Compression of Color-mapped Images*

A.C. Hadenfeldt
University of Nebraska Medical Center
Omaha, Nebraska 68198

and

K. Sayood
Department of Electrical Engineering
and
Center for Communication and Information Science
University of Nebraska-Lincoln
Lincoln, NE 68588
Telephone: (402) 472–6688
FAX: (402) 472–4732
email: ksayood@eecomm.unl.edu

COMSOC Technical Committee: Signal Processing and Communication Electronics
Hot topic session number: HT28

Abstract

In a standard image coding scenario, pixel-to-pixel correlation nearly always exists in the data, especially if the image is a natural scene. This correlation is what allows predictive coding schemes (e.g., DPCM) to perform efficient compression. In a color-mapped image, the values stored in the pixel array are no longer directly related to the pixel intensity. Two color indices which are numerically adjacent (close) may point to two very different colors. The correlation still exists, but only via the colormap. This fact can be exploited by sorting the color map to reintroduce the structure. In this paper we study the sorting of colormaps and show how the resulting structure can be used in both lossless and lossy compression of images.

* This work was supported by the NASA Goddard Space Flight Center (NAG 5–1612) and the NASA Lewis Research Center (NAG 3–806).

1 Introduction

Many lower-cost image display systems use color-mapped (or pseudo-color) displays. While there has been considerable attention devoted to the compression of monochrome and full-color images, the compression of color-mapped images has not received similar attention.

The human eye can distinguish hundreds of thousands of different colors in a color space, depending on viewing conditions [1]. A full-color (also called true-color) frame buffer provides a means of displaying this wide range. Such a system is illustrated in Figure 1.

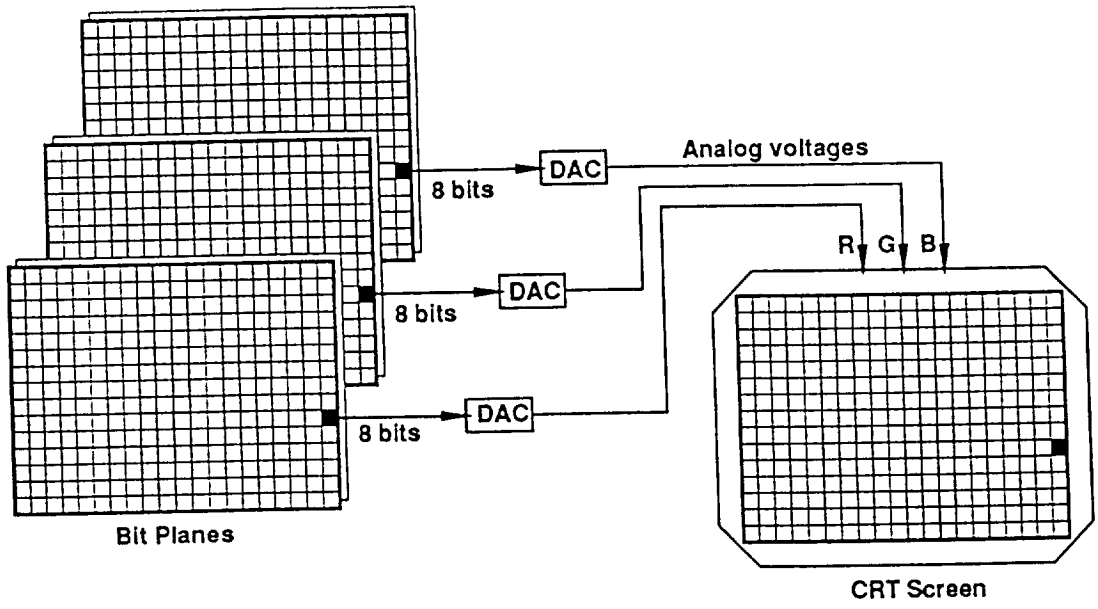

Figure 1 Full-Color Frame Buffer

Many applications of digital images benefit from, or require, color capabilities to be effective. If a full-color display is used, an application may become too costly to implement practically. Also, the images involved require large amounts of storage space, whether in display memory or on a mass-storage device. A less expensive solution is needed.

