Accelerated k-means clustering algorithm for colour image quantization

Article *in* Imaging Science Journal The · February 2008

DOI: 10.1179/174313107X176298

CITATIONS
48

READS
7,429

2 authors, including:

Yu-Chen Hu
Providence University, TaiChung City, Taiwan
128 PUBLICATIONS **2,323** CITATIONS

SEE PROFILE

Accelerated *k*-means clustering algorithm for colour image quantization

Y-C Hu* and **B-H Su**
Department of Computer Science and Information Management, Providence University, Taichung 433, Taiwan

Abstract: The *k*-means clustering algorithm is a commonly used algorithm for palette design. If an adequate initial palette is selected, a good quality reconstructed image of a compressed colour image can be achieved. The major problem is that a great deal of computational cost is consumed. To accelerate the *k*-means clustering algorithm, two test conditions are employed in the proposed algorithm. From the experimental results, it is found that the proposed algorithm significantly cuts down the computational cost of the *k*-means clustering algorithm without incurring any extra distortion.

Keywords: colour image quantization, *k*-means clustering algorithm, vector quantization, LBG algorithm

1 INTRODUCTION

Colour image quantization (CIQ) is an important technique[1] because of the limitations of displaying colour images, data communication and storage. Generally, CIQ consists of three procedures: palette design, image encoding and image decoding. The goal of the palette design procedure is to generate a set of representative colours used in image encoding/decoding. The set of representative colours is also called the colour palette.

The image encoding procedure is also called the pixel mapping procedure. The closest colour in the palette for each colour pixel in the colour image is decided. The closest colour in the palette is the one with the minimal squared Euclidean distance to the input colour pixel. The index of the closest colour of each colour pixel is then recorded. The compressed codes of the colour image are the indices of the closest colours in the palette.

The MS was accepted for publication on 4 October 2006.
* *Corresponding author: Dr Hu, Yu-Chen, Department of Computer Science and Information Management, Providence University, Taichung 433, Taiwan; e-mail: ychu@pu.edu.tw*

In the image decoding procedure, the same palette as used in the image encoding procedure is stored and used. Each received index is used to retrieve its corresponding colour in the palette. By repeatedly recovering every compressed colour pixel by its closest colour pixel in the palette, the compressed colour image is generated. The image decoding procedure is very simple and requires little computational cost.

The palette design procedure is very important in CIQ. If a set of representative colours is generated, a good reconstructed image quality of compressed colour image is obtained. Until now, several palette design methods have been proposed. They can be classified into two categories: splitting methods and clustering methods.

Splitting algorithms such as the median cut method,[2] the mean cut method,[3] the variance-based colour quantization method,[4] the radius weighted mean cut (RWM-cut) method,[5] and the reduction of colour space dimensionality (RCSD) method[6] divide the RGB colour space into disjoined regions. Then a representative colour is chosen to represent each region. The major difference between these proposed splitting methods is that different splitting rules are used to preserve different properties of the colour image.

Another category of palette design methods performs clustering of the colour pixels in the given colour image. The k-means algorithm[7] is the most commonly used algorithm in this category. It was originally proposed for pattern recognition. The k-means clustering algorithm is a sub-optimal quantization approach that is highly affected by the initial patterns used. To solve this problem, a genetic k-means clustering algorithm[8] has been proposed. The major drawback of this method is that it requires a great deal of computational cost. To reduce the computational cost of the palette design process, a reduction of RGB colour space[9] has been proposed. In addition, the fuzzy clustering colour quantization algorithm[10] has been introduced.

Generally speaking, the clustering algorithms provide better performance of colour palettes than that of the splitting algorithms. The clustering algorithms usually consume more computational cost than do the splitting algorithms. Therefore, research towards cutting down the computational cost of clustering algorithms is very important. The rest of the paper is organized as follows. In section 2, the newly proposed scheme is introduced. The experimental results are listed in section 3, to show whether the proposed scheme indeed cuts down the computational cost of the k-means algorithm. Finally, some discussions and conclusionsare given in sections 4 and 5.

2 PROPOSED SCHEME

The goal of the proposed algorithm is to accelerate the k-means clustering algorithm for palette design. The k-means algorithm is a commonly used algorithm for palette design. If a good initial palette is selected, the palettes designed by the k-means algorithm provide a better representation than those in the splitting methods.

Based on the study of vector quantization (VQ),[11,12] it is found that VQ and CIQ have the same encoding/decoding structures. The commonly used method for codebook design in VQ, i.e. the LBG algorithm,[11] is equivalent to the k-mean algorithm. The main difference between VQ and CIQ is that an image block of $n \times n$ pixels and a colour pixel of three dimensions are processed, respectively. The study of the fast algorithms in references [13–17] for VQ codebook design can help to speed up the k-means clustering algorithm for CIQ palette design.

The following section briefly describes some fast algorithms for VQ used in the proposed scheme. They are originally used to find the closest codeword in the codebook for each image block of $n \times n$ pixels. To employ them to accelerate the pixel mapping procedure of CIQ, the vector dimensions need to be changed from $n \times n$ to 3.

