



Available online at [www.sciencedirect.com](http://www.sciencedirect.com)

SCIENCE @ DIRECT®

JOURNAL OF  
VISUAL  
Communication &  
IMAGE  
Representation

J. Vis. Commun. Image R. 16 (2005) 311–332

[www.elsevier.com/locate/jvcir](http://www.elsevier.com/locate/jvcir)

## Novel fast color reduction algorithm for time-constrained applications

Kiattisin Kanjanawanishkul<sup>a,\*</sup>, Bunyarit Uyyanonvara<sup>b,1</sup>

<sup>a</sup> Department of Electrical Engineering, Kasetsart University, Siracha Campus, 199, Moo 6, Sukhumvit Road Thung-Sukhla, Siracha, Chonburi 20230, Thailand

<sup>b</sup> Department of Information Technology, Sirindhorn International Institute of Technology (SIIT), Thammasat University, 131 Moo 5, Tiwanont Road Bangkadi, Muang Pathumthani 12000, Thailand

Received 29 January 2004; accepted 2 July 2004

Available online 12 October 2004

---

### Abstract

In this paper, we propose a new adaptive approach of color quantization. It can significantly reduce the time consumption during the process compared with available methods but still maintains a good quality (greater than 30 dB of PSNR). It is implemented as a part of the media stream compression algorithms for a True Color Signboard System. We adapt and create some techniques to speed up the process. We start with a sampling technique on an RGB color space before constructing the 3D histogram of color distribution, and then we use the dynamic programming based on Wu's algorithm to construct the cumulative moment distribution. Then, we put the cutting plane through the centroid of that box. This plane is perpendicular to the axis, on which the sum of the squared Euclidean distances between the centroid of both of sub-boxes and the centroid of the box is the greatest. The sub-box, which contains the greatest value representing variance, is repeatedly sub-divided into the smaller sub-boxes until reaching the desired number of the representative colors. From our whole process, we gain approximately up to 50% less time consumption than Wu's quantizer [ACM Trans. Graph. 11 (1992) 348] and it is significantly faster than existing algorithms as shown in the result.

© 2004 Elsevier Inc. All rights reserved.

---

\* Corresponding author. Fax: +66 3835 4849.

E-mail addresses: [kiattisin@eng.src.ku.ac.th](mailto:kiattisin@eng.src.ku.ac.th) (K. Kanjanawanishkul), [bunyarit@siit.tu.ac.th](mailto:bunyarit@siit.tu.ac.th) (B. Uyyanonvara).

<sup>1</sup> Fax: +66 2501 3524.

*Keywords:* Color quantization; Sub-sampling quantizer; Value representing variance; Centroid mapping; Squared euclidean distance

---

## 1. Introduction

Like many time-constrained application, graphical signboard system which is most modern and effective among various available media for advertising and information display requires a fast and effective color reduction algorithm. This board is a microprocessor based system with LEDs arranged in a combination of rows and columns (Matrix). By adjusting the intensity of LEDs, different color shades can be displayed on the board. The animation can be made by displaying individual frames instantaneously on the board. Furthermore, the boards are interfaced to other hardware equipment to produce any customized application. Graphics and special effects like curtain rising, sliding, scrolling, shooting, zooming, flashing, etc. makes the visibility of the advertisement lively and effective on the display. The display system is capable of displaying the live information caught by video camera that accepts live frames (up to 20 frames) through the computer. The video signal from the video camera is received, processed with the frame grabber, video digitizer card, and transferred to the computer system located at the display board.

The main technique used in this system is the data compression. Although there are a number of hardware accelerators to achieve these requirements, it builds up the investment cost. Optimal software design is an alternative to solve this issue. Since one major limitation of the LEDs is that it is incapable of emitting as many colors as required by most applications. Color quantization is used as a first stage of our whole optimal algorithms.

Color quantization is one of the most useful lossy compression methods to find an acceptable set of palette color (typically 256) that can be used to represent the original colors of a digital image. Actually, the human eye can only distinguish less than a thousand of the color, therefore 8-bit indexed color is sufficient for human perception. This makes the color quantization process be fluently exploited for many applications especially in computer graphics and image processing.

Generally, full color digital display systems use red, green, and blue channel, each is 8-bit resolution, to specify the color of each pixel on the screen. The image usually is composed of a large number of distinguishable colors. Therefore the color quantization problem can be modeled as the cluster analysis problem. The number of ways of sorting  $n$  objects into  $k$  clusters is given by Liu (1968)

$$N(n, k) = \frac{1}{k!} \sum_{i=0}^k (-1)^i \binom{k}{i} (k-i)^n. \quad (1)$$

For example, to sort 25 objects into five clusters, there are  $N(25,5) = 2,436,684,974,110,751$  ways. Clearly, it is impractical for an algorithm to exhaustively search the best solution. Therefore, our algorithm is proposed to provide a tradeoff between acceptable quality and running time.

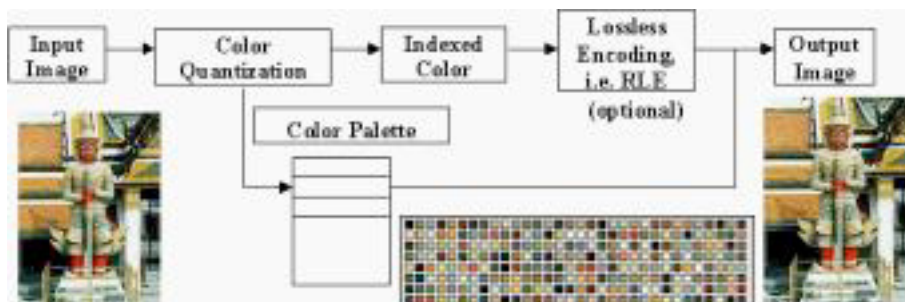


Fig. 1. Color quantization process.

In this paper, Euclidean distance is adopted in RGB color space because RGB color space is the hardware oriented color space used by most display devices. Quantization of colors is generally performed in the RGB color space since it has been proven by Heckbert (1982) that no particular advantage is gained by using other color spaces.

The entire process of color quantization, shown in Fig. 1, is applied in the Bitmap file format.

This paper is organized as follows. In Section 2, we introduce the literature survey of color quantization. Section 3 explains our proposed approach. Experimental results are introduced in Section 4. Finally, we conclude and give some future works in Section 5.

## 2. Literature survey

Over the past decades, there are a large number of approaches, proposed to produce the highest quality quantized images. While some approaches proposed to exploit the excellent algorithms or theoretical issues, such as fuzzy logic, neural network and genetic algorithm, etc., to their quantizers. Unlike these quantizers, we emphasize on the execution time. The main objective of our approach is to compromise between the acceptable quantization quality, which affects to the perceived distortion in the resulting image, and minimization of computational cost, which affects to the time consumption.

