Invisible Modification of the Palette Color Image Enhancing Lossless Compression*

Jaroslav Fojtík, Václav Hlaváč
Czech Technical University, Faculty of Electrical Engineering
Center for Machine Perception
121 35 Prague 2, Karlovo náměstí 13, Czech Republic
{fojtik,hlavac}@vision.felk.cvut.cz

Abstract. We address the problem of pseudocolor image compression. Image values represent indices into a look up table (palette). Due to quantization, the neighbouring pixel values (indices) change too much. This deteriorates performance of both lossless and lossy image compression methods.
We suggest a preprocessing phase that (a) analyses statistics of the adjacency relations of index values, (b) performs palette optimization, and (c) permutes indices to palette to achieve more smooth image. The smoother image causes that the lossless image compression methods yield less output data.
The task to optimally permute palette indices is a NP complete combinatorial optimization. Instead of checking all possibilities, we suggest a reasonable initial guess and a fast suboptimal hill climbing optimization.
The proposed permutation of indices should enhance performance of most lossless compression method used after it. To our knowledge, the proposed reordering followed by our own nonlinear compression technique [HF97b, HF97a] yields the best compression. Experiments with various images show that the indices reordering provides data savings from 10% to 50%.

1 Introduction

Let us assume that the palette image (image with a palette) is mapping $color = [R, G, B] = palette(f(x, y))$, where $[R, G, B]$ are individual color components i.e. three intensity images. The output of the function $f(x, y)$ is an index. This is a reason why we call this function *palette index function* in sequel. The *palette* is the look up table with [R,G,B] entries.

This paper discusses a lossless compression of pseudo color images. Some redundancies must be found in the image for doing so. The lossless compression methods for gray level images (including our method [HF97b, HF97a], which is based on the original Schlesinger's idea [Sch89]) are usually based on the assumption about the continuity of the image function. This assumption cannot be used for pseudocolor images because a typical palette functions break this assumption. The basic idea of our approach is to re-establish the smoothness of the palette index function $f(x, y)$ so that it can be compressed in the same way as image function of gray level images. Both newly created palette index function $f'(x, y)$ and the look up table *palette'* are modified in the way that the resulting colors, i.e. $[R, G, B]$ values, remain the same for all pairs of corresponding pixels from both images: $palette(f(x, y)) = palette'(f'(x, y))$.

* This research was supported by the Czech Ministry of Education grant VS96049, the Grant Agency of the Czech Republic 102/97/0480, 102/97/0855.

The palette is usually created from the true color RGB image (camera, color scan) or by an interactive painting program. The algorithm that quantizes the original true color image causes discontinuities in its output palette index function $f(x, y)$. This effect can significantly decrease the compression ratio.

2 Related works

There are many available compression algorithms for gray level images. Pseudo color images are usually compressed by the same algorithms as gray level ones. The compression ratio depends on a first order entropy (we will call it smoothness in the sequel) in these cases.

The reordering of palette is useful mainly in two cases mentioned in [HS94] (a) Sort indices to increase available compression ratio. (b) Sort indices in order to enhance human perception. The requirements for these approaches are not fully in contradiction. In this paper we will discuss the first case only.

The most related work to our contribution is [MV96]. The linear predictor is used as a lossless compression technique. Indices reordering is formulated as an optimization task. Three heuristic solutions are proposed to it. Two of them are very expensive due to used simulated annealing and third, based on a greedy algorithm, produces worse results.

The paper [AL93] describes lossy compression of palette images. Proposed method starts with construction of the optimal shortest route among colors in the RGB (or LUV) color space. Colors close each other are grouped into clusters. Each cluster corresponds to just one new color.

3 Creating neighbourhood relations table

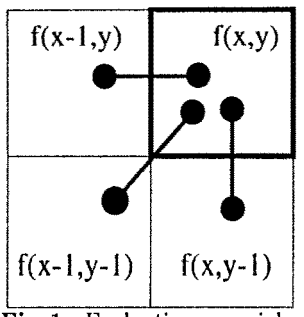

Fig. 1. Evaluating a neighbourhood relations.