In 1985, Bei and Gray[13] proposed the partial distance search (PDS) technique to speed up the codebook search algorithm of VQ. The idea of PDS is that, if the partially calculated squared Euclidean distance between the image vector and the checked codeword is greater than or equal to the minimal squared Euclidean distance found so far, the checked codeword cannot be the closest codeword for the input image vector.

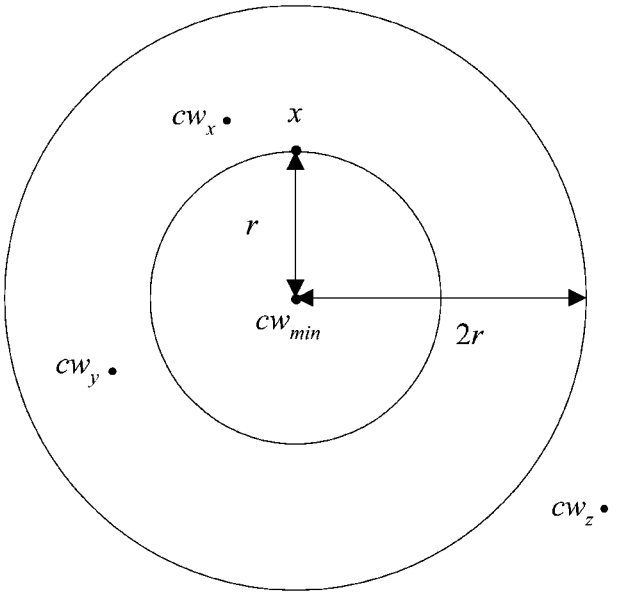
In 1990, Huang and Chen[14] proposed the fast search algorithm for VQ based on the triangular inequality (TIE). TIE is a basic concept in geometric analysis. It can be incorporated into the codebook search process of VQ. The triangular inequality rule for the squared Euclidean distance can be described as

$$\sqrt{d(cw,cw_{\min})} \leqslant \sqrt{d(x,cw)} + \sqrt{d(x,cw_{\min})} \qquad (1)$$

Suppose that $cw_{\min}$ is the closest codeword of the current image vector x found so far. Any codeword cw which satisfies $2\sqrt{d(x,cw_{\min})} < \sqrt{d(cw,cw_{\min})}$ is safely rejected because the result $d(x, cw_{\min}) < d(x, cw)$ can be induced by substituting the above inequality. In order to find the distance between each pair of codewords, the TIE method needs to store a distance table that records the squared Euclidean distances of all pairs of codewords in the codebook.

An example of the TIE method is given in Fig. 1. The codewords cw_x and cw_y are within that circle centred at $cw_{\min}$ with the radius equal to $2\sqrt{d(x,cw_{\min})}$. By contrast, the codeword cw_z is outside the circle centred at $cw_{\min}$ with the radius equal to $2\sqrt{d(x,cw_{\min})}$. In this example, only cw_x and cw_y pass the TIE test. Two codewords are the possible candidates of the closest codeword for x.

Some additional memory space is required in the TIE method. The selection of the initial searched candidate also has a great influence on the performance of the TIE method. If a tighter minimal squared Euclidean distance can be early found in the TIE method, less computational cost is required to find the closest codewords in the codebook for the image blocks.



1 Example of triangular inequality rule

In 1993, Ra and Kim[15] proposed the mean-distance-ordered partial codebook search (MPS) algorithm. It was proposed to accelerate the image encoding procedure of VQ using a codebook that was previously sorted by the mean values of the codewords of $n \times n$ pixels. One test condition exploring the relationship between the squared Euclidean distance (SED) and the squared sum distance (SSD) is introduced. The SED between the image vector x and the codeword cw in the codebook is defined as follows

$$d_{\mathrm{E}}(x,cw) = \sum_{\mathrm{j}=1}^{\mathrm{n}\times\mathrm{n}} \left(x_{\mathrm{j}} - cw_{\mathrm{j}}\right)^2 \tag{2}$$

In addition, the SSD between the image vector x and the codeword cw in the codebook is defined as follows

$$d_{\mathrm{S}}(x,cw) = \left(\sum_{\mathrm{j}=1}^{\mathrm{n}\times\mathrm{n}} x_{\mathrm{j}} - \sum_{\mathrm{j}=1}^{\mathrm{n}\times\mathrm{n}} cw_{\mathrm{j}}\right)^2 \tag{3}$$

The inequality exploiting the relationship between the SED and SSD for the image block x and the codeword cw is described as follows

$$d_{\mathrm{S}}(x,cw) \leqslant (n \times \mathrm{n}) \times d_{\mathrm{E}}(x,cw) \tag{4}$$

The MPS test condition that makes use of the derived relationship in equation (5) can be used to reject some impossible codewords in the codebook.

$$(n \times \mathrm{n}) \times d_{\min} < d_{\mathrm{S}}(x,cw) \tag{5}$$

Here $d_{\min}$ denotes the minimal squared Euclidean distance between the input vector x and the closest codeword $cw_{\min}$ searched for so far. If the squared sum distance between the input vector x and the codeword cw to be searched is greater than $n \times n$ times the value of $d_{\min}$, the codeword cw cannot be the closest codeword, because the inequality $d_{\min} < d_{\mathrm{E}}(x,cw)$ can be derived from equations (4) and (5). In other words, codeword cw can be safely rejected without actually computing the squared Euclidean distance between codeword cw and input vector x. Using the MPS test condition described in equation (5), some impossible codewords can be safely rejected to cut down the computational cost.