These applications naturally lead to the pseudo-color or color-mapped frame buffer, shown in Figure 2. This type of display is typical of those found on personal computers and workstations. A smaller amount of image memory is required, one-third that of the full-color system example. The values stored in memory are used as indices into a 24-bit table, the colormap.

Each entry in the colormap consists of 8-bit values for the red, green, and blue portions of the pixel. These three values are then passed through DACs to the red, green, and blue electron guns of the CRT, as with full-color system. The color-mapped system allows the display of a small number of colors at a time, 2^8 for the system shown in the figure, which can be selected from a larger set of colors (2^{24} for this example). By careful selection of the colors in the colormap, a large variety of images can be displayed, often with quality approaching that of a full-color display system.

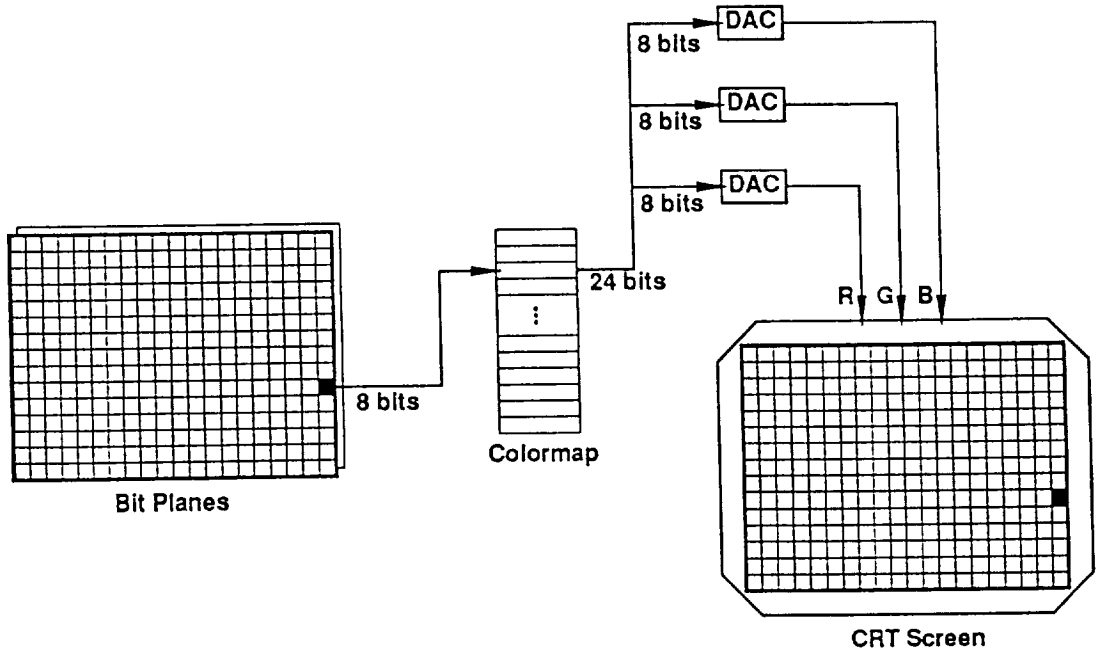

Figure 2 Color-Mapped Frame Buffer

The use of the colormap, however, disguises the spatial structure in the image. An indication of this can be obtained by calculating the zero-th and first order entropies of the image. These quantities were computed using the index arrays for the four test images shown in Figure 3, and are listed in Table 1.

Table 1 Entropies of the Source Images

Image	H_0	H_1
Lena	7.617	7.413
Park	7.470	7.797
Omaha	7.242	7.165
Lincoln	5.916	6.674

The large values of H_1 in the table verify that the spatial correlation in the image pixel values has been reduced by the color-mapping process. The values of H_0 are also relatively large, a direct result of selecting an 8-bit colormap. The data compression due to the color quantization process implies that the color index values stored in the image are more critical than similar values in, for example, an achromatic image. To further verify this, an experiment was conducted in which errors were introduced in the least significant bit of a color-mapped image, similar to what might be encountered if the color index values were quantized. The resultant images were of poor subjective quality at best, and often completely unrecognizable. Since quantization is a part of many popular source coding schemes, the available choices for compression schemes become limited.