Let u, v be two values (i.e. indices into *palette*) of the palette index function $f(x, y)$ in two different but neighboring pixels. The used neighborhood of the current pixel (x, y) is shown on Fig. 1(a). The symmetric relation $clasp(u, v) = clasp(v, u)$ tells whether two indices u, v are adjacent. Let us call *clasp* the *index adjacency relation*. We use the intuitive name *clasp* as the notion of a thing that fastens two regions with intensities u, v (index values) together. Simple statistics of the local index adjacency relation will serve as a measure of the smoothness of the palette index function $f(x, y)$.

We are interested in the number of occurrences of indices adjacency relations *rel* in the whole image $f(x, y)$. This information is stored in the *index adjacency table* $\mathbf{S}(i, j)$. The table $\mathbf{S}(i, j)$ has the same number of rows and columns that is equal to range (number of values) of the palette index function $f(x, y)$. The size of $\mathbf{S}(i, j)$ is 256×256 for typical palette

images. The table entries tell us how many times indices u and v are adjacent in the image. The statistical information stored in $\mathbf{S}(i,j)$ resembles more general co-occurrence matrix used often in the texture analysis [HS92].

The indices adjacency table $\mathbf{S}$ is symmetric and only lower (or upper) triangular part of it needs to be stored. The values on the main diagonal (identity relation) are not needed by the proposed algorithm.

The indices adjacency table $\mathbf{S}$ is created by one pass traversal of the image according to the neighborhood mask. The statistics stored in the intensity adjacency table $\mathbf{S}$ will be used to find optimal palette index function $f'(x,y)$.

4 Formulation of the optimization task

Let us start with an informal description of the optimization task that should be performed. A pseudocolor image depicting a park with a lake was chosen as an example, see first column in Fig. 3, where the pseudocolor image is shown in intensity values only. The difficulty with the lossless compression algorithms for pseudocolor images is that there are too many discontinuities in corresponding indices in the palette index function $f(x,y)$. Let us show them in individual bit planes of $f(x,y)$ for the park image. Three bit planes #8, #5 and #1 from all eight bit planes of the palette index function $f(x,y)$ are displayed in the top row of the Fig. 3. Even the most significant bit plane #8 changes often too.

Our aim is to reorder indices of the palette index function $f(x,y)$ in such a manner that resulting binary images in individual bit planes will consist of the smallest number of large regions. Larger and smoother regions ease the further image compression. Number of possible indices rearrangements is huge - $n!$, where n is a number of indices.

The top bit plane is most significant and thus it is processed first. Our algorithm is designed in such a way that lower bit planes could be modified similarly but the already modified bit planes above it should remain intact.

Processing of bit planes is a combinatorial optimization problem. To simplify it, we assume that all K indices are divided into two groups G_1 and G_2 consisting of half of the entries, i.e. $k = \frac{K}{2}$ each. The number of possible combinations is still tremendous, i.e. 10^{76} in typical case of 256 indices and the top bit plane.

5 Initial guess

Let us assume that there is some initial division of indices into two groups G_1 and G_2. A single candidate index in both groups is found that fits the least to the current group than other indices. These two found indices are swapped between groups. The process is repeated until the the global criterion describing division into two groups is minimized. Theoretically, this minimization would perform perfectly if we had a space without a lot of local extrema. Unfortunately, it seems that the number of local extrema is very high for real images and our optimization space. The algorithm may get stuck in a local minimum. On the other hand, experiments have shown that even this simple minimization yields much smoother palette index function $f'(x,y)$. Moreover we use additional heuristic rules to avoid local minima.

The good initial estimate of the division of indices k into groups G_1 and G_2 helps us to be quite near to the global minimum. The initial division is based on the strategy that light colors create one group and dark colors the second one. Therefore such initial division is very close to the situation when indices are sorted according to their intensity.

The adjacency relations between values of palette index function in the local neighborhood (recall Fig. 1) are used to define the optimization criterion. The index adjacency relation was denoted $clasp(u, v)$ in Section 3. Indices are split into two disjoint groups G_1 and G_2 with the same cardinality.

The number of relations $clasp(u, v)$, $u \in G_1$, $v \in G_2$, $u \neq v$ informs how many relations are between (in our case two) distinct groups. The number of existing relations $clasp(u, v)$, $u \in G_1$, $v \in G_1$, $u \neq v$ tells how many relations are within group the G_1, similarly for G_2. All three numbers can build up a quality measure of the grouping of indices into G_1 and G_2. They help to find best candidates for swapping between G_1 and G_2. The good news is that all needed statistics can be efficiently extracted from the indices adjacency table $\mathbf{S}$.

