-LS raster-scan an image and use the intensity values of some encoded local pixels to predict the intensity value of the pixel being encoded. The estimation error between the actual value and the estimated value is then encoded with entropy coding algorithms [Moffat95] to compress the image. In general, the smaller the average prediction error over the image, the higher compression ratio can be achieved.

However, predictive coding techniques do not perform well when they are used to encode a color-quantized image. Encoding a color-quantized image implies encoding its color index map. In general, a palette is generated in a way without concerning the order of the colors in the resultant palette. Accordingly, the numerical values of two indices that point to similar colors may be very different. Most predictors assume that neighboring pixels have similar attributes and, when these attributes are quantitatively measured, their values are similar. This assumption is valid for the intensity values of an image, but not for the index values in a color index map due to the aforementioned reason.

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. In general, palette reordering attempts to minimize the index difference between adjacent pixels such that the prediction error would be as small as possible [Hadenfeldt94]. Various reordering methods were proposed for this purpose. Some of them assign indices to palette colors based on the attributes of the palette colors [Zaccarin93] or the distance among the palette colors [Po94]. Some of them assign indices to palette colors based on the number of occurrences of having two particular palette colors in two spatially adjacent pixels [Memon97, Zeng00, Pinho04, Battiato01]. All of them can effectively improve the compression rate when their outputs are encoded with JPEG-LS.

Inspired by these palette reordering methods, we proposed a new reordering method in this work. This reordering method is named as adaptive palette reordering as it raster scans the image and adaptively reorders the palette based on both the palette and the index map. As a result, it is able to produce an index map of very low zero-order entropy and little spatial correlation. The index map can then be encoded with well-developed JPEG-LS or any other standard lossless coding techniques such as JBIG and JPEG-2000 efficiently.

The index map can also be encoded by some other non-standard coding schemes. A coding scheme was developed in this work based on the idea of bit-plane coding and context-based arithmetic coding to encode the output index map. The resultant coding performance is superior to those achieved by other state-of-art lossless image compression methods including those using techniques other than palette reordering [Po94, Spira01, Zaccarin93, Memon97, Zeng00, Pinho04, Battiato01].

1.2 Organization of this thesis

The proposed adaptive palette reordering technique is composed of two basic functional components including *index prediction* and *color reordering*. These two components can be realized with different schemes and, accordingly, their different combinations result in different adaptive palette reordering methods. Chapter 3 shows one of the realizations of adaptive palette reordering. Simulation results are provided to show its outstanding performance in producing a new index map of lower variance, lower zero-order entropy and lower spatial correlation as compared with other conventional palette reordering methods.

Chapter 4 puts its focus on the realization of *index prediction* in adaptive palette reordering. Besides the one used in the example presented in Chapter 3, *index prediction* can be realized with different schemes. In Chapter 4, four of them are studied and their prediction performance is evaluated. These schemes exploits MED[Weinberger00] or GAP[Wu97] to carry out the prediction in either the index plane or individual color intensity planes. Performance is evaluated in terms of various measures such as the entropy and the variance of their resultant reindexed index maps.

The focus of Chapter 5 is on the realization of *color reordering* in adaptive palette reordering. Five color reordering schemes are evaluated in Chapter 5 to see how well they can support adaptive palette reordering. In particular, we have two distance-oriented sorting schemes, one history-oriented scheme and two hybrid mode sorting schemes in our evaluation. Simulation results are presented. Performance is again evaluated in terms of both zero-order entropy and variance of the reindexed index maps.

In adaptive palette reordering, it collects statistical information in the course and uses it as supporting information to improve the performance of color reordering at a later stage. The information is stored in a data structure called DF-Table in our study. The DF-Table is immature when it contains a lot of zero entries, and this happens when the number of processed pixels is too small. Two solutions are proposed in Chapter 6 to solve this problem. Their performance is also compared and reported in Chapter 6.

Adaptive palette reordering produces a reindexed index map of low entropy, low variance and little spatial correlation. This index map can be encoded with various index encoding schemes easily. In Chapter 7, six different index encoding schemes are proposed to encode the output of adaptive palette reordering. Simulation results are presented to compare the compression performance of various lossless image coding methods including those exploit palette reordering and those do not. In other words, this chapter presents a direct comparison between the proposed adaptive-palette-reordering-based image coding methods and state-of-art lossless coding methods in terms of output bit rate.

Chapter 8 provides a brief summary of the work we have done. Some possible directions of the extension of the work are also suggested in this final chapter.

Chapter 2 – Literature Reviews

2.1 Introduction

This chapter provides reviews on some existing work that is relevant to our work. In particular, Section 2.2 provides a review on lossless image coding standards. In general, a color-indexed image is composed of an index map and a palette. The index map can be treated as a grey level image and then encoded with any lossless image coding techniques. A off-the-shelf lossless image coding standard is a natural choice for coding a color-indexed image.

In Section 2.3, some palette reordering algorithms are introduced. These algorithms reorder the palette of a color-indexed image such that the output index map associated with the reordered palette can be encoded with standard lossless image codecs such as JPEG-LS [JPEG99] and lossless JPEG-2000 [JPEG00] more effectively.

Color-indexed images can be encoded directly without reordering their palettes. In that case, off-the-shelf image coding standards may not work effectively and dedicated coding techniques have to be used for removing the redundancy. Section 2.4 presents a review on some state-of-art lossless image coding methods which do not solely or even do not rely on palette reordering.

2.2 Lossless image coding standards

The Joint Photographic Experts Group (JPEG) is a joint ISO/ITU committee responsible for developing standards for continuous-tone still-picture coding. It deals with both lossy and lossless standards for image compression. There are two standards for lossless image coding and both of them are based on predictive coding.

Predictive coding makes use of the property of natural images to achieve compression. The basic idea of predictive coding is to predict a pixel's intensity value with its neighbors' and then encode the prediction error with entropy coding. In general, intensity values of neighboring pixels are similar and hence the prediction error can be very small. This removes the redundancy of the image.

The old lossless standard of JPEG is referred to as "JPEG lossless" [Wallace91]. It provides eight different predictive schemes for a user to select. The new standard is referred to as JPEG-LS [JPEG99]. Medium Edge Detector (MED) is employed in its predictor to handle both edge and smooth regions. In particular, neighboring processed pixels are used to determine the current pixel value. An error compensation scheme is then applied to refine the predicted value. Finally, the prediction error is encoded with Golomb code [Elias75, Golomb66].

JPEG 2000 is a wavelet-based image compression standard [JPEG00]. It was created with the intention of replacing the previous DCT-based JPEG standard. It can operate at a higher compression ratio without generating the blocky and blurry artifacts introduced by the DCT-based JPEG standard. It provides both lossless and lossy compression in a single compression architecture. Lossless compression is provided by the use of a reversible integer wavelet transform and a quantization step size of 1. All bit planes have to be encoded by the Embedded Block Coding with Optimized Truncation (EBCOT) scheme [Taubman00] in lossless JPEG-2000.

Besides international coding standards like JPEG-LS and lossless JPEG-2000, there are some other defacto lossless image coding standards such as GIF and PNG. CompuServe introduced the GIF format in 1987 to provide a color image format [GIF87], and it is now one of the most popular formats on the Internet. The LZW compression scheme [Ziv78, Welch84] is used by GIF to compress an image. The most updated format version is GIF89a [GIF89a]. It supports transparency and interlacing.

Portable Network Graphics (PNG) format is an image format for storing bitmapped images on computers [PNG95]. It was designed to replace the GIF format in 1995 when royalties were required for Unisys' patent on the LZW compression method used in GIF. The compression scheme used in the PNG format is again based on predictive coding.

One can see that all these image coding standards were basically designed for encoding natural gray level images or color images. There is spatial correlation among the intensity values of neighboring pixels in a natural image and hence one can make use it to remove the redundancy with a predictive coding technique. In general, in the index map of a color-indexed image, the index values among neighboring pixels do not bear a similar extent of spatial correlation. The compression performance is generally not attractive when these coding standards are exploited to encode the index map directly.

2.3 Palette reordering

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. In general, palette reordering attempts to minimize the index difference between adjacent pixels such that the prediction error would be as small as possible.

Various reordering methods were proposed for this purpose. Basically they can be divided into two major categories. The first category is palette-based. They assign indices to palette colors based on the attributes of the palette colors [Zaccarin93, Po94] or the distance among the palette colors [Spira01]. The second category is index-map-based. They assign indices to palette colors based on the number of occurrences of having two particular palette colors in two spatially adjacent pixels [Battiato01, Memon96, Pinho04, Zeng00]. All of them can effectively improve the compression rate when their outputs are encoded with lossless image coding standards such as JPEG-LS and lossless JPEG-2000.

2.3.1 Palette-based palette reordering methods

Palette-based reordering methods extract information from the palette and then make use of it to reorder the palette. They may consider the luminance intensity values of the palette colors, or the Euclidean distance between the palette colors. The index map is not taken into account when they reorder the palette. Accordingly, it is not necessary to scan the index map for the reordering. As compared with index-map-based reordering methods, they are fast and the complexity is much lower, but the performance is generally poorer.

Zaccarin and Liu's method [Zaccarin93] adopts a straightforward approach to permute a given color palette. It is developed based on the assumption that pixels in a local region have similar luminance value and hence colors of similar luminance values should have similar indices. In their work, the luminance is defined as $Y=0.299R + 0.587G + 0.114B$, where R, G and B denote the intensity values of the red, the green and the blue components respectively. Palette colors are sorted by their luminance values. This method is simple and fast.

The development of Po and Tan's method [Po94] is based on an observation that pixels of similar colors are generally close to each other in natural images. In other words, it considers the similarity of colors instead of the similarity of the luminance of colors. The method assigns index value zero to the palette color of the lowest energy. The remaining indices are then assigned to the remaining palette colors one by one. In practice, it assigns index value j to the color which is the closest to the color assigned index value j-1 in terms of Euclidean distance in the color space.

Spira and Malah's method [Spira01] is based on a similar assumption that objects in an image are constructed with similar colors. Accordingly, the method tries to reorder the palette colors in a way that, when one visits the palette colors sequentially in the color space according to the index values and then goes back to the starting palette color, the path he travels is the shortest. This turns a palette reordering problem into a traveling salesman problem (TSP). The complexity for reaching an optimal solution is considerably high, and hence a suboptimal solution is pursued instead by making use of the Farthest Insertion Algorithm (FIA) [Lawler85].

2.3.2 Index-map-based palette reordering methods

Index-map-based reordering methods extract statistical information from the index map to reorder the palette. The exact colors in the palette itself are seldom taken into account in these methods.

The main idea behind index-map-based reordering methods is that colors that occur frequently close to each other should have close indices. If one can reorder the colors to minimize the average index difference between adjacent pixels in the index map, there will automatically be a gain in compression performance when predictive coding is used.

To achieve this, index-map-based reordering methods scan the index map to collect corresponding data before reordering the palette. The data collected can be the frequency of the occurrence that a pixel with color *i* is spatially adjacent to a pixel with color *j* in the index map, or the frequency of the occurrence that a pixel with color *i* is used to predict the color of a pixel with color *j*. Without losing generality, these data are denoted as C(*i,j*) in this section for elaborating how these reordering methods work.

As a matter of fact, by defining the cost as a function of C(*i,j*), an optimal reordering solution for achieving the minimum cost can be found with a full search. However, it is generally impractical because the number of possible permutations of *N* objects is *N*! and hence the computation complexity is extremely high. Many sub-optimal solutions with much lower complexity were proposed by different researchers accordingly.

Memon and Venkateswaran [Memon96] proposed two methods to reassign indices to palette colors by trying to minimize objective function $J = \left\{ \sum_{i=1}^{N} \sum_{j=1}^{N} C(i, j)|i - j| \right\}$, where *N* is the palette size. One of them is based on Simulated Annealing (SA) while the other is based on a heuristic approach named as Pairwise Merge (PM) heuristic.

In the SA-based approach, the indices of two randomly selected palette colors are interchanged and the cost is reevaluated to check if the exchange is justified. The exchange will be confirmed if it is. Otherwise it will be rejected conditionally. The likeliness of being rejected increases as the number of attempts of interchange increases. Attempts are repeated until a particular termination criterion is satisfied. The realization complexity is huge.

The PM heuristic approach was proposed to reduce the complexity. The idea behind this approach is to merge two selected ordered sets of colors in a limited number of ways and then pick the merge that minimizes cost function $\sum_{i=1}^{M}\sum_{j=1}^{M} C(i,j)|i-j|$, where i, j are the indices in the resultant ordered set and M is the total number of the colors in the resultant ordered result. The realization complexity is significantly reduced but its performance is not significantly lowered.

Zeng, Li and Lei [Zeng00] proposed a palette reordering method of even lower complexity. Palette colors are selected one by one to construct an ordered list. The list is initialized to be the pair of palette colors which are most frequently located adjacent to each other. The list grows by attaching one of those palette colors not in the list to either the left or the right end of the list. The picked color and the picked end should maximize $\sum_{c_j \in L} \log_2\left(1+\frac{1}{d(c_i,c_j)}\right)\cdot C(c_i,c_j)$ for all $c_i \notin L$, where $C(c_i,c_j)$ is the number of occurrences that palette colors c_i and c_j are neighbors in the image, L is the current ordered list, and $d(c_i,c_j)$ is the position displacement between palette colors c_i and c_j in the expanded list after c_i is attached to list L. Indices are assigned to the ordered list accordingly after all palette colors are included in the list. This method tries to reduce the overall index difference of adjacent pixels in the resultant index map.

Pinho and Neves [Pinho04] made a theoretical analysis on Zeng et al.'s Method [Zeng00] for the case of Laplacian distributed differences of neighboring pixels. The way how to select a color to build the ordered list was then modified. It was found that, under the Laplacian model, the picked color should maximize $\sum_{c_j \in L} C(c_i,c_j)$ for all $c_i \notin L$ and the picked end

should be determined by the sign of $\Delta = \sum_{c_j \in L} \left(M + 1 - 2p_{c_j}\right) \cdot C\left(c_i, c_j\right)$, where $p_{c_j} \in \{1,2...M\}$ is the position of c_j in list *L*, and *M* is the number of palette colors in list *L*. The selected c_i should be attached to the left end of list *L* if Δ>0. Otherwise it should be attached to the right end.

Battiato, Gallo, Impoco and Stanco's method [Battiato01] constructs a weighted graph based on $C\left(c_i, c_j\right)$ and then reformulates the reordering problem as a problem of finding the Hamiltonian path of maximum weight in the weighted graph. To solve this problem, they proposed a greedy strategy to find a sub-optimal solution by sequentially adding the heaviest nonvisited edges to the path. The growth of the path starts with the edge of the largest weight.

2.4 Coding methods which are not based on palette reordering

Kuroki, Yamane and Numa's method [Kuroki04] is actually a preliminary version of adaptive palette reordering. For a particular pixel, the palette colors are sorted by their distance to the color of one of the pixel's neighbors, and the position of the pixel's color in the sorted queue determines the new index of the pixel's color. In this method, the probability of encountering a particular palette color is not taken into account in the adaptation of the palette and hence the performance is limited.

Natale, Desoli, Giusto and Vemazza's method [Natale89] was originally proposed for coding vector-quantized images but the same idea can be applied in compressing color-quantized images. To handle color-indexed images, this method predicts the color of a pixel with its four neighboring pixels and then selects a subset of palette colors based on the prediction result to form a small palette. The pixel's color can be encoded with either the small dynamic palette or the original static palette to save bits.

Arnavut and Sahin adopted an approach [Arnavut06] similar to Battiato et al.'s method [Battiato01] to reindex the palette with a sub-optimal solution for traveling salesman problem. Unlike Battiato et al.'s approach, a block-sorting transformation (e.g. Burrows-Wheeler transformation (BWT) [Burrows94] or the linear order transformation (LOT) [Arnavut99]) using inversion ranks [Arnavut05] is further applied to the reindexed output before entropy coding is performed.

In Chen, Kwong and Feng's method [Chen02], a binary-tree structure of colors is constructed. The tree is traversed in a specific order during encoding. At each node, the colors of its child nodes and the locations of the pixels associated with the child nodes are encoded. The encoding of these pixel locations is performed by a context-based arithmetic encoder with variable size contexts. Pinho and Neves's method [Pinho05] is an improved version of [Chen02] in which the context adaptation model is modified. Both methods can provide an average output bit rate lower than 4 bpp. This implies a compression ratio higher than 2:1.

Ratnakar's method [Ratnakar98] classifies patterns of neighborhood pixels to predict and code a pixel. The prediction rules are adaptively learned during the coding process itself. An average output bit rate close to 4 bpp can be achieved.

The methods presented so far were mainly proposed or developed for handling color-indexed images. Recently, Robinson proposed a universal codec for coding images of various kinds [Robinson06]. This codec exploits the correlation among different color channels during compression and the achieved output bit rate is around 4.5 bpp when coding color-indexed images. Being a recently-proposed state-of-art codec, it is also included in our simulation study as a reference for comparison.

Chapter 3 – A framework of adaptive palette reordering

3.1 Introduction

As mentioned in Chapter 2, various reordering methods have been proposed to make the resulting color index map more suitable for predictive coding. Some of them assign indices to palette colors based on the attributes of the palette colors [Zaccarin93] or the distance among the palette colors [Po94]. Some of them assign indices to palette colors based on the number of occurrences of having two particular palette colors in two spatially adjacent pixels [Memon97, Zeng00, Pinho04, Battiato01]. In practice, all of them can effectively improve the compression rate when their outputs are encoded with JPEG-LS [Weinberger00].

In general, when a palette reordering method is exploited, it is required that the resultant palette is a static palette shared by all elements in the resultant index map such that the index map can be viewed as the original color image directly with the help of the palette. The resultant property is obviously good for convenience and transparency. However, from the compression point of view, this constraint is an additional burden to the codec and lowers the coding efficiency to a certain extent. For example, the zero-order entropy of the resultant index map cannot be reduced under this constraint as a bijective mapping has to be used in the reordering process.

By considering this, an adaptive palette reordering method is proposed in this chapter. This method does not take the aforementioned constraint into account and hence gets rid of the burden. It raster scans the image and adaptively reorders the palette to produce an index map of lower zero-order entropy. By so doing, the entropy of the resulting index map can be significantly reduced, and can then be easily encoded with a number of standard lossless coding techniques such as JPEG-LS and JPEG-2000 efficiently. In terms of zero-order

entropy, the resultant coding performance achieved by the proposed method is much lower as compared with other conventional lossless color-indexed image compression methods [Po94, Spira01, Zaccarin93, Memon96, Memon97, Zeng00, Pinho04, Battiato01].

3.2 Adaptive palette reordering

The proposed method processes a given color index map and its associated palette to generate a dynamic palette and a reindexed index map. The palette is dynamic in a way that the indices assigned to the palette colors are pixel position dependent and the assignment changes in the course of the processing. Accordingly, each element in the resultant index map is used as an index to find a color in the palette corresponding to the current pixel position. As a consequence, the resultant index map cannot be viewed as the original color image directly with the help of any static palette in our case.

Since the palette is reordered adaptively, the proposed reordering method is referred to as adaptive palette reordering method so as to discriminate it from the conventional palette reordering method 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 $\Omega=\{\vec{c}_k \mid k=0,1\ldots N-1\}$, where N is the size of the palette. 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

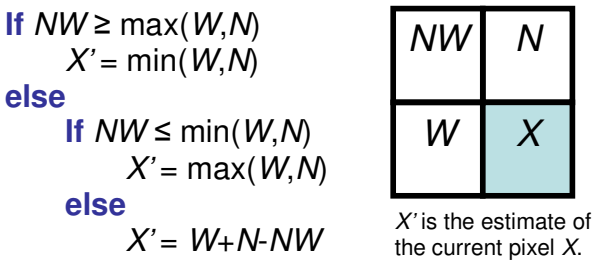


Figure 3.1 MED estimation scheme used in JPEG-LS

using a MED predictor. MED is used in JPEG-LS [Weinberger00] and its operation can be summarized as in Figure 3.1.

Suppose the prediction results of the red, green and 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 quantized with palette Ω. Let the quantization result be $\vec{c}_p$, the p^{th} palette color in Ω. $\vec{c}_p$ could be different from the original color-indexed color of the pixel, $\vec{c}_r$, which is assumed to be the r^{th} palette color in Ω without losing the generality.

In the proposed scheme, the occurrence of this discrepancy is recorded and cumulated for improving the prediction performance in the future. In particular, a table is constructed for storing the values of $\{H(m,n) | m,n=0,1\ldots N-1\}$, where $H(m,n)$ is defined as the number of occurrences when the quantized predicted color and the original color of a pixel are, respectively, $\vec{c}_m$ and $\vec{c}_n$. All $H(m,n)$ values are initialized to zero at the very beginning and the table is updated after a pixel is processed. For reference, this table is referred to as *discrepancy frequency table* (DF-Table) hereafter.

Table of {H(*m*,*n*)}

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
k	0	1	2	3	4	5	6	7
H(0,k)	29	7	6	5	4	3	2	1
H(1,k)	0	88	1	2	0	0	0	1
H(2,k)	0	2	65	1	2	1	0	0
H(3,k)	0	2	10	56	1	1	0	1
H(4,k)	3	1	0	**8**	8	8	0	0
H(5,k)	0	0	3	2	3	23	5	1
H(6,k)	0	0	0	1	1	2	23	2
H(7,k)	0	2	0	3	0	2	6	9

(a)

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
Prediction Error, $\|\vec{c}_k - \vec{v}(i,j)\|^2$	0.6	0.2	0.3	0.3	0.2	0.1	0.2	0.1

(b)

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
New index	3	4	7	2	1	0	6	5

(c)

Figure 3.2 Example of how to assign indices to a dynamic palette when $\vec{v}(i,j)$, $\vec{c}_4$ and $\vec{c}_3$ are, respectively, the predicted, the quantized predicted and the real colors: (a) current status of the DF-Table, (b) given prediction error $\|\vec{c}_k - \vec{v}(i,j)\|^2$ in the example, (c) index assignment of the proposed method

After $\vec{v}(i,j)$ and its quantization result $\vec{c}_p$ are determined, the colors in palette Ω are adaptively reordered based on H(p,k) and $\|\vec{c}_k - \vec{v}(i,j)\|^2$ for k=0,1…N-1. In particular, $\vec{c}_k$'s are sorted according to the values of {H(p,k)|k=0,1…N-1} in descending order. If there exist two different colors $\vec{c}_l$ and $\vec{c}_d$ such that H(p,l)=H(p,d), $\vec{c}_l$ and $\vec{c}_d$ will be sorted according to their Euclidean distances to $\vec{v}(i,j)$. 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, H(*p,r*) is incremented by 1 to update the frequency count of this event.

For each pixel, 3 MED prediction processes, an *N*-codeword VQ process and a sorting process are required. In practice, the sorting effort can be neglected as a sequence of $\vec{c}_k$'s which are sorted by {H(*p,k*)| *k*=0,1…*N*-1} can be easily updated when H(*p,r*) is updated after processing a pixel. As all we need is the position of $\vec{c}_r$ in the queue, in most cases it is not necessary to sort $\vec{c}_k$'s by $\| \vec{c}_k - \vec{v}(i,j) \|^2$.

Figure 3.2 shows an example of how an index is adaptively determined for a pixel when the current status of H(*m,n*) is shown in Figure 3.2a. In this example, the palette Ω is of size 8. Assume that the predicted color, the quantized predicted color and the real color of the pixel are, respectively, $\vec{v}(i,j)$, $\vec{c}_4$ and $\vec{c}_3$. In such a case, $\{\vec{c}_k | k=0,1\ldots7\}$ are sorted by H(4,*k*) and then by $\| \vec{c}_k - \vec{v}(i,j) \|^2$. It results in $\{\vec{c}_5, \vec{c}_4, \vec{c}_3, \vec{c}_0, \vec{c}_1, \vec{c}_7, \vec{c}_6, \vec{c}_2\}$. The position of $\vec{c}_3$ in the sorted sequence is 2 and hence the output index for $\vec{c}_3$ is 2. We note that the sequence order, and hence the index, is counted from 0 here.

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 use a fixed common palette such as Ω to generate a full-color image, or even (ii) the full-color image directly.

Figure 3.3 summaries the flow of the proposed adaptive palette reordering method in pseudo code.

Initialize DF-Table {H(m,n)} by H(m,n)=0 for m,n=0,1…N-1.
Raster scan the image
FOR each pixel (i,j)
Predict the red component of pixel (i,j) with MED in the red color plane to get r(i,j).
Predict the green component of pixel (i,j) with MED in the green color plane to get g(i,j).
Predict the blue component of pixel (i,j) with MED in the blue color plane to get b(i,j).
Quantize $\vec{v}(i,j) = (r(i,j), g(i,j), b(i,j))$ with the reference palette Ω.
% Assume the quantization result is $\vec{c}_p$, *the* p^{th} *palette color in reference palette Ω*
Sort palette colors in Ω ≡{ $\vec{c}_k$ |k=0,1…N-1} by H(p,k) and then by $\| \vec{c}_k - \vec{v}(i,j) \|^2$ and then by k.
% The sorted palette colors forms the transient version of the palette
% Assume the real color of pixel (i,j) is $\vec{c}_r$, *the* r^{th} *palette color in reference palette Ω*
Assign the position of $\vec{c}_r$ in the current sorted palette color queue to be the new index of $\vec{c}_r$.
Update {H(m,n)} by H(p,r)++.
END

Figure 3.3 Pseudo code of the proposed adaptive palette reordering method

3.3 Properties of the reindexed output

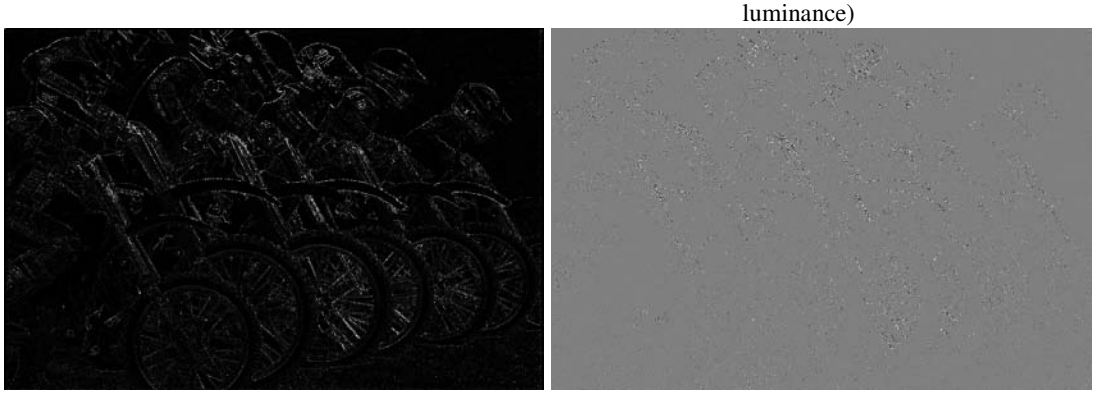
Simulations were carried out to evaluate the performance of the proposed reordering method. A set of 24 standard full-color testing images as shown in Figure 3.4 were color-quantized to 256-color images with MATLAB function RGB2IND. No dithering was performed in the quantization. The color-quantized images were then processed to produce their corresponding reindexed index maps with the proposed adaptive palette reordering method for analysis.

