8th IPPR Conference on
Computer Vision, Graphics and Image Processing
Tao-Yuan, Taiwan, R.O.C.
Aug. 20~23, 1995

Color Image Compression Using Quantization, Thresholding, and Edge Detection Techniques All Based on the Moment-Preserving Principle ♰

Chen-Kuei Yang@ (楊健貴) and Wen-Hsiang Tsai* (蔡文祥)

Department of Computer and Information Science
National Chiao Tung University
HsinChu, Taiwan 300, Republic of China
E-mail:whtsai@cis.nctu.edu.tw

Abstract

A new approach to color image compression with high compression ratios and good quality of reconstructed images using quantization, thresholding, and edge detection all based on the moment-preserving principle is proposed. An input image with 24 bits per pixel is quantized into 8 bits per pixel using a new color quantization method. The quantized image is then divided into $n \times n$ non-overlapping square blocks. Two representative colors for each block are computed by moment-preserving thresholding. A bit-map is then generated, consisting of 0's and 1's indicating whether the block pixels are assigned to the first color or the second according to the Euclidean distance measure. A moment-based edge detector is performed further on the bit-map of each non-uniform block. The two parameters ℓ and θ of a line edge with the equation of $x\cos\theta + y\sin\theta = \ell$ are obtained. The image is finally coded with a codebook of a 256-color palette; a 1-bit indicator for each block which specifies whether the block is uniform or not; an 8-bit color index for a uniform block, or two 8-bit color indices, a 3-bit index for θ, and a 2-bit or 3-bit index for ℓ for a non-uniform block. An average compression ratio of 22.49 or 33.32 can be obtained for 4×4 or 5×5 image blocks, respectively. Experimental results show the feasibility and efficiency of the proposed approach for color image compression.

1. Introduction

The objective of an image compression technique is to remove as much redundant information as possible without destroying the image integrity. In this study, the moment-preserving principle is used for color image compression to keep the quality of the resulting image good and acceptable for the human vision system.

The moment-preserving principle has been used in many compression methods mostly for gray-scale images. The block truncation coding (BTC) method developed by Delp and Mitchell [1] includes the use of a two-level non-parametric quantizer that adapts over the local regions of an input image. Lema and Mitchell [2] used absolute moments in the BTC technique to compress the three components of a color image separately. Wu and Coll [3] used a single bit-map to quantize all the three color planes. Kurita and Otsu [4] used the mean vector and the covariance matrix of color vectors to compute the principal score for the pixels in an image block, and classified the pixels in the block into two classes. Yang et al. [5] preserved some moments of the color components and computed two sets of desired color vectors accordingly using analytic formulas. In these methods, a compression ratio of no larger than 6 can be obtained for a 4×4 block.

In this study, a color image compression approach yielding high compression ratios and good reconstructed image quality is proposed. The approach consists of three steps, namely, quantization, thresholding, and edge detection, which are all based on the moment-preserving principle. First, an input color image with 24 bits per pixel is quantized into 8 bits per pixel. The quantization process is based on the moment-preserving thresholding technique [6]. Second, the quantized image is divided into $n \times n$ non-overlapping square blocks. Each block is requantized into two representative colors (R_1, G_1, B_1) and (R_2, G_2, B_2) by applying the moment-preserving thresholding technique to each color component. A bit-map is then generated, consisting of 0's and 1's indicating whether the block pixels are assigned to X_1 or X_2 where $X = R$,

* To whom all correspondence should be send.

@Also with the Department of Information Management, Ming Chuan College, Taipei, Taiwan 111, Republic of China.

♰This work was supported partially by National Science Council, Republic of China under Grant NSC83-0408-E009-010.

G, or B. Third, a moment-based subpixel edge detection technique [7] is performed on the bit-map of each non-uniform block. The purpose is to divide the block into two regions separated by a line edge. The image is finally coded with a codebook of 256 color palettes, one 1-bit indicator for each block which specifies whether the block is uniform or not, and one or two 8-bit indices of the color palette according to the block's uniformity. When the block is not uniform, additionally required are one 3-bit index for the line direction, and one 2-bit or 3-bit index for the line intercept for 4×4 or 5×5 window size, respectively. An average compression ratio of 22.49 or 33.32 can be obtained for a 4×4 or 5×5 image block, respectively.