In MPS, the initially searched codeword in the codebook for each input vector x is the codeword with the closest mean value to x. That is because the authors claimed that the closest codeword in the codebook for the input vector x typically had a very close mean value to x. The binary search technique is employed in MPS to find the initially searched codeword in the codebook.

In the proposed scheme, the test conditions of TIE and MPS are used to accelerate the k-means algorithm for palette design. In addition, the PDS technique is employed to accelerate the closest colour searching process. The colour palette used in the proposed scheme is sorted previously by the mean values of colours in the palette. The distance table that records the squared Euclidean distance between all possible pairs of colours is calculated and stored. The detailed descriptions of the proposed algorithm are given below.

3 PROPOSED ALGORITHM

Step 1. Generate the initial palette of k colours.

Step 2. Sort the palette by the mean values of colours. Calculate the distance table for TIE.

Step 3. Perform the clustering process for each colour pixel cp.

Step 3.1. Determine the initial searched colour c_{init} in the palette for cp.

Step 3.2. Calculate the squared Euclidean distance between cp and c_{init}. The calculated distance is stored in $d_{\min}$.

Step 3.3. Sequentially check the other colours in the palette to find the closest colour for cp. Here, MPS and TIE techniques are sequentially employed to speed up the searching process. In addition, the PDS technique is used to rejected impossible colours in the palette during the

calculation of squared Euclidean distances between *cp* and some palette colours.

Step 4. Compute the mean vector of the colour pixels belonging to each group. All these mean vectors in these k groups form the new colour palette.

Step 5. If the difference between previous palette and the new palette is small, then the algorithm stops. Otherwise, go to Step 2.

To generate the initial palette in Step 1, one can randomly select k colours from the colour image or use splitting methods, such as the median cut method, mean cut method, the VBA method, and so on, to generate an initial palette. The initial palette is then sorted by the mean values of colours, because the test condition derived in MPS is used in the proposed scheme. The distance table needed for TIE is then calculated in Step 2.

In Step 3.1, the initial searched colour in the palette is determined for each colour pixel to be compressed. Since the colours in the palette are previously sorted by their mean values, the binary searching technique that was used in the MPS algorithm can also be used to determine the initial searched colour in the palette. Using the binary searching technique, the time complexity is about $O(\log k)$ for finding the initial searched colour in a palette of k colours.

To speed up the initial colour selection process in Step 3.1, a table lookup mechanism was designed by constructing an array recording the indices of the relatively closest palette colour of each possible average component value ranging from 0 to 255. The initial searched colour for each colour pixel can be obtained by simply calculating the average component value of the input colour pixel and then following a simple array access operation. The detailed descriptions of constructing the array are given below.

Step 1. Let k denote the palette size, and *Index* be an array sized 256.

Step 2. Compute the average component value acv_{j} of the jth colour in the palette, where $1 \leqslant j \leqslant k$.

Step 3. Compute the *Index* array by the following three rules:

$$\mathrm{Index[i]} \begin{cases} = 1, \text{ if } 0 \leqslant i < (acv_1 + acv_2)/2 \\ = j, \text{ if} (acv_{\mathrm{j}-1} + acv_{\mathrm{j}})/2 \leqslant \mathrm{i} < (acv_{\mathrm{j}} + acv_{\mathrm{j}+1})/2 \\ \text{and } 2 \leqslant j \leqslant k-1 \\ = k, \text{ if} (acv_{\mathrm{k}-1} + (acv_{\mathrm{k}})/2 \leqslant \mathrm{i} < 256 \end{cases}$$

In Step 3.2, the squared Euclidean distance between the colour pixel and the predicted initial searched colour c_{init} is calculated and stored in $d_{\min}$. After the initial searched colour in the palette for each colour pixel has been decided, the MPS and TIE test conditions are sequentially checked to filter out impossible palette colours in Step 3.3. That is, the TIE test condition is used to improve the performance of the MPS test condition further. In addition, the PDS technique is used to accelerate the closest codeword searching process.

The searching order of the colours in the palette has a great influence on the performance of the proposed scheme. If a much closer palette colour can be searched earlier, more impossible candidates are rejected in the proposed scheme. In Step 3.3, a simple rule is used to determine the next colour to be examined. The one with the closest mean difference value to the input colour pixel is selected. In other words, the searching order of the colours in the palette in the proposed scheme is different from that in the MPS technique.