2 Colormap Sorting

In the previous section, several problems unique to color-mapped images were discussed. The root of these problems is that the colormap indices stored in the image have little relationship with each other, which complicates coding for progressive transmission. In this section, methods of restoring this relationship are discussed.

Colormap sorting is a combinatorial optimization problem. Treating the K colormap entries as vectors, the problem is defined as follows. Given a set of vectors $\{a_1, a_2, ..., a_K\}$ in a three-dimensional vector space and a distance measure $d(i,j)$ defined between any two vectors a_i and a_j, find an ordering function $L(k)$ which minimizes the total distance D:

$$D = \sum_{k=1}^{K-1} d(L(k),\ L(k+1)) \tag{1}$$

The ordering function L is constrained to be a permutation of the sequence of integers $\{1, ..., K\}$. Another possibility results when the list of colormap entries is considered as a ring structure. That is, the colormap entry specified by $L(K)$ is now considered to be adjacent to the entry specified by $L(1)$. In this case, an additional term of $d(L(K), L(1))$ is added to the distance formula D.

The sorting problem is similar to the well-known travelling salesman problem, and is identical if the colormap is considered as a ring structure. As such, the problem is known to be NP-complete [3], and the number of possible orderings to consider is $1/2[(K-1)!]$[4]. Algorithms exist which can solve the problem exactly [4][5]; however, these algorithms are computationally feasible only for K no greater than about 20. Efficient algorithms for locating a local minimum exist [4] for $K \leq 145$. For large colormaps such as $K = 256$, another approach is necessary. Two techniques were tested. The first is a "greedy" technique, discussed in Section 2.1. The second is an algorithm which has performed well in practice, known as *simulated annealing*. Simulated annealing was chosen as the sorting method for the colormaps in this work, and is described in more detail in Section 2.2.

The distance metric d was chosen to be (unweighted) Euclidean distance, and different color spaces were investigated. Three color spaces were selected: the NTSC RGB space, the CIE L*a*b* space, and the CIE L*u*v* space. The NTSC RGB space was chosen since it corresponds to the color primaries of the original images. Color spaces which can be linearly transformed to the NTSC RGB space were not considered, since the use of an unweighted Euclidean distance measure would give similar results for such a color space. The two CIE color spaces were selected since they provide a means to measure perceptual color differences.

2.1 Sorting Using Simulated Annealing

Simulated annealing [3][6] is a stochastic technique for combinatorial minimization. The basis for the technique comes from thermodynamics and observations concerning the properties of materials as they are cooled. The technique described in this section is based on the implementation in [6].

To illustrate this concept, consider an iron block. At high temperatures, the iron molecules move freely with respect to each other. If the block is *quenched* (cooled very quickly), the molecules will be locked together in a high-energy state. On the other hand, if the block is *annealed* (cooled very slowly), the molecules will tend to redistribute themselves as they lose energy, with the result being a lower energy lattice which is much stronger. The distribution of molecular energies is characterized by the Boltzmann distribution:

$$P(E) \sim e^{-E/kT} \tag{2}$$

where E is the energy state, T is the temperature, and k is Boltzmann's constant. The significance of this distribution is that even at low temperatures, there is some probability that a molecule will have a high energy. In a combinatorial optimization situation, the Boltzmann distribution can be used to temporarily allow increases in the cost function, while still generally striving to achieve a minimum.

Solving the colormap sorting problem involves selecting each color only once while minimizing the sum of the distances between the colors. To find a solution using simulated annealing, an initial path through the nodes (colors) is chosen, and its cost computed. The algorithm then proceeds as follows:

1. Select an initial temperature T and a cooling factor α.
2. Choose a temporary new path by perturbing the current path (see below), and compute the change in path cost, $\Delta E = E_{\text{new}} - E_{\text{old}}$. If $\Delta E \leq 0$, accept the new path.
3. If $\Delta E > 0$, randomly decide whether or not to accept the path. Generate a random number r from a uniform distribution in the range [0, 1), and accept the new path if $r < \exp(-\Delta E/T)$.
4. Continue to perturb the path at the current temperature for I iterations. Then, "cool" the system by the cooling factor: $T_{\text{new}} = \alpha T_{\text{old}}$. Continue iterating using the new temperature.
5. Terminate the algorithm when no path changes are accepted at a particular temperature.