The remainder of this paper is organized as follows. In Section 2, the proposed color image quantization and compression methods are presented. Several experimental results showing the feasibility of the proposed approach are described in Section 3. Finally, some conclusions are given in Section 4.

2. Proposed Approach to Color Image Compression

2.1. Proposed Color Quantization method

A color image can be defined as follows:

$$f: M \times M \rightarrow C \subseteq \Psi \qquad (1)$$

where $\Psi = \{(r, g, b) \mid 0 \leq r, g, b \leq 255\}$ is the RGB color space, $(x, y) \in M \times M$ are the spatial coordinates of a pixel, M is the integer set, and $C = \{\vec{c}_1, \vec{c}_2, \cdots, \vec{c}_N\}$ is a set of colors used in the image. There are $256 \times 256 \times 256$ combinations of red, green, and blue components. It puts heavy burden to deal with so many colors! By color quantization, not only the complexity can be reduced but the compression rate is also increased. A quantized image may be regarded as a mapping defined by

$$q: M \times M \rightarrow R \subseteq \Psi \qquad (2)$$

where $R = \{\vec{r}_1, \vec{r}_2, \cdots, \vec{r}_k\}$ is a set of representative colors used in the quantized image.

According to the above definition, the process of color image quantization basically can be divided into two major steps. The first step, called color palette design, selects the best match set of colors for a specific image. The second step, called pixel mapping, associates each pixel of the image with a best match color in the color palette to yield an image with the highest quality. The optimization goal is to make the perceived difference between the original image and its quantized version as small as possible. It is very difficult to formulate a definitive solution to meet this goal in terms of perceived image quality. In fact, there is no good objective criterion available for measuring the perceived image quality. In this study, the mean absolute error is used to measure the difference between the original image f and its quantized reproduction q. Accordingly, the average quantization error is defined as follows:

$$d_M(f, q) = \frac{1}{T} \sum_{(x,y) \in M \times N} |f(x, y) - q(x, y)| \qquad (3)$$

where T is the total number of pixels of the image.

2.1.1. Subsampling for Histogram Creation

A histogram gives the relative frequency of occurrences of colors in an image. The original color image has 24 bits per pixel. There are thus 2^{24} possible colors. To save memory space and computation time, it is desirable to reduce the histogram resolution. Heckbert [11] has suggested that the required resolution be at least 15 bits per pixel or 5 bits for each color component.

In our approach, to speed up histogram creation, not only the histogram resolution is reduced but also only the even-row-odd-column pixels are taken into account for the histogram. Hence, only a quarter of the original image pixels are subsampled. The corresponding histogram was found still a good approximation of the color distribution of the original image.

2.1.2. Design of Color Palette

Many algorithms [11-21] have been proposed for the design of color palettes. The popularity algorithm [11] selects the N most frequently occurring colors in the image as the representative colors according to the image color distribution. In order to prevent the concentration of too many colors in one neighborhood, an improved popularity algorithm was proposed by Braudaway [12]. When a most frequently occurring color is selected, the frequency count of its neighboring colors is reduced. The median cut algorithm [11] uses a splitting technique to repeatedly sub-divide the color histogram into smaller and smaller boxes which contain approximately equal numbers of color occurrences, and pick the mean values of the boxes as the representative palette colors. The mean-split algorithm [13] splits the box at the mean value instead of at the median, and recomputes the low mean and high mean for the two separated boxes as the representative palette colors. The variance-based algorithm [14] is the same as the median cut algorithm except that it splits the box whose color distribution has the largest values of variances, and the centroids of the sub-boxes are chosen as the representative palette colors. The maxmin algorithm [15] tries to minimize the distances between the selected and the original image colors. The initial P palette colors are selected by the frequencies of occurrences which exceed a predetermined threshold. The choice of the K representative colors is recursively made where K < P. One choice for the initial color value could be the most frequently occurring color found in the image to be quantized. The new representative color value, selected from the unused colors, is the one whose minimum distance to the representative colors selected thus far is maximum. There are still a lot of other algorithms [16-21] for the design of color palettes, but they are less relevant to this study and so are not described here.