In general, color image quantization involves two steps, i.e., palette design and pixel mapping. The first step selects the set of colors for a particular image. The second step associates each pixel of the image with a color from this palette to yield the optimal quality image.

### 2.1. Palette design

There are two general classes of quantization methods: fixed or universal palette and adaptive or custom palette. For fixed quantization, it is independent from the image contents, while for the adaptive quantization, a palette is designed adaptively to the image contents.

Fixed quantization is very fast, but sacrifices the quantization quality. Uniform quantization is an example of the fixed quantization. Generally, this kind of quantization can be made for the red axis and for the green axis into 8 levels each, and the blue axis (the human eyes are less sensitive to blue) into 4 levels. So that  $8 \times 8 \times 4 = 256$  colors are uniformly spread over the color space are available. A important drawback of this method is the potential artifact of appearing edges in the image, e.g., color-shift or false contour. And it also fails to capture the distribution of colors, i.e., some cells may be empty, and are wasted. On the other hand, adaptive quantization is the task of finding an optimal set of representative colors. It may be formulated as a large scale-clustering problem. An image-dependent set of display colors and a corresponding mapping from image colors to display colors are used. They practically always give much better quantization results, but take much more time than fixed quantization. The existing adaptive techniques to design a color palette can be divided into three categories by Sangwine and Horne Sangwine and Horne, 1998 as follows:

(a) *Hierarchical scheme or pre-clustering*. Most of the proposed algorithms are based on statistical analysis of the color distribution of image pixels within the color space. Hierarchical clustering methods can be further sub-divided into agglomerative (bottom-up) and divisive (top-down). An agglomerative clustering starts with one-point clusters and recursively merges two or more most appropriate clusters. A divisive clustering starts with one cluster of all data points and recursively splits the most appropriate cluster. The process continues until a stopping criterion (frequently, the requested number  $k$  of clusters) is achieved. The popularity and median cut proposed by Heckbert (1982) were popular in the past. After that other approaches were proposed such as the variance minimization (proposed by Wan et al. (1990)), binary splitting (proposed by Orchard and Bouman (1991)), greedy tree growing (proposed by Zeng et al. (1995)), Octree (proposed by Gervautz and Purgathofer (1990)), merging vector quantization approach (reported by Balasubramanian and Allebach (1991)), and Principal Analysis Algorithm (proposed by Wu (1992)). Lately, Sirisathitkula et al. (2004) proposed a new algorithm. They sorted colors based on their components along the principal axis, the one with the highest variance of color distribution, then the Euclidean distances between any adjacent colors' along the 1D axis are calculated and used to find the cutting plane.

Some of these algorithms, for example binary tree splitting (Orchard and Bouman, 1991) try to maintain a tree structure for the code to ensure more efficient pixel mapping and allow arbitrary orientations of the plane used to split a cluster.

(b) *Iterative scheme or post-clustering*. It involves an initial selection of a palette followed by iterative refinement of this palette using the K-Means algorithm to minimize the Mean Square Error. K-Means Algorithm can be called LBG (Linde et al., 1980) or Generalized Lloyd Algorithm (GLA) because Linde, Buzo, and Gray extended Lloyds' basic design (Lloyd, 1982) of scalar quantization to the general case of vector quantization. The K-Means Algorithm is by far the most popular clustering tool used in scientific and industrial applications. The wide popularity of K-Means Algorithm is well deserved. It is simple, straightforward, and is based on the firm foundation of analysis of variances. However, the result strongly depends

on the initial guess of centroids and its time consumption is very high. Fuzzy C-mean proposed by Lim and Lee (1990) is an extension of the LBG algorithm. But the selections of initial condition of this method still affect the results. In order to overcome this drawback, Scheunders proposed the Hierarchy Competitive Learning (HCL) (Scheunders, 1997) and Genetic C-means Algorithm (GCMA) (Scheunder, 1997). The hierarchical approach is a hybrid structure between competitive learning and splitting of the color space. A genetic approach is a hybrid structure between a genetic algorithm and C-means clustering.

Additionally, Kohonen's Self-Organizing Map (SOM), introduced by Kohonen (1989) is another vector quantization-related algorithm in the neural network category. Dekker's approach (Dekker, 1994) is very famous for exploiting the Kohonen Self-Organizing-Maps. His algorithm operates using a one-dimensional self-organizing Kohonen Neural Network. Two more hybrid methods are Local K-Means (LKM) and Adaptive Color Reduction (ACR) based on neural network approaches. LKM proposed by Verevka and Buchanan (1995) combines a K-Means Quantization and Self-Organizing Map, while ACR proposed by Papamarkos et al. (2002) consists of a principal component analyzer and a Kohonen Self-Organizing Feature Map (SOFM).

(c) Finally, there is a group of very simple algorithm which can be classified as based on improved scalar quantization rather than on vector quantization. They attempt to exploit within scalar quantization the statistical dependency and dimensionality of image data in a way similar to that use in vector quantization. Sequential scalar quantization proposed by Balasubramanian et al. (1995) is an example of this scheme.

## 2.2. Pixel mapping

After obtaining the color palette, the next step is the pixel mapping in which each color pixel is assigned to one of the colors in the palette. In general, finding the closest palette color is a computationally intensive operation, as it involves a brute force algorithm, which visits every representative to find the best mapping. So it is wasted a lot of time, it is impractical to use for the pixel mapping phase. There are some available methods for this step as follows:

(a) *k-d trees*. The classical k-d tree method for nearest-neighbor search by orthogonal partitioning was proposed by Bentley (1975). The k-d tree data structure provides a mechanism for examining only those points closest to the query point, considerably reducing the computation required to find its nearest-neighbor. This algorithm performs well for a large number of uniformly distributed data.

(b) *Locally sorted search*. Heckbert (1982) proposed this method to overcome the disadvantage of brute-force algorithm when the number of representative colors is large and when the colors in the input image have a wide distribution.

(c) *Hash tables*. The RGB color value is used as a hash key to find the corresponding color table index. Hashing is the method of searching appropriate bucket by hashing function using the key obtained data under consideration. The advantage of hashing does not need to search the whole table, but it obtains the location via

hashing function. When hashing applied to the clustering algorithm, this property allows nearest cluster to be found without computing the all clusters.

(d) *Centroid mapping*. A faster, sub-optimal alternative to nearest-representative mapping is to use the partition obtained during the color palette design process and map the image colors to the representative, which is the centroid of the cluster containing the color.

### 3. Our proposed techniques

Clearly, a large number of papers prove that the hierarchical method is faster than the iterative method but it gives less quality than the latter. Most hierarchical algorithms do not revisit once constructed (intermediate) clusters with the purpose of their improvement. Unlike traditional hierarchical methods, the iterative algorithms gradually improve clusters. With appropriate data, this results in high quality clusters. Our quantizer belonging to the hierarchical scheme is based on Wu's quantizer proposed in 1992 (Wu, 1992). His quantizer outperforms the early quantizers with respects to both computation time and quantization quality. In this paper, our techniques, which can reduce the computational time with an acceptable quality, are presented in Section 3 and the quality metrics are investigated in order to point out the performance of the quantizers in Section 3.2.