The decision-making process is known as the *Metropolis algorithm.* Note that the decision process will allow some changes to the path which increase its cost. This makes it possible for the simulated annealing method to avoid easily being trapped in a local minimum of the cost function. Hence, the algorithm is less sensitive to the initial path choice.

For the images of this work, initial values of T ranged from 80 to 500, depending on the color space used. The cooling factor α was usually chosen as 0.9. The simulated annealing algorithm seemed to be most sensitive to the choice of this value, as values outside the range [0.85, 0.95] caused the cooling to occur too slowly or too quickly. The number of iterations per temperature I was chosen as 100 times the number of nodes (colors), or 25,600. However, to improve the execution speed of the algorithm an improvement suggested by [6] was added, which causes the algorithm to proceed to the next temperature if (10)(number of nodes) = 2560 successful path changes are made at a given temperature.

Also, a method for perturbing the path must be selected. In this work, the perturbations were made using the suggestions of Lin [4][6]. At each iteration, one of two possible changes to the path are made, chosen at random. The first is a *path transport*, which removes a segment of the current path and reinserts it at another point in the path. The location of the segment, its length, and the new insertion point are chosen at random. The second perturbation method, called *path reversal*, removes a segment of the current path and reinserts it at the same point in the path, but with the nodes in reverse order. The location and length of the segment are again randomly chosen.

The algorithm outlined in the previous paragraphs formulates colormap sorting as a travelling salesman problem. This type of problem usually assumes a complete tour will be made (i.e., the salesman desires to return to the original city). Hence, the colormap is assumed to have a ring-like structure. However, the simulated annealing technique can also be used if this is not the case, allowing the colormap to be considered as a linear list structure. Experiments using both structures were conducted.

3 Colormap Sorting and Lossless Compression

The results of sorting the colormaps of the test images using simulated annealing are shown in the following tables. Table 2 shows results for sorting the colormap as a circular ring structure, while Table 3 shows the results of sorting the colormap as a linear structure. Given in the tables are values for the resulting first-order entropy and the final path cost (the distance measure D).

Table 2 Resultant Images With Circularly Sorted Colormaps

Image Name	RGB Space		L*a*b Space		L*u*v* Space	
	Cost	H_1	Cost	H_1	Cost	H_1
Lena	13.88	5.641	857.80	5.627	208.49	5.480
Park	19.32	6.325	1609.46	6.330	310.41	6.218
Omaha	11.04	6.209	1081.82	6.303	363.21	6.178
Lincoln	10.62	5.513	1193.88	5.831	224.06	5.478

Table 3 Resultant Images With Linearly Sorted Colormaps

Image Name	RGB Space		L*a*b Space		L*u*v* Space	
	Cost	H_1	Cost	H_1	Cost	H_1
Lena	11.68	5.575	847.29	5.933	200.31	5.512
Park	15.66	6.260	1509.25	6.775	292.29	6.546
Omaha	10.81	6.532	1004.69	6.554	283.66	6.199
Lincoln	10.61	5.774	1177.80	6.120	204.64	5.735

Note that the zero-order entropy H_0 is not changed by the sorting process, since permuting the colormap entries does not change the frequency of occurrence of a particular color. The lower first-order entropies of the resultant images indicate that some of the spatial correlation between color indices has been restored in each case. The sorting results for the NTSC RGB space show that sorting in this space yields good results, if entropy reduction (the first goal stated above) is the goal. However, the L*u*v* space sorting gives better results, with the added advantage that the perceptual differences between colormap entries has been considered. Hence, the resultant images from this sort should also be able to accept quantization errors while maintaining good subjective quality, the second goal stated previously. We examine this further in the next section. In terms of lossless compression, the sorting has resulted in a drop of 2 bits per pixel for the

Lena image and 1 to 1.5 bits per pixel for the other images. For a 512x512 image this translates to a savings of between 32,768 to 65,536 bytes per image. For a large database of images this could be a considerable saving.