In our approach, the entire color space is regarded initially as a single cube whose representative color is the mean vector of the R, G, and B components. The histogram resolution is reduced to 5 bits per color component and the even rows and odd columns of the image pixels are subsampled. The variances of the color components are computed. The color plane with the largest variance is split and the color space becomes two separated boxes. Moment-preserving thresholding is employed to compute two representative color values h_1 and h_2 for the two boxes and to choose as the cut point the value of the distribution closest to the p_1-tile of the corresponding histogram. Splitting of the color planes is repeated until the desired number of representative colors are achieved. K-1 times of splitting are required if K representative palette colors are needed. After the splitting process is completed, all the color coordinates in each box are regarded as identical and are represented by the representative color value computed by the moment-preserving thresholding method. Finally, all the representative colors are collected to form a color palette and are stored in a one-dimensional array as a color map. The computation time of the proposed method is less than that of the maxmin, the mean-split, and the variance-based algorithms because some steps of the proposed method are performed using analytic formulas. The quantization error is less than that of the median-cut algorithm.

2.1.3. Pixel Mapping

When a palette has been designed, the remaining step is to map the original color of each pixel in the input image to their best match in the color palette. Several techniques for this purpose have been introduced in the literature [11, 16, 19]. The simplest way is to compute the distance between the original color vector and all of the representative color vectors, and then choose the one with the minimum distance. This exhaustive search method was proposed by Heckbert [11]. But the computation is slow. A locally sorted search method was also reported in [11]. The computation time is reduced but it needs additional time to create a list of color vectors and sort the list by the distance key. The list is made by eliminating the representative color vectors whose distances to the centre of the box is above a threshold. Other faster methods include binary tree search [18] and k-d tree search [22]. These algorithms are based on the arrangement of the K representative colors in a binary tree structure. By a proper arrangement of the tree, a search time of O(log K) can be achieved.

In our approach, after the color palette is designed, each box's region in the 3D histogram contains a set of color values which are represented by a single color. Each representative color is included in the color map mentioned previously. Regarding each color value in the box as a cell, we fill up all the cells in each box with identical pointers which all point to the corresponding representative color in the color map. After the cells in all boxes in the 3D histogram are filled with pointers, the histogram may be regarded as a lookup index table. The color value of each input pixel then can be mapped to the corresponding cell in the 3D lookup index table, and in turn the pointer in the lookup table cab be retrieved and used to find the corresponding representative color in the color map. This is very convenience for pixel mapping because no search is needed. This direct pixel mapping method is shown in Fig. 1.

An algorithm is given below to summarize the proposed color quantization method using moment-preserving thresholding and splitting techniques (abbreviated as CQMPTS) as described above.

Algorithm 1. Color quantization by moment-preserving thresholding and splitting (CQMPTS)

Input. A digital color image.

Output. A quantized image with K colors.

Method.

Stage 1: Initialization

Step 1: Create a 3D histogram of the color component distribution with quarterly reduced color resolution for each color component using the subsampled pixels of the even rows and odd columns of the input image.

Step 2: Compute the means of the R, G, and B components individually to form the initial representative color vector for the entire color space which is the only cubic box so far.

Step 3: Compute the variances of each color component.

Stage 2: Color palette design

Step 4: Select the color plane which has the largest variance value among all of the color planes of the currently existing cubic boxes. The cubic box from which the selected color plane comes is called the selected cubic box.

Step 5: Compute the first three moments of the pixels on the selected color plane obtained in Step 4, and compute h_1, h_2, p_1, and p_2 according to [6].

Step 6: Split the selected cubic box into two subcubes using as the cut point the p_1-tile of the color component distribution of the selected color plane.

Step 7: Substitute the color component values of the selected color plane of the two newly-created subcubes with h_1 and h_2, respectively (e. g., if the selected cubic box's representative color vector is (r, g, b), and the red plane of the box is selected to split, then the two new representative colors for the two newly- created boxes are (h_1, g, b) and (h_2, g, b), respectively).

Step 8: Recompute the variances of the color component values of the selected color plane of the new distributions of the two newly-created subcubes.

Step 9: Repeat Steps 4 to 8 until K representative colors are obtained.

Stage 3: Pixel mapping

Step 10: Collect the representative color vectors of all the resulting cubic boxes to form a 1D color map.

Step 11: Build a 3D lookup index table by filling up the cells of each resulting cubic box with identical pointers which point to the corresponding representative color in the color map.