#### 3.1. Our techniques for color quantization

Our new algorithms are divided into four phases i.e., (a) histogram construction; (b) cumulative moment construction; (c) cutting plane positioning, and (d) pixel mapping.

##### 3.1.1. Histogram construction phase

First, a  $32 \times 32 \times 32$  3D-color histogram, based on Heckbert's approach (Heckbert, 1982) is created from a 24-bit/pixel color image by reducing the resolution of each color component to 5 bits. The histogram, which gives the relative frequency of occurrences of colors in an image, is then used as the color space for extracting representative colors. Although a  $32 \times 32 \times 32$  3D histogram is employed, the human eyes cannot distinguish the pre-quantized image. However, the time consumption of this phase rather depends on the size of an input image. We found that it is not necessary to use every pixel from the original image to construct this 3D histogram. Often images can have many pixels removed and still represent image contents. Actually, images, which come from the external sources, not created by using the computer applications are sampled and quantized to the digital format. However, we can shrink the input image by the sampling method. This shrunken image still represents the original one. An Eq. (2) below is used to create a new version of the image.

$$x[n_1, n_2] = x_i[n_1 T_1, n_2 T_2], \quad (2)$$

where  $x_i[t_1, t_2]$  is the original image,  $x[n_1, n_2]$  is the sampled version, and  $T_1$  and  $T_2$  are the sampling intervals on the horizontal and vertical directions, respectively.

Fig. 2 shows one of the test images. We investigate the sampling rate from 2 to 10 and then representative colors are extracted from its 3D histogram. The numbers of pixels are reduced to  $250 \times 250$ ,  $167 \times 167$ ,  $125 \times 125$ ,  $100 \times 100$ ,  $84 \times 84$ ,  $72 \times 72$ ,



Fig. 2. Different sampling rates on the test image.



(J) 50 × 50 pixels (Sampling Rate = 10).

Fig. 2. (continued)

63 × 63, 56 × 56, and 50 × 50 pixels as shown in Figs. 2B–J, respectively. However, this sampling technique is not suitable for the low-resolution image, of which either width or height is less 200 pixels because it may lose some important details. From our evaluation on five images of each different size: 200 × 200, 300 × 300, 400 × 400, 500 × 500, and 600 × 600 pixels as shown on Table 1 (only average value), we can conclude in Table 2 to point out the proper sampling rate. This table shows how to categorize an input image before performing the image sampling. We also test one hundred of test images by providing the sampling in both horizontal and vertical directions separately as shown on the experimental results in Section 4.

### 3.1.2. Cumulative moment construction phase

To avoid re-computation of the mean and our value representing variance when finding the position of the cutting plane, we exploit the dynamic programming based on Wu's algorithm (Wu, 1992) to construct the bottom-up cumulative distribution. This process is applied after 3D-color histogram is obtained. Unlike Wu's method (Wu, 1992), we cumulate only the zeroth-order and first-order moment distribution because we find that the second-order moment calculation produces the very high execution time. Therefore, we eliminate this portion and then we exploit the zeroth-order moment (to calculate the frequency) and the first-order moment (to calculate the mean) in order to position the cutting plane and to create our value representing variance, which is mentioned in the next subsection. For example, Table 3(a)–(d) show the frequency, the zeroth-order moment cumulating, frequency multiplied by index and the first-order moment cumulating respectively.

We can calculate the mean value of the one-dimensional table as follows:

$$\text{Mean value} = \frac{201 - 0}{37 - 0} = 5.432$$

and then we can split this table into two groups at the index of 5. Therefore the mean values of both groups are:

$$\text{Mean value of group 1} = \frac{63 - 0}{18 - 0} = 3.5,$$

Table 1  
Sampling rate on five evaluated images

Size	Sampling rate	Average	
		PSNR	MSE
200 × 200	1	30.924003	164.08068
	2	30.979865	161.95594
	3	30.281332	189.24258
	4	30.072077	196.72182
	5	29.887949	205.33402
	6	29.797156	210.57235
	7	28.969527	251.60649
	8	28.251832	304.67027
	9	28.050119	317.9819
	10	27.653101	347.10135
300 × 300	1	31.701111	142.76942
	2	31.756758	140.20458
	3	31.69193	139.36602
	4	31.543253	144.65232
	5	31.089796	157.67834
	6	30.955965	166.26824
	7	30.6282	179.91056
	8	30.179792	195.18495
	9	30.324697	191.52442
	10	29.797657	215.54889
400 × 400	1	32.959819	88.218455
	2	32.279817	102.65903
	3	32.273255	101.88269
	4	32.872394	90.836139
	5	32.186435	104.69531
	6	32.515237	99.266287
	7	32.329405	106.59152
	8	32.504314	99.263862
	9	31.91914	111.86165
	10	31.094241	145.35089
500 × 500	1	33.632811	85.543397
	2	33.576089	87.393161
	3	33.6026	86.421406
	4	33.281728	94.916303
	5	33.414886	90.163226
	6	33.714541	83.931453
	7	33.30547	95.341785
	8	33.485643	89.199342
	9	32.903576	100.77072
	10	33.233182	112.57386
600 × 600	1	32.448038	119.2235
	2	32.490815	116.90895
	3	32.471155	116.65567
	4	32.437329	117.57594
	5	32.544161	115.03906

(continued on next page)

Table 1 (continued)

Size	Sampling rate	Average	
		PSNR	MSE
	6	32.127766	126.44677
	7	32.304155	119.03329
	8	32.020274	130.68298
	9	32.253072	119.57259
	10	31.592238	147.66887

Table 2  
Sampling rate according to the size of images

Size of input images	Sampling rate
Width or height ≤ 200 pixels	No sampling
200 pixels < width or height ≤ 400 pixels	Sub-sampled by a factor of 4
400 pixels < width or height	Sub-sampled by a factor of 8

Table 3  
An example of one-dimensional dynamic programming evaluation (a) frequency, (b) the zeroth-order moment cumulating, (c) frequency multiplied by index, and (d) the first-order moment cumulating

(a)											
Index		1	2	3	4	5	6	7	8	9	10
Frequency		4	0	4	3	7	8	4	3	2	2
(b)											
Index		1	2	3	4	5	6	7	8	9	10
Cumulative frequency	0	4	4	8	11	18	26	30	33	35	37
(c)											
Index		1	2	3	4	5	6	7	8	9	10
Frequency multiplied by index		4	0	12	12	35	48	28	24	18	20
(d)											
Index		1	2	3	4	5	6	7	8	9	10
Cumulative frequency multiplied by index	0	4	4	16	28	63	111	139	163	181	201

$$\text{Mean value of group 2} = \frac{201 - 63}{37 - 18} = 7.263.$$

The basic idea behind dynamic programming is the organization of work in order to avoid repetition of work already done. The first trick is to describe the solution to the entire problem in terms of solutions to smaller problems. This defines a recursive algorithm. The second trick is to see that the recursive algorithm tries to repeatedly compute the same subproblems over and over again, so storing them in a table can lead to an efficient algorithm.