4 Colormap Sorting and Lossy Compression

The sorting of the colormap restores some perceptual structure to the colormap indices in the sense that indices close in numerical value are also close in some perceptual sense. Therefore it should be possible to introduce errors into the indices without destroying the image. To verify this hypothesis, we dropped the three least significant bits of the L*u*v*-sorted Park images. Good subjective results were obtained using quantization levels down to as low as 5 bits/pixel from the 8-bit original. Figure 4 shows the colormap for the Park image, before and after sorting. The sorted colormap shown was sorted as a linear list in L*u*v* space. Figure 5 shows the result of quantizing the Park image to 5 bits/pixel, before and after the colormap has been sorted. A caveat is in order here. While the distance between the eight-bit indices have more perceptual meaning, the sorted colormap image should not be assumed to have the same properties as an eight-bit monochrome image. In some cases, if the distance between the original and reconstructed (compressed and decompressed) indices is large enough, there might be a drastic change in color between those pixels in the original and reconstructed image. In the monochrome case large distances would correspond to changes in shading which might be overlooked by the viewer.

To see how well the sorted color-mapped images lend themselves to lossy compression we compress them using particular implementations of two popular lossy compression techniques, the Discrete Cosine Transform (DCT) and Differential Pulse Code Modulation (DPCM).

4.1 DCT Coding of Color-mapped Images

In Figure 6 we coded the Lena image with the unsorted colormap at two bits per pixel using the unsorted color map. As can be seen from the figure, the original image is totally lost and all that remains is seemingly random colors. It should be noted that for eight-bit monochrome images,

DCT coding at two bits per pixel generally provides a reconstruction which is indistinguishable from the original.

In Figure 7 we show the same image, this time with the sorted color map, coded at two bits per pixel with the fixed bit allocation. [7] The images in Figure 8 were coded at two and one bit per pixel using the JPEG [8] algorithm.[1] Note that while the image coded using the fixed bit allocation shown in Figure 7 is far superior to the image in Figure 6 there are still quite a few annoying artifacts. This is because of the nonadaptive nature of the algorithm which, while it minimizes the *average* error, may permit the introduction of large errors in individual blocks. As the color-mapped images are particularly sensitive to large errors, this could account for the low quality reproduction. The JPEG algorithm adapts its bit allocation on a block-by-block basis. Therefore, the image in Figure 8(b) which is coded at half the rate of the image in Figure 7 still provides superior quality.

4.2 DPCM Coding of Color-mapped Images

Standard DPCM coding of color-mapped images is problematic because in the busy regions of images, especially edges, the prediction error is generally large, leading to large overload noise values. In monochrome images these noise values result in a blurred look around edges, which may be acceptable for certain application. However, in color-mapped images these noise values will result in splotches of different colors. The Edge Preserving DPCM (EPDPCM) system avoids this problem by the use of a recursively indexed quantizer [9,10], in which the magnitude of the quantization error is always bounded by $\frac{\Delta}{2}$. This attribute makes it ideal for application to the coding of color-mapped images. Another advantage of the EPDPCM system is that, as the quantizer output alphabet can be kept small without incurring overload error, the output is amenable to entropy coding.

Results using the EPDPCM system are shown in Figure 10. The image in Figure 10(a) was coded at a rate of 2 bits per pixel, while the image in Figure 10(b) was coded with 1.35 bpp.

---

1 The JPEG coded images were coded using software from the independent JPEG foundation.

The advantage of DPCM systems over transform coding systems is their low complexity and higher speed. However, the reconstruction quality obtained using transform coding systems is generally significantly higher than that of DPCM systems at a given rate. Comparing Figure 10(a) and 8(a), this is obviously not the case for the sorted colormapped images. In fact, the quality of the two-bit EPDPCM coded image is actually somewhat higher than the two-bit DCT coded image. Thus using the EPDPCM system provides advantages both in terms of complexity and speed, and reconstruction quality.