As an example, Figure 3.5a shows the color quantization result of one of the testing natural full-color images. Figure 3.5b shows a typical color index map of Figure 3.5a. This index map is associated with a palette whose colors are sorted by luminance (i.e. palette Ω, the reference palette in our method). For reference, we refer to this index map as the reference index map of the color-quantized image. Figure 3.5c is the processing result of the proposed adaptive palette reordering method. One can see that very few indices are of large values in the reindexed index map.

Figure 3.4 Original Kodak full-color images (Refers as image 1 to image 24, from top-to-bottom and left-to-right)

Figure 3.6 shows the histograms of Figure 3.5b and Figure 3.5c. One can see that the peak of the histogram of our reindexed output is very sharp while that of the reference index map is not. As a matter of fact, the index histogram of our processing result, approximately, 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 result is significantly reduced as compared with that of the reference. In particular, the zero-order entropy values of the reference index map and our processing result are, respectively, 7.176 and 4.238 bpp (bits per pixel).

(a) Color-quantized image (b) Reference index map (colors are sorted by luminance)



(c) Our processing output (d) Mapping result of (c) using mapping (3.1)

Figure 3.5 Processing result of the proposed adaptive reordering method

As making the resultant reindexed index map directly displayable is not the concern in our case, the proposed reordering method does not exploit a bijective mapping as in conventional reordering methods [Po94, Spira01, Zaccarin93, Memon96, Memon97, Zeng00, Pinho04, Battiato01] to reindex the palette colors, which allows it to change the sorted index distribution and reduce the zero-order entropy remarkably. Figure 3.7 shows the reduction in zero-order entropy that the proposed method can achieve when processing the color quantization results of the testing images shown in Figure 3.4. As a bijective mapping function is used to reindex palette colors in conventional palette reindexing algorithms such as [Po94, Spira01, Zaccarin93, Memon96, Memon97, Zeng00, Pinho04, Battiato01], the zero-order entropy values of their reindexed results are exactly identical to the zero-order entropy value of the reference.

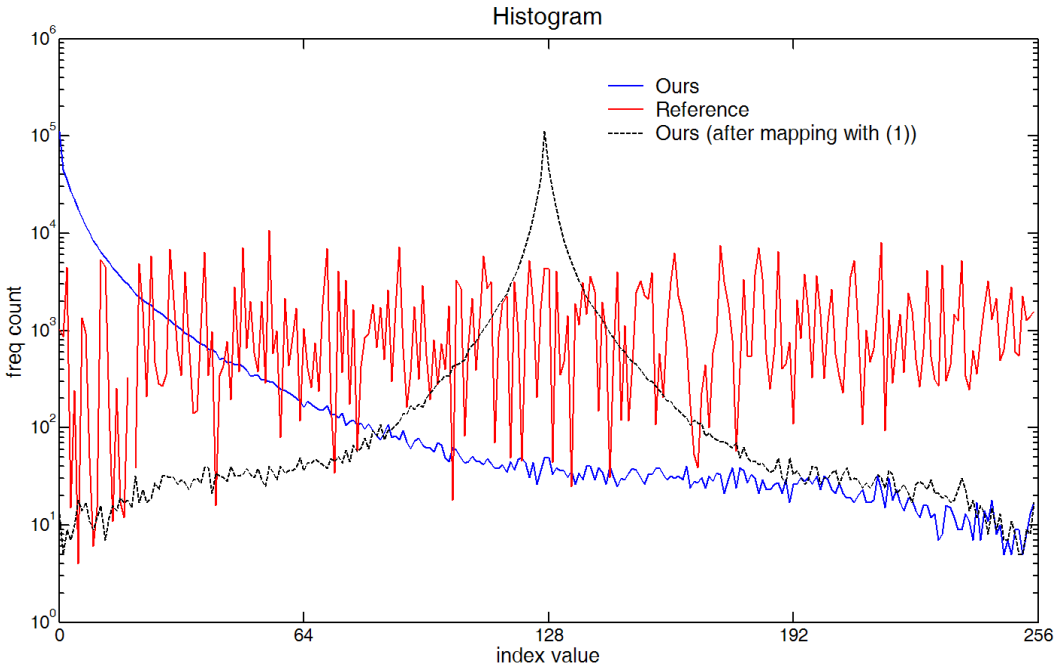


Figure 3.6 Histograms of color index maps of *Kodak-05*

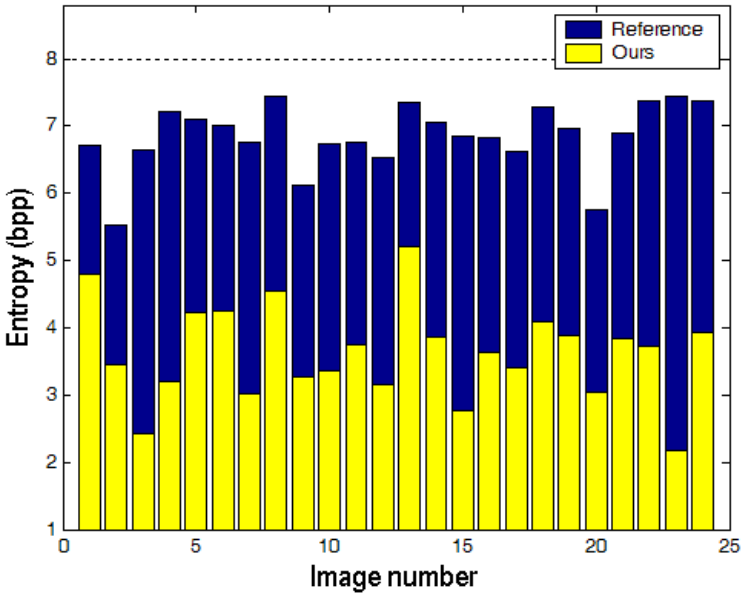
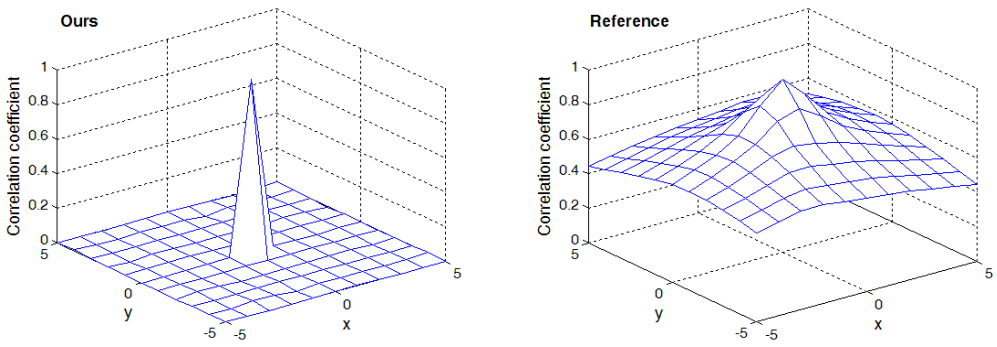


Figure 3.7 Reduction in zero-order entropy when the proposed reordering method is used



(a) (b)

Figure 3.8 Correlation between pixel (m,n) and pixel $(m+x,n+y)$ in (a) our reindexed output, and (b) the reference index map

In general, a significant number of indices in our processing result are of zero values. It drags the mean value of the indices to the zero side significantly, and hence large index values (e.g. those close to N) derivate from the mean very much. This results in a large variance of the indices. To solve this problem, the indices of the index map are further mapped to other values with a bijective mapping $M(\bullet)$ as follows.

$$M(i) = \lceil N/2 \rceil - 1 - (-1)^i \lceil i/2 \rceil \qquad \text{for } i = 0,1 \ldots N\text{-}1 \tag{3.1}$$

This mapping shifts the peak of the index distribution from 0 to $\lceil N/2 \rceil$-1. Accordingly, the maximum derivation of an index from the mean is more or less $N/2$, which reduces the variance of the indices. Note that this bijective mapping does not alter the lowered zero-order entropy of the resultant index map. The segmented curve in Figure 3.6 shows the histogram of the new mapping result and Figure 3.5d shows the mapping result of Figure 3.5c.

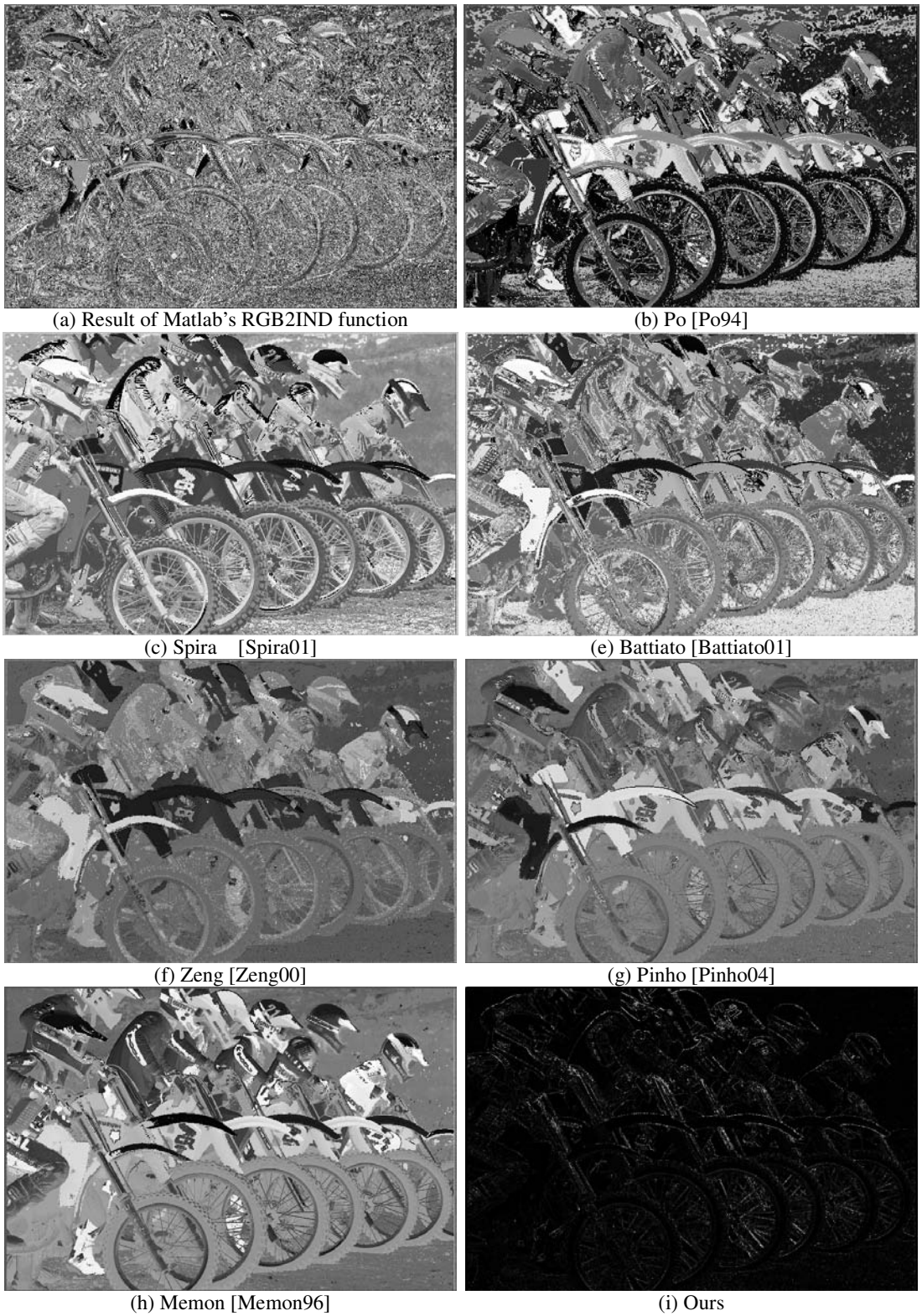
Another analysis was carried out to study the effectiveness of the proposed reordering method in eliminating the spatial statistical redundancy of indices. Figure 3.8 shows the correlation coefficient of the index values of two pixels which are (x,y) apart in an index map. It was evaluated with the results shown in Figures 3.5b and 3.5d. One can see that the remapped indices in the output of our approach are highly uncorrelated after mapping. This implies that most redundancy is removed by our reordering method. This removal helps a lot for one to encode the resultant index map efficiently.

This analysis shows that the proposed method can remove a lot of spatial correlation in the reindexed index map. In data compression, effectiveness in removing redundancy usually implies compression performance.

For comparison, Figure 3.9 shows some reindexed index maps generated with some conventional palette reordering methods. As shown in Figure 3.9, the energy retained in our reindexed index map is much lower as compared with that in the others. Table 3.1 shows the mean square values of the indices in different reindexed index maps while Table 3.2 shows the variances of the indices in different reindexed index maps. Here we note that our results presented in Table 3.2 are obtained with the help of the bijective mapping defined in eqn. (3.1). The small variance of the data in our outputs is a compression-friendly feature introduced by the proposed dynamic palette reordering method.

Figure 3.10 shows the histograms of the index map shown in Figure 3.9. Unlike other reindexed index maps, the histogram of our reindexed index map is very unequalized, which results in a low zero-order entropy.

Figure 3.11 shows the correlation among adjacent pixels in these reindexed index maps. One can see that a significant amount of correlation is retained in the outputs of the conventional static palette reordering algorithms while it is almost removed completely in our result.



(a) Result of Matlab's RGB2IND function
(b) Po [Po94]
(c) Spira [Spira01]
(e) Battiato [Battiato01]
(f) Zeng [Zeng00]
(g) Pinho [Pinho04]
(h) Memon [Memon96]
(i) Ours

Figure 3.9 Reordered index-maps of *Kodak-05* obtained with different methods

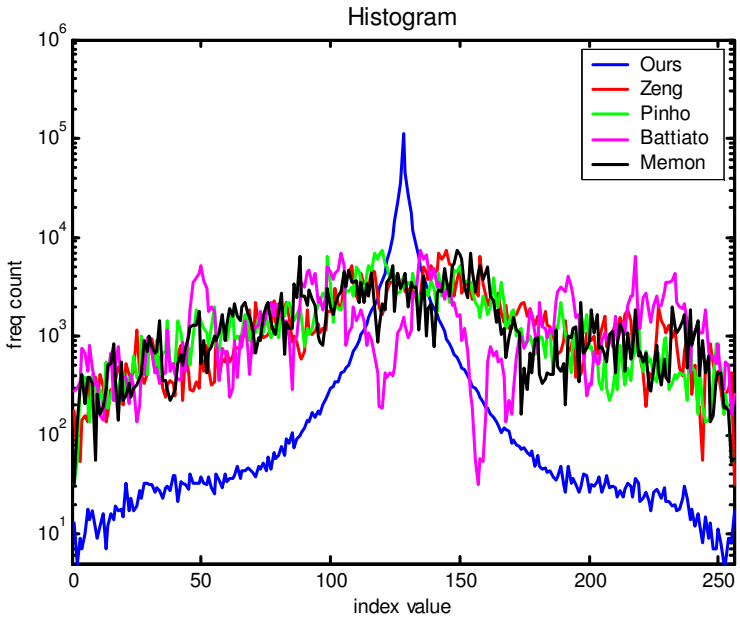


Figure 3.10 Histograms of different reordering results (*Kodak-05*)

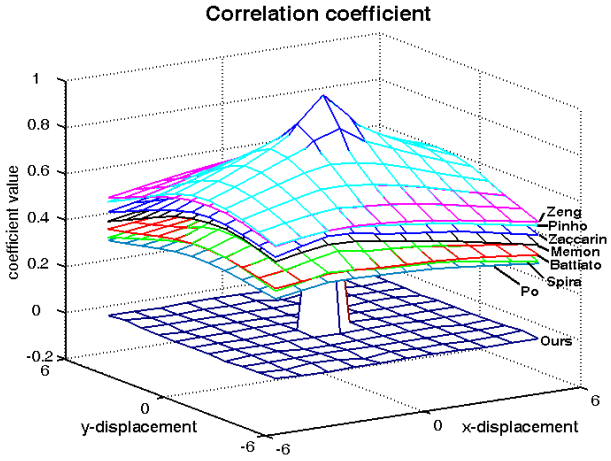


Figure 3.11 Performance of different algorithms in terms of correlation among pixels in their reindexed results (*Kodak-05*)

Group	Image	Size (pixel^2)	Palette reordering method						
			Po94	Spira01	Zeng00	Pinho04	Memon96	Battiato01	Ours
CG image	pool	510x383	3264.9	32086.0	16474.0	13505.0	16945.0	24862.0	49.8
	watch	1024x768	13651.0	27129.0	17420.0	16440.0	15871.0	24258.0	129.4
	water	1024x768	16546.0	20401.0	17892.0	20114.0	24342.0	23028.0	735.5
Natural image	Kodak 01	768x512	17403.0	23951.0	12358.0	16780.0	23743.0	23729.0	509.4
	Kodak 02	768x512	17321.0	13722.0	22765.0	24967.0	17174.0	6561.2	193.9
	Kodak 03	768x512	13244.0	18683.0	13119.0	12943.0	19484.0	18459.0	53.9
	Kodak 04	768x512	19439.0	21966.0	20053.0	19330.0	18565.0	16875.0	92.9
	Kodak 05	768x512	14958.0	22761.0	20570.0	17827.0	19141.0	22489.0	585.1
	Kodak 06	768x512	22120.0	24824.0	15255.0	15247.0	25367.0	23393.0	360.1
	Kodak 07	768x512	11375.0	13548.0	21687.0	11919.0	23470.0	16878.0	197.1
	Kodak 08	768x512	17796.0	20235.0	19384.0	20621.0	23641.0	22154.0	555.7
	Kodak 09	768x512	9956.7	29466.0	20969.0	10791.0	15770.0	17224.0	135.1
	Kodak 10	768x512	12723.0	20369.0	20861.0	22242.0	15459.0	18815.0	224.5
	Kodak 11	768x512	16342.0	18570.0	25170.0	28440.0	23991.0	23249.0	315.4
	Kodak 12	768x512	12779.0	34198.0	17183.0	23043.0	18752.0	18134.0	110.0
	Kodak 13	768x512	16249.0	22470.0	21638.0	15837.0	22619.0	20463.0	1202.8
	Kodak 14	768x512	13962.0	15828.0	20256.0	13425.0	28530.0	18215.0	278.2
	Kodak 15	768x512	12755.0	23743.0	16177.0	16249.0	24012.0	16501.0	77.9
	Kodak 16	768x512	16830.0	23727.0	14993.0	17500.0	27255.0	18552.0	181.4
	Kodak 17	768x512	13334.0	16882.0	17185.0	22560.0	12743.0	12204.0	232.1
	Kodak 18	768x512	16275.0	17694.0	30762.0	31107.0	27989.0	21631.0	468.4
	Kodak 19	768x512	15501.0	19188.0	28991.0	29021.0	15689.0	26163.0	208.0
	Kodak 20	768x512	22053.0	34122.0	9182.3	9170.1	12763.0	18871.0	210.2
	Kodak 21	768x512	22372.0	18671.0	12707.0	12944.0	14479.0	16616.0	344.4
	Kodak 22	768x512	16639.0	24568.0	27912.0	14674.0	16555.0	20351.0	272.8
	Kodak 23	768x512	17252.0	17698.0	17937.0	19620.0	20278.0	17556.0	64.8
	Kodak 24	768x512	14573.0	19360.0	20551.0	16003.0	23452.0	21593.0	581.0
		Average	15433.8	22068.9	19238.9	18234.0	20299.2	19586.1	310.0

Table 3.1 Mean square values of the indices in different reindexed index maps produced with different palette reordering methods

Group	Image	Size (pixel²)	Palette reordering method						
			Po94	Spira01	Zeng00	Pinho04	Memon96	Battiato01	Ours
CG image	pool	510x383	1800.3	1277.5	504.7	494.0	631.5	1477.5	48.6
	watch	1024x768	4064.8	3645.7	1350.2	1300.3	1473.1	2234.9	122.4
	water	1024x768	5536.3	4225.4	3547.4	3270.8	4948.3	6323.8	609.3
Natural image	Kodak 01	768x512	3873.5	3692.6	2515.3	2477.8	3814.3	5800.6	403.3
	Kodak 02	768x512	3817.1	2991.9	861.9	941.4	1212.2	3479.0	171.6
	Kodak 03	768x512	4465.6	3984.6	3225.5	3284.6	3531.3	5493.1	49.8
	Kodak 04	768x512	5412.5	4837.0	3082.2	3297.0	4375.8	4799.3	79.2
	Kodak 05	768x512	5906.3	3487.0	2353.9	2302.2	2656.0	4084.0	496.8
	Kodak 06	768x512	6008.0	4257.9	2887.3	3708.1	5475.4	6009.6	300.8
	Kodak 07	768x512	4283.8	4699.1	2647.6	2490.6	2834.3	4371.8	183.5
	Kodak 08	768x512	5206.9	4972.0	3801.6	3966.8	4820.7	4723.0	472.4
	Kodak 09	768x512	3100.6	2960.8	1352.0	1745.0	1812.2	3415.8	119.8
	Kodak 10	768x512	3765.6	3394.2	2271.4	2046.4	3381.8	5718.1	202.1
	Kodak 11	768x512	4600.6	4313.5	3076.1	3108.7	5838.7	5087.8	281.5
	Kodak 12	768x512	3902.2	5258.3	2297.5	2546.1	2677.0	4381.5	98.3
	Kodak 13	768x512	4837.1	4532.3	3352.8	3835.9	4240.7	4586.9	916.7
	Kodak 14	768x512	4914.5	3921.4	2354.9	2893.3	4083.9	5373.8	240.9
	Kodak 15	768x512	5058.8	5371.6	3156.9	2850.7	5356.5	5022.9	69.8
	Kodak 16	768x512	5250.9	5841.5	2695.5	3660.2	3552.3	5234.4	157.2
	Kodak 17	768x512	4883.1	3188.6	1856.6	1758.7	3344.8	4631.6	206.8
	Kodak 18	768x512	5878.9	3069.6	3506.7	3336.6	4311.4	4273.2	399.6
	Kodak 19	768x512	3671.0	4729.0	3327.6	4061.5	4363.4	6759.0	177.8
	Kodak 20	768x512	4541.1	6282.9	3206.5	3306.6	2560.0	2217.5	191.2
	Kodak 21	768x512	4596.6	3809.8	3904.8	4511.7	4122.6	4469.8	302.7
	Kodak 22	768x512	4538.4	3921.1	3554.8	3520.4	4206.3	5495.5	234.4
	Kodak 23	768x512	6104.8	3931.5	3771.6	3851.6	4080.7	6167.1	61.2
	Kodak 24	768x512	5304.2	3759.6	2726.0	2625.0	3176.3	5793.2	509.1
		Average	4641.6	4087.3	2710.7	2859.0	3588.2	4719.4	263.2

Table 3.2 Variances of the indices in different reindexed index maps produced with different palette reordering methods

Table 3.3 and table 3.4 show the upper bound of computational complexity of the proposed dynamic palette reordering algorithm in pre-processing stage and post-processing stage, respectively. There are 3 stages processes: Color Prediction, Color Quantization and Palette reordering. *N* is the size of palette Ω. The complexity is proportional to the number of colors in palette.

Operation	Stage	Operations / pixel
CMP	Color Prediction	6
ADD		6
CMP	Color Quantization	N
ADD		2N
CMP	Reordering	N

Table 3.3 The computational complexity of the proposed dynamic palette reordering algorithm (Pre-Processing)

Operation	Stage	Operations / pixel
CMP	Color Prediction	6
ADD		6
CMP	Color Quantization	N
ADD		2N
CMP	Reordering	N(N+1)/2

Table 3.4 The computational complexity of the proposed dynamic palette reordering algorithm (Post-Processing)

3.4 A generalized framework

An adaptive palette reordering method is proposed in Section 3.3. Unlike those conventional palette reordering methods [Po94, Spira01, Zaccarin93, Memon97, Zeng00, Pinho04, Battiato01], it reorders the palette on the fly when accessing the pixels in an image such that the indices to the colors in the palette is spatial variant. To complete the compression, an encoder is required to encode the indices in the end. The index encoder works with the proposed adaptive palette reordering method to form a coding system for compressing a color-quantized image.

Figure 3.12 shows the basic structure of such a coding system. Though a number of details of the proposed adaptive palette reordering method have been provided in the previous section, the whole system can be generalized with the simplified model shown in Figure 3.12. In particular, pixels are processed one by one. For each pixel, a prediction is performed to estimate its color. The prediction error and, if necessary, some other available information as well, is then exploited to reorder the color in the palette. The index to the color in the reordered palette is encoded after all.

The generalized structure shown in Figure 3.12 forms a framework for coding color-quantized images. As a matter of fact, based on this structure, a number of variants of the proposed palette reordering methods can be developed by varying the actual implementation of a particular functional block. In the following chapters, we will investigate some of these variants and evaluate the impacts of different factors.

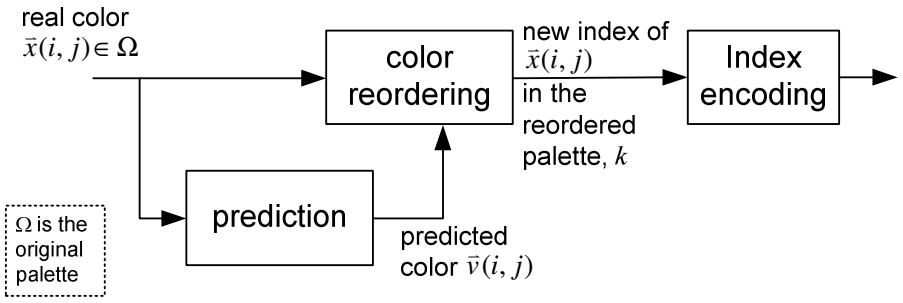


Figure 3.12 Structure of a codec using adaptive palette reordering method

3.5 Summary

An adaptive palette reordering method is proposed in this Chapter to reshape the statistical properties of a color index map. This method uses the color planes instead of the index plane to do prediction such that the color spatial property can be exploited. Unlike other reordering methods, this method adaptively reorders the palette based on both the palette and the index map to produce a new index map of low variance, low zero-order entropy and little spatial correlation.

Based on this proposed palette reordering method, a coding system for compressing color-quantized images is proposed. It forms a framework for one to develop other coding schemes by modifying the actual realization of its functional blocks. In the following chapters, we are going to investigate some of these variants and evaluate the impacts of different factors in the realization of this coding system.