After the three-dimensional cumulative distribution is constructed, we can obtain the centroid of each box by using the addition and subtraction as defined in Eq. (3).

$$\begin{aligned} \text{Frequency}[r_0 \text{ to } r_1, g_0 \text{ to } g_1, b_0 \text{ to } b_1] &= \text{Freq}[r_1, g_1, b_1] - \text{Freq}[r_0, g_1, b_1] \\ &\quad - \text{Freq}[r_1, g_0, b_1] - \text{Freq}[r_1, g_1, b_0] \\ &\quad - \text{Freq}[r_0, g_0, b_0] + \text{Freq}[r_0, g_0, b_1] \\ &\quad + \text{Freq}[r_0, g_1, b_0] + \text{Freq}[r_1, g_0, b_0], \end{aligned} \quad (3)$$

$$\begin{aligned} \text{Sum of Red}[r_0 \text{ to } r_1, g_0 \text{ to } g_1, b_0 \text{ to } b_1] &= \text{R}[r_1, g_1, b_1] - \text{R}[r_0, g_1, b_1] \\ &\quad - \text{R}[r_1, g_0, b_1] - \text{R}[r_1, g_1, b_0] \\ &\quad - \text{R}[r_0, g_0, b_0] + \text{R}[r_0, g_0, b_1] \\ &\quad + \text{R}[r_0, g_1, b_0] + \text{R}[r_1, g_0, b_0]. \end{aligned}$$

Therefore,

$$\text{MeanOfRed} = \frac{\text{Sum of Red}}{\text{Frequency}},$$

where R[] is the cumulative Red pixels to that point (first-order moment of Red axis). Freq[] is the cumulative frequency of that point (zeroth-order moment). Green and blue are calculated in the same way.

### 3.1.3. Cutting-plane positioning phase

The main difference among the proposed approaches is how to position the cutting plane, which gives the optimal result. Median-cut algorithm Heckbert, 1982 uses the method that provides the same number of each box. Variance-based algorithm Wan et al., 1990 tries to find the position, which keeps the minimization of variance of resulting sub-boxes. Wu Wu, 1992 proposed an improved method, which exploits the principal axis along which the image data has a maximum variance rather than normal to one of the three coordinate axes.

For our approach, we want to reduce the time consumption as much as possible while the PSNR, which we used as a criterion, should be more than 30-dB. In this section, we show that it does not need to use the multiple cutting planes to obtain the minimization of variance. We can put the cutting plane passing the centroid of the box to be perpendicular to the coordinate axis. Although, it does not obtain the best position, the quality is still acceptable as displayed in the results. However, it does not satisfy us in terms of time consuming, we propose a value representing variance instead of the actual variance which requires the high computational time. Because variance is not actually used in calculation, a simpler indicator of data scatterness would speed up the process. We make use of the dynamic programming and the mean value to create this value. First of all, we evaluate the centroid of one box, which encloses the colors of all pixels from the original image. After that we put the three planes to be perpendicular to red-axis, green-axis, and blue-axis. Consequently,

eight sub-boxes are obtained. Then each centroid of these sub-boxes is calculated. These centroids are used as the representatives of the data in each sub-box. Eq. (4) showed the value representing variance of the box.

$$\text{Value representing variance} = \sum_{i=1}^8 \|X_i - \bar{X}\|, \tag{4}$$

where  $\| \cdot \|$  stands for the Euclidean distance norm;  $X_i$  is the centroid of each sub-box;  $\bar{X}$  is the centroid of the box.

Instead of finding the actual variance using all points in that box, we require only 8 points representing the centroid of each box to calculate our value representing variance. Certainly, computational time can be much reduced.

Fig. 3 shows an example of two-dimensional clustering diagram. Although this value which is used to represent the actual variance cannot be equal to the actual variance, it can be used to point out how much the data scatter.

This formula is also performed to find the largest value representing variance of the available divided box. The next step is to find the axis to be perpendicular. Clearly, that axis, which gives us the minimization of the sum of MSE of both divided boxes, should be selected. This guarantees that the overall MSE is reduced automatically. We split this box at the centroid to be perpendicular to Red-axis first. The centroid of each sub-box is obtained by using Eq. (3) and then we evaluate the sum of the squared Euclidean distances between the centroid of both sub-boxes and the centroid of the box as defined in Eq. (5). We also do this calculation on Green-axis and Blue-axis. The cutting plane is perpendicular to the axis which has the greatest distance value.

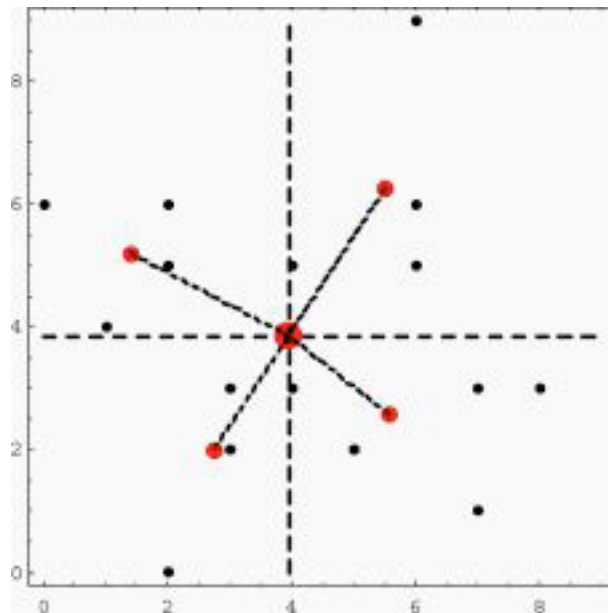


Fig. 3. Two-dimensional clustering diagram to illustrate how to find the value representing variance.