5 Conclusion

In this paper we have shown that use of sorted colormaps makes color-mapped images amenable to both lossless and lossy compression. For lossy compression conventional wisdom dictates the use of DCT coding for most types of images. However, for color-mapped images DPCM coding might be more advantageous.

6 References

[1] Foley, J.D., van Dam, A., S.K. Feiner, and J.F. Hughes, *Computer Graphics: Principles and Practice (Second Edition)*, Reading, MA: Addison-Wesley, 1990.

[2] *Graphics Interchange Format (GIF) Specification*, CompuServe, Inc., Columbus, OH, June 1987.

[3] Aarts, E., and J. Korst, *Simulated Annealing and Boltzmann Machines*, New York: John Wiley and Sons, 1989.

[4] Lin, S., "Computer Solutions of the Traveling Salesman Problem," *Bell System Technical Journal*, pp. 2245–2269, December 1965.

[5] Bellman, R.E., and S.E. Dreyfus, *Applied Dynamic Programming*, Princeton, NJ: Princeton University Press, 1962.

[6] Press, W.H., B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes in C*, New York: Cambridge University Press, 1988.

[7] Jayant, N.S. and P. Noll, *Digital Coding of Waveforms*, Prentice-Hall, 1984.

[8] Wallace, G.K., "The JPEG still picture compression standard," *Communications of the ACM*, 34(4):31–44, April 1991.

[9] Rost, M.C. and K. Sayood, "An Edge Preserving Differential Image Coding Scheme," *IEEE Transactions on Image Processing*, 1:250–256, April 1992.

[10] Sayood, K. and S. Na. "Recursively Indexed Quantization of Memoryless Sources," *IEEE Transactions on Information Theory*, IT-38, November 1992.