5.1 Quality of the index k.

Each index k has associated a quality w_k. The quality w_k can be calculated as a number of adjacent indices w_k^+ in its own group minus the number of adjacent indices w_k^- in the other group. We call numbers w_k^+ and w_k^- as components of w_k.
If the index k belongs into the group G_1:

$$w_k = w_k^+ - w_k^- = \sum_{i \in G_1;\, i \neq k} \mathbf{S}(i,k) - \sum_{i \notin G_1;\, i \neq k} \mathbf{S}(i,k)\,; \quad k \in G_1\,. \tag{1}$$

It can be seen that w_k^+ is a number of clasps (satisfied index adjacency relations) within own group G_1 for index k, w_k^- is number of clasps between own group G_1 and the alien group G_2. Their difference w_k is a relative measure of the index k quality. The index with minimal quality within group G_1 will be selected as a candidate for swapping. Similar equation holds, if the the index k belongs to G_2.

Qualities of all indices constitute a indices quality vector $\mathbf{w} = (w_1, w_2, ...w_k)$.

5.2 Global optimization criterion

We have seen how quality w_k enables to find candidates for swapping. A global optimization criterion is needed to know when to stop swapping iterations. We use slightly modified hill climbing optimization.

The pre-calculated indices adjacency table $\mathbf{S}(i, j)$ contains needed information, see Fig. 2(a), where four distinct parts W_1, W_{21}, W_{12}, and W_2 are illustrated. Note that $W_{12} = W_{21}$ due to the symmetry of the table $\mathbf{S}$. All entries of the table $\mathbf{S}(i, j)$ with same labeling are summed together:

$$W_1 = \sum_{i \in G_1} w_i^+\,; \quad W_{12} = \sum_{i \in G_1} w_i^-\,; \quad W_2 = \sum_{i \in G_2} w_i^+\,; \quad W_{21} = \sum_{i \in G_2} w_i^-\,. \tag{2}$$

If any labeling occurs more than once in the table **S** then its value is a sum of all areas with this label into the matrix. This situation occurs in the case of bit planes below the top one.

When we know a quality of each isolated index, we can compute a quality of the whole group of indices as a sum of qualities of their members. The quality of whole group of indices, i.e. $Q(G_1)$ and $Q(G_2)$ is defined as:

$$Q(G_1) = W_1 - W_{12}\,, \quad Q(G_2) = W_2 - W_{21}\,. \tag{3}$$

The global optimization criterion, i.e. the overall quality Q of splitting indices into two groups G_1 and G_2 is defined as

$$Q = Q(G_1) + Q(G_2) = W_1 - W_{21} + W_2 - W_{12}\,. \tag{4}$$

The global optimization criterion Q is the the sum of qualities of all indices in both groups. It is evaluation of the whole area of the index adjacency table **S** with same labeling.

5.3 Algorithm for swapping indices

The worst index in each group (with the lowest value w_k) is found. The worst indices are swapped between groups. Let us suppose that we have two indices, each one from different group $a \in G_1$ and $b \in G_2$. We want to swap them.
New values of qualities of swapped indices are:

$$w'_a = w'^{+}_a - w'^{-}_a = -w_b - 2\mathbf{S}(a,b)\,; \quad w'_b = w'^{+}_b - w'^{-}_b = -w_a - 2\mathbf{S}(a,b)\,. \tag{5}$$

All actions which must be done after swapping two indices are called transactions. The transaction includes the recomputation of the indices quality vector **w** and global criterion Q.

If the transaction decreases the global criterion then ($Q' < Q \quad \rightarrow \quad \Delta = Q' - Q < 0$). The Δ value means how many clasps will appear when $\Delta > 0$ or disappear when $\Delta < 0$ in the image.
The value Δ for swapping indices a and b is:

$$\frac{(Q' - Q)}{4} = \Delta = w_a + w_b + 2\,\mathbf{S}(a,b)\,. \tag{6}$$

When indices are swapped, all items in the indices quality vector **w** must be updated too.

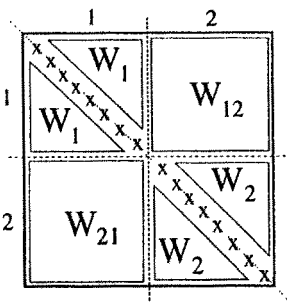

Fig. 2. Created domains in **S**

5.4 Lower bit planes

The algorithm proceeds from the highest to less significant bit planes. The task is to split both groups G_1 and G_2 into four groups in the bit plane below the top one. The approach is similar to that described above. The computation is slightly more complicated.