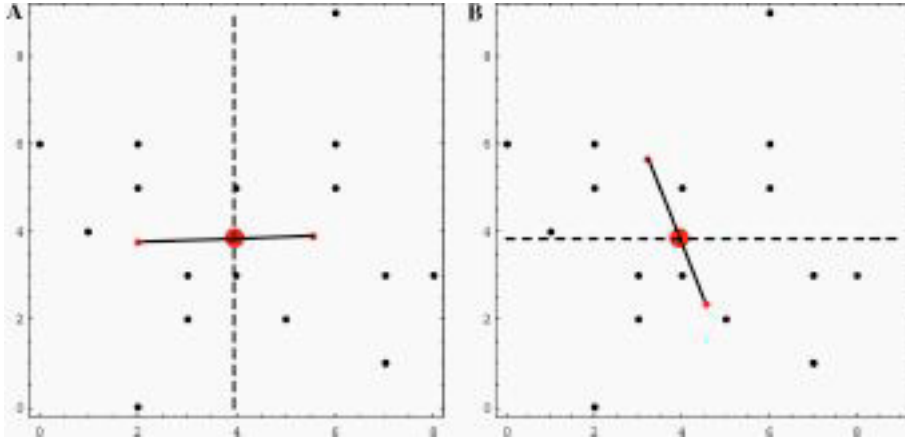


Fig. 4. Two-dimensional clustering diagram to illustrate how to find Squared Euclidean Distance when (A) the cutting plane is perpendicular to the  $x$ -axis and (B) the cutting plane is perpendicular to the  $y$ -axis; the cutting plane which creates the largest distance will be chose.

$$\begin{aligned} \text{Squared Euclidean Distance of Red} = & f_1(r_1 - \bar{r})^2 + f_1(g_1 - \bar{g})^2 + f_1(b_1 - \bar{b})^2 \\ & + f_2(r_2 - \bar{r})^2 + f_2(g_2 - \bar{g})^2 + f_2(b_2 - \bar{b})^2, \end{aligned} \tag{5}$$

where  $r_1, g_1,$  and  $b_1$  are the centroid point of the lower box;  $r_2, g_2,$  and  $b_2$  are the centroid point of the upper box;  $\bar{r}, \bar{g}, \bar{b}$  are the centroid point of the box; and  $f_1$  and  $f_2$  are the number of the pixels in the lower box and upper box respectively divided by  $(f_1 + f_2)$

The largest summation among three axes of the squared Euclidean distance means that the upper box and the lower box clusters are dissimilar as shown in Fig. 4. Therefore, it should be separated into two boxes orthogonally on this axis.

Fig. 5 shows an example of the automatic reduction of total MSE of a test image. This tree guarantees that the global MSE of the quantized image is reduced although this method cannot find the optimal cutting position. Certainly, Red-axis is selected because it produces less MSE than the others and its squared Euclidean distance is the largest.

### 3.1.4. Pixel mapping phase

This problem can be viewed as searching a nearest neighbor in three-dimensional spaces. There are some algorithms as mentioned above. For example, kd-tree is a data structure based on recursively subdividing a set of points with alternating axis-aligned hyperplanes. However, in this phase, we use the same algorithm as Wu’s quantizer (Wu, 1992), that is, the centroid mapping. The centroid mapping is very fast. It maps every color, which belong to that box, to its centroid. But it is effective when the number of colors in the original image is smaller than the number of pixels in the image. Since the pre-quantization to 15 bits is used, the number of colors is under 32768.

From Sections 3.1.1–3.1.4, the entire algorithms of our adaptive color quantization method can be programmed as the following pseudo-code.

```

3D-Color Histogram with sampling techniques //First phase
Zeroth-Order Moment Cumulative Distribution //Second phase
First-Order Moment Cumulative Distribution
No = 0; //Third phase: Cutting plane positioning
While (No < MaxColor)
{
    WhichBox = Largest value representing variance box among Boxes [No-1]
    Find MeanofRed, MeanofGreen, MeanofBlue of BoxesWhichBox]
    Red = Squared Euclidean Distance of Red //Red Axis
    Green = Squared Euclidean Distance of Green //Green Axis
    Blue = Squared Euclidean Distance of Blue //Blue Axis
    If Max(Red,Green,Blue) = Red
    {
        CuttingPoint = MeanofRed
        AXIS = RED
    }
    Else If Max(Red,Green,Blue) = Green
    {
        CuttingPoint = MeanofGreen
        AXIS = GREEN
    }
    Else
    {
        CuttingPoint = MeanofBlue
        AXIS = BLUE
    }
    Split Boxes[WhichBox] at CuttingPoint, to be perpendicular to AXIS
    Update Value representing variance of divided boxes
    No++
}
Centroid Mapping // Fourth phase: Pixel Mapping

```

### 3.2. Quality measures

Image quality assessment is an important and yet unsolved problem in image processing. In fact, there is no good objective criterion available for measuring the perceived image similarity. However the simplest and most widely used full reference quality metric is the mean squared error (MSE), computed by averaging the squared intensity differences of distorted and reference image pixels, along with the related quantity of peak signal-to-noise ratio (PSNR). These are appealing because they are simple to calculate, have clear physical meanings, and are mathematically convenient in the context of optimization.

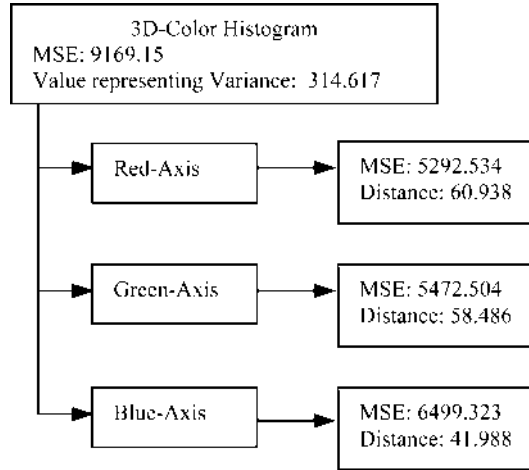


Fig. 5. Variance Reduction Tree on “Mandrill” standard image.

In this paper, we propose a new scoring system to indicate the performance of quantizers and comparison purpose, based on its time consumption and quantization quality (PSNR). 20 frames per second or 50 ms interval between each frame is selected to be the time target and PSNR is the quality criterion of the quantizer. As a rough rule of thumb provided by McGowan McGowan, 2000, an image with a PSNR of 25 dB is usually pretty poor. Anything below 25 dB is usually unacceptable. Perceived quality usually improved from 25 dB about to 30 dB. Above around 30 dB images look pretty good and are often indistinguishable form the uncompressed original image. Therefore, PSNR of greater than 30 dB is our quality criterion lower limit. Therefore a scoring system, based on both target variables, can be defined as Eq. (6).

$$\text{Score} = \frac{1}{1 + 10^{2[\ln(15) - \ln(x)]}} + \left( \frac{50}{\text{Total of time consuming (ms)}} \right) \left( \frac{\text{image size}}{640 \times 480} \right). \tag{6}$$

Table 4  
Public domain quantizers

Quantizer	Proposed by	Year
(1) Median-cut	P. Heckbert <a href="http://www.programmersheaven.com/d/click.aspx?ID=F15370">http://www.programmersheaven.com/d/click.aspx?ID=F15370</a>	1982
(2) Variance-based	S.J. Wan et al. <a href="http://www.programmersheaven.com/d/click.aspx?ID=F15366">http://www.programmersheaven.com/d/click.aspx?ID=F15366</a>	1988
(3) Octree	M. Gervautz and W. Purgathofer <a href="http://www.cs.ubc.ca/spider/ladic/quantize.html">http://www.cs.ubc.ca/spider/ladic/quantize.html</a>	1990
(4) Wu's Quantizer	X. Wu <a href="http://www.ece.mcmaster.ca/~xwu/cq.c">http://www.ece.mcmaster.ca/~xwu/cq.c</a>	1992
(5) NeuQuant	A. Dekker <a href="http://members.ozemail.com.au/~dekker/NEUQUANT.HTML">http://members.ozemail.com.au/~dekker/NEUQUANT.HTML</a>	1994

We propose to utilize the sigmoidal function to evaluate the quality due to its shape. If PSNR is much more than 30 dB, the first term approaches to one. This scoring system is derived from PSNR calculation, time consuming and image size.