Chapter 4 – Impact of prediction to adaptive palette reordering

4.1 Introduction

The proposed adaptive palette reordering method presented in Chapter 3 can be generalized with the model shown in Figure 4.1. This model forms a framework for carrying out adaptive palette reordering. In other words, one can just treat the method proposed in Chapter 3 as a particular example of the realization of adaptive palette reordering method. By using different approaches to realize the prediction module or the color reordering module in the framework, one can develop a number of variants of the proposed adaptive palette reordering method.

The MED predictor is exploited to realize the prediction module in the method proposed in Chapter 3. In particular, the prediction is carried out in the red, the green and the blue color intensity planes separately. In this chapter, we are going to explore some other variants for the realization of the prediction module. In particular, we will investigate the impact of (1) using Gradient Adjusted Predictor (GAP), the predictor used in CALIC codec [Wu97], instead of the MED predictor used in JPEG-LS [Weinberger00] and (2) carrying out the prediction in the index plane directly instead of individual color intensity planes.

This chapter is organized as follows. First, a brief introduction of the difference between MED and GAP is given in Section 4.2. An evaluation is then made in Section 4.3 to compare the prediction performance of MED and GAP in the case when the prediction is carried out in the index domain directly. In Section 4.4, we show their prediction performance in the case when the prediction is carried out in the intensity domain. The actual impact of these different prediction schemes to the performance of adaptive palette reordering is discussed in Section 4.5. A summary is finally given in Section 4.6.

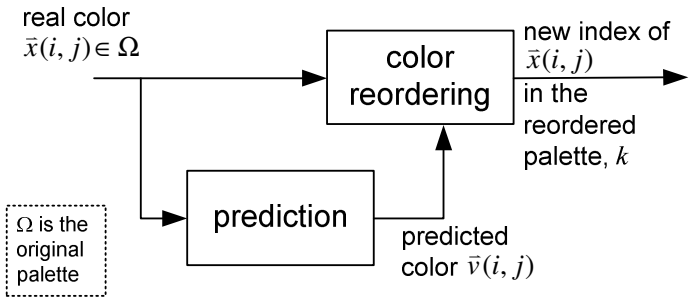


Figure 4.1 Structure of adaptive palette reordering

4.2 MED vs GAP in prediction

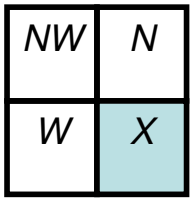
Median Edge Detection (MED) is a simple edge detector originally proposed for JPEG-LS. It tries to classify a pixel by detecting whether it is in a smooth region or on a vertical or horizontal edge. The detection is based on 3 neighboring pixels of the pixel of interest as shown in Figure 4.2. The detection result then determines the prediction result of pixel X.

The MED predictor is designed to be simple so as to reduce the realization complexity. It is used in the adaptive palette reordering method proposed in Chapter 3 to realize the prediction module.

As compared with MED, Gradient Adjusted Predictor (GAP) is a more complicated predictor in which a more sophisticated edge detection scheme is performed in the prediction. A larger support region of the pixel of interest is used to classify the edge that the pixel is on into more classes according to its nature.

Figure 4.3 shows the causal context template used for predicting the value of pixel x. The pseudo code shown in Figure 4.4 summaries the working principle of GAP. The predictor coefficients and the thresholds involved in the pseudo code were determined empirically [Wu97]. As compared with MED, the number of pixels covered by the support

```
If NW ≥ max(W,N)
    X' = min(W,N)
else
    If NW ≤ min(W,N)
        X' = max(W,N)
    else
        X' = W+N-NW
```



X' is the estimate of the current pixel *X*.

Figure 4.2 MED estimation scheme used in JPEG-LS

region increases from 3 to 7 to collect more information of the neighborhood of the predicted pixel. Besides, its edge detector is not only able to classify the orientation of an edge, but tell if it is a sharp edge, a strong edge or a weak edge as well.

GAP is the predictor used in CALIC. One can see that more effort is required to realize GAP than MED. This is expected as GAP is designed for a better performance. As a consequence, its prediction performance is expected to be higher in return.

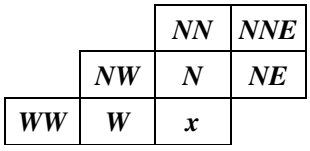


Figure 4.3 The causal context template used in CALIC for pixel ***x***

As a matter of fact, the pseudo code presented in Figure 4.4 is only the core part of GAP. The original version of GAP[Wu07] exploits some error feedback and compensation schemes to refine the prediction result so as to improve the performance. This refinement step is dropped when GAP is used in our study as the marginal improvement does not justify the required realization complexity of the refinement step. Accordingly, in this work, GAP is referred to as the version without refinement as it is presented in Figure 4.4.

Pseudo Code	*Remarks*
$d_v = \lvert I_W - I_{NW} \rvert + \lvert I_N - I_{NN} \rvert + \lvert I_{NE} - I_{NNE} \rvert$	Criteria for vertical direction edge
$d_h = \lvert I_W - I_{WW} \rvert + \lvert I_N - I_{NW} \rvert + \lvert I_{NE} - I_N \rvert$	Criteria for horizontal direction edge
$J = (I_W + I_N)/2 + (I_{NE} - I_{NW})/4$	
IF $(d_v - d_h > 80)$, $\hat{I} = I_W$	CASE: Sharp Horizontal Edge
ELSEIF $(d_v - d_h < -80)$, $\hat{I} = I_N$	CASE: Sharp Vertical Edge
ELSEIF $(d_v - d_h > 32)$, $\hat{I} = (J + I_W)/2$	CASE: Strong Horizontal Edge
ELSEIF $(d_v - d_h < -32)$, $\hat{I} = (J + I_N)/2$	CASE: Strong Vertical Edge
ELSEIF $(d_v - d_h > 8)$, $\hat{I} = (3J + I_W)/4$	CASE: Weak Horizontal Edge
ELSE $(d_v - d_h < -8)$, $\hat{I} = (3J + I_N)/4$	CASE: Weak Vertical Edge
END	

Figure 4.4 Pseudo code for classifying an edge with Gradient Adjusted Predictor (GAP) to predict a pixel value

4.3 Prediction in index domain

As mentioned in Chapter 1, it is difficult to make a prediction in the index plane of a color-quantized image as two similar colors can be of completely different index values in a palette. At the same time, two completely different colors can be of similar index values in a palette. Figure 4.5b shows a typical index plane for representing the color quantized image shown in Figure 4.5a. It is generated with Matlab's RGB2IND function. One can see that the spatial correlation among pixels that exists in the color intensity planes can hardly be found in the index plane. Accordingly, it makes prediction meaningless in the index plane to a certain extent.

To solve this problem, one can reorder the colors in the palette to establish a connection between the color luminance value and the index value. Since there is correlation among adjacent pixels in their luminance values, this reordering in palette colors is able to introduce a correlation among the indices of adjacent pixels.

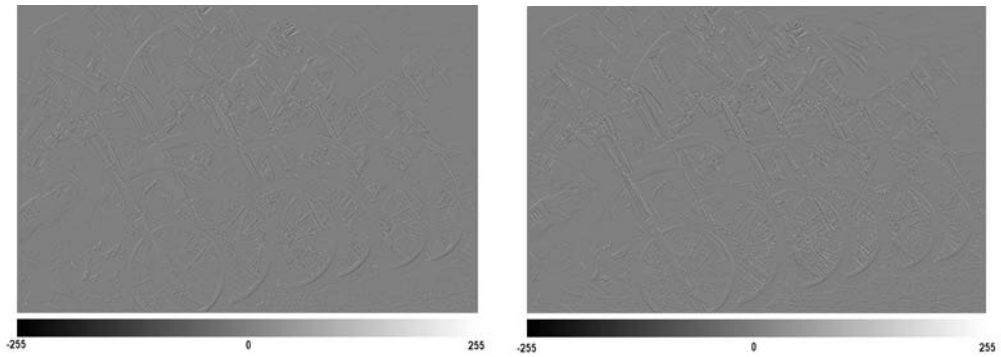
(a) Color-quantized image (b) Result of Matlab's RGB2IND function

(c) Luminance plane of (a) (d) Reference index map (colors are sorted by luminance)

Figure 4.5 Color-quantized image *Kodak-05* and some of its index maps: (a) color-quantized image, (b) the index map generated with Matlab's RGB2IND function, (c) luminance plane of (a), and (d) the index map associated with the palette sorted by luminance.

In our study, palette colors are sorted by their luminance values Y=0.299R+0.587G+0.114B, where R, G and B are, respectively, the intensity values of the red, the green and the blue color components. The sorted palette is then used as a reference palette to update the index map. Figure 4.5d shows the index map associated with the reference palette. As mentioned in Chapter 3, this index map is referred to as the reference index map.

The reference index map appears as a grey-level image and the pixels are now highly correlated. Figure 4.5c shows the luminance plane of Figure 4.5a for comparison. With this spatial correlation, prediction can be performed in the reference index plane.



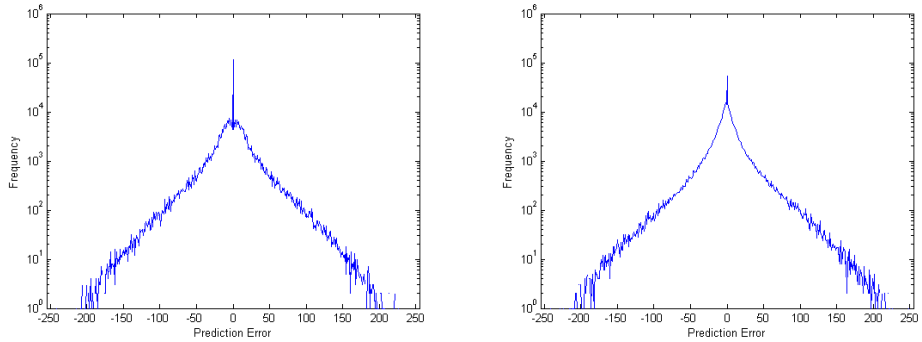
(a) MED (b) GAP

Figure 4.6 Index residue planes of *Kodak-05* when the prediction is done is the index plane directly (a) MED and (b) GAP

As a remark, we note that GAP may produce a floating-point prediction result. When a prediction is carried out in the index plane with GAP, the prediction result is actually the rounding result of the prediction result as an index must be an integer.

Simulation was carried out to evaluate the prediction performance of MED and GAP in predicting an index in the index plane. Figure 4.6a and Figure 4.6b show, respectively, MED's and GAP's prediction residue planes of *Kodak-05*. Here, the residue means the difference between the predicted index p and the real index r of a pixel. Apparently, the two residue planes are more are less the same.

Figure 4.7 shows the histograms of the residue planes shown in Figure 4.6. One can see that there are more pixels whose indices are exactly estimated in MED's prediction result. In particular, 29.3% of pixels are accurately predicted with MED while only 13.8% of pixels are accurately predicted with GAP. However, the variances of MED's and GAP's residue planes are, respectively, 705.17 and 694.58. In other words, the average performance of GAP is a bit better but the superiority is not that obvious. Similar observations were obtained in the simulation results when using other testing images. By considering these findings and the lower complexity of MED, MED seems better to be used when handling the index plane.



(a) MED (b) GAP

Figure 4.7 Histograms of Figure 4.6 (in log scale)

Table 4.1 shows some more data for comparing the prediction performance of MED and GAP in predicting a pixel in the index plane. Their performance is compared in three aspects including the energy of the index prediction error (E_I), the entropy of the index prediction error ($Entropy_I$) and the energy of the color prediction error (E_C). In particular, they are defined as follows.

$$E_I = 10\log_{10}\left(\sum_{(i,j)}\left(r(i,j)-p(i,j)\right)^2\right) \quad \text{in dB} \tag{4.1}$$

$$Entropy_I = \sum_{k=0}^{N-1}\left(\frac{n_k}{n_{all}}\right)\log_2\left(\frac{n_k}{n_{all}}\right) \quad \text{in bpp (bits per pixel)} \tag{4.2}$$

$$E_C = 10\log_{10}\left(\sum_{(i,j)}\left\|\vec{c}_r(i,j)-\vec{c}_p(i,j)\right\|^2\right) \quad \text{in dB} \tag{4.3}$$

where $r(i,j)$ and $p(i,j)$ are, respectively, the real index and the predicted index of pixel (i,j), n_{all} is the total number of pixels in an image and n_k is the number of pixels whose predicted indices are different from whose real indices by k. $\vec{c}_r(i,j)$ and $\vec{c}_p(i,j)$ are, respectively, the real color and the predicted color of pixel (i,j). They are determined by the real index $r(i,j)$ and the predicted index $p(i,j)$ of pixel (i,j). As a remark, all three color components of a color is normalized such that their intensity values are bounded in [0,1].

Group	Image	Size (pixel^2)	Energy of the index prediction error, E_I (dB)		Entropy of the index prediction error, $Entropy_I$ (bpp)		Energy of the color prediction error, E_C (dB)	
			MED	GAP	MED	GAP	MED	GAP
CG image	pool	510x383	71.095	70.273	2.149	2.816	29.278	37.275
	watch	1024x768	81.366	81.357	2.831	3.440	35.096	37.783
	water	1024x768	85.881	85.760	5.617	6.020	39.889	41.728
Natural image	Kodak 01	768x512	83.022	83.140	5.879	6.216	36.218	37.761
	Kodak 02	768x512	80.851	80.682	4.982	5.702	32.213	35.958
	Kodak 03	768x512	77.570	77.421	3.654	4.521	35.295	42.714
	Kodak 04	768x512	79.518	79.105	4.668	5.305	33.959	39.020
	Kodak 05	768x512	84.429	84.364	5.665	6.113	39.186	41.457
	Kodak 06	768x512	80.966	80.936	5.244	5.559	35.249	36.898
	Kodak 07	768x512	79.560	79.292	4.139	4.822	34.320	39.304
	Kodak 08	768x512	81.738	81.817	5.359	5.731	37.215	38.795
	Kodak 09	768x512	76.711	76.373	4.400	4.856	33.686	39.730
	Kodak 10	768x512	78.146	77.993	4.508	5.018	33.679	38.007
	Kodak 11	768x512	82.111	82.085	5.042	5.595	35.856	39.148
	Kodak 12	768x512	76.467	76.261	4.194	4.912	31.687	36.527
	Kodak 13	768x512	86.077	86.083	6.573	6.750	40.239	40.917
	Kodak 14	768x512	81.638	81.535	5.324	5.756	38.373	42.629
	Kodak 15	768x512	77.611	77.016	4.182	4.706	34.911	40.108
	Kodak 16	768x512	78.514	78.423	4.706	5.197	31.626	34.261
	Kodak 17	768x512	78.160	77.765	4.593	4.927	33.461	36.378
	Kodak 18	768x512	83.193	83.072	5.789	6.169	37.414	39.411
	Kodak 19	768x512	78.996	78.728	4.941	5.320	34.580	39.058
	Kodak 20	768x512	77.642	77.481	3.810	4.004	33.447	36.638
	Kodak 21	768x512	82.035	82.003	5.254	5.721	35.699	38.619
	Kodak 22	768x512	80.875	80.600	5.131	5.647	35.086	39.465
	Kodak 23	768x512	77.307	76.870	3.380	4.365	34.696	43.497
	Kodak 24	768x512	83.056	82.999	5.099	5.616	38.184	39.206
		Average	80.168	79.979	4.708	5.215	35.205	38.974

Table 4.1 Prediction performance of MED and GAP when the prediction is carried out in the index domain

From Table 4.1, one can see that the performance of MED and GAP is similar in terms of E_I. In terms of E_C, MED is better than GAP. The gap between MED and GAP is much larger in the E_C as compared with that in E_I because smaller difference in two indices does not imply smaller Euclidean distance between their associated colors. There is no linear relationship between index difference and color difference, and it is not even monotonic.

MED's prediction performance is better than GAP's in terms of the entropy of index prediction error. This measure describes the distribution of the prediction error in the index

plane and reflects its statistical property. In case the prediction residue of MED is directly encoded with entropy coding at this point, the achievable average compression rate is around 4.7 bpp.

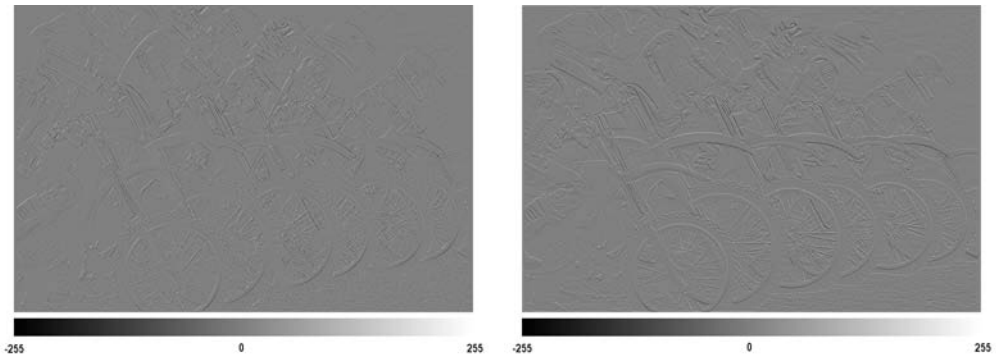
The focus of this section is put on the prediction performance of GAP and MED when the prediction is carried out in an index plane. Evaluation of their performance is directly carried out in their prediction results. The impact of their prediction results to the overall performance of adaptive palette reordering has still not yet been discussed. This issue will be addressed in Section 4.5.

4.4 Prediction in color intensity domain

Prediction can also be done in each of the three color planes. For each pixel (*i,j*), its three predicted color components form a vector denoted as $\vec{v}(i, j)$ which defines the predicted color of the pixel. After color quantizing $\vec{v}(i, j)$ with the palette associated with the image, the predicted index, say $p(i,j)$, can be determined as the index of the quantized predicted color in the palette.

Both MED and GAP can be used for the prediction. As a matter of fact, the adaptive palette reordering method presented in Chapter 3 exploits MED to predict the intensity values of individual color components of (*i,j*). This section presents some simulation results showing the performance of MED and GAP when the prediction is carried out in individual color planes.

Figure 4.8a and Figure 4.8b show, respectively, MED's and GAP's index prediction error planes of *Kodak-05*. Figure 4.9 shows their corresponding histograms. 30.8% of pixels are accurately predicted with MED while only 27.4% of pixels are accurately predicted with



(a) MED (b) GAP

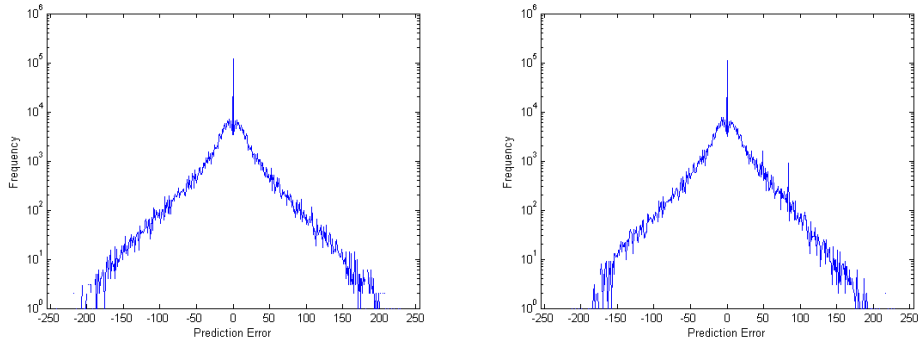
Figure 4.8 Index residue planes of *Kodak-05* when the prediction is done is individual color planes (a) MED and (b) GAP

GAP. The variances of MED's and GAP's residue planes are, respectively, 677.72 and 675.34. Their performance is actually more or less the same when the prediction is done in individual color planes. Similar observations were obtained when the simulation was done with other testing images.

Table 4.2 shows the prediction performance of MED and GAP in terms of the energy of the index prediction error (E_I), the entropy of the index prediction error ($Entropy_I$) and the energy of the color prediction error (E_C). The definition of E_I and $Entropy_I$ is exactly the same as that in eqn. (4.1) and (4.2), while E_C is defined as

$$E_C = 10\log_{10}\left(\sum_{(i,j)}\left\|\vec{c}_r(i,j)-\vec{v}(i,j)\right\|^2\right) \quad \text{in dB} \tag{4.4}$$

where $\vec{c}_r(i,j)$ and $\vec{v}(i,j)$ are, respectively, the real color and the predicted color of pixel (i,j). Unlike the case where the prediction is preformed in the index plane, the predicted color may not be a color in the palette and hence the definition of E_C is different in the two cases.



(a) MED (b) GAP

Figure 4.9 Histograms of Figure 4.8 (in log scale)

The prediction performance of MED is a bit better than that of GAP in all three measures. The achievable average compression rate is around 4.6 bpp when the index prediction residue of MED is directly encoded with entropy coding, which is around 0.2 bpp lower than when GAP is used.

By comparing Table 4.1 and Table 4.2, one can find that the performance gap between GAP and MED is narrowed when the prediction is done in individual color planes. To a certain extent, it implies that the prediction is more robust to the predictor to be used when it is done in individual color planes. Another observation we have is that MED is a bit better than GAP in reducing the prediction error and lowering the zero-order entropy of the index prediction error.

Like the discussion we have in section 4.2, the focus of this section is put on the prediction performance of GAP and MED only. In the following section, their actual impact to the overall performance of adaptive palette reordering will be discussed.

Group	Image	Size (pixel^2)	Energy of the index prediction error, E_I (dB)		Entropy of the index prediction error, $Entropy_I$ (bpp)		Energy of the color prediction error, E_C (dB)	
			MED	GAP	MED	GAP	MED	GAP
CG image	pool	510x383	71.195	70.902	2.114	2.160	24.696	24.537
	watch	1024x768	81.174	81.809	2.748	3.071	33.073	33.408
	water	1024x768	85.929	85.848	5.555	5.718	38.899	38.713
Natural image	Kodak 01	768x512	83.010	84.345	5.806	6.224	35.275	36.507
	Kodak 02	768x512	80.848	81.545	4.881	5.098	30.034	30.448
	Kodak 03	768x512	77.451	78.090	3.574	3.738	30.199	30.436
	Kodak 04	768x512	79.491	79.481	4.586	4.687	30.947	30.604
	Kodak 05	768x512	84.259	84.242	5.560	5.715	37.589	37.483
	Kodak 06	768x512	80.955	82.391	5.189	5.551	34.134	35.280
	Kodak 07	768x512	79.426	79.981	4.056	4.461	31.026	31.361
	Kodak 08	768x512	81.762	83.290	5.297	5.923	36.579	38.057
	Kodak 09	768x512	76.657	78.000	4.357	4.511	30.065	31.142
	Kodak 10	768x512	77.945	78.445	4.451	4.590	31.198	31.528
	Kodak 11	768x512	82.033	82.626	4.962	5.282	33.613	33.981
	Kodak 12	768x512	76.382	77.513	4.135	4.429	29.459	30.415
	Kodak 13	768x512	86.082	85.886	6.516	6.565	39.866	39.569
	Kodak 14	768x512	81.594	81.878	5.224	5.419	34.660	34.678
	Kodak 15	768x512	77.605	78.148	4.125	4.160	31.619	31.687
	Kodak 16	768x512	78.488	79.875	4.646	5.024	30.459	31.744
	Kodak 17	768x512	78.174	78.301	4.548	4.620	32.105	31.879
	Kodak 18	768x512	83.171	82.892	5.707	5.698	36.308	35.753
	Kodak 19	768x512	78.936	80.808	4.884	5.145	32.470	34.433
	Kodak 20	768x512	77.566	78.047	3.773	3.829	32.044	32.898
	Kodak 21	768x512	81.981	82.481	5.194	5.296	34.393	34.875
	Kodak 22	768x512	80.886	81.207	5.059	5.109	33.070	33.144
	Kodak 23	768x512	77.062	77.133	3.331	3.263	30.793	30.346
	Kodak 24	768x512	82.876	82.838	5.033	5.200	37.495	37.418
		Average	80.109	80.667	4.641	4.833	33.040	33.419

Table 4.2 Prediction performance of MED and GAP when the prediction is carried out in the color intensity domain

4.5 Impact to the overall reordering performance

In Sections 4.3 and 4.4, the prediction performance of MED and GAP in different scenarios is evaluated. Specifically, the following four schemes are evaluated:

(i) Using MED in the index plane;

(ii) Using GAP in the index plane;

(iii) Using MED in individual color planes and

(iv) Using GAP in individual color planes.

The evaluation is directly made on the prediction error in either the index plane or the color plane. As shown in Figure 4.1, prediction is only one of the components in our proposed adaptive palette reordering framework. The actual impact of the four prediction schemes to the overall adaptive palette reordering performance has to be investigated and the investigation result is presented in this section.

Figure 4.10 shows the pseudo code for realizing adaptive palette reordering with either prediction scheme (i) or prediction scheme (ii), while Figure 4.11 shows the pseudo code for realizing adaptive palette reordering with prediction scheme (iii) or prediction scheme (iv). The lines in blue highlight the difference between the four corresponding adaptive palette reordering methods.

For reference, the adaptive palette reordering methods using prediction schemes (i), (ii), (iii) and (iv) are, respectively, referred to as APR-I-MED, APR-I-GAP, APR-C-MED and APR-C-GAP hereafter. Here, ‘APR’ stands for adaptive palette reordering. ‘I’ and ‘C’ are used to specify whether the prediction is carried out in the index plane or individual color intensity planes. ‘MED’ and ’GAP’ specify the prediction scheme used in the prediction.

Initialize DF-Table {H(m,n)} by H(m,n)=0 for m,n=0,1…N-1.
Raster scan the image
FOR each pixel (i,j)
Predict index at (i,j) with MED/GAP† in the index map.
% Assume the predicted index value is p
Sort palette colors in $\Omega \equiv \{ \vec{c}_k \mid k=0,1 \ldots N-1 \}$ by H(p,k) and then by $\| \vec{c}_k - \vec{v}(i,j) \|^2$ and then by k.
% The sorted palette colors forms the transient version of the palette
% Assume the real color of pixel (i,j) is $\vec{c}_r$, *the* r^{th} *palette color in reference palette* Ω
Assign the position of $\vec{c}_r$ in the current sorted palette color queue to be the new index of $\vec{c}_r$.
Update {H(m,n)} by H(p,r)++.
END

† MED is used in prediction scheme (i) while GAP is used in scheme (ii).

Figure 4.10 Pseudo code of an adaptive palette reordering method in which prediction is carried out in the index plane.

As a remark, we note that APR-C-MED is actually the adaptive palette reordering method presented in Chapter 3.