In Step 4, the mean values of these k groups are calculated to form the new palette. If the difference of the incurred image distortions in two consecutive iterations is small, the algorithm stops. The following equation is used to determine whether the algorithm is to be terminated.

$$\frac{|SED_{\mathrm{current}} - SED_{\mathrm{previous}}|}{SED_{\mathrm{current}}} \leqslant TH \tag{6}$$

Here, SED_{current} and SED_{previous} denote the squared Euclidean distance incurred in the current iteration and the previous iteration, respectively. According to the above descriptions, it is found that the k-means algorithm requires a great deal of computational cost, because several iterations are executed for palette design.

Once the palette of k colours is generated, the image encoding procedure is also finished. Remember that a palette of k colours is designed using the input colour image only as the training image; all pixels in this image are classified into k groups in each iteration of the k-means algorithm. By using the grouping information in the last iteration of the k-means algorithm, the corresponding index of each colour pixel is taken to encode it. Finally, the compressed codes consist of the colour palette, and the indices are transmitted to the decoder.

4 SIMULATION RESULTS

A variety of simulations were performed to verify the efficiency of the accelerated k-means algorithm. All

2 Four 512×512 pixel testing images: (a) 'Airplane'; (b) 'Lena'; (c) 'Peppers'; (d) 'Splash'

these experiments were performed on a PC with an Intel Pentium 4 2.4 GHz CPU and 512 MB RAM. The operating system was Red Hat Linux 9.0, and the ANSI C programming language was used. For any $M \times N$ colour image, the mean squared error (MSE) between the original image and the reconstructed image is defined as

$$MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} \sum_{h=1}^{3} \left(x_{ijh} - x'_{ijh} \right)^2 \tag{7}$$

Here x_{ij} and x'_{ij} denote the original and the recovered colour pixels of three dimensions, respectively. Besides, the quality of the reconstructed image is measured by means of the peak signal-to-noise-ratio (PSNR), which is defined as

$$PSNR = 10 \times \log_{10}\left(255^2 / MSE\right) \tag{8}$$

Note that MSE is a commonly used measurement for evaluating the image quality. Some other measurements[18–20] based on the human visual system (HVS) have been proposed to measure the degree of similarity between two given images.

In the simulations, five palettes sized 16, 32, 64, 128 and 256 were designed. Four 512×512 pixel colour image were used to measure the performance of the proposed scheme. These testing images are listed in Fig. 2.

To verify that the *k*-means algorithm indeed provides better image qualities than the splitting methods, the comparative results of the image qualities of the median cut method,[2] the mean cut method[3] and the RCSD method[6] are given in Table 1. These three methods belong to the splitting-based palette design algorithms. According to the results, the mean cut method achieves the best image quality.

The experimental results of the image qualities and the number of iterations of the *k*-means algorithm using different termination threshold values are listed in Table 2. Five threshold values 0.1, 0.01, 0.001, 0.0001 and 0.00001 are used in the simulations. The results indicate that more rounds are executed when a smaller threshold value is used in the *k*-means algorithm. In addition, more rounds are executed in the *k*-means algorithm to design a colour palette of a larger size when the threshold value used is fixed.

According to the results in Table 2 the image qualities do not strictly increase with the decrease in threshold values, but more rounds are needed in the *k*-means algorithm to design the colour palette. To balance the image quality and the required computational cost of the *k*-means algorithm, it is suggested that the control threshold value for the termination of the *k*-means algorithm used in the following simulations should be 0.001.

In the *k*-means algorithm, the selection of initial colours in the palette plays an important role in the performance of the designed colour palette. In the simulations, the colours in the initial palette were selected from the diagonal line of the colour image, with the limitation that colour pixels with the same values appear only once. In fact, some additional simulations have been performed to evaluate the performance of the proposed scheme using some other initialization methods such as random selection, the RCSD method and so on. It was found that the improvements of the proposed scheme compared with the conventional *k*-means algorithm using different initialization methods are quite similar. To generate the initial colour palette efficiently, the palette initialization method described above is used in the simulations.

From the results in Tables 1 and 2, it is found that the *k*-means algorithm achieves better image qualities than the splitting-based methods. However, the mean cut method and the RCSD method consume less computational cost than does the *k*-means algorithm. The median cut method consumes approximately the same computational time as the *k*-means algorithm, because the search for the median value is time consuming.

The experimental results of the execution times of the *k*-means algorithm with the full search algorithm, the *k*-means algorithm with PDS technique, the *k*-means algorithm with the TIE technique, the *k*-means algorithm with the MPS technique and the proposed scheme are given in Table 3. In these comparative methods, the look-up table that records the squared values ranging from 0^2, 1^2, 2^2, …, 255^2 is employed to avoid the multiplication operation needed in calculating the squared Euclidean distance.

