MERGE–BASED COLOR QUANTIZATION AND CONTEXT TREE MODELING FOR COMPRESSION OF COLOR QUANTIZED IMAGES

Alexey Podlasov and Pasi Fränti

Speech and Image Processing Unit
Department of Computer Science
University of Joensuu
P.O.Box 111, 80101 Joensuu, Finland
{apodla, franti}@cs.joensuu.

ABSTRACT

A two-stage lossless compression method based on a binary tree representation of colors and on context-based arithmetic coding has been recently proposed. We propose two improvements to this method: merge-based color quantization instead of the original splitting strategy, and context tree modeling optimized for each layer separately. The proposed method achieves better compression performance, and a better reproduction quality in the color progression.
Keywords: progressive image coding.

1. INTRODUCTION

Color-quantized and palette images are widely used in web and on low-cost devices, which are typically restricted by a low number of colors displayed simultaneously. Lossy compression is not always applicable since the degradation of the quality cannot be tolerated in many applications. On the other hand, lossless algorithms optimized for compression of photographic images lack the compression efficiency since the correlation between image indexes in palette images can be lost.

Significant progress has been made in recent years in palette- and color-quantized-oriented lossless compression. Typically, two principal approaches have been considered. The first approach uses color map reordering [1], which revives index correlation, followed by compression with existing techniques such as JPEG-LS [2] or CALIC [3]. The second approach concentrates on the development of specific, well-tuned coding techniques. Successful examples of such algorithms are PWC [4] and EIDAC [5].

Chen *et al*. [6] has proposed an algorithm following the second approach. Besides the efficient compression it also provides lossy-to-lossless progressive encoding allowing to stop the transmission when the desired quality level is reached. The algorithm consists of two main stages: color indexing and context-based arithmetic coding. At the indexing stage, the palette of the image is represented by a binary tree. The root of the tree corresponds to all colors appeared in the image. Two children of the root divide the root's color set into two subsets. Every node is divided in the same manner until every single color gets its own leaf node. The tree is constructed minimizing the distortion caused by the color quantization. At the coding stage, the tree is traversed starting from the root. For each node, the encoder sends the weighted average color of each of its two children and a bitmap indicating the location of the pixels having a color change. The bitmap is encoded by a binary context-based arithmetic coder.

In this paper, we follow the principles proposed by Chen *et al.* and consider two improvements of the original algorithm. First, we propose merge-based tree construction [7]. Then, we consider improved encoding of pixel locations based on highly-optimized context tree modeling [8]. We show that the applied techniques achieve improvement both in compression performance and in quality of the color progression.

2. COMPRESSION SCHEME OUTLINE

In this section, we briefly outline Chen's compression algorithm, and its improved variant [9].

2.1. Binary tree color indexing

For the description of tree construction scheme, we use the formulation as presented in [9]. We denote the color palette of an image as a set of RGB triplets $C = \{c_1, \ldots, c_M\}$, where M is the total number of colors. Let $S_0 = \{1, 2, \ldots, M\}$ be the set of indexes identifying the colors in the palette, where index i corresponds to the color c_i. The number of pixels of color c_i is denoted p_i. Each node j of the binary tree represents a certain subset of the color palette. This node is referred using the corresponding set of indexes s_j. Every color set is associated with a *representative color* q_j, which is given by the weighted average color of the node,

$$q_j = \frac{\sum_{i \in s_j} p_i c_i}{\sum_{i \in s_j} p_i}$$

The tree is constructed in a top-down manner starting from the root. Whenever the color set associated with a node contains more than one color, the node is split into two, in such a way that the two new subsets of colors are separated by a hyperplane that is perpendicular to the principal direction of the data and passes through the average color value q_j. The principal direction of the data is given by the eigenvector of largest eigenvalue of the covariance matrix of the weighted colors in s_j, C_j, where

$$C_j = \sum_{i \in s_j} p_i (c_i c_i^t - q_i q_i^t).$$

When the tree is constructed, every color of the image palette is associated with a unique variable-length binary sequence representing the path in the tree. When decompressing the sequence, we know that the first bits represent the most important part of the color information. In this way, the binary tree representation assigns indexes to the image colors so that neighboring indexes are assigned to neighboring colors, converting the correlation of colors into the correlations of indexes. This property is utilized for obtaining higher compression efficiency for color-quantized images.