Unlike the approach presented in Sections 4.3 and 4.4, to evaluate the contribution of the four prediction schemes to the overall adaptive palette reordering performance, we compare the reindexed index maps obtained with APR-I-MED, APR-I-GAP, APR-C-MED and APR-C-GAP directly. Figure 4.12 shows the corresponding reindexed index maps of *Kodak-05* and Figure 4.13 shows their histograms. Bijective mapping (3.1) was used in the final stage of the production of the index maps shown in Figure 4.12. From Figure 4.13, one can see that the performance of APR-C-MED and APR-C-GAP is comparatively better in the four evaluated adaptive palette reordering methods. Specifically, the zero-order entropy values of the results provided by APR-I-MED, APR-I-GAP, APR-C-MED and APR-C-GAP are, respectively, 4.766, 5.620, 4.237 and 4.238 bpp.

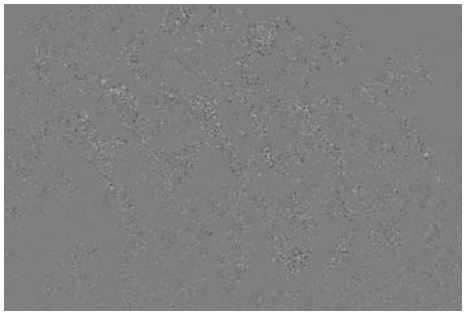
Initialize DF-Table {H(m,n)} by H(m,n)=0 for m,n=0,1…N-1.
Raster scan the image
FOR each pixel (i,j)
Predict the red component of pixel (i,j) with MED/GAP† in the red color plane to get $r(i,j)$.
Predict the green component of pixel (i,j) with MED/GAP† in the green color plane to get $g(i,j)$.
Predict the blue component of pixel (i,j) with MED/GAP† in the blue color plane to get $b(i,j)$.
Quantize $\vec{v}(i,j) = (r(i,j), g(i,j), b(i,j))$ with the reference palette Ω.
% Assume the quantization result is $\vec{c}_p$, the p^{th} palette color in reference palette Ω
Sort palette colors in $\Omega \equiv \{ \vec{c}_k \mid k=0,1\ldots N-1 \}$ by H(p,k) and then by $\| \vec{c}_k - \vec{v}(i,j) \|^2$ and then by k.
% The sorted palette colors forms the transient version of the palette
% Assume the real color of pixel (i,j) is $\vec{c}_r$, the r^{th} palette color in reference palette Ω
Assign the position of $\vec{c}_r$ in the current sorted palette color queue to be the new index of $\vec{c}_r$.
Update {H(m,n)} by H(p,r)++.
END

† MED is used in prediction scheme (iii) while GAP is used in scheme (iv).

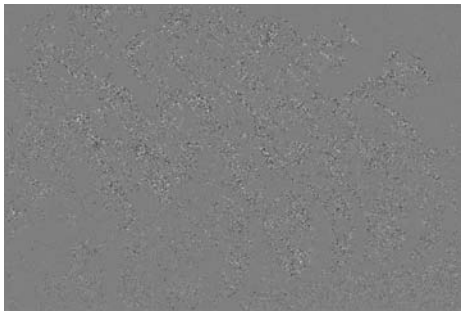
Figure 4.11 Pseudo code of an adaptive palette reordering method in which prediction is carried out in individual color intensity planes.

Figure 4.14 shows the correlation among adjacent pixels in the reindexed index maps shown in Figure 4.12. One can see that little correlation is retained in all four reindexed index maps. Among the four adaptive palette reordering methods, the decorrelation performance of APR-C-MED is the best.

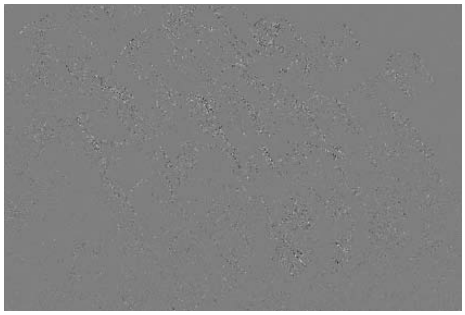
Tables 4.3 and 4.4 summarize the performance of APR-I-MED, APR-I-GAP, APR-C-MED and APR-C-GAP when they were used to process a set of testing images. In particular, Table 4.3 shows the variance of the indices in a reindexed index map while Table 4.4 shows the zero-order entropy of the indices in a reindexed index map. On average, performing prediction in individual color intensity planes provides a reindexed index map of lower entropy and smaller variance as compared with performing prediction in the index plane. Another observation is that the performance of APR-C-MED and APR-C-GAP is more or less the same. By considering that MED is of lower complexity, it seems that APR-C-MED is the best one of the four evaluated adaptive palette reordering methods.



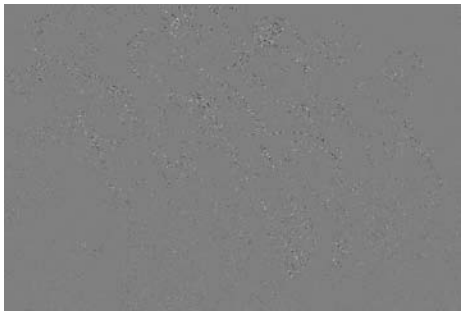
(a) APR-I-MED



(b) APR-I-GAP



(c) APR-C-MED



(d) APR-C-GAP

Figure 4.12 *Kodak-05*'s reindexed index-maps obtained with different methods

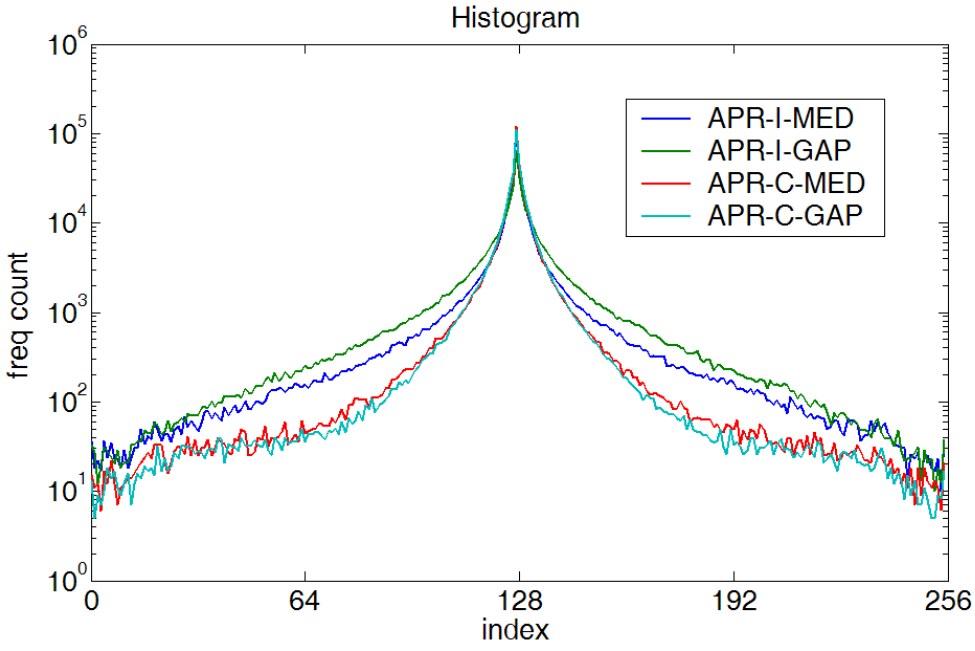


Figure 4.13 Histograms of Figure 4.12 (in log scale)

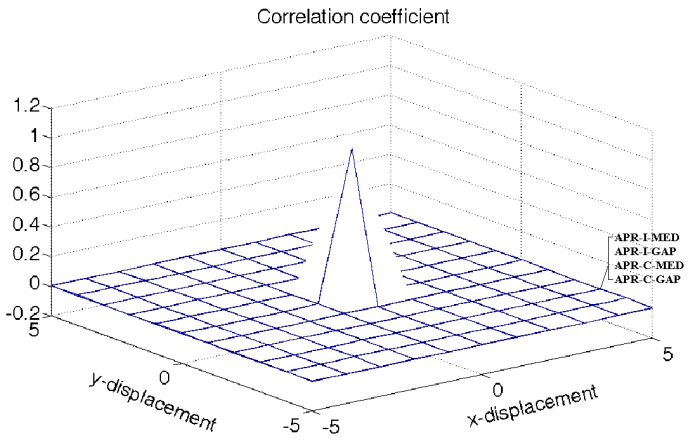


Figure 4.14 Performance of different algorithms in terms of correlation among pixels in their reindexed results

Group	Image	Size (pixel²)	Variance of the reindexed index map			
			APR-I-MED	APR-I-GAP	APR-C-MED	APR-C-GAP
CG image	pool	510x383	36.91	74.57	12.76	10.82
	watch	1024x768	72.50	97.25	33.06	32.10
	water	1024x768	306.47	383.90	186.74	177.09
Natural image	Kodak 01	768x512	206.00	255.11	129.98	153.07
	Kodak 02	768x512	113.22	146.06	49.72	52.41
	Kodak 03	768x512	59.86	121.74	14.03	12.34
	Kodak 04	768x512	96.74	172.98	24.21	20.68
	Kodak 05	768x512	319.14	452.60	148.68	127.00
	Kodak 06	768x512	174.72	219.64	92.01	96.92
	Kodak 07	768x512	106.62	168.87	50.25	43.29
	Kodak 08	768x512	234.81	300.84	141.27	150.25
	Kodak 09	768x512	58.25	78.25	34.81	43.90
	Kodak 10	768x512	94.83	117.49	57.35	55.37
	Kodak 11	768x512	199.52	280.62	80.36	77.20
	Kodak 12	768x512	50.93	75.36	28.41	28.81
	Kodak 13	768x512	530.03	705.67	304.99	261.29
	Kodak 14	768x512	182.79	273.23	71.15	63.33
	Kodak 15	768x512	66.30	120.01	20.24	15.58
	Kodak 16	768x512	80.59	105.67	46.63	51.14
	Kodak 17	768x512	95.64	120.17	59.34	51.72
	Kodak 18	768x512	269.42	406.00	119.23	102.70
	Kodak 19	768x512	90.81	124.10	53.42	71.30
	Kodak 20	768x512	84.61	110.39	53.68	56.56
	Kodak 21	768x512	181.59	254.30	87.79	86.51
	Kodak 22	768x512	148.49	234.77	69.82	63.02
	Kodak 23	768x512	68.38	138.64	16.73	15.42
	Kodak 24	768x512	279.48	348.72	147.42	124.79
		Average	155.88	218.03	79.04	75.73

Table 4.3 Variance of the indices in a reindexed index map produced with adaptive palette reordering using a particular prediction scheme (with mapping (3.1))

Group	Image	Size (pixel^2)	Zero-order entropy of the reindexed index map (bpp)			
			APR-I-MED	APR-I-GAP	APR-C-MED	APR-C-GAP
CG image	pool	510x383	1.516	1.824	1.401	1.376
	watch	1024x768	2.380	2.880	2.094	2.266
	water	1024x768	4.945	5.691	4.499	4.558
Natural image	Kodak 01	768x512	4.921	5.333	4.564	4.787
	Kodak 02	768x512	3.792	4.306	3.404	3.462
	Kodak 03	768x512	2.750	3.791	2.403	2.428
	Kodak 04	768x512	3.723	4.817	3.238	3.200
	Kodak 05	768x512	4.766	5.620	4.237	4.238
	Kodak 06	768x512	4.479	4.999	4.070	4.241
	Kodak 07	768x512	3.270	4.036	2.882	3.011
	Kodak 08	768x512	4.658	5.426	4.202	4.541
	Kodak 09	768x512	3.431	3.802	3.244	3.275
	Kodak 10	768x512	3.662	4.465	3.380	3.354
	Kodak 11	768x512	4.096	4.965	3.599	3.739
	Kodak 12	768x512	3.350	4.234	3.054	3.146
	Kodak 13	768x512	5.776	6.355	5.277	5.202
	Kodak 14	768x512	4.355	5.090	3.807	3.866
	Kodak 15	768x512	3.215	4.120	2.859	2.773
	Kodak 16	768x512	3.793	4.326	3.484	3.642
	Kodak 17	768x512	3.777	4.251	3.480	3.416
	Kodak 18	768x512	4.809	5.792	4.191	4.090
	Kodak 19	768x512	4.036	4.651	3.704	3.888
	Kodak 20	768x512	3.224	3.587	3.020	3.044
	Kodak 21	768x512	4.235	5.028	3.796	3.831
	Kodak 22	768x512	4.251	5.249	3.783	3.725
	Kodak 23	768x512	2.520	4.085	2.260	2.182
	Kodak 24	768x512	4.323	5.161	3.927	3.951
		Average	3.854	4.588	3.476	3.527

Table 4.4 Zero-order entropy of the indices in a reindexed index map produced with adaptive palette reordering using a particular prediction scheme

4.6 Summary

Index prediction is one of the critical components of adaptive palette reordering, and it can be realized with different schemes. Four schemes are studied and their prediction performance is evaluated in this chapter. These schemes exploits MED or GAP to carry out the prediction in either the index plane or individual color intensity planes. In terms of various measures related to prediction error, the scheme which exploits MED to predict individual color components appears to be better.

The actual impact of these prediction schemes to the overall adaptive palette reordering performance is then evaluated. Again it is found that the scheme which exploits MED to predict individual color components is the best as it can provide reindexed index maps of lower variance and lower entropy on average at a lower complexity cost.

Chapter 5 – Impact of color reordering to adaptive palette reordering

5.1 Introduction

There are two functional components in adaptive palette reordering. One is index prediction and the other is color reordering. Each one of them can be realized with different schemes and, accordingly, results in different palette reordering outputs. In Chapter 4, we investigates four different schemes for realizing index prediction, and their impact to the overall performance of adaptive palette reordering is reported. In this chapter, we put our focus on the realization of color reordering. Several color reordering schemes will be introduced, and their contribution to the overall palette reordering performance will be evaluated.

The actual implementation of color reordering can be done with different approaches. In conventional palette reordering methods, the resultant palette is static and the palette colors can be sorted by a measure which is not pixel-dependent. For example, one can sort the colors by luminance to obtain the reference palette used in this thesis. However, in adaptive palette reordering, palette colors are sorted by a measure of some pixel-dependent properties such that the resultant palette is pixel dependent as well.

Since performing MED prediction in individual color intensity planes is found to be the best prediction scheme in the schemes evaluated in Chapter 4, it is used in our study of the performance of various color reordering schemes to produce prediction results for the evaluated color reordering schemes.

This chapter is organized as follows. Two distance-oriented sorting schemes and one history-oriented sorting schemes are, respectively, introduced in Section 5.2 and Section 5.3 for realizing color reordering. Then, two hybrid sorting schemes which combine distance-oriented and history-oriented sorting schemes together are proposed in Section 5.4.

In Section 5.5, simulation results are presented for comparing the performance of using various color sorting schemes in realizing adaptive palette reordering. Finally, a brief summary is given in Section 5.6.

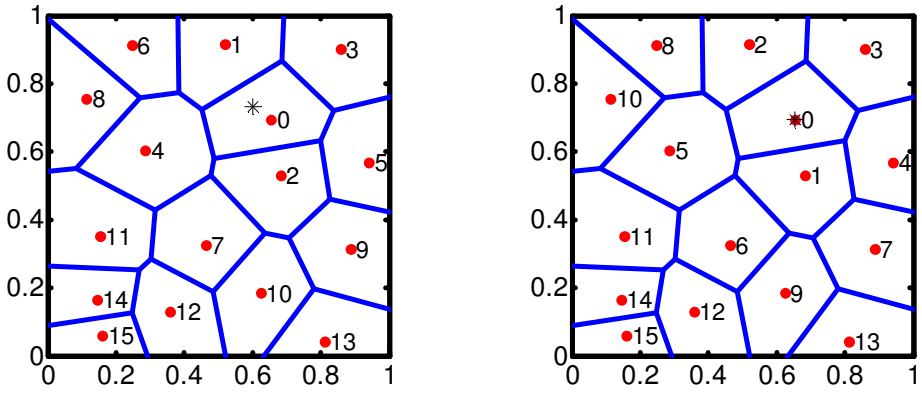
5.2 Distance-oriented sorting

For each pixel (*i,j*), the prediction module in adaptive palette reordering provides a predicted color $\vec{v}(i, j)$ as its output. One can sort the palette colors by their Euclidean distances to $\vec{v}(i, j)$. The closer one is put in the front. As $\vec{v}(i, j)$ is pixel-dependent, the resultant palette is also pixel dependent, and it can be considered as a transient version of the original palette. It is possible that some palette colors are equally distant away form $\vec{v}(i, j)$. In that case, these colors are sorted by their luminance and then by their original index values in the original palette. The sorting is done in ascending order.

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. The newly sorted color sequence forms a transient version of palette Ω.

For reference, the adaptive palette reordering method using this color reordering scheme is referred to as APR-D-PC, where 'D-PC' means palette colors are sorted by their *distances* to the *predicted color* of pixel (*i,j*).

To reduce the realization complexity of APR-D-PC, one can quantize $\vec{v}(i, j)$ with the original palette and then sort the palette colors by their distances to the quantized $\vec{v}(i, j)$. In that case, as the quantized $\vec{v}(i, j)$ is one of the palette colors and the distances among palette colors can be presorted, all possible reordered palettes are well ready and the one for pixel (*i,j*) can be determined as soon as the quantized $\vec{v}(i, j)$ is determined. For reference, this simplified version of APR-D-PC is referred to as APR-D-QPC, which implies palette colors are sorted by their *distances* to the *quantized predicted color* of pixel (*i,j*).

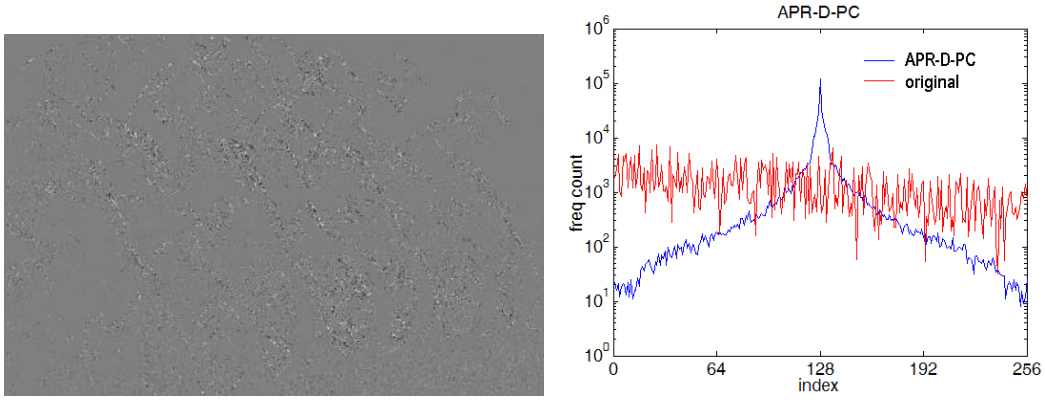


(a) APR-D-PC (b) APR-D-QPC

Figure 5.1 Examples showing how palette colors are sorted in (a) APR-D-PC and (b) APR-D-QPC

Figure 5.1 graphically shows an example showing how palette colors are sorted in APR-D-PC and APR-D-QPC. To make the example simple enough to understand, the color space is condensed to a two-dimensional space. The red dots denote the palette colors and the blue lines shows the boundary of their Voronoi regions. The asterisks in Figure 5.1a and Figure 5.1b respectively show $\bar{v}(i, j)$ and the quantized $\bar{v}(i, j)$. In either case, palette colors are sorted by their distances to the asterisk and the number associated with a particular dot specifies the dot's position in the sorted result.

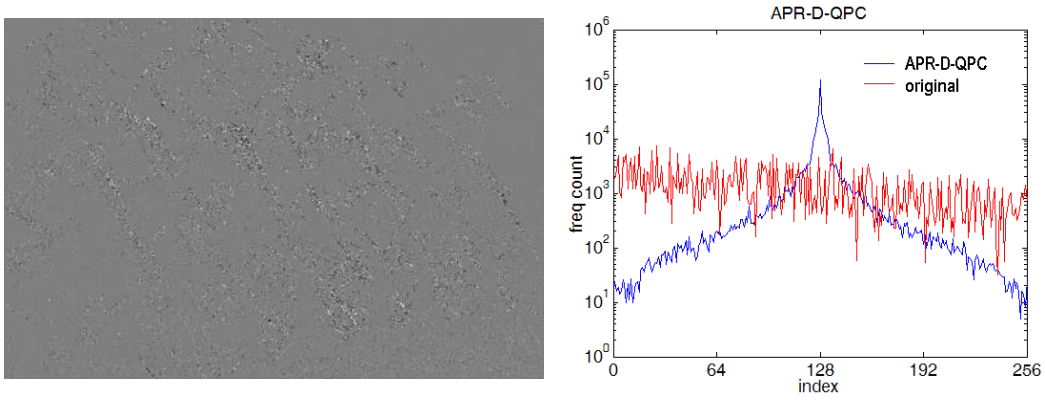
Figure 5.2 shows a reindexed index map produced with APR-D-PC and its corresponding histogram. The red curve shows the histogram of the index-map of the original testing image whose palette was generated with MATLAB function RGB2IND. This histogram serves as a reference for comparison. As expected, APR-D-PC changes the index distribution and turns it into a highly unequalized one. This action significantly reduces the zero order entropy of the index map. As a matter of fact, the entropy of the reindexed index map is only 4.722 while that of the original index map is 7.469.



(a) Reindexed index map (b) Histogram

Figure 5.2 Output of APR-D-PC: (a) reindexed index map of *Kodak-05* and (b) histogram

Figure 5.3 shows the processing result of APR-D-QPC. Again, the histogram in red serves as a reference for comparison. APR-D-QPC also changes the index distribution and significantly reduces the zero order entropy of the index map. Specifically, the entropy of the reindexed index map produced by APR-D-QPC is 4.740. The performance of APR-D-QPC and APR-D-PC is more or less the same in terms of entropy. A more detailed comparison is carried out in Section 5.5.



(a) Reindexed index map (b) Histogram

Figure 5.3 Output of APR-D-QPC: (a) reindexed index map of *Kodak-05* and (b) histogram

5.3 History-oriented sorting

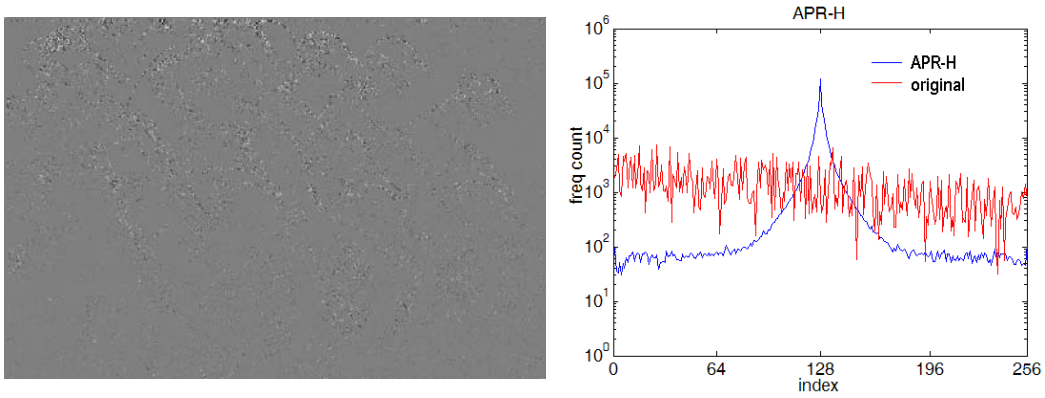
In practice, we learn from our experience. Similar idea can be applied to color reordering. Let the quantized predicted error of pixel (*i,j*) be $\vec{c}_p$, the p^{th} color in reference palette Ω. In general, $\vec{c}_p$ is different from the real color of pixel (*i,j*). The occurrence of this discrepancy is a valuable experience. It can be recorded and cumulated for improving the prediction performance in the future.

To achieve this objective, a table is constructed to store the values of {H(*m*,*n*)|*m*,*n*=0,1…N-1}, where H(*m*,*n*) is defined as the number of occurrences when the quantized predicted color and the original color of a pixel are, respectively, $\vec{c}_m$ and $\vec{c}_n$. All H(*m*,*n*) values are initialized to zero at the very beginning and the table is updated after a pixel is processed. As mentioned in Chapter 3, this table is referred to as *discrepancy frequency table* (DF-Table).

After $\vec{c}_p$ is determined, palette colors $\{\vec{c}_k | k=0,1\ldots N-1\}$ are sorted according to the values of {H(*p*,*k*)|*k*=0,1…*N*-1} in descending order. When there exist two different palette colors $\vec{c}_l$ and $\vec{c}_d$ such that H(*p*,*l*)=H(*p*,*d*), they are sorted by their luminance and then by their original index values in the original palette. The sorting is done in ascending order.

The newly sorted queue forms a transient version of palette Ω. The position of $\vec{c}_r$ in the newly sorted queue can be used as an index to the queue and is used to represent the pixel in the reindexed index map. After processing this pixel, H(*p*,*r*) is incremented by 1 to update the frequency count of this event.

For reference, the adaptive palette reordering method using this color reordering scheme is referred to as APR-H, where 'H' means palette colors are sorted according to the frequency of occurrence of a particular pair of predicted color and real color so far in the *history*.



(a) Reindexed index map (b) Histogram

Figure 5.4 Output of APR-H: (a) reindexed index map of *Kodak-05* and (b) histogram.

Figure 5.4a shows the resultant reindexed index map of APR-H and Figure 5.4b shows its corresponding histogram. One can see that, as compared with the histogram of APR-D-PC's result, the histogram of APR-H's result provides a sharper peak but a flatter tail. A sharper peak implies the distribution is more unequalized in the range of indices which are more likely to happen. This is good from entropy point of view. At the same time, a flatter tail means the entropy of the indices which are unlikely to happen is higher.

The existence of the flat tail in APR-H's result can be explained by the fact that color sorting in APR-H is actually based on the likeliness of the occurrence of a particular color when the quantized predicted color $\vec{c}_p$ is given and it is in turn estimated based on the history. For indices which are unlikely to happen, their occurrences in the history are too few to provide any useful information for APR-H to guess which one of them is more likely to happen. When guess does not work, all possible indices are equally likely to happen. This results in a flat tail.

It happens that the effect of the sharper peak is stronger than that of the flatter tail, and hence the overall performance of APR-H is better. In particular, the entropy of Figure 5.4a is 4.315 bpp, which is lower than the entropy of Figure 5.3a.