Table 1 Results of image qualities of splitting-based palette design algorithms with different palette sizes

Palette sizes	Images	Median cut method[2]	Mean cut method[3]	RCSD method[6]
16	Airplane	27.053	29.882	26.893
	Lena	28.272	28.910	28.117
	Pepper	25.781	25.971	23.729
	Splash	25.583	27.226	27.229
	Average	**26.672**	**27.997**	**26.492**
32	Airplane	29.322	31.883	28.041
	Lena	30.690	31.331	30.608
	Pepper	27.520	28.096	25.735
	Splash	29.353	31.090	28.475
	Average	**29.221**	**30.600**	**28.215**
64	Airplane	31.757	35.384	31.374
	Lena	32.967	33.509	32.576
	Pepper	29.274	30.381	28.644
	Splash	31.758	33.950	30.483
	Average	**31.439**	**33.306**	**30.769**
128	Airplane	33.364	37.758	34.526
	Lena	35.074	35.495	34.116
	Pepper	30.988	32.618	31.177
	Splash	34.392	36.513	33.475
	Average	**33.455**	**35.596**	**33.324**
256	Airplane	35.956	39.772	36.557
	Lena	36.998	37.346	35.936
	Pepper	33.003	34.700	33.385
	Splash	36.337	38.960	35.924
	Average	**35.574**	**37.695**	**35.451**

Table 2 Results of image qualities (PSNR) and required iteration number (INO) of proposed scheme using different control threshold values

		Values									
		0.1		0.01		0.001		0.0001		0.00001	
Palette sizes	Images	PSNR	INO	PSNR	INO	PSNR	INO	PSNR	INO	PSNR	INO
16	Airplane	30.182	6	30.684	11	31.393	35	31.390	35	31.866	105
	Lena	28.701	3	29.392	10	29.443	14	29.452	17	29.452	17
	Pepper	25.948	3	26.455	9	26.628	17	26.661	33	26.661	33
	Splash	29.066	4	29.277	7	29.448	17	29.449	19	29.450	23
	Average	**28.474**	**4.00**	**28.952**	**9.25**	**29.228**	**20.75**	**29.238**	**26.00**	**29.357**	**44.50**
32	Airplane	32.310	4	32.663	10	34.697	55	34.709	72	34.519	144
	Lena	31.282	4	31.784	9	31.857	16	31.855	21	31.815	49
	Pepper	28.191	3	28.513	7	29.074	26	29.075	29	29.069	50
	Splash	31.030	3	31.501	8	32.306	23	32.543	50	32.540	53
	Average	**30.703**	**3.50**	**31.115**	**8.50**	**31.984**	**30.00**	**32.046**	**43.00**	**31.986**	**74.00**
64	Airplane	34.586	4	36.346	10	36.647	31	36.761	49	36.242	187
	Lena	33.241	4	33.628	10	33.706	16	33.677	38	33.381	154
	Pepper	31.212	6	31.306	8	31.500	22	31.501	24	31.493	34
	Splash	34.140	4	35.045	14	35.068	18	35.132	38	35.132	38
	Average	**33.295**	**4.50**	**34.081**	**10.50**	**34.230**	**21.75**	**34.268**	**37.25**	**34.062**	**103.20**
128	Airplane	37.421	5	38.197	11	38.641	23	38.604	30	37.237	334
	Lena	35.002	4	35.535	10	35.580	14	35.588	19	35.592	24
	Pepper	33.044	5	33.367	10	35.508	20	33.513	24	33.330	117
	Splash	36.811	5	37.791	14	37.392	22	37.918	32	37.918	32
	Average	**35.570**	**4.75**	**36.223**	**11.25**	**36.780**	**19.75**	**36.406**	**26.25**	**36.019**	**126.75**
256	Airplane	39.189	5	40.152	14	40.237	19	40.437	35	39.972	77
	Lena	36.628	3	37.214	9	37.384	23	36.344	175	34.731	562
	Pepper	34.723	4	35.243	10	35.450	26	34.924	166	34.898	192
	Splash	38.478	5	39.771	14	40.008	25	40.008	25	37.813	429
	Average	**37.255**	**4.250**	**38.095**	**11.75**	**38.270**	**23.25**	**37.928**	**100.25**	**36.854**	**315.00**

Table 3 Results of execution times (unit: seconds) for original *k*-means algorithm, *k*-means algorithm with PDS, TIE and MPS, and proposed scheme for palette design

		k-means Algorithm				
Palette sizes	Images	Full search	PDS[13]	TIE[14]	MPS[15]	Proposed scheme
16	Airplane	16.000	11.233	10.317	8.900	6.217
	Lena	6.783	4.967	4.333	3.917	2.900
	Pepper	8.417	6.500	5.367	4.717	3.450
	Splash	8.117	5.783	4.917	4.317	3.200
	Average	**9.829**	**7.121**	**6.234**	**5.463**	**3.942**
32	Airplane	45.383	29.883	26.183	19.750	12.083
	Lena	14.150	9.850	7.900	6.033	4.033
	Pepper	23.967	17.067	13.017	9.517	6.217
	Splash	20.600	13.983	10.700	7.833	5.100
	Average	**26.025**	**17.696**	**14.450**	**10.783**	**6.858**
64	Airplane	48.167	30.117	15.400	9.350	8.733
	Lena	27.117	17.517	13.150	8.117	4.950
	Pepper	40.283	25.950	18.283	11.100	6.700
	Splash	31.483	19.550	14.033	8.000	4.617
	Average	**24.582**	**15.999**	**10.851**	**7.300**	**5.181**
128	Airplane	71.700	42.083	33.383	16.217	8.583
	Lena	47.567	28.433	21.683	10.350	5.267
	Pepper	72.583	44.333	29.033	14.550	7.917
	Splash	76.867	45.250	29.233	13.183	7.033
	Average	**67.179**	**40.025**	**28.333**	**13.575**	**7.200**
256	Airplane	114.983	67.117	98.333	39.567	10.017
	Lena	153.317	90.533	60.417	23.017	11.050
	Pepper	184.650	109.050	73.183	28.700	13.467
	Splash	171.567	97.967	69.300	24.533	10.133
	Average	**156.129**	**91.167**	**75.308**	**28.954**	**11.167**