Mean square error or MSE measures the average amount of difference between pixels of an image and its reconstructed image. If the MSE is small, the reconstructed image closely resembles the original. Below is the formula of the MSE calculation.

$$\text{MSE} = \frac{\sum_{x=1}^M \sum_{y=1}^N \|I(x, y) - I'(x, y)\|^2}{NM}, \quad (7)$$

Table 5

Performance of five downloaded quantizers, the uniform quantization and our quantizer on (a) “Mandrill” (512 × 512 pixels, 230,427 colors), (b) “Fish” (200 × 250 pixels, 37,515 colors), and (c) “Bird” (500 × 370 pixels, 108,331 colors), “Floating Market” (640 × 480 pixels, 178,201 colors)

24-bit images	Quatizers	Exe. Time (ms)	MSE	PSNR (dB)	Score
(1) Mandrill	Uniform	10	2056.02	18.6	5.00 <sup>a</sup>
	Median-Cut	55	696.02	23.3	1.66
	Variance-Based	53	178.02	29.22	1.76
	Wu’s Quantizer	35	118.63	30.98	2.18
	Octree	1027	196.4	28.79	0.99
	NeuQuant (SF = 1)	1274	116.44	31.07	1.00
	NeuQuant (SF = 30)	403	197.97	28.76	1.06
	Our Approach	20	169.22	29.44	3.09
(2) Fish	Uniform	5	1981.64	19.91	2.41 <sup>a</sup>
	Median-Cut	22	263.21	28.68	1.32
	Variance-Based	22	69.94	34.43	1.35
	Wu’s Quantizer	14	47.68	36.1	1.56
	Octree	245	77.42	33.99	1.01
	NeuQuant (SF = 1)	216	87.54	33.46	1.01
	NeuQuant (SF = 30)	48	160.83	30.82	1.13
	Our Approach	9	118.7	32.13	1.88
(3) Bird	Uniform	10	1900.16	20.08	3.80 <sup>a</sup>
	Median-Cut	41	331.26	27.67	1.68
	Variance-Based	39	85	33.57	1.75
	Wu’s Quantizer	26	64.04	34.8	2.14
	Octree	833	96.88	33.01	1.01
	NeuQuant (SF = 1)	814	65.92	34.68	1.02
	NeuQuant (SF = 30)	231	271.65	28.53	1.08
	Our Approach	16	99.65	32.88	2.86
(4) Floating Market	Uniform	10	1812.14	20.32	5.80 <sup>a</sup>
	Median-Cut	60	450.93	26.36	1.76
	Variance-Based	55	125.52	31.91	1.88
	Wu’s Quantizer	40	76.31	34.08	2.23
	Octree	1308	141.85	31.38	1.01
	NeuQuant (SF = 1)	1293	74.12	34.2	1.02
	NeuQuant (SF = 30)	290	135.07	31.6	1.14
	Our Approach	22	134.58	31.61	3.24

<sup>a</sup> Notes. Means that uniform quantizer clearly is unacceptable due to poor quality.

where  $I(x,y)$  is the original image;  $I'(x,y)$  is the approximated version;  $M,N$  is the dimension of the image  $\| \cdot \|$  stands for the euclidean distance norm.

Peak signal to noise ratio or PSNR measures the amount of useful data versus the amount of noise introduced into the image. Therefore, the higher the number, the more accurate the reconstruction. PSNR is a good measure for comparing restoration results for the same image, but between-image comparisons of PSNR are meaningless. Below is the formula of PSNR calculation.

$$\text{PSNR} = 10 \log \left( \frac{\text{Max(Original Image)}^2}{\text{MSE}} \right). \tag{8}$$

For example, in three-dimensional spaces (RGB),  $\text{Max(Original Image)}$  is equal to  $\sqrt{255^2 + 255^2 + 255^2}$  and PSNR is 30 dB and then

$$30 = 10 \log \left( \frac{\left( \sqrt{255^2 + 255^2 + 255^2} \right)^2}{\text{MSE}} \right), \tag{9}$$

$$\text{MSE} = \left( \frac{255^2 + 255^2 + 255^2}{1000} \right) = 195.075, \tag{10}$$

that is, PSNR of 30 dB is equal to MSE of 195.075.

#### 4. Experimental results

All quantizers are conducted on an Intel Pentium-4 2.4 GHz PC with 256 MB RAM, using Visual C++ 6.0 Compiler on Windows XP. The execution time, the

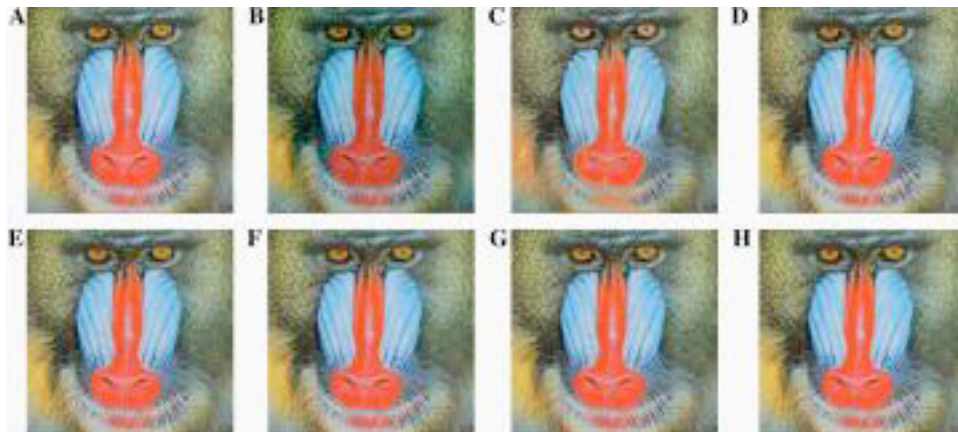


Fig. 6. (A) Original Mandrill image and its compressed versions, quantized to 256 colors by (B) uniform, (C) median-cut, (D) variance-based, (E) Octree, (F) Wu's quantizer, (G) NeuQuant (Sampling Factor = 1), and (H) our approach.

quantization error and our scoring system are adopted as the factors for evaluating the performance of the color quantization algorithms.