5.4 Hybrid mode sorting

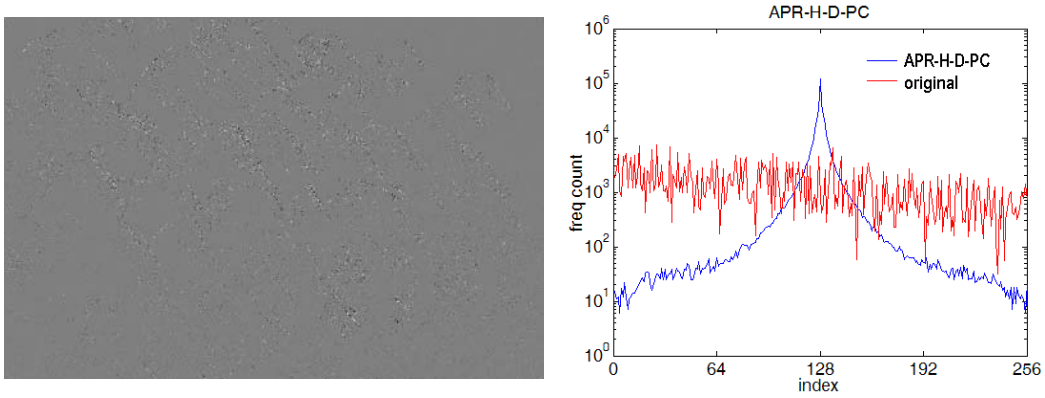
When APR-H is used, if the likeliness of the occurrence of a particular palette color cannot be guessed effectively with the quantized predicted color $\vec{c}_p$ based on the history, color cannot be sorted effectively and all possible indices will be considered equally likely to happen. This makes APR-H do not work properly.

In APR-H, when there exist two different palette colors $\vec{c}_l$ and $\vec{c}_d$ such that H(*p*,*l*)=H(*p*,*d*), $\vec{c}_l$ and $\vec{c}_d$ are sorted by their luminance. Luminance is solely color-dependent and it is nothing related to the predicted color $\vec{v}(i, j)$. In such a case, history and the prediction color $\vec{v}(i, j)$ do not contribute to the decision of the final color reordering result any more at this stage.

To solve this problem, one can adopt a hybrid color reordering scheme which is both history- and distance-oriented. Basically, this hybrid scheme follows the same steps of APR-H excepts that, when there exist two different palette colors $\vec{c}_l$ and $\vec{c}_d$ such that H(*p*,*l*)=H(*p*,*d*), $\vec{c}_l$ and $\vec{c}_d$ are sorted by their Euclidean distances to the predicted color $\vec{v}(i, j)$. The closer is put in the front. They are only sorted by their luminance and then by their original index values in the original palette when they are still not distinguishable. By doing so, $\vec{v}(i, j)$ may still contribute to color reordering when history cannot be used.

For reference, the adaptive palette reordering method using this color reordering scheme is referred to as APR-H-D-PC, where 'H-D-PC' means palette colors are sorted according to the *history* and then their *distances* to the *predicted color* of pixel (*i*,*j*). Note that APR-H-D-PC is actually the adaptive palette reordering method presented in Chapter 3.

The complexity of APR-H-D-PC can be reduced by sorting $\vec{c}_l$ and $\vec{c}_d$ according to their Euclidean distances to the quantized predicted color $\vec{c}_p$ instead of the predicted color $\vec{v}(i, j)$ whenever H(*p*,*l*)=H(*p*,*d*) happens. As $\| \vec{c}_p - \vec{c}_k \|^2$ for all *k* can be presorted, the



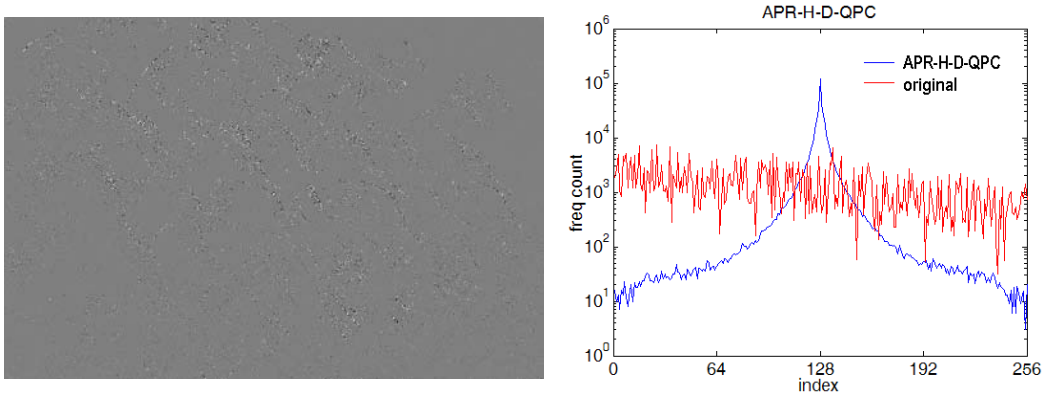
(a) Reindexed index map (b) Histogram

Figure 5.5 Output of APR-H-D-PC: (a) reindexed index map of *Kodak-05* and (b) histogram

final color reordering result can be determined by table lookup as soon as $\vec{c}_p$ is ready. This saves an amount of its realization effort. For reference, this simplified version of APR-H-D-PC is referred to as APR-H-D-QPC, which implies palette colors are sorted according to the *history* and then their *distances* to the *quantized predicted color* of pixel (*i*,*j*).

Figure 5.5 shows the output of APR-H-D-PC. The red curve is the histogram of the index-map of the original testing image. The histogram of Figure 5.5a is highly unequalized. The shape of the distribution is something in the middle of that of APR-D-QPC shown in Figure 5.3b and that of APR-H shown in Figure 5.4b. Its waist is slimmer as compared with the histogram of APR-D-QPC while its tail is not as flat as the histogram of APR-H. The zero-order entropy of Figure 5.5a is 4.237.

Figure 5.6 shows the result of APR-H-D-QPC. By comparing the histograms of APR-H and APR-H-D-PC, one can expect that their performance should be more or less the same. In particular, the zero-order entropy of Figure 5.6a is 4.239. A more detailed comparison among various color sorting schemes will be presented in Section 5.5.



(a) Reindexed index map (b) Histogram

Figure 5.6 Output of APR-H-D-QPC: (a) reindexed index map of *Kodak-05* and (b) histogram

5.5 Performance Study

This section presents a more detailed study of the performance of the five adaptive palette reordering methods introduced in Sections 5.3, 5.4 and 5.5. For convenience, Table 5.1 summaries the sorting criteria and the rules adopted in the color reordering schemes exploited in these five adaptive palette reordering methods. Again, we note that APR-H-D-PC is actually the adaptive palette reordering method presented in Chapter 3.

For an easier comparison of the histograms of their reindexed index maps of *Kodak-05*, Figure 5.7 groups the plots shown in Figures 5.2b, 5.3b, 5.4b, 5.5b and 5.6b together. From Figure 5.7, one can easily see that the histograms of the results of APR-H-D-PC and APR-H-D-QPC are more unequalized. Figure 5.8 shows the correlation among adjacent pixels in their reindexed index maps of *Kodak-05*. All of them can successfully remove the spatial correlation of the indices.

Adaptive palette reordering method	Color sorting criteria/order used
APR-D-PC	(1) $\| \vec{c}_k - \vec{v}(i,j) \|^2$, (2) luminance of $\vec{c}_k$, (3) k
APR-D-QPC	(1) $\| \vec{c}_k - \vec{c}_p \|^2$, (2) luminance of $\vec{c}_k$, (3) k
APR-H	(1) H(p,k), (2) luminance of $\vec{c}_k$, (3) k
APR-H-D-PC	(1) H(p,k), (2) $\| \vec{c}_k - \vec{v}(i,j) \|^2$, (3) luminance of $\vec{c}_k$, (4) k
APR-H-D-QPC	(1) H(p,k), (2) $\| \vec{c}_k - \vec{c}_p \|^2$, (3) luminance of $\vec{c}_k$, (4) k

Table 5.1 Summary of the sorting criteria and rules adopted in different adaptive palette reordering methods

Table 5.2 shows the performance of the five adaptive reordering methods in terms of the index variance of their reindexed index maps. Bijective mapping (3.1) was used in the final stage to remap the reindexed index maps in the simulation and Table 5.2 shows the data obtained with this final mapping results. One can see that the adaptive palette reordering methods using hybrid schemes provide the best average performance among the evaluated adaptive palette reordering methods. The one using history-oriented scheme is the poorest one in terms of this measure.

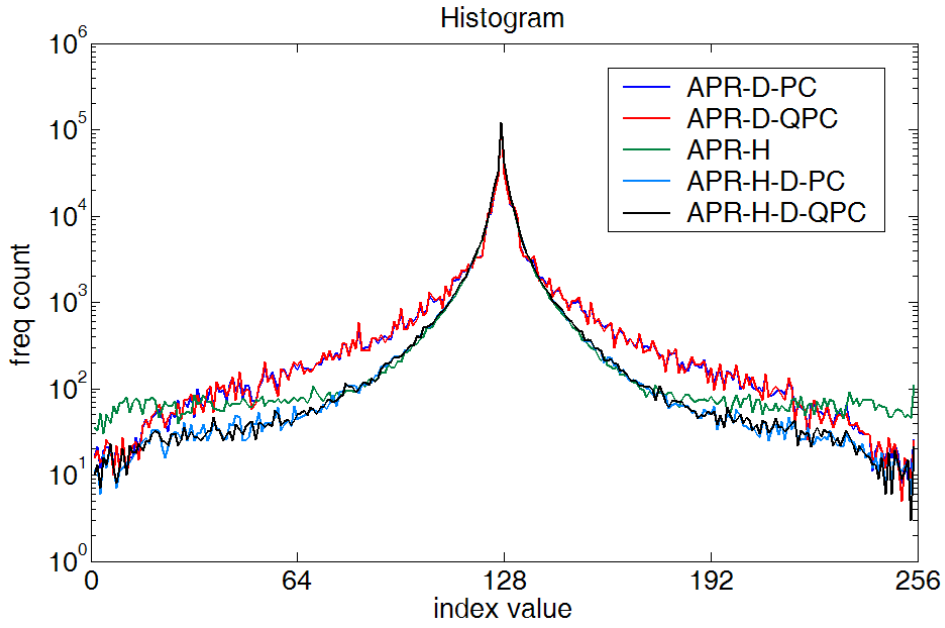


Figure 5.7 Histograms of outputs of different adaptive palette reordering methods

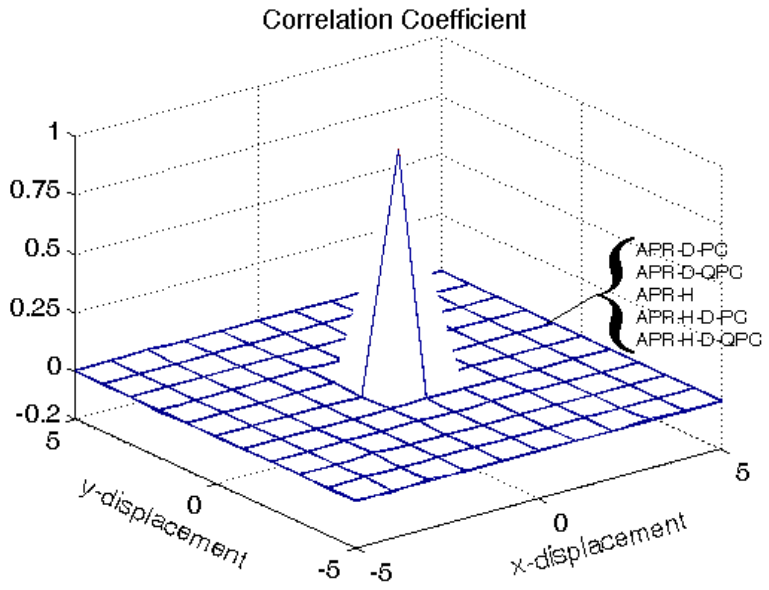


Figure 5.8 Performance of different adaptive palette reordering methods in terms of correlation among pixels

Table 5.3 shows the performance of the evaluated adaptive reordering methods in terms of the zero-order entropy of their reindexed index maps. Again, the adaptive palette reordering methods using hybrid schemes in color reordering provide the best performance. Though APR-H provides a reindexed index map of larger variance on average as compared with APR-D-X (X can be either PC or QPC.), it provides a reindexed index map of lower entropy on average. Since entropy is a measure more directly related to the compression performance, one may consider that APR-H is actually better than APR-D-X and hence history-oriented scheme performs better than distance-oriented schemes in supporting adaptive palette reordering.

The performance of APR-H-D-PC and APR-H-D-QPC is more or less the same in both measures. By considering that the realization complexity of APR-H-D-QPC is lower, it seems that APR-H-D-QPC is the best option at this stage.

Group	Image	Size (pixel^2)	Adaptive reordering method				
			APR-D-PC	APR-D-QPC	APR-H	APR-H-D-PC	APR-H-D-QPC
CG image	pool	510x383	20.44	20.90	97.28	12.76	13.08
	watch	1024x768	81.82	82.49	83.84	33.06	33.24
	water	1024x768	252.92	255.71	299.19	186.74	187.27
Natural image	Kodak 01	768x512	344.40	345.17	297.62	129.98	130.23
	Kodak 02	768x512	106.16	106.90	156.67	49.72	49.99
	Kodak 03	768x512	27.90	28.62	89.81	14.03	14.38
	Kodak 04	768x512	43.46	44.44	110.41	24.21	24.63
	Kodak 05	768x512	327.73	330.52	269.68	148.68	149.52
	Kodak 06	768x512	188.41	189.83	253.17	92.01	92.49
	Kodak 07	768x512	84.33	85.41	147.00	50.25	50.70
	Kodak 08	768x512	246.31	248.06	286.78	141.27	141.68
	Kodak 09	768x512	50.14	50.71	153.16	34.81	35.06
	Kodak 10	768x512	81.27	82.18	190.14	57.35	57.66
	Kodak 11	768x512	150.48	151.84	193.88	80.36	80.84
	Kodak 12	768x512	42.47	43.14	139.01	28.41	28.69
	Kodak 13	768x512	686.17	689.36	479.74	304.99	305.77
	Kodak 14	768x512	116.67	118.14	185.97	71.15	71.74
	Kodak 15	768x512	33.74	34.45	123.05	20.24	20.57
	Kodak 16	768x512	97.30	97.97	154.44	46.63	46.84
	Kodak 17	768x512	92.30	92.98	180.34	59.34	59.57
	Kodak 18	768x512	254.32	258.36	260.19	119.23	120.29
	Kodak 19	768x512	92.80	93.60	171.28	53.42	53.70
	Kodak 20	768x512	84.30	84.78	154.93	53.68	53.89
	Kodak 21	768x512	186.70	187.75	197.21	87.79	88.13
	Kodak 22	768x512	124.31	126.89	191.60	69.82	70.56
	Kodak 23	768x512	27.32	28.04	90.13	16.73	17.03
	Kodak 24	768x512	330.91	333.73	296.88	147.42	148.24
		Average	154.63	156.00	194.57	79.04	79.47

Table 5.2 Performance comparison of APR-D-PC, APR-D-QPC, APR-H, APR-H-D- PC and APR-H-D-QPC in terms of variance of the indices in a reindexed index map

Group	Image	Size (pixel^2)	Adaptive reordering method				
			APR-D-PC	APR-D-QPC	APR-H	APR-H-D-PC	APR-H-D-QPC
CG image	pool	510x383	1.647	1.653	1.477	1.401	1.403
	watch	1024x768	2.397	2.400	2.129	2.094	2.095
	water	1024x768	4.859	4.878	4.560	4.499	4.500
Natural image	Kodak 01	768x512	5.421	5.421	4.658	4.564	4.565
	Kodak 02	768x512	4.108	4.109	3.478	3.404	3.405
	Kodak 03	768x512	2.684	2.698	2.466	2.403	2.404
	Kodak 04	768x512	3.577	3.593	3.326	3.238	3.240
	Kodak 05	768x512	4.722	4.740	4.315	4.237	4.239
	Kodak 06	768x512	4.648	4.660	4.158	4.070	4.071
	Kodak 07	768x512	3.136	3.159	2.954	2.882	2.884
	Kodak 08	768x512	4.675	4.694	4.290	4.202	4.204
	Kodak 09	768x512	3.684	3.694	3.326	3.244	3.246
	Kodak 10	768x512	3.722	3.743	3.459	3.380	3.381
	Kodak 11	768x512	4.028	4.043	3.680	3.599	3.601
	Kodak 12	768x512	3.422	3.439	3.133	3.054	3.055
	Kodak 13	768x512	6.072	6.080	5.360	5.277	5.279
	Kodak 14	768x512	4.186	4.205	3.895	3.807	3.809
	Kodak 15	768x512	3.122	3.142	2.939	2.859	2.861
	Kodak 16	768x512	4.069	4.076	3.564	3.484	3.485
	Kodak 17	768x512	3.926	3.939	3.563	3.480	3.481
	Kodak 18	768x512	4.715	4.747	4.278	4.191	4.194
	Kodak 19	768x512	4.198	4.212	3.788	3.704	3.705
	Kodak 20	768x512	3.335	3.345	3.091	3.020	3.020
	Kodak 21	768x512	4.356	4.372	3.873	3.796	3.797
	Kodak 22	768x512	4.110	4.149	3.872	3.783	3.786
	Kodak 23	768x512	2.367	2.390	2.330	2.260	2.262
	Kodak 24	768x512	4.436	4.453	3.997	3.927	3.928
		Average	3.912	3.927	3.554	3.476	3.478

Table 5.3 Performance comparison of APR-D-PC, APR-D-QPC, APR-H, APR-H-D- PC and APR-H-D-QPC in terms of zero-order entropy of the indices in a reindexed index map

5.6 Summary

Five color reordering schemes were evaluated to see how well they can support adaptive palette reordering. In particular, we have two distance-oriented sorting schemes, one history-oriented scheme and two hybrid mode sorting schemes in our evaluation. These five color reordering schemes work with the prediction scheme that uses MED to predict individual color planes to form five adaptive palette reordering methods. The details of each of them are summarized in Table 5.1.

Simulation results show that hybrid mode sorting schemes are better than history-oriented and distance-oriented sorting schemes in terms of both zero-order entropy and variance of the reindexed indices.

Chapter 6 – DF table merging

6.1 Introduction

As shown in Chapters 3 and 5, the DF-Table plays a significant role in adaptive palette reordering. Based on the values of {H(*m*,*n*)|*n*=0,1..*N*-1}, the table shows how likely that $\vec{c}_n$ is the real color when $\vec{c}_m$ is the quantized predicted color. With a given prediction result based on the neighboring pixels of pixel (*i*,*j*), the DF-Table helps one to predict the likeliness of the occurrence of a particular color in pixel (*i*,*j*). One can then assign an index of smaller value to a color which is more likely to happen. Eventually, the reindexed index map contains a number of indices of small values. This makes the histogram of the reindexed index map very unequalized and hence reduces the entropy of the reindexed index map significantly.

At the early stage of adaptive palette reordering, most of H(*m*,*n*) entries are of zero value. A DF-Table of such a property is immature as it provides no or little statistical information. From another point of view, when most of the values in {H(*m*,*n*)|*n*=0,1..*N*-1} are zero, one cannot sort $\vec{c}_n$ by H(*m*,*n*) for *n*=0,1..*N*-1. In such a case, the DF-Table cannot contribute to the reordering performance.

One of the solutions to solve this problem is to reduce the size of the DF-Table by merging some of its entries. Each entry of the DF-Table records the frequency count of a particular event. The more events it takes care of, the more entries it has. Obviously, at any particular instant of the construction of the DF-Table, the sum of all entry values, *L*, equals to the total number of pixels already processed. At the early stage, only a few pixels have been processed. When *L* is shared by the entries of a large DF-Table, most of the entries are of zero value. However, when the same *L* is shared by the entries of a small DF-Table, there would be fewer zero entries and hence it would be easier for one to discriminate palette colors with {H(*m*,*n*)|*n*=0,1..*N*-1}. That qualitatively explains why merging DF-Table can improve the situation.

In this chapter, two DF-Table merging schemes are proposed to solve this problem. The rest of this chapter is organized as follows. Section 6.2 and Section 6.3 introduce the two proposed DF-Table merging schemes. In particular, we will show how these two DF-Table merging schemes work with the adaptive palette reordering method proposed in Chapter 3. Their performance is then evaluated in Section 6.4. A summary is finally provided in Section 6.5.

6.2 Absorbing pre-clustered colors

In this DF-Table merging scheme, palette colors are pre-clustered before carrying out adaptive palette reordering. When the DF-Table is found to be not mature, palette colors in the same cluster are merged to form a smaller palette. A smaller DF-Table is then generated based on the frequency count of the occurrence that $\vec{c}_n$ is the real color when the quantized predicted color belongs to a particular cluster. Accordingly, this DF-Table merging scheme is referred to as Absorbing-preclustered-colors. The details of the scheme are as follows.

In the proposed scheme, by making use of LBG algorithm[Linde80], a palette of a size smaller than the original palette Ω is generated with all colors in Ω as the training vectors. All colors in Ω are then color quantized with this smaller palette. In consequence, all $\vec{c}_k$ in Ω are clustered into a few groups.

When the quantized predicted color $\vec{c}_p$ is determined, $H_p = \sum_{k=0}^{N-1} \mathrm{H}(p,k)$ is checked against a predefined threshold value T. If it is smaller than T, which implies insufficient samples were collected for predicting the real color $\vec{c}_r$ based on $\vec{c}_p$, the

Table of {H(*m*,*n*)}

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
k	0	1	2	3	4	5	6	7
H(0,k)	29	7	6	5	4	3	2	1
H(1,k)	0	88	1	2	0	0	0	1
H(2,k)	0	2	65	1	2	1	0	0
H(3,k)	0	2	10	56	1	1	0	1
H(4,k)	3	1	0	**8**	8	8	0	0
H(5,k)	0	0	3	2	3	23	5	1
H(6,k)	0	0	0	1	1	2	23	2
H(7,k)	0	2	0	3	0	2	6	9

(a)

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
Prediction Error, $\|\vec{c}_k - \vec{v}(i,j)\|^2$	0.6	0.2	0.3	0.3	0.2	0.1	0.2	0.1

(b)

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
* H(4,k)+H(7,k)	3	3	0	11	8	10	6	9
New index	6	5	7	0	3	1	4	2

* Assume that $\vec{c}_4$ and $\vec{c}_7$ are in the same group.

(c)

Figure 6.1 Example of how to assign indices to a dynamic palette when (i) $\vec{v}(i,j)$, $\vec{c}_4$ and $\vec{c}_3$ are, respectively, the predicted, the quantized predicted and the real colors and (ii) Absorbing pre-clustered colors is used in DF-Table merging: (a) current status of the DF-Table, (b) given prediction error $\|\vec{c}_k - \vec{v}(i,j)\|^2$ in the example, (c) index assignment with DF-Table merging

statistics of all colors in the same group with $\vec{c}_p$ will be merged to determine the new index of $\vec{c}_r$.

Without loss of generality, let us assume that $\vec{c}_p$ belongs to cluster $\Phi_p \subset \Omega$. In that case, $\{\vec{c}_k \mid k = 0, 1\ldots N\text{-}1\}$ are sorted according to the values of $\{\sum_{\vec{c}_l \in \Phi_p} \mathrm{H}(l,k) \mid k = 0,1\ldots N\text{-}1\}$ in descending order. If there exist two different colors $\vec{c}_u$ and $\vec{c}_v$ such that $\sum_{\vec{c}_l \in \Phi_p} \mathrm{H}(l,u) = \sum_{\vec{c}_l \in \Phi_p} \mathrm{H}(l,v)$, $\vec{c}_u$ and $\vec{c}_v$ will be sorted according to their Euclidean distance to $\vec{v}(i,j)$, the predicted color of pixel (i,j). If they are still not distinguishable, their order will be determined by their ranking in Ω.

Partition all palette colors in reference palette $\Omega \equiv \{ \vec{c}_k \mid k=0,1 \ldots N-1 \}$ into $N/2$ groups with LBG algorithm.
Initialize DF-Table $\{H(m,n)\}$ by $H(m,n)=0$ for $m,n=0,1 \ldots N-1$.
Raster scan the image
FOR each pixel (i,j)
  Predict the red component of pixel (i,j) with MED in the red color plane to get $r(i,j)$.
  Predict the green component of pixel (i,j) with MED in the green color plane to get $g(i,j)$.
  Predict the blue component of pixel (i,j) with MED in the blue color plane to get $b(i,j)$.
  Quantize $\vec{v}(i,j) = (r(i,j), g(i,j), b(i,j))$ with the reference palette Ω.
  % Assume the quantization result is $\vec{c}_p$, the p^{th} palette color in reference palette Ω
  IF $H_p = \sum_{k=0}^{N-1} H(p,k) \geq$ threshold T
    Sort all palette colors in $\Omega \equiv \{ \vec{c}_k \mid k=0,1 \ldots N-1 \}$ by $H(p,k)$ and then by $\| \vec{c}_k - \vec{v}(i,j) \|^2$ and then by k.
  ELSE
    Sort all palette colors in $\Omega \equiv \{ \vec{c}_k \mid k=0,1 \ldots N-1 \}$ by $\sum_{\vec{c}_l \in \Phi_p} H(l,k)$, where $\Phi_p \subset \Omega$ is the group to which $\vec{c}_p$ belongs, and then by $\| \vec{c}_k - \vec{v}(i,j) \|^2$ and then by k.
  END
  % The sorted palette colors forms the transient version of the palette
  % Assume the real color of pixel (i,j) is $\vec{c}_r$, the r^{th} palette color in reference palette Ω
  Assign the position of $\vec{c}_r$ in the current sorted palette color queue to be the new index of $\vec{c}_r$.
  Update $\{H(m,n)\}$ by $H(p,r)$++.
END

Figure 6.2 Pseudo code of an adaptive palette reordering method which uses *Absorbing pre-clustered colors* to merge the DF table

Let's consider the example shown in Figure 3.2 of Chapter 3 again. Figure 6.1 shows how index is assigned when Absorbing-preclustered-colors is used to merge the DF-Table. Assume that $\vec{c}_4$ and $\vec{c}_7$ belong to the same group and H_4 is now smaller than the threshold. After sorting $\{ \vec{c}_k \mid k=0,1 \ldots 7 \}$ by $H(4,k)+H(7,k)$ and then by $\| \vec{c}_k - \vec{v}(i,j) \|^2$, the new queue is $\{ \vec{c}_3, \vec{c}_5, \vec{c}_7, \vec{c}_4, \vec{c}_6, \vec{c}_1, \vec{c}_0, \vec{c}_2 \}$ and the new index of $\vec{c}_3$ is 0. One can compare Figure 3.2 and Figure 6.1 to contrast their difference in determining the index of the real color.

Figure 6.2 summaries the flow of an adaptive palette reordering method which uses Absorbing-preclustered-colors to merge a DF-Table. The pseudo code provided in Figure 6.2 shows the case when the method supports a DF-Table merging scheme in which N palette colors are divided into $N/2$ groups.

A merged DF-Table can further be merged into an even smaller DF-Table in the same manner when it is necessary. Figures 6.3, 6.4 and 6.5 show how this can be done with an example in which three DF-tables are constructed such that one can use the most appropriate one whenever it is necessary. Note that the color space is reduced to a two-dimensional space for simplicity here. In this example, a palette of 8 palette colors is clustered into 4 clusters and 2 clusters with LBG algorithms separately as shown in Figure 6.3. Based on the clustering results shown in Figure 6.3, palette colors belong to the same cluster are grouped together and the color space is then redivided according to the grouping results as shown in Figure 6.4.