We omit the full description of this due to the lack of space. The more detailed description can be found in [FH98].

6 Experimental results

The proposed method was tested on six pseudocolor images obtained from the web. The results are summarized in Table 1. First column gives image names. Second column shows image sizes of input uncompressed images, i.e. rows × columns × number of bits per pixel. The third column depicts number of bytes of uncompressed images including palette stored in BMP format. The fourth column with header FH gives the number of bytes after our own FH compression [HF97b, HF97a] was applied. The fifth column gives the size of images if palette was first reordered according to intensity [Fry93]. The sixth column shows the influence of indices and palette modification that is the contribution of the palette rearranging method suggested in this paper. The length of the files is in bytes including palette after palette modification and FH compression. Gain of our method according to the equation (7) is written in the rightmost column.

Image name	Image size	Uncompressed	FH	Y + FH	Opt. + FH	gain
Descent	320x200x8	65078	24334	27115	21132	13.6%
Garfield	640x480x4	153718	3323	-	2955	13.4%
Lena, color	512x512x8	263222	235579	185275	154401	52.2%
Lynne	320x200x8	65078	59016	43506	38769	52.2%
Peppers	512x512x8	263222	206349	165315	129513	59.3%
Tartan	256x256x4	32886	1596	-	1478	11.0%

Table 1. Compression performance on palette colour images.

The *gain* in compression for palette color images is measured as a ratio:

$$Gain = \frac{\text{total\_output\_bytes (no\_optimization)} - \text{total\_output\_bytes (optimal)}}{\text{total\_output\_bytes (optimal)}} \cdot 100\% \quad (7)$$

Let us visualize results of the proposed palette modification algorithm in pictorial form. The image called park with a lake was chosen as an example for further processing. Three different palette index functions are shown in the first column of Figure 3. These images were obtained by truncating palette and adding monotonic gray palette. The gray level image that looks most similarly to the original pseudocolor image is displayed in Figure 3(b), first column. Colors were converted to intensities $Y = R+G+B$ and the palette index function was sorted according to intensity Y. Figure 3(a), first column, illustrates the original palette index function. Notice very many changes in it. The palette index function that is the outcome of the proposed optimization algorithm is shown in Figure 3(c), first column.

Nine binary images in Figure 3, columns $2 \div 4$ provide more intuitive insight into results. Bit planes #8, #5, #1 corresponding to intensity images in the first column in Figure 3 are shown in each row. The top row (a) shows three bit planes of the palette index function. The middle row (b) visualizes performance of the much simpler re-arrangement of the palette according to intensity Y [Fry93]. The bottom row (c) visually demonstrates results of the palette optimization described in this paper. Notice that even bit plane #5 is relatively smooth if our modification was used. Do not forget that these three images produce the same color image.