Step 12: For each pixel of the input image, perform the following steps:

(1) recompute for the pixel a new color vector (R', G', B') with quarterly-reduced color resolution according to the previously-mentioned reduction of the histogram resolution;

(2) map the color vector (R', G', B') to the corresponding cell in the 3D lookup index table, and in turn follow the index pointer in the cell to get a representative color in the color map;

(3) replace the pixel's color with the color obtained in the last step.

Some experimental results and performance evaluations of the above algorithm are given in Section 3.

2.2. Color Image Compression

After the input color image is quantized into 256 colors using the previously-described algorithm CQMPTS, the image is divided into $n \times n$ non-overlapping blocks. Each block is requantized in this study into two colors by applying the moment-preserving thresholding to each color component again. Each image block can thus be coded with two indices of colors and one $n \times n$ bits bit-map. However, the resulting bit-maps still occupy more than 50% of the output codes. In order to increase the compression ratio, the moment-based edge detector is performed on the bit-map. Hence, a non-uniform image block can be coded with two indices of colors and two edge parameters without using the bit-map. And a uniform block can be coded with just a single index of color. Furthermore, since the block size is small, the range of the possible orientations θ of an edge in an image block is limited and so is the range of the intercept value ℓ. The values of these two edge parameters can thus be fixed to certain specified values. Then, an edge block can be coded with two indices of colors and two indices of specified edge parameter values. An algorithm using the proposed color quantization algorithm (CQMPTS), moment-preserving thresholding, and moment-based edge detection to detect edges in bit-maps for color image compression (abbreviated as CICQTE) is given below.

Algorithm 2. Color image compression using quantization, thresholding and edge detection (CICQTE).

Input. A test image f.

Output. A compressed image with each block being coded with one 1-bit indicator; one 8-bit index of a color in a 256-color palette for a uniform block, or two 8-bit indices of colors in a 256-color palette and two 3-bit indices of the edge parameters for a non-uniform block.

Method.

Step 1: Quantize f into a 256-color image f' using Algorithm CQMPTS with the output a 256-color palette codebook (i. e., the color map mentioned in CQMPTS).

Step 2: Divide f' into $n \times n$ non-overlapping blocks where n=4 or 5.

Step 3: For each block B of f', perform the following steps:

(1) requantize each block into two colors (R_1, G_1, B_1) and (R_2, G_2, B_2) by applying the moment-preserving thresholding technique to each color component individually;

(2) compute the minimum Euclidean distance between the color vector (r, g, b) of each pixel in B and (R_1, G_1, B_1) and (R_2, G_2, B_2) to generate a binary bit-map for B (e. g., if the distance between (r, g, b) and (R_1, G_1, B_1) is smaller, then the bit in the bit-map for the pixel is assigned 0);

(3) assign value 0 to the block indicator b (indicating that the block is uniform) if the value of the sum of all the bits in the bit-map is greater than $n \times n - 4$ or less than 4; otherwise assign the value 1 to b;

(4) apply the moment-based edge detector on the bit-map if b=1; otherwise output the index of the uniform color in the block, and the value of b, and skip to the next block.

(5) quantize the parameter values θ and ℓ of the edge detected in the last step into the specified values and look up their indices from a predefined codebook for θ and ℓ.

(6) output the indices of the two colors in the non-uniform block, the value of b, and the two indices of θ and ℓ.

Block truncation coding (BTC) [1] was originally proposed for gray-scale images to quantize an image block into two gray levels. The requantization of a color image block into two colors in Step 3(1) of Algorithm CICQTE is of the same idea as BTC. However, the two resulting color vectors may not be found in the 256-color palette. Thus, the minimum value of the Euclidean distance is used as a criterion to select the best match color in the 256-color palette. This criterion is implemented in Step 3(2) of Algorithm CICQTE to generate a bit-map.

From our experimental experience, in general there are very few cases in which all the values of the bits are one or zero in the bit-map. However, the variation of the bit values in a 4×4 or 5×5 block will become noticeable when there exist four or more bits which form a cluster and show a shape of line and edge. According to this concept of visual discontinuity, a block may be assumed uniform when the value of the sum of all the bit values in the bit-map is greater than $n \times n - 4$ or less than 4 (see Step 3(3)). Note that as the number of uniform blocks increases, the compression ratio is improved.