2.2. Image encoding

The encoding starts from the first node of the tree for which its representative color value q_0 is transmitted. The following procedure is then applied for every node of the tree:

1. Choose the node m for processing.
2. Transmit the index of the node and the representative colors of its two children $q_m{}^l$ and $q_m{}^r$.
3. Encode the location of the pixels, whose colors belong to the sets $s_m{}^l$ and $s_m{}^r$.

The node to process is chosen according to the largest associated eigenvalue of C_j. The values $q_m{}^l$ and $q_m{}^r$ are the representatives of the color sets $s_m{}^l$ and $s_m{}^r$, associated with m's two child nodes (note, that $s_m = s_m{}^l \cup s_m{}^r$).

The pixel locations are encoded by a context-based arithmetic coder. Chen *et al.* proposed to use an 8-pixel fixed-order context template (see Figure 1, left context). Since the location of the pixels whose color belongs to s_m is known to the decoder, we only need to encode, for each of those pixels, which now belongs to $s_m{}^l$ and which to $s_m{}^r$. The information to be encoded is binary and for every pixel which color $q \in s_m$ we encode a bit b_0, defined as

$$b_0 = \begin{cases} 0, & q \in s_m{}^l \\ 1, & q \in s_m{}^r \end{cases}.$$

The context is constructed using a sequence of bits $b_1, \ldots, b_8$, where

$$b_i = \begin{cases} 0, & \left\| q^i - q_m{}^l \right\| \le \left\| q^i - q_m{}^r \right\| \\ 1, & otherwise \end{cases},$$

and q^i denotes the color of the pixel from the reconstructed image corresponding to the position i of the context template.

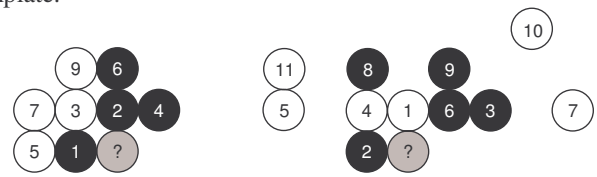

Figure 1. Sample contexts obtained by Chen's algorithm (left) and a sample context constructed by the context-tree approach (right).

In order to achieve better compression performance and avoid context dilution problem, Chen *et al*. considered a variable size context. Instead of using all 8 template locations, only first k are used, where k is defined by the relation

$$k(n) = 9 - \lfloor \log_2 n \rfloor,$$

where n-1 is the number of colors already encoded.

Recently, Pinho *et al.* [9] proposed another context adaptation model for Chen's algorithm. The generalization of context size function $k(n)$ is considered as

$$k(n) = \lceil \alpha(N) - \log_2 n \rceil,$$

where $\alpha(N) = m\log_2 N + b$ is a function of the number of pixels in the image. The values m and b are tuned using test images. This scheme takes into account the size of the image and results in about 4% compression improvement over Chen's algorithm.

3. PROPOSED APPROACH

The original algorithm assumes split-based (top-down) tree construction method. Though this scheme allows combining the tree construction with the encoding into one process, this algorithm lacks quantization quality. Its visual appearance is illustrated on a sample color images (see Figure 2, upper row). We consider another simple and popular quantization heuristics using a bottom-up approach [7]. The algorithm is based on agglomerate (tree-structured) clustering and it provides more precise quantization and better visual appearance (see Figure 2, lower row). Besides that, we apply *free-tree* context modeling [8], instead of a variable-size static-order modeling used by Chen *et al*. The overall scheme is illustrated on Figure 3.