Assume that the current status of the full-size DF-table is given as in Figure 6.5a and the quantized predicted color is $\vec{c}_7$. Entries in the DF-Table shown in Figure 6.5a are merged according to the clustering results or, to be more precise, the partition results shown in Figure 6.4 to produce two smaller DF-Tables. Figures 6.5b and 6.5c show the DF-Tables corresponding to the partition in Figure 6.4b and the partition in Figure 6.4c respectively. Whenever $\vec{c}_7$ is determined, the following steps are executed to sort the palette color.

IF $H_7 = \sum_{k=0}^{N-1} \mathrm{H}(7,k) > \mathrm{T}$

Use the DF-Table shown in Figure 6.5(a). (i.e. Sort $\vec{c}_k \in \Omega$ by H(7,*k*) and then by $\| \vec{c}_k - \vec{v}(i,j) \|^2$ and then by *k*.)

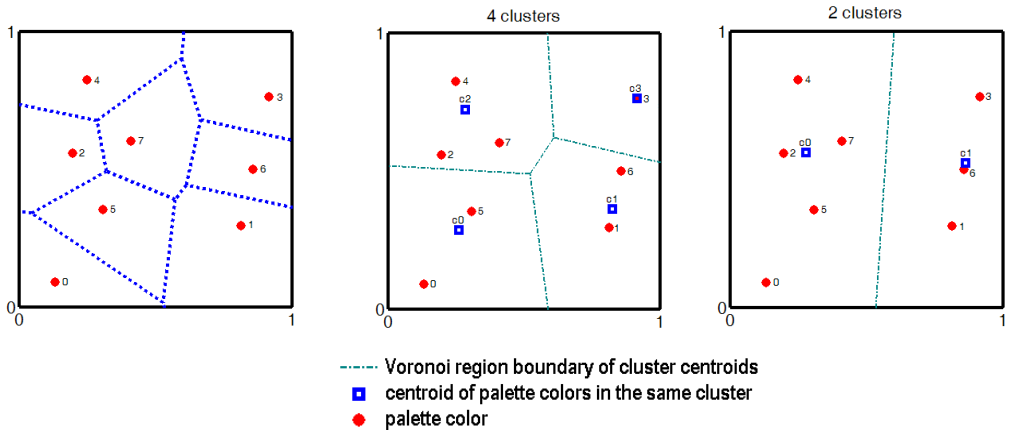
ELSEIF $H_2 + H_4 + H_7 = \sum_{k=0}^{N-1} \{\mathrm{H}(2,k) + \mathrm{H}(4,k) + \mathrm{H}(7,k)\} > \mathrm{T}$

Use the DF-Table shown in Figure 6.5(b). (i.e. Sort $\vec{c}_k \in \Omega$ by {H(2,*k*)+H(4,*k*)+H(7,*k*)} and then by $\| \vec{c}_k - \vec{v}(i,j) \|^2$ and then by *k*.)

ELSE

Use the DF-Table shown in Figure 6.5(c) (i.e. Sort $\vec{c}_k \in \Omega$ by {H(0,*k*)+H(2,*k*)+H(4,*k*)+H(5,*k*)+H(7,*k*)} and then by $\| \vec{c}_k - \vec{v}(i,j) \|^2$ and then by *k*.)

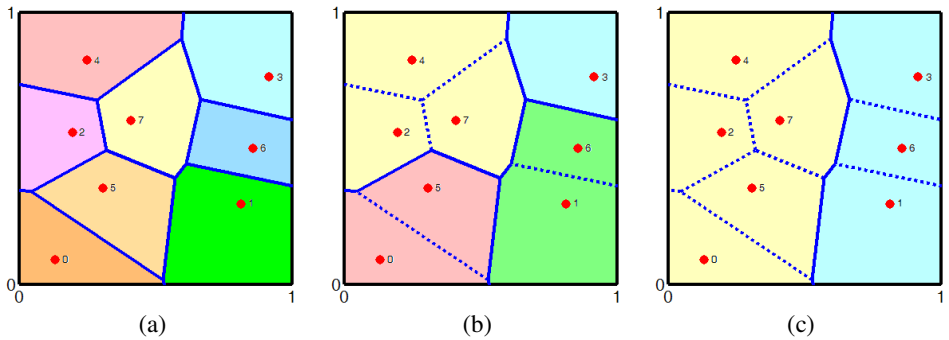
END



(a) (b) (c)

Figure 6.3 Clustering results of LBG algorithm: (a) original palette of 8 colors; (b) 4 clusters and (c) 2 clusters

By doing so, the codec can use a smaller DF-Table whenever a large DF-Table has not yet been mature. As it needs fewer samples to make a smaller DF-Table mature, the proposed palette reordering method can provide a reasonable and steady performance after processing a few samples. Its advantage can be seen even at a very early stage of the index reordering process.



(a) (b) (c)

Figure 6.4 Redivide the color space according to the clustering results of LBG algorithm: (a) partition result associated with the original palette; (b) 4-cluster case and (c) 2-cluster case

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
k	0	1	2	3	4	5	6	7
H(0,k)	2	0	0	0	0	0	0	0
H(1,k)	1	1	0	0	0	0	0	0
H(2,k)	0	1	3	1	0	1	0	0
H(3,k)	0	0	0	1	0	0	0	0
H(4,k)	0	0	0	**1**	0	2	0	0
H(5,k)	0	0	0	0	0	1	0	0
H(6,k)	0	0	0	0	1	0	0	0
H(7,k)	0	0	0	0	0	2	1	1

(a)

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
k	0	1	2	3	4	5	6	7
H(0,k)+ H(5,k)	2	0	0	0	0	1	0	0
H(1,k)+H(6,k)	1	1	0	0	1	0	0	0
H(2,k)+ H(4,k)+ H(7,k)	0	1	3	2	0	5	1	1
H(3,k)	0	0	0	1	0	0	0	0

(b)

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
k	0	1	2	3	4	5	6	7
F(0,k)+ H(2,k)+ H(4,k)+ H(5,k)+ H(7,k)	2	1	3	2	0	6	1	1
H(1,k)+ H(3,k)+ H(6,k)	1	1	0	1	1	0	0	0

(c)

Figure 6.5 A set of three DF-Tables associated with (a) the original palette, (b) the 4-cluster color space and (c) the 2-cluster color space

6.3 Absorbing the nearest colors

In the DF-Table merging scheme presented in Section 6.2, a pre-clustering process has to be carried out to determine which palette colors should be merged when a DF-Table of small size is required. Once the clusters are determined, how to merge a DF-Table is well-defined and fixed. The merging is independent of the quantized predicted color $\vec{c}_p$.

The pre-clustering process could be time-consuming especially when many DF-Tables of different sizes are required. One can get rid of this pre-clustering process by using another merging scheme as follows.

When the quantized predicted color $\vec{c}_p$ is determined, $H_p = \sum_{k=0}^{N-1} \mathrm{H}(p,k)$ is checked against a predefined threshold value T. The DF-Table will be considered to be not mature if it is smaller than T. In such a case, the nearest s palette colors to $\vec{c}_p$ are grouped with $\vec{c}_p$

together, where *s* is a predefined integer. The corresponding DF-Table entries associated with these grouped palette colors are then merged. Accordingly, this DF-Table merging scheme is referred to as Absorbing-the-nearest-colors. After merging, each of the resultant entries specifies the frequency count of the occurrence that $\vec{c}_n$ is the real color when $\vec{c}_p$ is one of the members in the merged group of palette colors. Unlike the case in the merging scheme presented in Section 6.2, the merging is now $\vec{c}_p$ -dependent.

Assume that $\vec{c}_p$ and its nearest *s* palette colors form a set denoted as $\Psi_p \subset \Omega$. In that case, $\{\vec{c}_k \mid k = 0, 1\ldots N\text{-}1\}$ are sorted according to the values of $\{\sum_{\vec{c}_l \in \Psi_p} \mathrm{H}(l,k) \mid k = 0,1\ldots N\text{-}1\}$ in descending order. If there exist two different colors $\vec{c}_u$ and $\vec{c}_v$ such that $\sum_{\vec{c}_l \in \Psi_p} \mathrm{H}(l,u) = \sum_{\vec{c}_l \in \Psi_p} \mathrm{H}(l,v)$, $\vec{c}_u$ and $\vec{c}_v$ will be sorted according to their Euclidean distance to $\vec{v}(i, j)$. If they are still not distinguishable, their order will be determined by their ranking in Ω.

Figure 6.6 shows how index is assigned when Absorbing-the-nearest-colors instead of Absorbing-preclustered-colors is used to handle the example shown in Figure 6.1. Assume that H_4 is now smaller than the threshold and *s* is predefined to be 2. It is given in Figure 6.6b that $\vec{c}_3$ and $\vec{c}_7$ are the nearest two colors to $\vec{c}_4$ and, hence, we have $\Psi_p = \{\vec{c}_3, \vec{c}_4, \vec{c}_7\}$. After sorting $\{\vec{c}_k \mid k=0,1\ldots7\}$ by H(3,*k*)+H(4,*k*)+H(7,*k*) and then by $\| \vec{c}_k - \vec{v}(i, j) \|^2$, the new queue is $\{\vec{c}_3, \vec{c}_5, \vec{c}_7, \vec{c}_2, \vec{c}_4, \vec{c}_6, \vec{c}_1, \vec{c}_0\}$ and the new index of $\vec{c}_3$ is 0. One can compare Figure 6.6 with Figures 6.1 and 3.2 to contrast their difference in determining the index of the real color.

Like the case presented in Section 6.2, a merged DF-Table can further be merged into an even smaller DF-Table when it is necessary. An example is given in Figures 6.7 and 6.8 to show how Absorbing-the-nearest-colors can be used to provide a set of DF-tables of

Table of {H(*m*,*n*)}

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
K	0	1	2	3	4	5	6	7
H(0,*k*)	29	7	6	5	4	3	2	1
H(1,*k*)	0	88	1	2	0	0	0	1
H(2,*k*)	0	2	65	1	2	1	0	0
H(3,*k*)	0	2	10	56	1	1	0	1
H(4,*k*)	3	1	0	**8**	8	8	0	0
H(5,*k*)	0	0	3	2	3	23	5	1
H(6,*k*)	0	0	0	1	1	2	23	2
H(7,*k*)	0	2	0	3	0	2	6	9

(a)

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
Prediction Error, $\|\vec{c}_k - \vec{v}(i,j)\|^2$	0.6	0.2	0.3	0.3	0.2	0.1	0.2	0.1
Distance to $\vec{c}_4$, $\|\vec{c}_k - \vec{c}_4\|^2$	0.3	0.7	0.5	0.2	0.0	0.9	0.6	0.1

(b)

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
*H(3,*k*)+H(4,*k*)+H(7,*k*)	3	5	10	67	9	11	6	10
New index	7	6	3	0	4	1	5	2

*$\vec{c}_7$ and $\vec{c}_3$ are the nearest 2 colors to $\vec{c}_4$.

(c)

Figure 6.6 Example of how to assign indices to a dynamic palette when (i) $\vec{v}(i,j)$, $\vec{c}_4$ and $\vec{c}_3$ are, respectively, the predicted, the quantized predicted and the real colors and (ii) Absorbing the nearest colors is used in DF-Table merging: (a) current status of the DF-Table, (b) given additional information in the example, (c) index assignment with DF-Table merging

different sizes. In this example, the palette shown in Figure 6.3a is used and the current status of the full-size DF-Table is given as in Figure 6.5a for consistency with the example shown in Figures 6.4 and 6.5. The full-size DF-table shown in Figure 6.8a is only a sorted version of that shown in Figure 6.5a.

Assume that the quantized predicted color is $\vec{c}_7$ again. To derive a DF-Table of appropriate size, palette colors are grouped with $\vec{c}_p$ one by one according to their distance to $\vec{c}_p$ until $\sum_{\vec{c}_l \in \Psi_p} \mathrm{H}_l > \mathrm{T}$ or $\Psi_p = \Omega$ is satisfied. The nearest palette colors are grouped with $\vec{c}_p$ first. Figure 6.7 shows how the color space is divided according to the grouping results as

the palette colors are grouped one by one in seven steps until $\Psi_p = \Omega$. In each step, corresponding entries in the DF-Table are merged and the resultant DF-Table is shown in Figure 6.8.

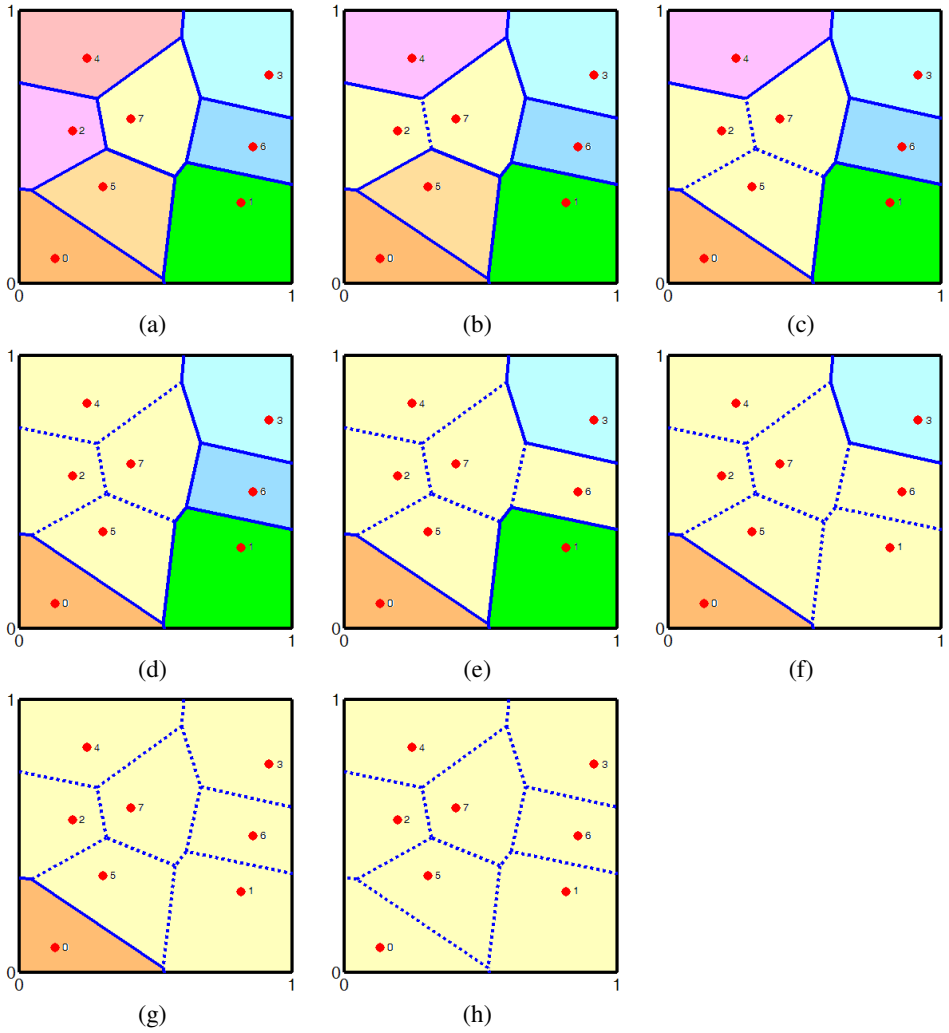


Figure 6.7 Merging steps of absorbing the nearest colors: (a) partition result associated with the original palette; (b) after absorbing the nearest palette color; (c) after absorbing 2 nearest palette colors; (d) after absorbing 3 nearest palette colors; (e) after absorbing 4 nearest palette colors; (f) after absorbing 5 nearest palette colors; (g) after absorbing 6 nearest palette colors; (h) after absorbing all other palette colors

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
K	0	1	2	3	4	5	6	7
H(7,k)	0	0	0	0	0	2	1	1
H(2,k)	0	1	3	1	0	1	0	0
H(5,k)	0	0	0	0	0	1	0	0
H(4,k)	0	0	0	1	0	2	0	0
H(6,k)	0	0	0	0	1	0	0	0
H(1,k)	1	1	0	0	0	0	0	0
H(3,k)	0	0	0	1	0	0	0	0
H(0,k)	2	0	0	0	0	0	0	0

(a)

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
K	0	1	2	3	4	5	6	7
H(7,k)+H(2,k)	0	1	3	1	0	3	1	1
H(5,k)	0	0	0	0	0	1	0	0
H(4,k)	0	0	0	1	0	2	0	0
H(6,k)	0	0	0	0	1	0	0	0
H(1,k)	1	1	0	0	0	0	0	0
H(3,k)	0	0	0	1	0	0	0	0
H(0,k)	2	0	0	0	0	0	0	0

(b)

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
K	0	1	2	3	4	5	6	7
H(7,k)+H(2,k)+H(5,k)	0	1	3	1	0	4	1	1
H(4,k)	0	0	0	1	0	2	0	0
H(6,k)	0	0	0	0	1	0	0	0
H(1,k)	1	1	0	0	0	0	0	0
H(3,k)	0	0	0	1	0	0	0	0
H(0,k)	2	0	0	0	0	0	0	0

(c)

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
K	0	1	2	3	4	5	6	7
H(7,k)+H(2,k)+H(5,k)+H(4,k)	0	1	3	2	0	6	1	1
H(6,k)	0	0	0	0	1	0	0	0
H(1,k)	1	1	0	0	0	0	0	0
H(3,k)	0	0	0	1	0	0	0	0
H(0,k)	2	0	0	0	0	0	0	0

(d)

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
K	0	1	2	3	4	5	6	7
H(7,k)+H(2,k)+H(5,k)+H(4,k)+H(6,k)	0	1	3	2	1	6	1	1
H(1,k)	1	1	0	0	0	0	0	0
H(3,k)	0	0	0	1	0	0	0	0
H(0,k)	2	0	0	0	0	0	0	0

(e)

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
K	0	1	2	3	4	5	6	7
H(7,k)+H(2,k)+H(5,k)+H(4,k) + H(6,k)+H(1,k)	1	2	3	2	1	6	1	1
H(3,k)	0	0	0	1	0	0	0	0
H(0,k)	2	0	0	0	0	0	0	0

(f)

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
K	0	1	2	3	4	5	6	7
H(7,k)+H(2,k)+H(5,k)+H(4,k) + H(6,k)+H(1,k)+H(3,k)	1	2	3	3	1	6	1	1
H(0,k)	2	0	0	0	0	0	0	0

(g)

$\vec{c}_k$	$\vec{c}_0$	$\vec{c}_1$	$\vec{c}_2$	$\vec{c}_3$	$\vec{c}_4$	$\vec{c}_5$	$\vec{c}_6$	$\vec{c}_7$
K	0	1	2	3	4	5	6	7
H(7,k)+H(2,k)+H(5,k)+H(4,k) + H(6,k)+H(1,k)+H(3,k)+H(0,k)	3	2	3	3	1	6	1	1

(h)

Figure 6.8 A set of eight DF-Tables each of which is associated with one of the merging results shown in Figure 6.7a-h

Obviously, when the predicted color is a color very different from $\vec{c}_p$, the probability that the real color is $\vec{c}_r$ is different from the case when the predicted color is $\vec{c}_p$. Hence, it is meaningless to merge the DF-Table by grouping two palette colors which are distant from each other in the color space. In practice, one can bound the size of Ψ_p or the distance of the colors in Ψ_p such that the merging process can be terminated in the course to avoid merging two distant palette colors.

6.4 Performance study

Simulations were carried out to evaluate the performance of the two proposed DF-Table merging schemes in supporting the adaptive palette reordering method proposed in Chapter 3. In the simulations, the value of threshold T was set to be $0.1N$ and the merging terminated when the number of merged colors in the group containing the quantized predicted color was equal to or larger than 8.

Figure 6.9 shows the difference between using DF-Table merging and not using DF-Table merging in the realization of the adaptive palette reordering method proposed in Chapter 3. It shows their difference in the so-far accumulated sum of square index values after processing $\boldsymbol{i}$ pixels. A positive value in the measure means that DF-Table merging provides more indices of small index values. It helps to reduce the overall variance of the indices.

Theoretically, the more significant the reduction in the accumulated sum of square index values, the more indices are packed into a range of small values. In view of this, DF-Table merging helps to produce more small indices, and Absorbing-preclustered-colors behaves better than Absorbing-the-nearest-colors.

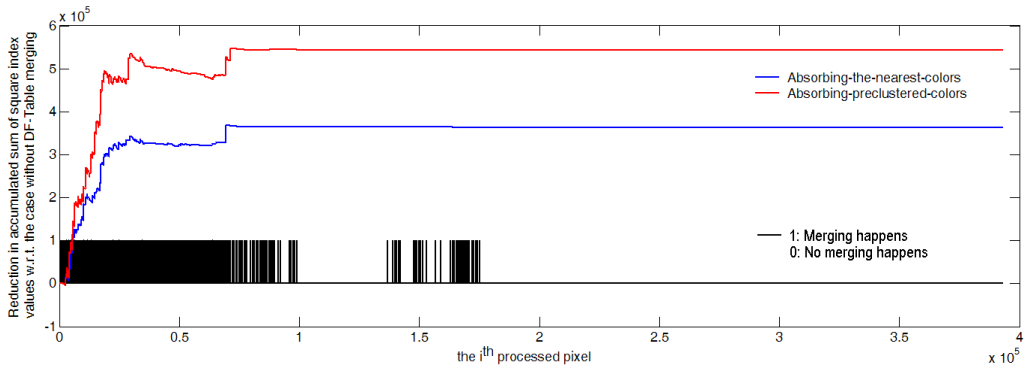


Figure 6.9 Reduction in accumulated sum of square values of indices with respect to the case without DF-Table merging when DF-Table merging is used

From Figure 6.9, one can see that DF-Table merging only happens in the early stage of adaptive palette reordering. As more pixels are processed, the DF-Table gets more mature and it is no longer necessary to merge the DF-Table. When no merging is necessary, Absorbing-preclustered-colors and Absorbing-the-nearest-colors behave in the same way as in the case without DF-Table merging. Processing a pixel in such a case does not result in any difference in the index value by using a DF-Table merging scheme and hence there is no further reduction in the accumulated sum of square index values. Accordingly, the corresponding portions of the curves in Figure 6.9 remain level whenever no merging is necessary.

In contrast to the plot shown in Figure 6.9, the plot shown in Figure 6.10 does not take all processed pixels into account but only those involve DF-Table merging when they are processed. Pixels which do not involve DF-Table merging when they are processed are removed in the plot as they do not contribute to the reduction or addition in the accumulated sum of square index values.

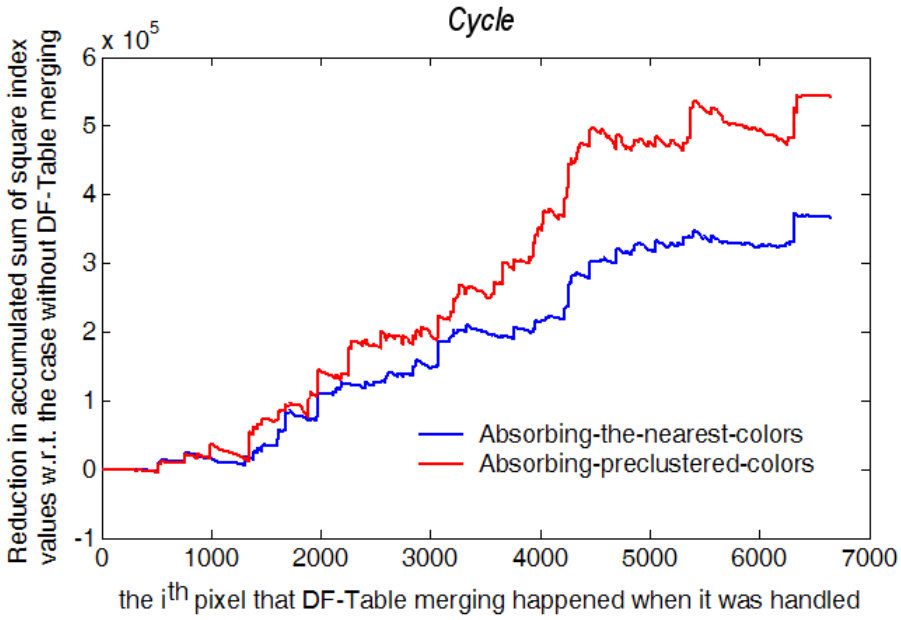


Figure 6.10 Focused portions of the plot shown in Figure 6.9

Table 6.1 shows the simulation results for all testing images in terms of SSIVe. SSIVe is defined to be the sum of the square index values of all effective pixels. Here, effective pixels means the pixels which involve DF-Table merging when they are processed.

Form Table 6.1, one can see that DF-Table merging helps to produce more small indices. The performance of the two proposed DF-Table merging schemes are more or less the same in terms of the average reduction in SSIVe. Absorbing-preclustered-colors is slightly better than Absorbing-the-nearest-colors. This can be observed by comparing the absolute difference between their SSIVe values.

Group	Image	Size (pixel^2)	SSIVe				
			No merging	Absorbing-the-nearest- colors		Absorbing-preclustered-colors	
CG image	pool	510x383	3069875	2675324	(87.1%)	2799979	(91.2%)
	watch	1024x768	9215808	7880299	(85.5%)	7674466	(83.3%)
	water	1024x768	21302066	20909971	(98.2%)	20858138	(97.9%)
Natural image	Kodak 01	768x512	6938379	6101089	(87.9%)	6039125	(87.0%)
	Kodak 02	768x512	7124694	6379827	(89.5%)	6338480	(89.0%)
	Kodak 03	768x512	5523060	5127955	(92.8%)	4725953	(85.6%)
	Kodak 04	768x512	3359119	3046236	(90.7%)	3067039	(91.3%)
	Kodak 05	768x512	7887582	7525028	(95.4%)	7345004	(93.1%)
	Kodak 06	768x512	16201335	12557494	(77.5%)	12516702	(77.3%)
	Kodak 07	768x512	3640236	3320252	(91.2%)	3321282	(91.2%)
	Kodak 08	768x512	8559655	8027324	(93.8%)	8082672	(94.4%)
	Kodak 09	768x512	3314721	3141663	(94.8%)	3243908	(97.9%)
	Kodak 10	768x512	9375159	7921324	(84.5%)	8106321	(86.5%)
	Kodak 11	768x512	6149399	5586054	(90.8%)	5488017	(89.2%)
	Kodak 12	768x512	6394072	5948635	(93.0%)	5698833	(89.1%)
	Kodak 13	768x512	21243252	18263697	(86.0%)	18509351	(87.1%)
	Kodak 14	768x512	4992044	4190197	(83.9%)	4204705	(84.2%)
	Kodak 15	768x512	3090025	2553231	(82.6%)	2628302	(85.1%)
	Kodak 16	768x512	10871548	9412486	(86.6%)	9135033	(84.0%)
	Kodak 17	768x512	7480895	6666806	(89.1%)	6729078	(90.0%)
	Kodak 18	768x512	8381187	7650564	(91.3%)	7745990	(92.4%)
	Kodak 19	768x512	5671096	5326916	(93.9%)	5374973	(94.8%)
	Kodak 20	768x512	11561203	10279368	(88.9%)	10382971	(89.8%)
	Kodak 21	768x512	11113901	10213911	(91.9%)	9986989	(89.9%)
	Kodak 22	768x512	8730668	7657395	(87.7%)	7612550	(87.2%)
	Kodak 23	768x512	908785	866923	(95.4%)	864153	(95.1%)
	Kodak 24	768x512	25911883	20980268	(81.0%)	21140460	(81.6%)
		Average	8815246.2	7785564.3	(88.3%)	7763721.3	(88.1%)

The figure in the bracket is the percentage w.r.t to case of no merging

Table 6.1 Sum of square index values of the pixels which involve DF-Table merging when they are handled

6.5 Summary

The DF-Table plays a significant role in adaptive palette reordering. It helps one to predict the likeliness of the occurrence of a particular color in pixel (i,j) such that an index of smaller value can be assigned to a color which is more likely to happen. This results in a reindexed index map having a number of small indices.