The difference between two edges with very close directions is not perceivable when the image resolution is high and the window size is small. Also, the number of the possible orientations of an edge in a small image block is limited. Therefore, the values of θ and ℓ can be predefined and fixed to be some specified values. The values of ℓ are predefined to be the distances from the center of the block to those pixels along the x-axis in the block, and the values of θ are selected to be those with the increments of $\frac{\pi}{6}$. These values of θ and ℓ in a 4×4 and 5×5 circular window are shown in Table 1. Because these values are predefined and fixed, it is not necessary to count them in the coding cost.

A 256-color palette codebook needs 3×8×256 bits. Each uniform block is coded with 1×8+1=9 bits and each edge block is coded with 2×8+1+2+3=22 bits or 2×8+1+2×3=23 bits for 4×4 or 5×5 window size, respectively. The overall compression ratio so can be computed as follows:

$$r = \frac{M \times N \times 3 \times 8}{256 \times 3 \times 8 + b_0 \times 9 + \left(\frac{M \times N}{n \times n} - b_0 \right) \times b} \qquad (4)$$

where M×N is the image size, b_0 the number of uniform blocks, n×n the block size, and $b = 5$ or 6 for the indices of the edge parameters with 4×4 or 5×5 window size, respectively.

3. Experimental Results

The proposed algorithms have been tested on an IRIS Indigo workstation on several color images. Each color image has 24 bits/pixel and is 512×512 in size. We use the mean absolute error (MAE) as the criterion for the performance evaluation of image quantization and compression. A smaller value of the MAE means that the quantized or the reconstructed image is less distorted. Table 2 shows the MAE values of the reconstructed images and the compression ratios of the images mentioned above with 4×4 and 5×5 window sizes, respectively. Fig. 2 shows the standard images of "Lena," "pepper," and "mandrill" of size 512×512 in left, middle, and right, respectively. Fig. 3 shows the results of the proposed quantization method with 256 colors using the images shown in Fig. 2. Figs. 4 and 5 show the reconstructed results of the proposed compression approach using the images shown in Fig. 2 with 4×4 and 5×5 window sizes, respectively. From Fig. 3, we can find that the image quality is almost the same as the original. It is difficult to find differences between the original images and the reproduced results by the human vision system. However, little differences can be distinguished by the computed MAE values. It can be seen from Figs. 4 and 5 that the reconstructed image quality is still good and acceptable. Only some slightly zigzag edges and a litter color difference in regions with gradual color changes can be seen. This is due to the use of fewer bits for the indices of ℓ and θ and the detection of line edges only.

4. Conclusions

A new color image compression algorithm using quantization, thresholding, and edge detection all based on the moment-preserving principle has been proposed. The major contributions of this study include the following. (a) A new approach to the design of color palettes is proposed. (b) A fast pixel mapping method for color quantization and reproduction is presented. (c) An edge operator is performed on the bit-map to avoid using the bit-map. (d) An efficient coding of the edge detection results has been proposed. (e) Reconstructed images with good quality and reasonable compression ratios have been obtained. (f) The computation is fast and thus suitable for real-time applications due to the use of analytic formulas for some of the quantization and compression steps. (g) The proposed approach is applicable not only to the RGB color model but also to any other trichromatic model. Some possible improvements and further research topics are as follows.

The human visual system is more sensitive to details in the luminance component than to details in the chrominance components. Hence, the RGB components can be first linearly transformed into the YIQ components, and the luminance and the chrominance components can be treated differently at the time of resampling the histogram with resolution reduction to save the memory space of storing the histogram. But this requires additional time for color model transformation. The size of an image block also need not be fixed. The use of variable block sizes in the compression scheme will yield generally higher compression ratios. The zigzag effect and image distortion in the compressed image can be improved by increasing the number of allowed specified orientations of the detected edges, or by detecting more complicated features in the bit-map such as curves and lines.