Five quantizers on Table 4 downloaded from the public domain for comparison purpose are adapted to conform to Visual C++ syntax. The timer starts at the pixel extraction and stops after completing pixel mapping phase. We present four test images in the different sizes. One is a standard image, i.e., “Mandrill,” sizing  $512 \times 512$  pixels and three more images with small size ( $200 \times 250$  pixels), medium

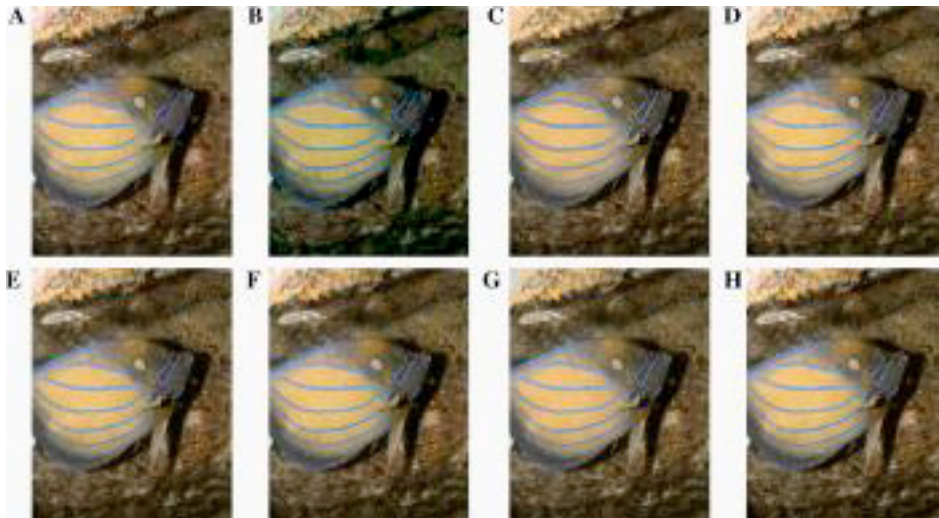


Fig. 7. (A) Original Fish image and its compressed versions, quantized to 256 colors by (B) uniform, (C) median-cut, (D) variance-based, (E) Octree, (F) Wu’s quantizer, (G) NeuQuant (sampling factor = 1), and (H) our approach.

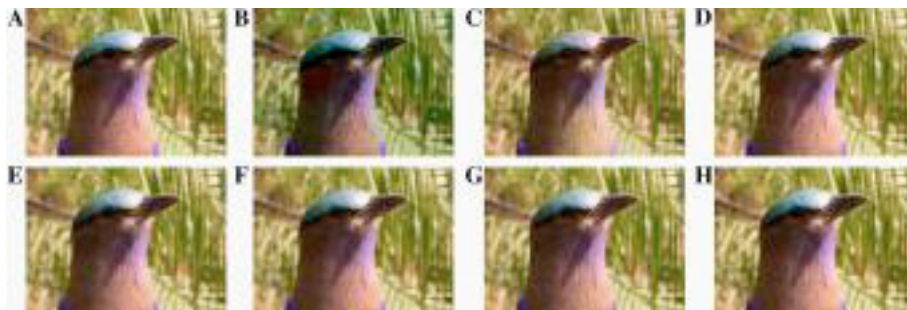


Fig. 8. (A) Original Bird image and its compressed versions, quantized to 256 colors by (B) uniform, (C) median-cut, (D) variance-based, (E) Octree, (F) Wu’s quantizer, (G) NeuQuant (sampling factor = 1), and (H) our approach.

size ( $500 \times 370$  pixels) and large size ( $640 \times 480$  pixels). We apply the sampling rate as defined in Table 2 to these test images on both horizontal and vertical directions separately. Table 5 shows the comparison of these five existing color quantization algorithms, the uniform quantization and our quantizer with respect to our scoring system, based on the computation time and the quantization quality. Each resulting image corresponding to the Table 5 is shown in Figs. 6–9. In addition, another small isolated-color image is brought to test in order to confirm that our algorithm can be applied to a small range of colors as shown on Fig. 10.

Next experiment, Table 6 presents the reduction of MSE after the number of representative colors is increased. Furthermore, to verify the efficiency of our quantizer, one hundred test images in the different sizes and contents are quantized by Wu’s



Fig. 9. (A) Original Floating Market image and its compressed versions, quantized to 256 colors by (B) uniform, (C) median-cut, (D) variance-based, (E) Octree, (F) Wu’s quantizer, (G) NeuQuant (sampling factor = 1), and (H) our approach.

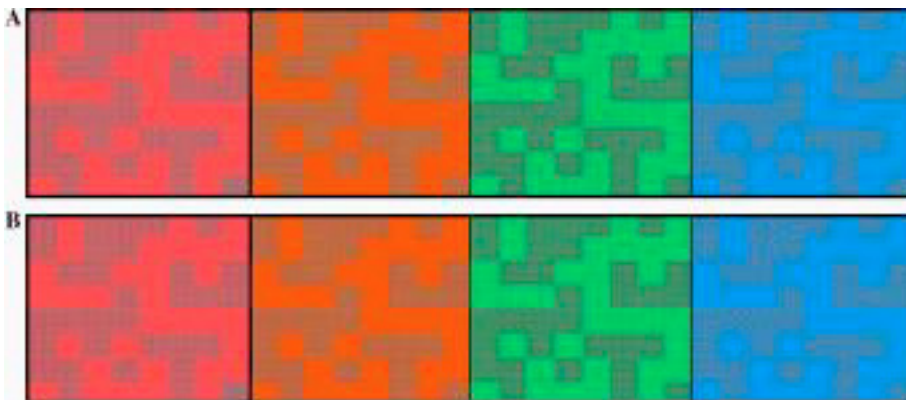


Fig. 10. (A) Original small isolated-color image with 35 colors and  $512 \times 110$  pixels, and (B) its quantized versions by our approach.