The average execution times of these schemes with different palette sizes are listed. The colour palettes used in the simulations were previously sorted by the mean values of colours. The binary searching technique was employed to determine the initial searched colour in the palette for the *k*-means algorithm with the TIE technique and the *k*-means algorithm with the MPS technique. According to the results, the proposed scheme significantly cuts down the computational cost of the *k*-means algorithm when a large palette is used for CIQ. An average 92.848% reduction in execution time is achieved when the palette of 256 colours is used. In addition, it consumes 10.718%, 21.076% and 26.353% of the execution times of the *k*-means algorithm when palettes of 128, 64 and 32 are used, respectively. The proposed scheme consumes 40.102% of the execution time of the *k*-means algorithm when the palette of 16 colours is used. It is obvious that the time reduction obtained using the proposed scheme decreases when a small-sized palette is used.

To explain why the proposed scheme provides a better performance than the comparative schemes, the experimental results of the average number of colours examined in the palette are listed in Table 4. In the *k*-means algorithm with the full search approach, the number of colours examined equals the size of the palette used. The *k*-means algorithm with the PDS technique also needs to examine all the colours in the palette. In PDS, the square Euclidean distances between some candidate colours and the input colours are partially calculated so that the execution time is reduced. In addition, the *k*-means algorithm with the TIE technique needs to exam each colour in the palette. The calculation of the squared Euclidean distance is avoided if it cannot pass the TIE test condition.

In Table 4, average numbers of colours examined in the *k*-means algorithm with the MPS technique and the proposed scheme are not equal to the palette sizes used. This is because the MPS test condition has the following properties. When a palette colour with a mean value less than the input colour pixel is rejected using the MPS test condition, those palette colours with mean values less that the rejected colour are also rejected. Similarly, once a palette colour with a mean value greater than the input colour pixel is rejected using the MPS test condition, those palette colours with mean values greater than the rejected colour are also rejected.

The average number of colours examined in the proposed scheme is less than that in the *k*-means algorithm with the MPS technique in Table 4. That is because the order of searching colours in the proposed scheme can find more suitable candidates faster than the search order of the MPS technique.

Table 5 lists the results of the average numbers of colours used in calculating the whole squared Euclidean distance of these comparative schemes. In addition, the average numbers of components used in PDS and the proposed scheme are also listed. According to the results in Table 3, PDS consumes more execution time than does TIE. In Table 5, the number of colours used in PDS is less than that in TIE. However, some colours are rejected early during the closest colour searching process in PDS.

As the palette size is incremented, the percentage of colours used in each comparative scheme decreases. For example, 10.41%, 4.06%, 1.67% of colours in the palettes of 16, 64 and 256 colours, respectively, are needed in the *k*-means algorithm with PDS. In addition, 9.61%, 3.47% and 1.28% of colours in the palettes of 16, 64, and 256 colours, respectively, are needed in the proposed scheme. In Table 5, 2.49% and 3.96% of colours in the palette sized 256 are used to calculate the whole squared Euclidean distance in the *k*-means algorithm with TIE and MPS, respectively. An average of 1.28% of colours in the palette of 256 colours are taken to calculate the squared

Table 4 Results of average number of examined colours of full search (FS) *k*-means algorithm, *k*-means algorithm with PDS, TIE and MPS, and proposed scheme

	k-means Algorithm				
Palette sizes	FS	PDS[13]	TIE[14]	MPS[15]	Proposed scheme
16	16	16	16	6.775	4.865
32	32	32	32	8.449	5.921
64	64	64	64	10.653	7.540
128	128	128	128	14.005	9.694
256	256	256	256	20.556	12.108

Table 5 Results of numbers of components and colours that are needed in calculation of squared Euclidean distance in *k*-means algorithm with PDS, TIE and MPS, and proposed scheme for palette design