References

1. E. J. Delp and O. R. Mitchell, "Image compression using block truncation coding," *IEEE Trans. on Commun.*, vol. COM-27, pp. 1335-1142, 1979.
2. M. D. Lema and O. R. Mitchell, "Absolute moment block truncation coding and its application to color images," *IEEE Trans. on Commun.*, vol. COM-32, pp. 1148-1157, 1984.
3. Y. Wu and D. C. Coll, "Single bit map block truncation coding for color images," *IEEE Trans. on Commun.*, vol. 35, pp. 352-356 1987.
4. T. Kurita and N. Otsu, "A method of block truncation coding for color image compression," *IEEE Trans. on Commun.*, vol. 35, pp. 352-356, 1987.
5. C. K. Yang, J. C. Lin, and W. H. Tsai, "Color image compression by moment-preserving and block truncation coding techniques," *Proc. IEEE ICIP-94*, Austin, TX, USA, vol. 3, pp. 972-976, 1994.
6. W. H. Tsai, "Moment-preserving thresholding: a new approach," *CVGIP*, vol. 29, pp. 377-393, 1985.
7. E. P. Lyvers, O. R. Mitchell, M. L. Akey, and A. P. Reeves, "Subpixel measurement using a moment-based edge operator," *IEEE Trans. on PAMI*, vol. 11, no. 12, pp. 1293-1309, 1989.
8. A. Rosenfeld and A. C. Kak, *Digit Picture Processing*, vol. I, NY: Academic Press, pp. 223-237, 1982.
9. J. S. Weszka, "A survey of threshold selection techniques," *CVGIP*, vol. 7, pp. 259-265, 1978.
10. L. H. Chen, and W. H. Tsai, "Moment-preserving techniques in image processing-a survey and new approaches," *Proc. Natl. Sci. Counc., ROC*, vol. 13, no. 5, pp. 280-301, 1989.
11. P. Heckbert, "Color image quantization for frame buffer display," *Computer Graphics*, vol. 16, no. 3, pp. 297-307, 1982.
12. G. Braudaway, "A procedure for optimum choice of a small number of colors from a large color palette for color imaging," *Proc. Electronic Imaging '86*, San Francisco, CA , pp. 75-79, 1986.
13. R. S. Gentile, J. P. Allebach, and E. Walowit, "Quantization of color images based on uniform color spaces," *J. Imaging Technology*, vol. 16, no. 1, pp. 12-21, 1990.
14. S. J. Wan, P. Prusinkiewicz, and S. K. M. Wong, "Variance-based color image quantization for frame buffer display," *Color Research and applications*, vol. 15, no. 1, pp. 52-58, 1990.
15. G. Houle and E. Dubois, "Quantization of color images for display on graphics

terminals," *Proc. IEEE Global Telecommun. Conf.*, pp. 1138-1142, 1986.
16. R. Balasubramanian, C. A. Bouman, and J. P. Allebach, "Sequential scalar quantization of color images," *J. Electronic Imaging*, vol. 3, no. 1, pp. 45-59, 1994.
17. S. S. Dixit, "Quantization of color images for display/printing on limited color output device," *Computer and Graphics*, vol. 4, no. 1, pp. 561-567, 1991.
18. M. T. Orchard and C. A. Bouman, "Color quantization of images," *IEEE Trans. on Signal Processing*, vol. 39, no. 12, pp. 2677-2690, 1991.
19. Naftaly Goldberg, "Color image quantization for high resolution graphics display," *Image and Vision Computing*, vol. 9, no. 5, pp. 303-312, 1991.
20. X. L. Wu and I. H. Witten, "A fast k-means type clustering algorithm," *Technical Report*, no. 85/197/10, Dept. of Computer Science, University of Calgary, Calgary, Canada, 1985.
21. X. Wu, "Color quantization by dynamic programming and principal analysis," *ACM Trans. on Graphics*, vol. 11, no. 4, pp. 348-372, 1992.
22. J. J. Friedman, J. L. Bentley, and R. A. Finkel, " An algorithm for finding best matches in logarithmic expected time," *ACM Trans. on Math. Software*, vol. 3, pp. 209-226, 1977.

Table 1. The predefined values of ℓ and θ for 4×4 and 5×5 window size.