An immature DF-Table cannot provide sufficient statistic information for one to estimate the likeliness of the occurrence of a particular color in pixel (i,j) with its predicted color. To solve this problem, two DF-Table merging schemes are proposed to build DF-Tables of smaller sizes such that the resultant DF-Tables contains fewer zero entries. Eventually, one can have a DF-Table of appropriate size in which there is sufficient statistical information for one to sort the palette colors.

In Absorting-preclustered-colors, palette colors are preclustered before processing the image. Based on the clustering result, palette colors are grouped and corresponding DF-Table entries merges together when the DF-Table is found to be immature. As for Absorting-the-nearest-colors, no preclustering is required. When the DF-Table is found to be immature, the entries associated with the quantized predicted color $\bar{c}_p$ and $\bar{c}_p$'s nearest palette colors are merged together.

The DF-Table merging schemes improve adaptive palette reordering at its early stage. However, as more and more pixels are processed, the DF-Table gets more and more mature and no merging is required any more. Accordingly, when the image being processed is very large, the effect of DF-Table merging can be insignificant.

Chapter 7 – Compression performance

7.1 Introduction

One can see form the previous chapters that adaptive palette reordering can effectively turn the index map of an color indexed image into another index map of little spatial correlation, low variance and low zero-order entropy. An index map of such properties is suitable for being compressed and good compression performance can be easily achieved with different coding techniques. In this chapter, we present several approaches to encode the reindexed index map and show how adaptive palette reordering can easily work with different coding schemes to achieve high compression ratios. In other words, the focus of this chapter is on the index encoding module of the lossless coding system proposed in this work. Figure 7.1 highlights this module in the proposed coding system.

This chapter is organized as follows. In Section 7.2, coders which compile with international coding standards such as JPEG-LS and Lossless JPEG-2000 are used to encode the reindexed index map. This approach makes adaptive palette reordering a preprocessing step and then allows one to use adaptive palette reordering to improve the performance of popular standard coders or off-the-shelf coding systems without modifying their structure.

In Sections 7.3 and 7.4, the constraint of being compatible with international coding standards is released. Bit-plane coding and context-based entropy coding techniques are used to encode reindexed index maps. Simulation results are provided in Section 7.5 to comapre the performance of the proposed coding system and some other conventional lossless coding algorithms. A brief summary is provided in Section 7.6.

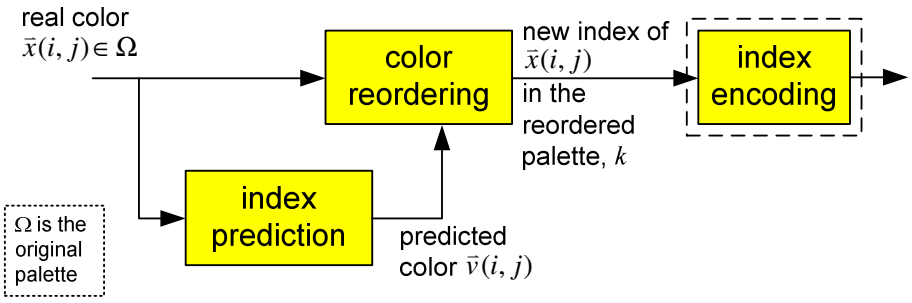


Figure 7.1 The focus of Chapter 7 is on the realization of the index encoding module

7.2 Encoding index maps with JPEG-LS/Lossless JPEG-2000

Since most of the spatial correlation is removed and indices are highly biased in our reordering result, our reindexed output can be easily compressed with a lot of well-developed lossless compression algorithms such as JPEG-LS and JPEG-2000.

JPEG-LS and lossless JPEG-2000 are well-developed lossless image coding standards and hence sometimes it would be convenient for one to make use of them to compress an image directly. Adaptive palette reordering can be used as a pre-/post-processing step. It pre-processes a given color index map to generate an input to a JPEG-LS/JPEG-2000 codec and, at the receiver, post-processes the output from a JPEG-LS/JPEG-2000 decoder to reconstruct the original color-quantized image. To the JPEG-LS/JPEG-2000 codec, the reindexed index map just appears as a grey-level image and it is encoded directly as if it were a grey-level image.

Adaptive palette reordering is fully compatible with JPEG-LS and JPEG-2000 in a way that no modification to these codecs is required to compress an image when the proposed method is exploited.

7.3 Encoding index maps with significance-based bit-plane coding

If being compatible with JPEG-LS/JPEG-2000 is not the concern, an even higher compression ratio can be achieved with some other means such as bit-plane coding. The index values of the reindexed index map are bounded by N, the total number of palette colors. By abandoning the use of bijective mapping (3.1) at the final stage of adaptive palette reordering so as not to shift the peak of the index distribution to *N*/2, most of the index values of the reindexed index map are of values equal or close to zero. This reindexed index map can be directly separated into $\lceil \log_2 N \rceil$ bit planes each of which carries the jth most significant bits of the binary representation of the indices in the reindexed output, where j=1,2…$\lceil \log_2 N \rceil$. Figure 7.2 shows the 8 bit-planes extracted from the adaptive palette reordering result of testing image Kodak-05.

In Chapter 3, we show that there is little correlation among the indices of neighboring pixels. However, it can be found from Figure 7.2 that there is still some spatial correlation among the pixels in individual bit planes. Another important finding is that the bits in the significant bit planes are highly biased to be zero. The entropies of these bit planes are very low. Comparatively, the entropies of insignificant bit planes are much higher but they are bounded by 1 bpp in practice. This bound is based on the fact that the bit plane can be represented without any compression at a bit rate of 1 bit per pixel.

The bit planes can be separately encoded with any bit-plane coding techniques. In this section, two approaches for encoding the bit planes are presented for examples.

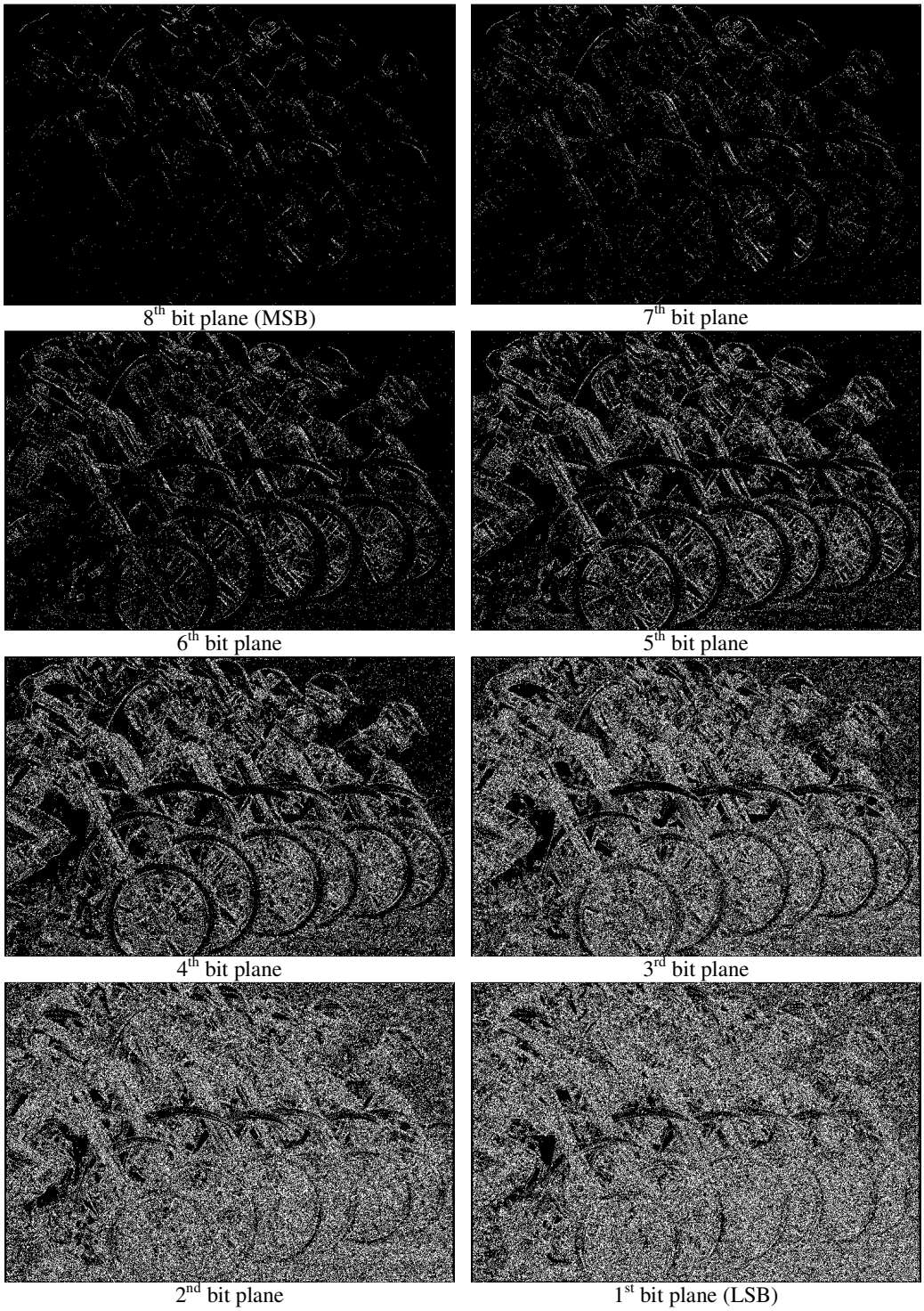


Figure 7.2 The 8 bit planes of the reindexed index map produced with adaptive palette reordering

A. Coding with JBIG

JBIG [JBIG1993] is a popular international coding standard for encoding binary images. To make use of it, each of the bit planes can be encoded with JBIG individually. Our simulation results shown that this approach would provide a better compression result as compared with the approach working with JPEG-LS or JPEG-2000.

B. Coding with a context-based binary arithmetic codec

A QM-coder [Pennebaker88] is used in JBIG to code a binary bit-plane. In approach B, a general context-based binary arithmetic coder is used instead to code a binary bit-plane. Figure 7.3 shows the context template used in this approach. 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

$$P(1 \mid context = binary\ pattern\ i) = (r_i + \Delta)/(s_i + 2\Delta) \quad \text{for } i=0,1\ldots4095 \tag{7.1}$$

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

$$r_i := \begin{cases} \alpha r_i + 1 & if\ the\ current\ pixel\ value\ is\ 1 \\ \alpha r_i & else \end{cases} \tag{7.2}$$

and

$$s_i := 1 + \alpha s_i \tag{7.3}$$

The initial values of r_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 Reavy et al.'s work on binary image compression [Reavy01].

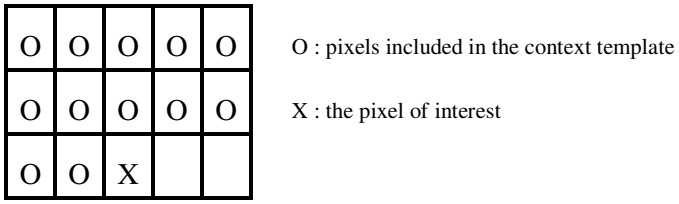


Figure 7.3 A context template used in context-based binary arithmetic codecs

7.4 Encoding index maps with value-based bit-plane coding

In Section 7.3, bit planes are constructed with particular bits of the binary representation of indices according to their bit significance. When the palette is of size 2^m, where *m* is a positive number, we have *m* bit planes. Otherwise, the bit planes cannot be fully utilized and redundancy exists in the bit planes.

As most indices are of values identical or close to zero and the distribution of the index values can be modeled with an exponential function, one can construct bit planes with another approach as

$$B_k(i,j) = \begin{cases} 1 & \text{if } I(i,j) > k \\ 0 & \text{if } I(i,j) = k \\ \text{don't care} & \text{if } I(i,j) < k \end{cases} \quad \text{for } k=0,1,\ldots N\text{-}2 \qquad (7.4)$$

where $B_k(i,j)$ is the $(i,j)^{\text{th}}$ element of the k^{th} bit plane and $I(i,j)$ is the index value of pixel (i,j). Totally, there are N-1 bit planes, which is different from the approach used in

Section 7.3. For reference, this approach of bit-plane separation is referred to as *value-based bit-plane separation* (VBS) while the approach presented in Section 7.3 is referred to as *significance-based bit-plane separation* (SBS). Figure 7.4 shows an example which highlights the difference between VBS and SBS in constructing their bit planes.

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. As bit planes of lower k values are encoded first, don't care $B_k(i, j)$ bits can be skipped. There must be $B_l(i, j)$ =0 for some l<k and it must be encoded already. The values of $I(i, j)$ are well-defined at the moment.

Though apparently VBS results in more bit planes as compared with SBS, the total number of bits to be encoded in VBS can be even less. For the example shown in Figure 7.4, the total number of bits to be encoded in VBS is 38 instead of 48 as in SBS.

The context-based entropy coding is carried out as it is in the second approach presented in Section 7.3 (Approach B) except that its context template is different. It is possible that the 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 variable size is used.

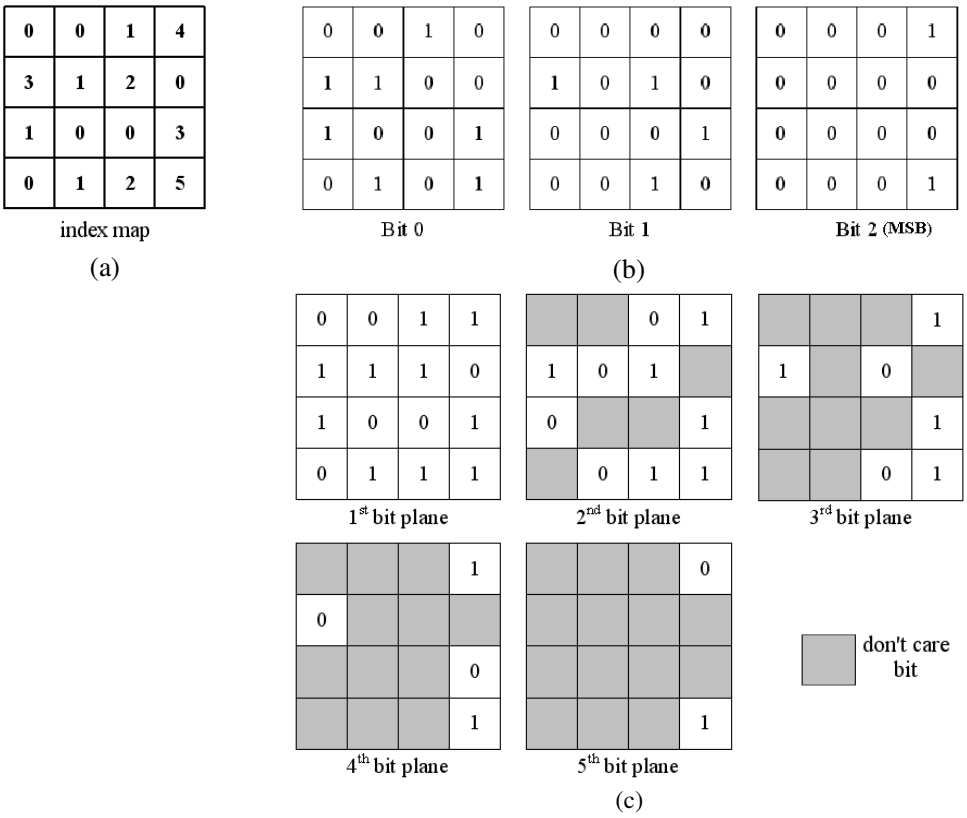


Figure 7.4 An example showing how a 4x4 index map is split into bit planes: (a) index map; (b) the bit planes obtained with SBS and (c) the bit planes obtained with VBS

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

X : the pixel of interest

Figure 7.5 The context template used in VBS

Figure 7.5 shows a context template whose pixel positions are numbered. Instead of using all template locations, only first L positions are used, where L is a function of k. Two functions are used in our study. The first function $f_1(k)$ is defined as

$$f_1(k) = \min\left(\left\lceil \log_2 \frac{N_o{}^n 2^b}{k+1} \right\rceil, 12\right) \qquad \text{for } k=0,1,\ldots N\text{-}2 \qquad (7.5)$$

where N_o is the total number of pixels of the image, n=0.671 and b= -0.859. The values of n and b are selected based on Pinho's work [Pinho05]. The second function $f_2(k)$ is defined as

$$f_2(k) = \lceil 9 - \log_2(k+1) \rceil, \qquad \text{for } k=0,1,\ldots N\text{-}2 \qquad (7.6)$$

For the purpose of reference, the index coding scheme using VBS to construct bit planes and function $f_1(k)$ to define the context template for entropy coding is referred to as VBS1 hereafter. Accordingly, the index coding scheme using VBS to construct bit planes and function $f_2(k)$ to define the context template for entropy coding is referred to as VBS2.

7.5 Simulation Results

When working with adaptive palette reordering, the various coding schemes presented earlier in this chapter form corresponding lossless coding methods for coding color-indexed images. Simulations were carried out to evaluate the performance of these coding schemes in coding the resultant reindexed index maps of adaptive palette reordering. Besides, a comparison among various conventional and the proposed lossless image coding methods were done. This section presents the evaluation results and the comparison results in our simulations.

For the purpose of reference, the corresponding lossless coding methods of the six index coding schemes presented in Sections 7.2, 7.3 and 7.4 are, respectively, referred to as

APR+Jls	:	Encoding adaptive palette reordering output with JPEG-LS
APR+J00	:	Encoding adaptive palette reordering output with lossless JPEG-2000
APR+SBSa	:	Encoding adaptive palette reordering output with Approach A presented in Chapter 7.3
APR+SBSb	:	Encoding adaptive palette reordering output with Approach B presented in Chapter 7.3
APR+VBS1	:	Encoding adaptive palette reordering output with VBS1 presented in Chapter 7.4
APR+VBS2	:	Encoding adaptive palette reordering output with VBS2 presented in Chapter 7.4

As discussed in Chapters 4, 5 and 6, various prediction schemes and color reordering schemes can be used to realize adaptive palette reordering. In the study we reported in this chapter, adaptive palette reordering was realized in a way similar to the method proposed in Chapter 3. The only difference was that DF-Table merging was on in this study. Starting from 256, the number of groups into which all palette colors were clustered was halved progressively until either it reached 8 or $\sum_{\vec{c}_l \in \Phi_p} \sum_{k=0}^{N-1} \mathrm{H}(l,k) > T$, where $\Phi_p \subset \Omega$ is the group to which $\vec{c}_p$ belongs, was satisfied. The threshold T was selected to be 0.1*N*. Figure 7.6 shows the pseudo code for realizing adaptive palette ordering in this study.

A set of testing color-indexed images were generated as mentioned in Chapter 3. They were used for comparing the proposed coding methods with various lossless coding methods. The evaluated coding methods cover both methods which exploit palette reordering [Po94, Spira01, Zaccarin93, Memon97, Zeng00, Pinho04, Battiato01] and methods which do not [Natale89, Ratnakar98, Chen02, Kuroki04, Pinho05, Robinson06, Arnavut06].

Partition all palette colors in reference palette $\Omega \equiv \{ \vec{c}_k \mid k=0,1\ldots N-1 \}$ into $N/2$ groups with LBG algorithm.

% The N/2 groups are denoted as $\Psi_v^{N/2} \subset \Omega$ *for v=0,1…N/2-1*

Partition all palette colors in reference palette Ω into $N/4$ groups with LBG algorithm.

% The N/4 groups are denoted as $\Psi_v^{N/4} \subset \Omega$ *for v=0,1…N/4-1*

…

Partition all palette colors in reference palette Ω into 8 groups with LBG algorithm.

% The 8 groups are denoted as $\Psi_v^{8} \subset \Omega$ *for v=0,1…7*

Initialize DF-Table $\{H(m,n)\}$ by $H(m,n)=0$ for $m,n=0,1\ldots N-1$.

Raster scan the image

FOR each pixel (i,j)

Predict the red component of pixel (i,j) with MED in the red color plane to get $r(i,j)$.

Predict the green component of pixel (i,j) with MED in the green color plane to get $g(i,j)$.

Predict the blue component of pixel (i,j) with MED in the blue color plane to get $b(i,j)$.

Quantize $\vec{v}(i,j) = (r(i,j), g(i,j), b(i,j))$ with the reference palette Ω.

% Assume the quantization result is $\vec{c}_p$, *the* p^{th} *palette color in reference palette Ω, and*

% $\Psi_p^g \subset \Omega$, *for g∈ {N/2,N/4…8}, are the groups to which* $\vec{c}_p$ *belongs.*

IF $H_p = \sum_{k=0}^{N-1} \mathrm{H}(p,k) \geq$ threshold T

Sort all palette colors in $\Omega \equiv \{ \vec{c}_k \mid k=0,1\ldots N-1 \}$ by $H(p,k)$ and then by $\| \vec{c}_k - \vec{v}(i,j) \|^2$ and then by k.

ELSE

IF $\sum_{\vec{c}_l \in \Psi_p^{N/2}} \sum_{k=0}^{N-1} \mathrm{H}(l,k) \geq$ threshold T; $\Phi_p = \Psi_p^{N/2}$

ELSEIF $\sum_{\vec{c}_l \in \Psi_p^{N/4}} \sum_{k=0}^{N-1} \mathrm{H}(l,k) \geq$ threshold T; $\Phi_p = \Psi_p^{N/4}$

ELSEIF …

ELSEIF $\sum_{\vec{c}_l \in \Psi_p^{16}} \sum_{k=0}^{N-1} \mathrm{H}(l,k) \geq$ threshold T; $\Phi_p = \Psi_p^{16}$

ELSE $\Phi_p = \Psi_p^{8}$

END

Sort all palette colors in $\Omega \equiv \{ \vec{c}_k \mid k=0,1\ldots N-1 \}$ by $\sum_{\vec{c}_l \in \Phi_p} \mathrm{H}(l,k)$ and then by $\| \vec{c}_k - \vec{v}(i,j) \|^2$ and then by k.

END

% The sorted palette colors forms the transient version of the palette

% Assume the real color of pixel (i,j) is $\vec{c}_r$, *the* r^{th} *palette color in reference palette Ω*

Assign the position of $\vec{c}_r$ in the current sorted palette color queue to be the new index of $\vec{c}_r$.

Update $\{H(m,n)\}$ by $H(p,r)$++.

END

Figure 7.6 Pseudo code of the adaptive palette reordering method used in APR+Jls, APR+J00, APR+SBSa, APR+SBSb, APR+VBS1 and APR+VBS2.

Group	Image	Size (pixel^2)	Bits per pixel									
			Zaccarin93	Po94	Spira01	Pinho04	Zeng00	Battiato01	Memon96	Memon97	Lai07	APR+Jls
CG image	pool	510x383	1.965	1.540	2.270	1.462	1.560	1.786	1.464	1.559	1.422	1.370
	watch	1024x768	2.436	2.658	2.432	2.169	2.273	2.632	2.093	2.269	2.085	1.956
	water	1024x768	5.673	6.287	5.641	5.621	6.207	7.089	5.487	5.350	5.291	4.878
Natural image	Kodak 01	768x512	5.956	6.481	6.000	5.599	5.969	7.055	5.403	5.346	5.375	4.911
	Kodak 02	768x512	5.277	6.581	6.311	4.383	4.605	5.418	4.234	4.252	4.208	3.734
	Kodak 03	768x512	3.587	3.361	3.264	2.742	2.875	3.548	2.723	2.836	2.731	2.509
	Kodak 04	768x512	4.765	5.139	5.801	4.312	4.953	5.447	3.920	4.010	3.879	3.451
	Kodak 05	768x512	5.669	6.332	5.710	5.204	5.390	6.408	5.029	5.108	4.962	4.469
	Kodak 06	768x512	4.997	5.633	5.405	5.153	5.413	5.921	4.976	5.079	4.853	4.232
	Kodak 07	768x512	4.208	4.339	4.421	3.857	4.069	4.698	3.667	3.450	3.627	3.065
	Kodak 08	768x512	5.554	6.242	5.872	5.852	5.965	6.404	5.797	5.545	5.518	4.587
	Kodak 09	768x512	4.423	4.720	4.484	4.100	4.514	5.506	3.952	4.131	3.712	3.481
	Kodak 10	768x512	4.551	4.870	4.840	4.826	5.339	5.969	4.505	4.580	4.389	3.652
	Kodak 11	768x512	5.024	5.177	4.786	4.458	4.890	4.799	4.334	4.347	4.294	3.771
	Kodak 12	768x512	4.316	5.118	4.934	3.960	4.403	5.329	3.868	3.786	3.740	3.302
	Kodak 13	768x512	6.482	6.976	6.585	6.203	6.509	7.016	5.968	6.381	5.881	5.592
	Kodak 14	768x512	5.369	5.629	5.345	4.717	5.140	6.482	4.497	4.406	4.454	4.090
	Kodak 15	768x512	4.089	4.629	4.350	3.636	3.883	4.332	3.537	3.374	3.526	3.046
	Kodak 16	768x512	4.558	4.977	5.032	4.386	4.817	5.995	4.254	4.282	4.132	3.645
	Kodak 17	768x512	4.510	5.252	4.371	4.818	5.526	5.763	4.335	4.321	4.074	3.717
	Kodak 18	768x512	5.906	6.674	6.149	5.220	5.777	6.762	5.276	5.276	4.991	4.568
	Kodak 19	768x512	5.018	5.363	5.824	4.845	5.104	5.987	4.656	4.661	4.466	3.972
	Kodak 20	768x512	3.201	3.465	3.726	3.053	3.326	3.690	3.144	3.118	3.039	2.897
	Kodak 21	768x512	5.231	5.622	4.747	4.544	4.916	5.563	4.422	4.697	4.297	3.995
	Kodak 22	768x512	5.242	5.869	5.327	5.119	5.708	5.969	4.990	4.865	4.838	4.081
	Kodak 23	768x512	3.489	3.578	3.228	2.890	3.252	3.297	3.091	3.165	2.835	2.432
	Kodak 24	768x512	5.084	6.122	5.525	5.017	5.367	5.951	4.830	4.921	4.712	4.224
		Average	4.688	5.135	4.903	4.376	4.731	5.364	4.239	4.264	4.123	3.690

Table 7.1 Performance of different palette reordering methods when working with JPEG-LS

Tables 7.1 and 7.2 list the compression performance of various coding methods which exploit palette reordering. They are common in a way that the palette associated with the image being encoded is reordered to generate a new index map for being encoded. The resultant index map will be treated as a grey level image and hence can be encoded with any lossless gray-level image coding standards such as JPEG-LS and JPEG-2000. Table 7.1 and Table 7.2 show, respectively, the case using JPEG-LS and the case using JPEG-2000.