List of Figures

Figure 3. Test Images

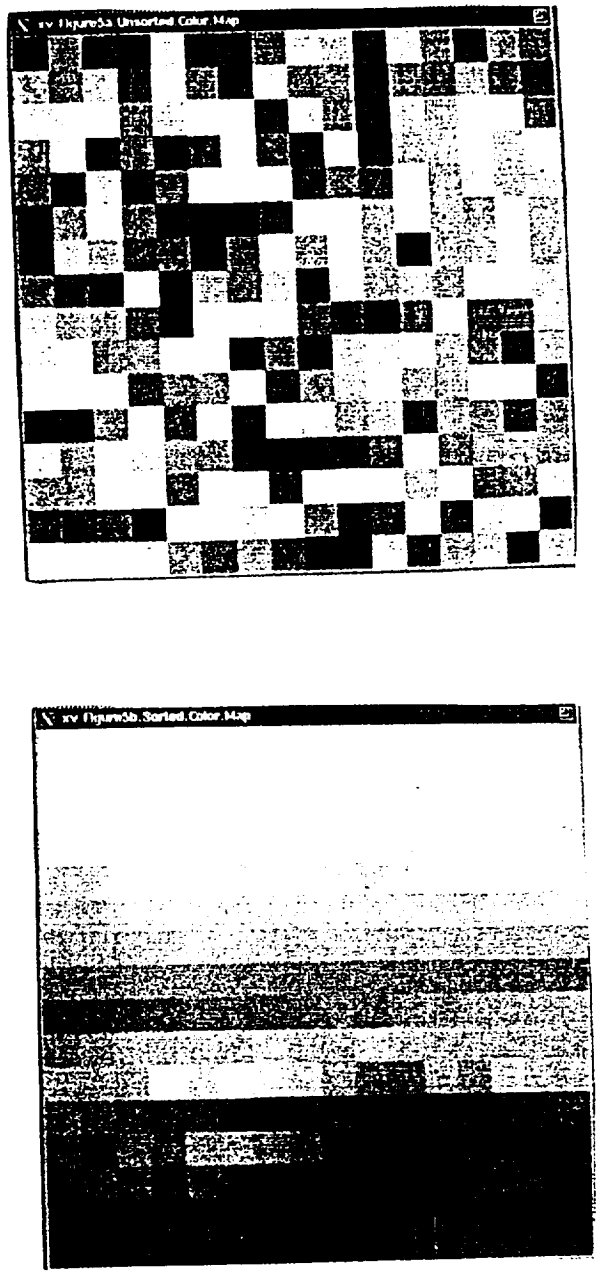

Figure 4. Colormap for Park Image (a) before and (b) after sorting

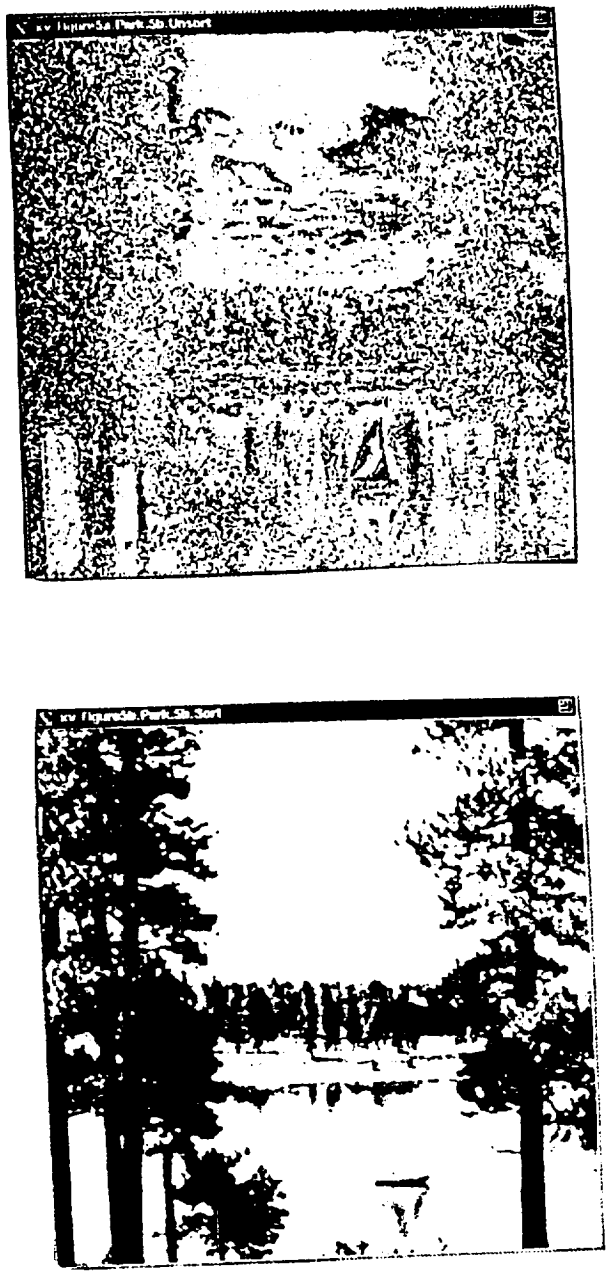

Figure 5. Park image quanitzed to five bits per pixel with (a) unsorted and (b) sorted colormaps

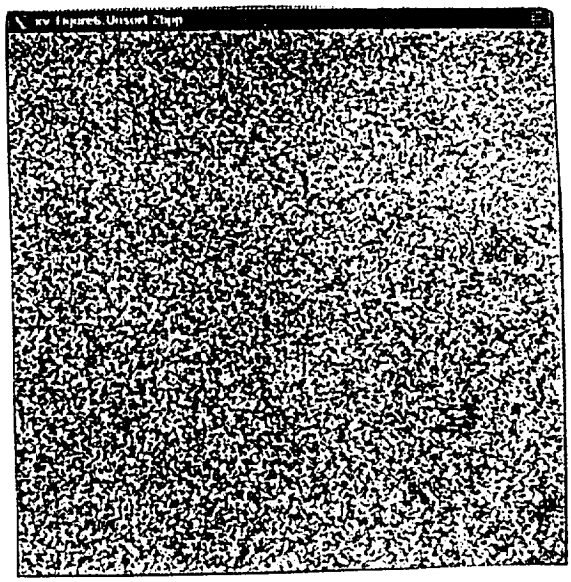

Figure 6. Lena image with unsorted colormap coded at two bits per pixel using JPEG DCT algorithm