		k-means Algorithm					
		PDS[13]				Proposed scheme	
Palette sizes	Images	Components	Colours	TIE[14]	MPS[15]	Components	Colours
16	Airplane	20.004	1.355	1.825	1.422	3.892	1.262
	Lena	22.653	1.769	2.503	2.068	5.151	1.628
	Pepper	23.635	2.018	2.601	2.406	5.614	1.763
	Splash	22.452	1.522	2.201	1.996	4.803	1.494
	Percentage	**46.22**	**10.41**	**14.27**	**12.33**	**10.14**	**9.60**
32	Airplane	37.313	1.624	2.326	1.852	4.639	1.462
	Lena	41.960	2.284	3.286	2.994	6.577	2.021
	Pepper	42.449	2.438	3.239	3.368	6.662	2.020
	Splash	41.317	1.808	2.740	2.704	5.805	1.761
	Percentage	**42.46**	**6.37**	**9.06**	**8.53**	**6.17**	**5.68**
64	Airplane	73.177	2.164	3.600	2.965	6.131	1.850
	Lena	79.712	2.905	4.372	4.492	8.242	2.430
	Pepper	80.729	3.075	4.285	5.405	8.504	2.476
	Splash	79.192	2.253	3.602	3.948	7.282	2.137
	Percentage	**40.73**	**4.06**	**6.19**	**6.57**	**3.93**	**3.47**
128	Airplane	144.374	2.904	4.618	4.344	8.172	2.352
	Lena	155.288	3.856	5.844	7.229	10.792	3.016
	Pepper	155.747	3.990	5.706	8.814	10.988	3.059
	Splash	150.448	2.718	4.571	5.798	8.822	2.515
	Percentage	**39.44**	**2.63**	**4.05**	**5.11**	**2.52**	**2.14**
256	Airplane	285.866	3.832	5.356	6.348	10.559	2.918
	Lena	294.017	4.894	7.185	11.250	13.261	3.593
	Pepper	298.183	5.022	7.136	14.090	13.620	3.624
	Splash	290.752	3.360	5.787	8.832	10.991	3.009
	Percentage	**1.67**	**2.49**	**3.96**	**1.58**	**1.28**	**38.05**

Table 6 Results of distribution of absolute distance (AD) between mean values of closest colours and original pixels

Palette sizes	Images	AD = 0	AD ⩽ 1	AD ⩽ 2	AD ⩽ 3	AD ⩽ 4	AD ⩽ 5
16	Airplane	27,028	80,707	132,760	182,167	212,501	225,789
	Lena	17,235	51,153	84,240	116,122	146,271	174,403
	Pepper	13,273	39,765	66,458	92,894	118,779	143,642
	Splash	21,681	63,412	102,058	136,494	166,384	190,560
	Percentage	7.55	22.41	36.77	50.32	61.41	70.04
32	Airplane	35,707	105,175	169,980	214,048	234,814	244,787
	Lena	23,065	68,558	112,202	152,793	188,066	215,040
	Pepper	19,103	57,558	94,653	127,765	158,016	184,258
	Splash	30,039	88,684	140,115	181,479	210,905	229,869
	Percentage	10.29	30.52	49.30	64.48	75.51	83.35
64	Airplane	58,063	162,430	213,886	237,211	248,276	254,601
	Lena	33,232	97,439	155,506	199,785	226,123	240,644
	Pepper	25,835	77,013	122,781	161,611	193,630	217,911
	Splash	42,402	121,459	183,344	219,061	236,457	246,987
	Percentage	15.21	43.71	64.42	77.98	86.26	91.57
128	Airplane	80,809	196,948	234,450	249,029	255,718	258,816
	Lena	43,687	125,162	188,658	225,693	243,379	252,472
	Pepper	34,008	98,509	153,046	195,864	225,504	242,867
	Splash	59,911	159,721	219,257	244,419	254,211	258,272
	Percentage	20.83	55.35	75.86	87.26	93.35	96.55
256	Airplane	96,827	210,541	243,702	253,453	258,654	260,416
	Lena	54,529	152,508	216,432	243,247	254,303	258,613
	Pepper	43,026	123,100	184,916	225,171	246,171	255,317
	Splash	76,244	192,015	239,522	253,357	257,929	259,933
	Percentage	25.81	64.67	84.36	93.00	96.99	98.64

3 Compressed images of CIQ using different palette sizes: (a) 'Airplane' with palette of 16 colours; (b) 'Lena' with palette of 16 colours; (c) 'Airplane' with palette of 64 colours; (d) 'Lena' with palette of 64 colours; (e) 'Airplane' with palette of 256 colours; (f) 'Lena' with palette of 256 colours

Euclidean distances between the colours in the proposed scheme.

The MPS technique was originally proposed to accelerate the codebook searching process of VQ for greyscale image quantization. A key reason for the success of the MPS technique is that the pixel values of real images in each image block are highly concentrated about their mean values. To see whether the MPS technique works well for colour images, an analysis of the differences in the mean values of the colour pixels and their corresponding quantized palette colours are shown in Table 6. The mean value of one colour pixel is the average value of its R, G and B component values.

There are 262,144 pixels in each testing image of 512×512 pixels. There are 7.55%, 15.21% and 25.81% of colours with the same mean values as their quantized colours when the palette sizes are 16, 64 and 256, respectively. The larger the palette size, the smaller the absolute difference between the mean value of a colour pixel and its quantized colour. There are 70.04%, 91.57% and 98.64% of colours with absolute distances $\leqslant 5$ when the palette sizes are 16, 64 and 256, respectively.

To understand the visual effects of the compressed images of CIQ, some resultant images of 'Airplane' and 'Lena' are given in Fig. 3. The blocking effects and the staircase effects can be observed in these compressed images of CIQ in the palettes sized 16. This is because these palette colours are not suitable for representing all these colour pixels in the testing images.