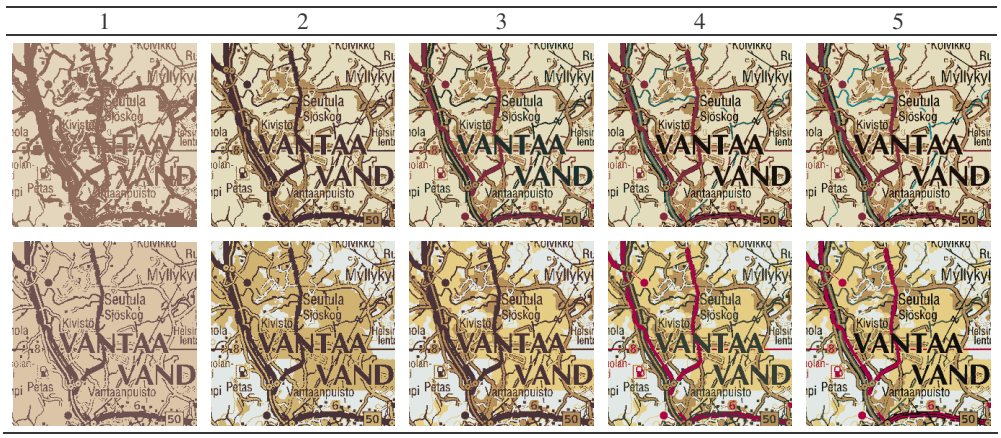

Figure 2. Illustration of the tree-structured quantization. Quantization steps are shown from left to right. Upper sequence corresponds to the split-based tree construction, and the lower sequence represents merge-based tree construction.

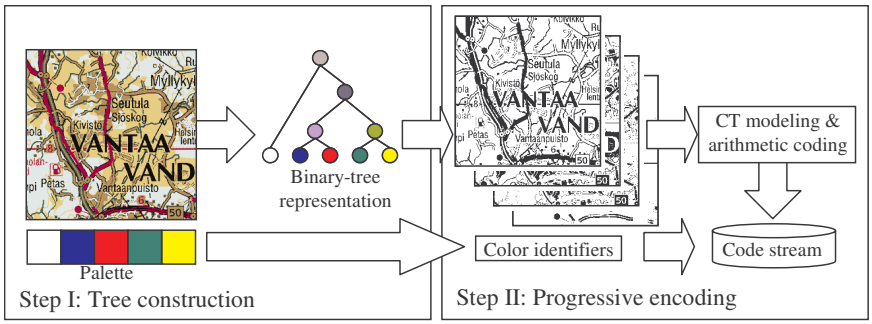

Figure 3. Two steps of the proposed algorithm.

3.1. Merge-based tree construction

The algorithm constructs the tree bottom–up using a sequence of merge operations. First, every color c_i of the image palette $C = \{c_1, \ldots, c_M\}$ is associated with a leaf node in the tree. At each step, two color nodes are merged. The nodes are selected so that their merging causes minimal increase in distortion. The distortion d_{ij} caused by the merging of two color sets s_i and s_j, is calculated by

$$d_{ij} = \frac{n_i n_j}{n_i + n_j} \left\| q_i - q_j \right\|^2,$$

where q_i and q_j are the color representatives of the corresponding sets, and n_i is the number of pixels that belongs to the set s_i, defined as

$$n_i = \sum_{i \in s_i} p_i$$

The value n_j is defined analogously. The parent node is associated with the merged color set. The process continues until only one node is left in the tree. After the tree is constructed, it is applied for color progression and compression in the same top-down manner as in the original algorithm.

3.2. Context tree modeling

Context tree is a highly optimized form of context modeling technique, which has shown to be an efficient tool in compression. It provides more flexible approach for modeling the contexts so that a larger number of neighbor pixels can be taken into account without the context dilution problem. The contexts are represented by a binary tree, in which the context is constructed pixel by pixel and the memory is allocated only for contexts that are really present in the image avoiding extensive memory consumption.

In free-tree variant [8], the location of the template pixels is also optimized. The position of the next context pixel is determined at each step. When a new child is constructed, all possible positions for the next context pixel are analyzed within a predefined search area, and the position resulting in maximal compression gain is chosen. The construction is stopped when increasing the context size does not give any further improvement in the probability estimate, giving the optimal context size. The sample context constructed by the free-tree is illustrated in Figure 1, right context.

Table 1. Compression performance of PNG, Pinho et al. and PNN-Free Tree algorithms on a palette image test set.