Specifically, the last column of Table 7.1 shows the performance of APR-Jls while the last column of Table 7.2 show the performance of APR-J00.

We have two observations from the Tables. First, the performance of the adaptive palette reordering is the best among the evaluated palette reordering methods. No matter JPEG-LS or JPEG-2000 is used, its palette reordering results can be encoded with minimum number of bits as compared with the others. Second, JPEG-LS is more suitable than JPEG-2000 to encode palette reordering results whatever palette reordering methods are used. JPEG-LS generally provides a better performance than JPEG-2000 in coding the reordering results in terms of bit rate of the output.

Table 7.3 compares the compression performance of the proposed coding methods (APR+SBSa, APR+SBSb, APR+VBS1 and APR+VBS2) with various state-of-art lossless coding methods which are not bounded to exploit palette reordering. One can see that APR+SBSb, APR+VBS1 and APR+VBS2 are the best three in terms of compression ratio while APR+SBSa is the fifth best. Without the constraint of being compatible with JPEG-LS or JPEG-2000, the output of adaptive palette reordering can be encoded at an even lower bit rate. This can be revealed by comparing the performance of APR-Jls and APR-J00 with APR+SBSa, APR+SBSb, APR+VBS1 and APR+VBS2. Among the six lossless image coding methods, APR+VBS2 provides the best bit rate. This implies that VBS2 is more effective in encoding the output of adaptive palette reordering.

Group	Image	Size (pixel^2)	Bits per pixel									
			Zaccarin93	Po94	Spira01	Pinho04	Zeng00	Battiato01	Memon96	Memon97	Lai07	APR+J00
CG image	pool	510x383	2.701	2.031	3.150	1.860	1.985	2.234	1.894	2.021	1.880	1.626
	watch	1024x768	3.287	3.568	3.360	2.620	2.760	3.432	2.770	2.971	2.777	2.316
	water	1024x768	6.112	7.128	6.477	6.023	6.563	7.735	5.976	5.840	5.881	5.170
Natural image	Kodak 01	768x512	6.257	6.972	6.373	5.529	5.846	7.500	5.712	5.683	5.764	5.144
	Kodak 02	768x512	5.764	7.150	6.824	4.704	5.094	6.435	4.679	4.746	4.684	4.008
	Kodak 03	768x512	4.449	4.000	4.002	2.996	3.118	4.139	3.318	3.495	3.338	2.756
	Kodak 04	768x512	5.242	5.946	6.156	4.685	5.359	6.042	4.466	4.616	4.368	3.654
	Kodak 05	768x512	6.136	7.285	6.451	5.721	5.750	6.969	5.658	5.744	5.575	4.839
	Kodak 06	768x512	5.274	5.996	5.898	5.207	5.579	6.581	5.400	5.498	5.292	4.450
	Kodak 07	768x512	4.872	5.075	5.193	4.197	4.494	5.491	4.267	4.062	4.210	3.372
	Kodak 08	768x512	5.917	6.654	6.703	6.351	6.254	6.792	6.386	6.117	6.082	4.896
	Kodak 09	768x512	4.824	5.098	4.862	4.136	4.474	5.958	4.347	4.560	4.091	3.658
	Kodak 10	768x512	5.043	5.624	5.379	4.962	5.236	6.661	5.062	5.159	4.957	3.891
	Kodak 11	768x512	5.511	5.664	5.309	4.809	5.003	5.281	4.756	4.819	4.718	3.999
	Kodak 12	768x512	4.841	5.706	5.506	3.995	4.328	5.943	4.330	4.250	4.259	3.532
	Kodak 13	768x512	6.679	7.196	7.062	6.374	6.579	7.539	6.279	6.784	6.248	5.773
	Kodak 14	768x512	5.741	6.237	5.798	5.195	5.500	7.252	4.849	4.757	4.882	4.323
	Kodak 15	768x512	4.458	5.255	4.876	3.911	4.156	4.787	3.963	3.888	4.017	3.249
	Kodak 16	768x512	5.028	5.502	6.061	4.567	5.109	6.900	4.737	4.827	4.648	3.875
	Kodak 17	768x512	4.831	5.749	4.804	4.679	4.881	6.075	4.735	4.732	4.488	3.929
	Kodak 18	768x512	6.157	7.387	6.619	5.808	6.421	7.200	5.614	5.625	5.301	4.740
	Kodak 19	768x512	5.313	5.859	6.282	4.945	5.099	6.407	5.013	5.051	4.830	4.104
	Kodak 20	768x512	3.469	3.926	4.102	3.267	3.477	4.049	3.535	3.484	3.418	3.109
	Kodak 21	768x512	5.652	6.003	5.478	4.978	5.241	6.163	4.695	5.134	4.630	4.184
	Kodak 22	768x512	5.605	6.339	5.900	5.693	5.838	6.504	5.483	5.300	5.338	4.286
	Kodak 23	768x512	4.467	4.386	4.380	3.727	3.941	4.203	3.865	3.968	3.579	2.722
	Kodak 24	768x512	6.780	6.215	5.592	5.923	6.945	5.343	5.474	5.558	5.325	4.568
		Average	5.200	5.702	5.504	4.699	5.001	5.912	4.713	4.766	4.614	3.932

Table 7.2 Performance of different palette reordering methods when working with Lossless JPEG-2000

Group	Image	Size (pixel^2)	Bits per pixel												
			GIF	PNG	Natale89*	Ratnakar98	Chen02	Pinho05	Robinson06	Kuroki04	Arnavut06	Adaptive palette reordering			
												+SBS		+VBS	
												+SBSa	+SBSb	+VBS1	+VBS2
CG image	pool	510x383	1.724	1.591	3.455	1.657	1.288	1.235	1.451	1.919	1.309	1.258	1.255	1.148	1.129
	watch	1024x768	2.619	2.036	3.986	2.221	1.936	1.913	2.238	2.590	1.674	1.880	1.803	1.661	1.625
	water	1024x768	6.571	5.696	5.211	5.244	4.943	4.923	5.731	4.926	4.484	4.801	4.505	4.493	4.360
Natural image	Kodak 01	768x512	6.365	5.491	5.433	5.393	4.905	4.870	5.757	5.776	4.750	4.831	4.574	4.591	4.430
	Kodak 02	768x512	4.967	4.433	4.588	4.229	3.952	3.939	4.602	4.433	3.611	3.591	3.412	3.365	3.274
	Kodak 03	768x512	3.328	2.858	3.727	2.745	2.580	2.507	2.845	2.695	2.161	2.317	2.267	2.176	2.139
	Kodak 04	768x512	4.967	4.285	4.441	3.997	3.817	3.741	4.386	3.873	3.309	3.355	3.198	3.136	3.063
	Kodak 05	768x512	6.203	5.244	5.396	4.921	4.776	4.725	5.229	5.026	4.372	4.424	4.192	4.169	4.024
	Kodak 06	768x512	5.284	4.644	4.673	4.686	4.410	4.377	5.154	4.599	3.817	4.154	3.955	3.972	3.829
	Kodak 07	768x512	4.403	3.733	4.321	3.423	3.353	3.263	3.778	3.376	2.843	2.932	2.797	2.725	2.659
	Kodak 08	768x512	6.750	5.680	5.479	5.298	5.051	5.010	5.798	5.549	4.785	4.513	4.244	4.216	4.068
	Kodak 09	768x512	4.933	4.265	4.293	3.880	3.593	3.517	4.289	4.111	3.345	3.369	3.207	3.159	3.091
	Kodak 10	768x512	5.196	4.487	4.454	4.009	3.870	3.773	4.671	4.131	3.552	3.553	3.376	3.331	3.240
	Kodak 11	768x512	4.790	4.291	4.517	4.235	3.959	3.900	4.683	4.253	3.556	3.659	3.486	3.460	3.349
	Kodak 12	768x512	4.073	3.626	3.826	3.510	3.340	3.272	4.058	3.423	2.904	3.174	3.025	2.962	2.892
	Kodak 13	768x512	7.143	6.131	5.861	5.978	5.683	5.659	6.312	6.188	5.268	5.557	5.265	5.383	5.168
	Kodak 14	768x512	5.560	4.851	4.766	4.639	4.483	4.429	4.963	4.356	3.873	4.002	3.806	3.790	3.677
	Kodak 15	768x512	4.537	3.880	4.206	3.499	3.371	3.294	3.642	3.519	2.908	2.904	2.771	2.702	2.653
	Kodak 16	768x512	4.305	3.895	4.222	3.988	3.682	3.639	4.614	3.943	3.210	3.548	3.385	3.365	3.257
	Kodak 17	768x512	5.486	4.774	4.695	4.283	3.918	3.817	4.742	4.278	3.644	3.616	3.441	3.416	3.321
	Kodak 18	768x512	6.335	5.593	5.077	5.133	5.110	5.051	5.435	4.965	4.275	4.467	4.232	4.245	4.108
	Kodak 19	768x512	5.857	5.099	4.780	4.656	4.235	4.171	4.827	4.595	3.867	3.881	3.683	3.670	3.569
	Kodak 20	768x512	3.522	3.185	3.968	3.365	2.892	2.833	3.383	3.404	2.669	2.785	2.681	2.647	2.547
	Kodak 21	768x512	4.818	4.385	4.371	4.383	4.043	3.979	4.922	4.385	3.620	3.898	3.714	3.716	3.583
	Kodak 22	768x512	5.505	4.916	4.710	4.460	4.347	4.274	5.182	4.577	3.975	3.954	3.765	3.747	3.624
	Kodak 23	768x512	3.640	3.017	3.934	2.517	2.589	2.478	2.855	2.609	2.333	2.267	2.176	2.084	2.048
	Kodak 24	768x512	5.767	4.939	4.963	4.573	4.4632	4.3927	6.452	4.956	4.125	4.107	3.903	3.837	3.690
		Average	4.987	4.334	4.553	4.090	3.874	3.814	4.518	4.292	3.490	3.585	3.412	3.376	3.275

* Sub-codebook size = 8

Table 7.3 Performance of various lossless image coding methods for coding color-indexed images

Table 7.4 shows the actual processing time of compressing color-indexed images with the proposed dynamic palette reordering algorithm. The simulation results were done on a 2.4GHz Celeron PC with 512MB RAM. The average processing time per pixel is about 2.83 microseconds. Palettes of the testing images are of size 256.

Group	Image	Size (pixel^2)	Time	
			Seconds	μs/pixel
CG image	pool	510x383	0.640	3.28
	watch	1024x768	1.296	1.65
	water	1024x768	2.453	3.12
Natural image	Kodak 01	768x512	1.235	3.14
	Kodak 02	768x512	1.156	2.94
	Kodak 03	768x512	0.890	2.26
	Kodak 04	768x512	0.641	1.63
	Kodak 05	768x512	0.844	2.15
	Kodak 06	768x512	1.047	2.66
	Kodak 07	768x512	1.000	2.54
	Kodak 08	768x512	1.109	2.82
	Kodak 09	768x512	0.875	2.23
	Kodak 10	768x512	1.546	3.93
	Kodak 11	768x512	0.953	2.42
	Kodak 12	768x512	0.844	2.15
	Kodak 13	768x512	1.375	3.50
	Kodak 14	768x512	1.172	2.98
	Kodak 15	768x512	0.985	2.50
	Kodak 16	768x512	0.937	2.38
	Kodak 17	768x512	1.484	3.77
	Kodak 18	768x512	1.266	3.22
	Kodak 19	768x512	1.813	4.61
	Kodak 20	768x512	0.875	2.23
	Kodak 21	768x512	1.391	3.54
	Kodak 22	768x512	0.922	2.34
	Kodak 23	768x512	1.062	2.70
	Kodak 24	768x512	1.500	3.81
		Average	1.160	2.83

Table 7.4 The actual processing time of compressing color-indexed images with the proposed dynamic palette reordering algorithm

7.6 Summary

After adaptive palette reordering, the output can be encoded with various index encoding schemes. Different index encoding schemes provide different compression performance in terms of bit rate. Six different index encoding schemes were studied and their performance in encoding reindexed index maps are presented in this chapter. It was found that the performance of VBS2 is the best among the six evaluated index encoding schemes.

Simulation was also carried out to compare the compression performance of various lossless image coding methods including those exploited palette reordering and those did not. Results reported in this chapter shows that APR-VBS2 is superior to all other coding methods in terms of bit rate.

There are some other interesting findings in the simulation results. First, JPEG-LS generally performs better that JPEG-2000 to encode palette reordering results. Second, adaptive palette ordering can significantly reduce the output bit rate when it is used to process a color-indexed image before compressing the image. Third, without concerning the constraint of being compatible with JPEG-LS/JPEG-2000, one can use an even better encoding scheme such as VBS2 to lower the output bit rate further.

Chapter 8 – Conclusions

8.1 Summary of the work

A lossless image coding technique for coding color-indexed images is proposed in this work. The core of the technique relies on a palette reordering technique which adaptively reorders the palette associated with a color-indexed image such that the resultant index map can be encoded more effectively. Unlike other conventional palette reordering techniques, the proposed adaptive palette reordering technique does not produce a directly-displayable index map. However, it is able to effectively reshape the statistical properties of a given color index map to produce a new index map of low variance, low zero-order entropy and low spatial correlation. As a result, this allows one to encode the index map at a very low bit rate easily.

The proposed adaptive palette reordering technique is composed of two basic functional components including *index prediction* and *color reordering*. These two components can be realized with different schemes and, accordingly, their different combinations result in different adaptive palette reordering methods. Chapter 3 shows one of the realizations of adaptive palette reordering. Simulation results show that, as compared with the other evaluated palette reordering methods, it is able to produce a new index map of lower variance, lower zero-order entropy and less spatial correlation.

Index prediction is one of the critical components of adaptive palette reordering, and it can be realized with different schemes. Four schemes are studied and their prediction performance is evaluated in Chapter 4. These schemes exploits MED or GAP to carry out the prediction in either the index plane or individual color intensity planes. In terms of various measures related to prediction error, the scheme which exploits MED to predict individual color components appears to be better. The same scheme is the winner when we inspect the entropy and the variance of their resultant reindexed index maps.

Color reordering is another critical component of adaptive palette reordering. Five color reordering schemes were evaluated in Chapter 5 to see how well they can support adaptive palette reordering. In particular, we have two distance-oriented sorting schemes, one history-oriented scheme and two hybrid mode sorting schemes in our evaluation. Simulation results show that hybrid mode sorting schemes are better than history-oriented sorting schemes and distance-oriented sorting schemes in terms of both zero-order entropy and variance of the reindexed index maps.

In adaptive palette reordering, when color reordering is realized with the proposed history-oriented sorting scheme or one of the hybrid mode sorting schemes, a DF-Table plays a significant role to provide some supporting information learnt from the history. It helps one to assign an index of smaller value to a color which is more likely to happen and hence produce a reindexed index map of a number of small indices.

The DF-Table is immature when it contains a lot of zero entries, and this happens when the number of processed pixels is too small. To solve this problem, two DF-Table merging schemes, namely, Absorting-preclustered-colors and Absorting-the-nearest-colors, are proposed in Chapter 6 to build DF-Tables of smaller sizes such that the resultant DF-Tables contains fewer zero entries. It allows one to select a DF-Table of appropriate size whenever it is necessary. This helps at the early stage of adaptive palette reordering. However, as more and more pixels are processed, the DF-Table gets mature and DF-Table merging is no longer required.

Adaptive palette reordering produces a reindexed index map of low entropy, low variance and little spatial correlation. This index map can be encoded with various index encoding schemes easily. In Chapter 7, we proposed six index encoding schemes to encode the output of adaptive palette reordering. It was found that the performance of VBS2 is the best among the six index encoding schemes in terms of output bit rate.

Simulation was also carried out to compare the compression performance of various lossless image coding methods including those exploited palette reordering and those did not. Results shows that APR-VBS2, which adaptively reorders the palette with DF-Table merging and then encodes the output of adaptive palette reordering with index encoding scheme VBS2, is superior to all other coding methods in terms of output bit rate. It achieves an average compression ratio of 2.44:1.

There are some other interesting findings in the simulation results. First, JPEG-LS generally performs better than JPEG-2000 to encode palette reordering results. Second, adaptive palette reordering can significantly reduce the output bit rate when it is used to process a color-indexed image before compressing the image. Third, without concerning the constraint of being compatible with JPEG-LS/JPEG-2000, one can pick a better encoding scheme such as VBS2 to achieve an even lower output bit rate.

8.2 Future research directions

As we mentioned in this thesis, dynamic palette reordering is composed of two critical components involving index prediction and color reordering. There are unlimited possible ways to realize index prediction and color reordering. After having the output of adaptive palette reordering, there are also unlimited possible ways to encode the index map. In this

work, we investigated some possible schemes for realizing index prediction, color reordering and index encoding. They were combined flexibly to form different lossless coding methods for coding color-indexed images, and it was found that APR-VBS2 could achieve an average output bit rate as low as 3.275bpp. However, we cannot exclude the possibility that there exist some other combinations of some other realizations of index prediction, color reordering and index encoding to provide an even better compression performance. To explore this possibility forms a direction for the extension of the present work.

In our study, the evaluation of the performance of various lossless coding methods was based on a set of testing color-indexed images. These color-indexed images were generated by color-quantizing a set of standard 24-bit full-color images. No halftoning [Floyd76] was performed in the color quantization. From our evaluation results, it can be find that the proposed lossless coding methods perform outstandingly for this kind of images. The question is whether the proposed methods perform equally remarkable when the input images are halftoned images. For halftoned images, there is little spatial correlation in the color intensity planes, and hence index prediction is not that easy. Dedicated solutions are required to take care of these images. It would be another interesting extension of the present work to sort out this problem.

All coding methods proposed in this work support lossless image compression. In Internet communications, sometimes it is advantageous to transmit an image progressively. This allows the receiver to have a lower quality version of the image and quit before the transmission is completed. It would be another meaningful direction for one to modify the proposed coding methods to support both near-lossless compression and progressive mode compression.

Bibliography

[Arnavut99] Z. Arnavut, “Lossless compression of pseudo-color images,” Optical Engineering, vol. 38, no. 6, 1999, pp.1001-1005.

[Arnavut05] Z. Arnavut and F. Sahin, “Inversion ranks for lossless compression of palette images,” in Proc IEEE International Conf. on Electro Information Technology, 22-25 M ay 2005.

[Arnavut06] Z. Arnavut and F. Sahin, “Lossless compression of color palette images with 1-dimensional techniques,” Journal of Electronic Imaging, vol. 15, no. 2, 023014, Apr-Jun 2006.

[Battiato01] S. Battiato, G. Gallo, G. Impoco, and F. Stanco, “A color reindexing algorithm for lossless compression of digital images,” in Proc. IEEE Spring Conf. Computer Graphics Budmerice, Slovakia, Apr. 2001, pp. 104-108.

[Burrows94] M. Burrows and D. J. Wheeler, “A block-sorting lossless data compression algorithm,” SRC Research Report 124 (1994)

(ftp site: ftp://gatekeeper.dec.com/pub/DEC/SRC/research-reports/ SRC-124.ps.Z).

[Chen02] 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.

[Elias75] P. Elias, “Universal Codeword Sets and Representations of the Integers,” IEEE Trans. on Information Theory, vol. 21, 1975, pp. 192-203

[Floyd76] R.W. Floyd and L. Steinberg, “An adaptive algorithm for spatial greyscale,” Proceedings of Society for Information Display, Vol. 17, No. 2, 1976, pp. 75-77

[Gervautz90] M. Gervautz , W. Purgathofer, A simple method for color quantization: octree quantization, Graphics gems, Academic Press Professional, Inc., San Diego, CA, 1990

[GIF87] http://www.w3.org/Graphics/GIF/spec-gif87.txt

[GIF89a] http://www.w3.org/Graphics/GIF/spec-gif89a.txt

[Golomb66] S.W. Golomb, “Run-Lengrh Encodings,” IEEE Trans. on Information Theory, IT-12, July 1966, pp. 399-401

[Hadenfeldt94] A.C. Hadenfeldt and K. Sayood, “Compression of Color-Mapped Images,” IEEE Trans. on Geoscience and Remote Sensing, Vol. 32, No. 3, May 1994, pp. 534-541

[Heckbert82] P. Heckbert, “Color Image Quantization for Frame Buffer Display,” Computer Graphics, vol. 16, No. 3, 1982, pp. 297-307.

[JBIG93] Information technology — Coded representation of picture and audio information — Progressive bi-level image compression, Int. Std. ISO/IEC 11 544:1993 and ITU-T Recommendation T.82 (1993), Mar. 1993.

[Joy93] G. Joy and Z. Xiang, “Center-cut for color-image quantization,” The Visual Computer: International Journal of Computer Graphics, Vol. 10. No.1, 1993, pp. 62-66

[JPEG00] Information technology — JPEG 2000 image coding system: Core coding system, ISO/IEC 15444-1 and -2, 2000.

[JPEG99] Information technology — Lossless and near-lossless compression of continuous-tone still images: Baseline, ISO/IEC 14495-1:1999.

[Kuroki04] N. Kuroki, T. Yamane, and M. Numa, “Lossless Coding of Color Quantized Images Based on Pseudo Distance,” in Proc. the 47th IEEE International Midwest Symposium on Circuits and Systems, Vol.1, 25-28 Jul 2004, pp. 245-247.

[Lai07] 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

[Lawler85] E.L. Lawler et al., ed., The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, Interscience Series in Discrete Mathematics. New York: Wiley, 1985.

[Linde80] Y. Linde, A. Buzo, and R. M. Gray, “An algorithm for vector quantizer design,” IEEE Trans. Commun., vol. COM-28, Jan. 1980, pp. 84-95.

[Lo03] K. C. Lo, Y. H. Chan and M. P. Yu, “Colour quantization by 3-dimensional frequency diffusion,” Pattern Recognition Letters, Vol. 24, July 2003, pp. 2325-34

[Memon96] 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.

[Memon97] 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.

[Moffat95] A. Moffat, R. Neal, and I. Witten, “Arithmetic coding revisited,” in Proc. Data Compression Conf., J. A. Storer and M. C. Cohn, Eds., 1995, pp. 202–211.

[Natale89] F. G. B. De Natale, G. S. Desoli, D. D. Giusto, and G. Vernazza, “A framework for high-compression coding of color images,” Visual Commun. Image Process. '89 and in Proc. SPIE, vol. 1199, Nov. 1989, pp. 1430-1439.

[Orchard91] M. T. Orchard, and C. A. Bouman, “Color quantization of images,” IEEE Trans. On signal Processing, Vol.39, No.12, Dec 1991, pp.2677-2690.

[Pennebaker88] W. B. Pennebaker, J. L.Mitchell, G. G.Langdon, and R. B.Arps, “An overview of the basic principles of the q-coder adaptive binary arithmetic coder,” IBM J. Res. Develop., vol. 32, Nov. 1988, pp. 717-726.

[Pinho04] 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.

[Pinho04A] A. J. Pinho and A. J. R. Neves, “A survey on palette recording methods for improving the compression of color-indexed images,” IEEE Trans. On Image Processing, Vol.13, No.11, Nov 2004, pp. 1411-1418.

[Pinho05] A. J. Pinho, and A. J. R. Neves, “A Context Adaptation Model for the Compression of Images with a Reduced Number of Colors,” Proceedings, IEEE ICIP’05, Vol.2, Sep. 2005, pp. 738-741.

[PNG95] http://www.libpng.org/pub/png/

[Po94] L. M. Po and W. T. Tan, “Block address predictive colour quantisation image compression,” E.Letter, vol.30, no.2, Jan. 1994, pp.120-121.

[Ratnakar98] V. Ratnakar, “RAPP - lossless image compression with runs of adaptive pixel patterns,” Signals, Systems & Computers, 1998. Conference Record of the Thirty-Second Asilomar Conference on. Vol 2, Nov. 1998, pp. 1251-1255

[Reavy01] M. D. Reavy, and C. G. Boncelet, “An algorithm for compression of bilevel images,”, Image Processing, Vol. 10, Iss. 5, May 2001, pp. 669-676.

[Robinson06] J. A. Robinson, “Adaptive Prediction Trees for Image Compression,” IEEE Trans. on Image Processing, vol. 15, no 8, Aug. 2006, pp. 2131-2145.

[Spira01] 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.

[Taubman00] D. Taubman, “High performance scalable image compression with EBCOT,” IEEE Trans. on Image Processing, Vol. 9, Jul. 2000, pp. 1158-1170.

[Wallace91] G. K. Wallace, "The JPEG still Picture Compression Standard," Communications of the ACM, vol. 34, no. 4, 1991, pp. 30-44.

[Weinberger00] M. Weinberger, G. Seroussi and G. Sapiro, “The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS,” Hewlett-Packard Laboratories Technical Report No. HPL-98-193R1, IEEE Trans. Image Processing, vol. 9, Aug. 2000, pp. 1309-1324.

[Welch84] T. A. Welch, “A technique for high-performance data compression,” Computer, Vol. 17, Jun. 1984, pp. 8-19.

[Wu97] X. Wu and N. D. Memon, “Context-based adaptive lossless image coding,” IEEE Trans. Commun., vol. 45, Apr. 1997, pp. 437-444.

[Zaccarin93] A. Zaccarin and B. Liu, “A novel approach for coding color quantized images,” IEEE Trans. Image Processing, vol. 2, Oct. 1993, pp. 442-453.

[Zeng00] W. Zeng, J. Li and S. Lei, “An efficient color re-indexing scheme for palette-based compression,” in Proc. 7th IEEE Int. Conf. Image Processing Vancouver, BC, Canada, vol. III, Sep. 2000, pp. 476-479.

[Ziv78] J. Ziv and A. Lempel, "Compression of Individual Sequences Via Variable-Rate Coding," IEEE Trans. on Information Theory, vol24, no. 5, Sep. 1978, pp. 530-536.