Table 6

MSE Reduction according to increasing the number of representative colors on “Mandrill” image, “Lena” image, “Peppers” images, “Airplane” image and “Floating Market” image

24 bit images	Number of colors	Proposed Algorithm		Median-cut Algorithm		Variance-based Algorithm		Octree Algorithm		Wu’s Algorithm		NeoQuant Algorithm (SF = 1)	
		MSE	Time (ms)	MSE	Time (ms)	MSE	Time (ms)	MSE	Time (ms)	MSE	Time (ms)	MSE	Time (ms)
Mandrill (512 × 512)	16	928	17	6751	44	878	38	1870	489	778	32	705	158
	32	587	18	4683	48	528	39	1094	657	468	34	420	249
	64	378	18	8163	51	386	42	545	779	288	34	272	413
	128	244	18	1364	52	266	46	315	898	187	35	165	688
	256	169	20	914	53	178	53	196	1007	119	35	116	1227
Lena (512 × 512)	16	348	16	10105	44	303	35	687	586	276	30	267	130
	32	213	17	3715	46	183	36	483	732	160	31	151	208
	64	140	17	2238	49	126	38	261	917	101	33	93	343
	128	99	18	2380	49	90	40	152	1009	65	33	55	592
	256	92	19	204	49	64	46	75	1186	42	34	39	1074
Peppers (512 × 512)	16	781	16	12007	43	577	35	1592	484	480	31	501	140
	32	394	16	2604	47	348	36	592	691	279	32	268	222
	64	267	17	7844	47	226	43	449	800	166	32	162	374
	128	171	18	3480	47	157	42	188	976	102	32	101	639
	256	112	19	1335	50	106	49	143	1068	66	33	74	1149
Airplane (512 × 512)	16	374	16	7207	44	285	35	394	673	158	30	223	123
	32	195	17	640	45	137	37	293	737	85	30	101	200
	64	146	19	2656	46	80	37	164	963	51	31	57	332
	128	100	18	1697	47	55	41	134	1064	33	32	30	587
	256	99	19	162	48	41	48	43	1271	22	33	24	1084
Floating Market (640 × 480)	16	728	18	2899	52	611	42	2886	561	564	37	498	171
	32	525	19	440	56	396	44	695	842	327	38	321	266
	64	316	20	1938	58	264	46	395	1021	202	39	194	439
	128	226	20	858	59	182	49	234	1129	123	39	106	729
	256	135	22	585	61	126	57	142	1290	76	40	74	1319

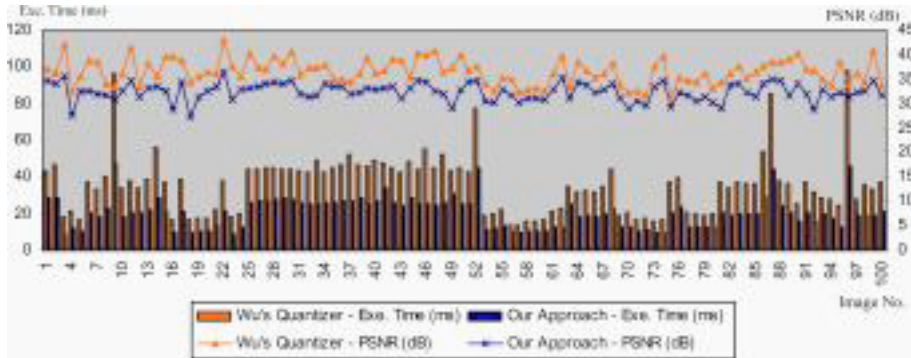


Fig. 11. Comparison between Wu's quantizer and our quantizer based on the execution time and PSNR on one hundred of test images.

quantizer and our quantizer. Fig. 11 shows the performance of both quantizers, based on our PSNR and the execution time respectively on these one hundred test images.

## 5. Conclusion and future work

We adapt and create some techniques to speed up the process. These are sampling technique on an RGB color space, putting the cutting plane perpendicular to the axis on which the sum of the squared Euclidean distances between the centroid of sub-boxes and the centroid of the box is the greatest, and employing the value representing variance rather than actual variance. Our new approach is implemented into our data compression algorithms for a True Color Graphical Signboard. It reduces execution time significantly compared to any other existing algorithms. Furthermore our algorithm is efficient enough to be adopted for color quantization on the various sizes of images. Not only our system but also other applications, which require fast execution time and ability to display a full-color image using limited number of colors at a time, can exploit our algorithms. However, some techniques such as lossy compression and lossless compression on both intra-frame and inter-frame coding of image sequences will be further investigated to reduce the processing time and to fulfill our requirement in the future work.

## References

- Balasubramanian, R., Allebach, J.P., 1991. A new approach to palette selection for color images. *J. Imag. Technol.* 17 (6), 284–290.
- Balasubramanian, R., Allebach, J.P., Bouman, C., 1995. Sequential scalar quantization of vectors: an analysis. *IEEE Trans. Image Process.* 4, 1282–1295.
- Bentley, J.L., 1975. Multidimensional binary search trees used for associative searching. *Commun. ACM* 18, 509–517.

- Dekker, A., 1994. Kohonen neural networks for optimal colour quantization. *Network: Comput. Neural Syst.* 5, 351–367.
- Gervautz, M., Purgathofer, W., 1990. A Simple Method for Color Quantization: Octree Quantization, *Graphic Gems*. Academic Press, New York, 287–293.
- Heckbert, P., 1982. Color image quantization for frame buffer displays. *Comp. Graph.* 16, 297–307.
- Kohonen, T., 1989. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin.
- Lim, Y.W., Lee, S.U., 1990. On the color image segmentation algorithm based on the thresholding and the fuzzy c-means techniques. *Pattern Recognit.* 23, 935–952.
- Linde, Y., Buzo, A., Gray, R., 1980. An algorithm for vector quantizer design. *IEEE Trans. Commun.* 28, 84–95.
- Liu, G.L., 1968. *Introduction to Combination Mathematics*. McGraw-Hill, New York.
- Lloyd, S.P., 1982. Least squares quantization in PCM. *IEEE Trans. Inform. Theory* 28, 129–137.
- McGowan, J.F., 2000. John McGowan's AVI Overview. Available from: <<http://www.jmcgowan.com>>.
- Orchard, M., Bouman, C., 1991. Color quantization of images. *IEEE Trans. Signal. Proc.* 39, 2677–2690.
- Papamarkos, N., Atsalakis, A., Strouthopoulos, C., 2002. adaptive color reduction. *IEEE Syst. Man Cybernet.—Part B* 32, 44–56.
- Sangwine, S., Horne, R., 1998. *The Colour Image Processing Handbook*. Chapman and Hall, London.
- Scheunder, P., 1997. A genetic C-means clustering algorithm applied to color image quantization, *Pattern Recognition* 30, 859–866.
- Scheunders, P., 1997. A comparison of clustering algorithms applied to color image quantization, *Pattern Recognit. Lett.*, 18, 1379–1384.
- Sirisathitkula, Y., Auwatanamongkola, S., Uyyanonvara, B., 2004. Color image quantization using adjacent colors' line segments. *J. Pattern Recognit. Lett.* 25 (9), 1025–1043.
- Verevka, O., Buchanan, J.W., 1995. Local k-means algorithm for color image quantization. *Proc. Graph. Interface*, 128–135.
- Wan, S.J., Prusinkiewicz, P., Wong, S.K.M., 1990. Variance-based color image quantization for frame buffer display. *Color Res. Appl.* 15, 52–58.
- Wu, X., 1992. Color quantization by dynamic programming and principal analysis. *ACM Trans. Graph.* 11, 348–372.
- Zeng, W., Huang, Y.F., Huang, S.C., 1995. Two greedy tree growing algorithms for designing variable rate vector quantizers. *IEEE Trans. Cir. Syst. Video Technol.* 5, 236–242.