Image	PNG	Chen *et al.*	Pinho *et al.*	Proposed
pc	360291	135051	117794	91220
books	20754	8102	8047	7950
music	2800	830	829	814
winaw	33182	13046	12980	11366
party8	10140	3341	3442	2933
netscape	40546	11176	11176	11146
sea_dusk	2712	1497	1595	1128
benjerry	6239	2847	2848	2921
gate	47446	15329	15302	15057
descent	39738	17911	17853	17157
sunset	136966	59806	59489	58335
yahoo	11912	5764	5771	6442
Total	712726	274700	257126	226469

Table 2. Compression performance of PNG, Pinho et al. and PNN-Free Tree algorithms on a natural image test set.

Image	PNG	Chen *et al.*	Pinho *et al.*	Proposed
airplane	381946	121424	121121	122337
anemone	575643	164523	164073	163486
arial	715586	238231	235740	228611
baboon	597818	178131	177768	175876
bike3	1074229	287320	278711	274248
boat	520232	150633	148546	151991
clegg	966435	344181	326807	337366
cweel	299722	122990	108225	102530
fractal	245480	241345	238026	239310
frymire	787184	323584	286396	267087
ghouse	699103	214900	208669	201592
girl	398164	134186	132364	135017
Total	7261542	2521448	2426446	2399451

The drawback of the approach is that the context tree should be transmitted to the decoder increasing the size of required side information. For some images this increase overweights the compression improvement. In order to overcome this drawback we implemented a combined scheme. On every encoding step, we compare the performance of Chen's and free-tree context modeling schemes, choose the best and transmit a flag bit indicating which modeling scheme is used.

4. EXPERIMENTS

We tested our algorithm on the set of images used in [9]. The images are separated into two classes: palette images (of synthetic nature) and natural images. The proposed algorithm is compared to the PNG compressor, the original Chen's algorithm [6], and Pinho's modification [9]. For palette images, our algorithm outperforms the Chen's algorithm by 17.5%, in total, and the Pinho's modification by 11.9%, though negative improvement was obtained on some individual images. The tests on natural image test set show that Chen's algorithm is outperformed by 4.8%; and minor improvement of 1.1% is obtained to Pinho's method. All three techniques clearly outperforms PNG compressor for both test sets.

5. CONCLUSION

We proposed bottom-up tree construction based on sequence of merging and tree-based context modeling for color-quantized image coding. The proposed algorithm improves the compression by about 12% for a set of palette images, and about 1% for a set of natural images.

6. REFERENCES

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

[2] M. Weinberger, G. Seroussi, G. Shapiro, "The LOCO-I lossless image compression algorithm: Principles and Standartization into JPEG-LS", *IEEE Trans. on Image Processing*, 9 (8), pp. 1309-1324, 2000.

[3] X. Wu, N. Memon, "Context-based, adaptive, lossless image coding", *IEEE Trans. on communications*, 45(4), pp. 437-444, 1997.

[4] Ausbeck, P.J., Jr. "The piecewise-constant image model" *Proceedings of the IEEE*, Vol. 88, Issue 11, pp. 1779–1789, Nov 2000.

[5] Y. Yoo, Y. G. Kwon, A. Ortega, "Embedded image-domain compression using context models", *Proc. of IEEE Int. Conf. on Image Processing 1999 (ICIP 99)*, Kobe, Japan, vol. 1, pp. 477–481, 1999.

[6] X. Chen, S. Kwong, and J.-F. Feng, "A new compression scheme for color-quantized images", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 12, no. 10, pp. 904-908, Oct. 2002.

[7] P. Fränti, T. Kaukoranta, D.-F. Shen and K.-S. Chang, "Fast and memory efficient implementation of the exact PNN", *IEEE Trans. Image Process*., vol. 9, no. 4 pp. 773-777, May 2000.

[8] B. Martins and S. Forchhammer, "Bi-level image compression with tree coding," *IEEE Trans. Image Process.*, vol. 7, no. 4, pp. 517-528, Apr. 1998.

[9] A.J. Pinho and A.J.R. Neves. A context adaptation model for the compression of images with a reduced number of colors. *Proc. of the IEEE Int. Conf. on Image Processing, ICIP-2005*, September 2005, Genova, Italy.