	4×4				5×5				
ℓ	-0.75	-0.25	0.25	0.75	-0.8	-0.4	0	0.4	0.8

	4×4 and 5×5						
θ	$-\pi/2$	$-\pi/3$	$-\pi/6$	0	$\pi/2$	$\pi/3$	$\pi/6$

Table 2. The values of MAE and the compression ratios of the resulting images mentioned in Table 2 with 4×4 and 5×5 block sizes, respectively

	4×4 block size		5×5 block size	
Images	MAE values	compression ratios	MAE values	Compression ratios
Lena	7.9226	22.06	8.4477	31.09
pepper	8.9569	21.74	10.1498	30.30
jet	8.1365	23.70	8.8268	32.79
house	5.2712	24.28	6.0135	37.69
mandrill	16.4548	19.18	19.3235	28.07
candy	4.9882	23.98	5.4691	39.98
average		**22.49**		**33.32**

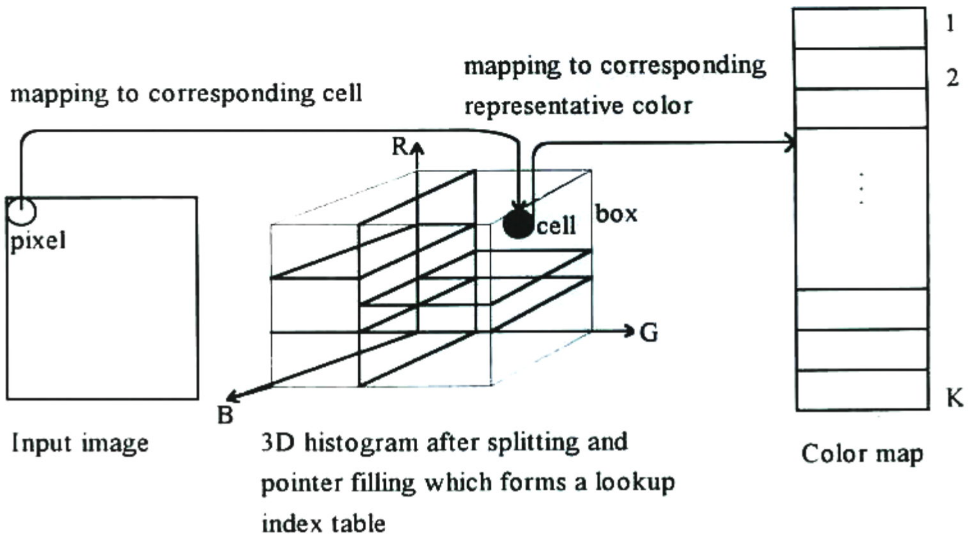
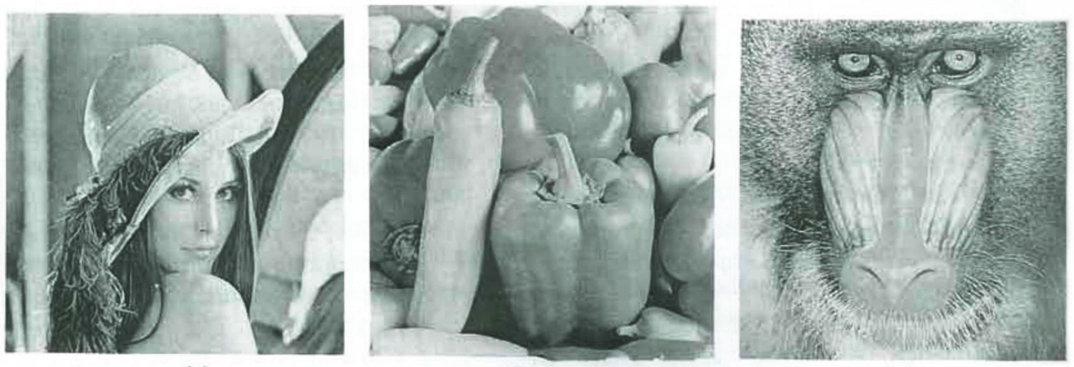
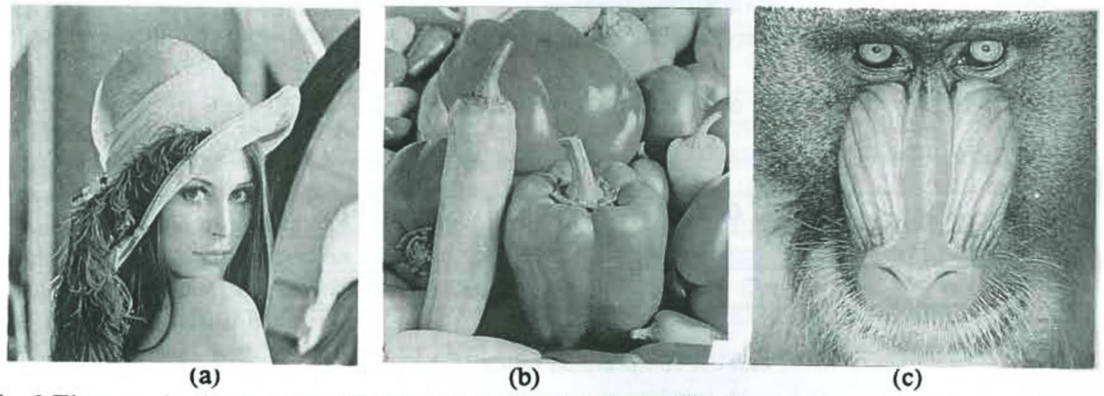