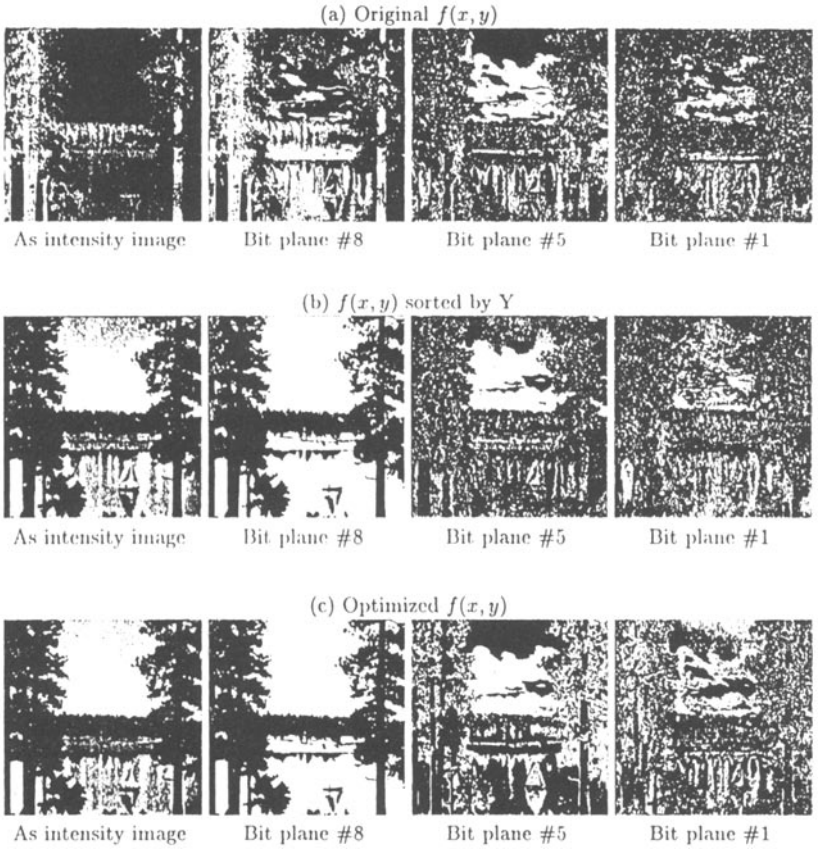

Fig. 3. Three modifications of palette index function $f(x,y)$ compared.

The proposed palette index function optimization algorithm is reasonably fast. It runs slightly around one second for an 512x512x8bit pseudocolor image on Intel Pentium with 200 MHz clock frequency.

7 Conclusion

The method that optimizes palette index function of the pseudocolor image was described. The proposed method can be used prior any lossless compression technique is applied. We believe that the suggested optimization of the palette index function can be easily incorporated into standard lossless pseudocolor images compression methods. The reverse step is not needed during decompression phase. Thus no additional software is needed in image viewers.

The method can be practically used for 4, 16 and 256 color palette images of any size. If there are many more colors (e.g. 16 bits per pixel) the indices adjacency table **S** would be huge to be stored in the memory. Theoretically, the proposed method should work even in this case.

Tests on real images demonstrate compression improvement between 10% - 50%. The actual improvement depends how the original color image was quantized when pseudocolor image was created and of course on the used compression method.

If the reader wants to test the method then she/he is advised to consult www page *http://cmp.felk.cvut.cz/~fojtik/* for our implementation.

The planned future work is to: (a) study more carefully the optimization and learn if there is not a computationally plausible way how to overcome local minima; (b) to formalize a good initial guess that is used prior the iterative optimization starts.

References

[AL93] Zaccarin André and Bede Liu. A novel approach for coding color quantized images. *IEEE Transactions on Image Processing*, 2(4):442–453, October 1993.

[FH98] Jaroslav Fojtík and Václav Hlaváč. Invisible modification of palette color image for increasing compression ratio of lossless compression methods. Technical Report K335/98/159, Czech Technical University, Faculty of Electrical Engineering, Karlovo Náměstí 13, Prague 2, May 1998.

[Fry93] Michael Frydrych. Image compression. Master's thesis, Charles University, Faculty of Mathematics Physics, Prague, Czech Republic, 1993.

[HF97a] V. Hlaváč and J. Fojtík. Adaptive non-linear predictor for lossless image compression. In G. Sommer, K. Daniilidis, and J. Pauli, editors, *Proceedings of the conference Computer Analysis of Images and Patterns'97, Kiel, Germany*, pages 279–288. Springer-Verlag, LNCS 1296, September 1997.

[HF97b] V. Hlaváč and J. Fojtík. Predictor based on frequency analysis of the local configurations used for lossless image compression. In *Proceedings of the 1st IAPR TC1 workshop on Statistical Techniques in Pattern Recognition, Prague, Czech Republic, June 9-11, 1997*, pages 73–78, Prague, Czech Republic, June 1997. Institute of Information Theory and Automation, Czech Academy of Sciences.

[HS94] Andrew C. Hadenfeldt and Khaid Sayood. Compression of color-mapped images. *IEEE Transactions on Geoscience and Remote Sensing*, 32(3):534–541, May 1994.

[HS92] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision, Volume I.* Addison Wesley, Reading, Ma., 1992.

[MV96] Nasir D. Memon and Ayalur Venkateswaran. On ordering color maps for lossless predictive coding. *IEEE Transactions on Image Processing*, 5(11):1522–1527, November 1996.

[Sch89] M.I. Schlesinger. *Matematicckie srcdstva obrabotki izobrazenij, in Russian, (Mathematic tools for image processing).* Naukova Dumka, Kiev, Ukraine, 1989.