The block effects can also be found in some compressed images of CIQ with a palette of 64 colours. In other words, the visual qualities of these compressed images of CIQ are still poor. The blocking effects and the staircase effects can be solved using a large-sized palette in CIQ. For example, when a palette of 256 colours was used to compress the test image 'Lena', the visual quality is approximately the same as the original image of 'Lena' in Fig. 2. That is why the palettes of 256 colours are usually used in CIQ to compress colour images.

5 CONCLUSIONS

A fast palette design algorithm that is equivalent to the *k*-means algorithm was proposed. In this algorithm, two test conditions based on the triangular equality[14] and the MPS technique[15] were employed to filter out impossible colours from the palette during the closest colour search process. In addition, the partial distance search technique[13] was used to early terminate the calculating of the squared Euclidean distances for impossible candidates.

To speed up the search of the initial colour in the palette in the closest colour search process, an array that records the indices of the relatively closest palette colour of each possible average component value ranging from 0 to 255 was constructed. Application of a table look-up operation was much faster than the binary search technique to determine the initial search colour in the palette.

According to the results, an average time reduction of about 92.848% was achieved when a palette of 256 colours was used in the proposed scheme compared with the conventional *k*-means algorithm. No extra image distortion was incurred in the proposed scheme. In other words, the proposed scheme significantly cut down the computational cost of the *k*-means algorithm for palette design. It is quite suitable for real-time multimedia applications based on CIQ.

ACKNOWLEDGMENT

This research was supported by the National Science Council, Taipei, under contract NSC 94-2213-E-126-009.

REFERENCES

1 Allebach, J. P. Digital color imaging bring color to the desktop. IEEE SP-Society Distinguished Lecture's Series*: Third Signal Processing Workshop*, Goreme, Turkey, May 1995.

2 Heckbert. P. Color image quantization for frame buffer display. *Comput. Graph.*, 1982, **16,** 297–307.

3 Wu, X. and Witten, I. H. A fast *k*-means type clustering algorithm. Technical Report, Department of Computer Science, University of Calgary, Calgary, Canada, 1995.

4 Wan, S. J., Prusinkiewicz, P. and Wong, S. K. M. Variance-based color image quantization for frame buffer display. *Color Res. Appl.*, 1990, **15**, 52–58.

5 Yang, C. Y. and Lin, J. C. RWM-Cut for color image quantization. *Comput. Graph.*, 1996, **20**, 577–588.

6 Cheng, S. C. and Yang, C. K. A fast novel technique for color quantization using reduction of color space dimensionality. *Pattern Recognition Lett.*, 2001, **22**, 845–856.

7 Tou, J. T. and Gonzalez, R. C. *Pattern Recognition Principles*, 1974 (Addison-Wesley, Reading, MA).

8 Scheunders, P. A genetic *c*-mean clustering algorithm applied to color image quantization. *Pattern Recognition*, 1997, **30**, 859–866.

9 Hsieh, I. S. and Fan, K. C. An adaptive clustering algorithm for color quantization. *Pattern Recognition Lett.*, 2000, **21**, 337–346.

10 Akarun, L. A fuzzy algorithm for color quantization of images. *Pattern Recognition Lett.*, 2002, **35**, 1785–1791.

11 Linde, Y., Buzo, A. and Gray, R. M. An algorithm for vector quantizer design. *IEEE Trans. Commun.*, 1980, **28**, 84–95.

12 Gersho, A. and Gray, R. M. *Vector Quantization and Signal Compression*, 1992 (Kluwer Academic, Boston, MA).

13 Bei, C. and Gray, R. M. An improvement of the minimum distortion encoding algorithm for vector quantization. *IEEE Trans. Commun.*, 1985, **33**, 1132–1133.

14 Huang, S. H. and Chen, S. H. Fast encoding algorithm for VQ-based image coding. *Electronics Lett.*, 1990, **26**, 1618–1619.

15 Ra, S. W. and Kim, J. K. A fast mean-distance-ordered partial codebook search algorithm for image vector quantization. *IEEE Trans. Circ. Syst. II*, 1993, **40**, 576–579.

16 Chang, C. C. and Hu, Y. C. A fast codebook training algorithm for vector quantization. *IEEE Trans. Cons. Electron.*, 1998, **44**, 1201–1208.

17 Hu, Y. C. and Chang, C. C. An effective codebook search algorithm for vector quantization. *Imag. Sci. J.*, 2003, **51**, 221–234.

18 Zhang, X. and Wandell, B. A. A spatial extension to CIELAB for digital color image reproduction. *Soc. Inf. Display Symp. Tech. Dig.*, 1996, **27**, 731–734.

19 Eckert M. P. and Bradley, A. P. Perceptually quality metrics applied to still image compression. *Signal Process.*, 1998, **70**, 177–200.

20 Pan, F., Lin, X., Rahardja, S., Lin, W., Ong, E., Yao, S., Lu, Z. and Yang, X. A locally adaptive algorithm for measuring blocking artifacts in images and videos. *Signal Process. Image Commun.*, 2004, **19**, 499–506.