mapping to corresponding cell
mapping to corresponding representative color
R
pixel
cell
box
G
B
1
2
K
Input image
3D histogram after splitting and pointer filling which forms a lookup index table
Color map

Fig. 1. The process of pixel mapping for reproducing a quantized image.



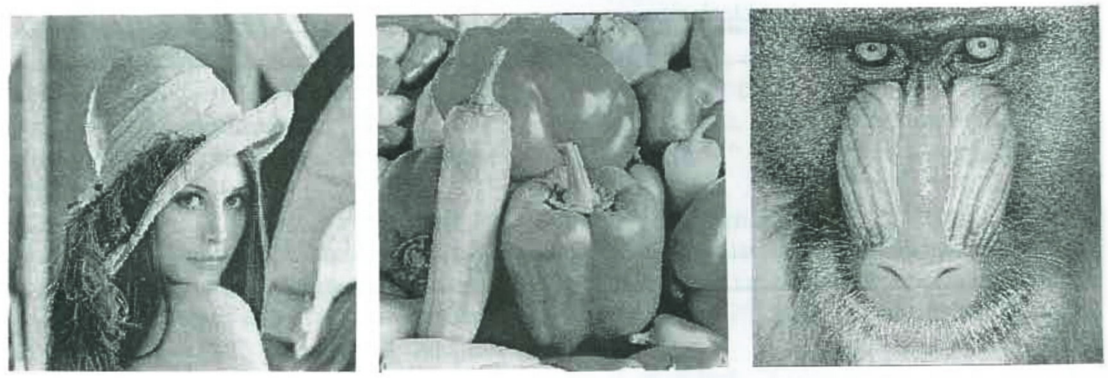
(a) (b) (c)

Fig. 2 The original images of "Lena," "pepper," and "mandrill" in left, middle, and right, respectively.



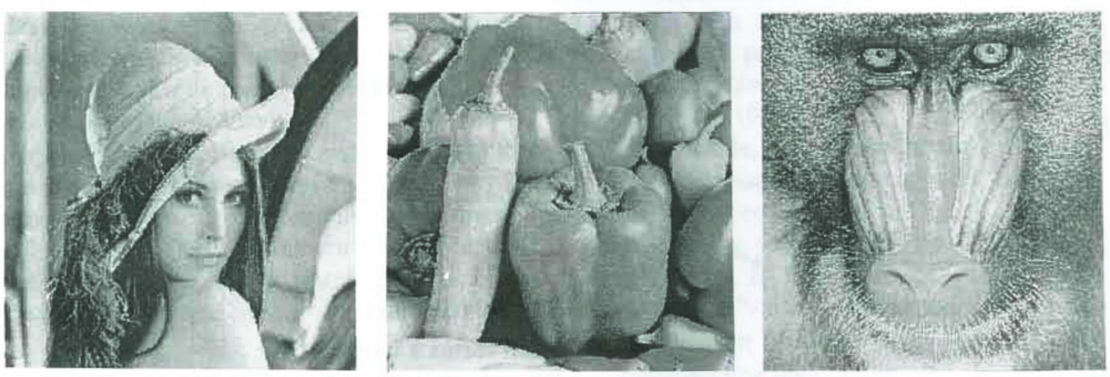
(a) (b) (c)

Fig. 3 The quantization results with 256 colors of the images in Fig. 2.



(a) (b) (c)

Fig. 4 The reconstructed results of the images in Fig. 2 with 4×4 windows.



(a) (b) (c)

Fig. 5 The reconstructed results of the images in Fig. 2 with 5×5 windows.