A new simple method for constructing a color palette that uses the radius weighted mean cut (RWM-cut) is proposed. The method is a hierarchically divisive method, and each two-class partition uses the centroid and the RWM only. Experiments show that the RWM-cut algorithm is feasible and visually acceptable. The algorithm can either be used alone or be used to create a good initial palette for the LBG algorithm. Besides the 3-D version, a 1-D version of the RWM-cut algorithm is also included in the paper for real-time color quantization. The quantization error is small and the processing speed is competitive. Dithered images are also provided.

1. INTRODUCTION

The process of reproducing a color image using a very limited number of colors suitably generated from that given image is called color quantization. In general, people quantize a full-color (typically 16 million colors) image into one with 256 (or fewer than 256) colors. A full-color image can then be displayed directly on commonly used output devices such as the monitor of a low-cost workstation or PC. The other application of color quantization is that: when a color image is to be compressed, the compression ratio can often become higher if the compression technique is combined with color quantization using, say, 64 colors only. Roughly speaking, color quantization includes two parts: the generation of an appropriate palette, and the correspondence that assigns to each input pixel one of the palette colors. Since the size of the palette is quite limited, it is critical to derive a good palette. The designer, when the palette size is fixed, should try to make the perceived quality of the reproduced image as close as possible to the original one [1].

A number of approaches [1-9] have been proposed to perform color image quantization. The LBG algorithm developed by Linde et al. [2] is a simple method that is easy to implement, but it has the following drawbacks: a poorly selected initial palette may lead to an undesirable final palette, and a complete design requires a large number of computations. Heckbert [3] proposed the median-cut algorithm for color quantization. The algorithm recursively divided the reduced (5-5-5) RGB color space (that is, the 24 bit image is first reduced to a 15 bit image in a pre-processing via discarding the three least significant bits for each RGB component) into two subboxes containing approximately equal number of colors until the desired number of subboxes is reached. Finally, the color palette is formed by collecting the centroids of these subboxes. Although the algorithm is simple, a greater visible distortion exists in the quantized image. Kurz [4] presented another simple algorithm called the uniform quantization algorithm to quantize colors. The principle of the algorithm is that the 3-D color space is divided into N rectangular subboxes of equal size. Then, the centers of these subboxes are used as the N representative colors. In spite of the simplicity of the uniform quantization technique, false contours severely exhibit in the quantized images, especially when the size of the palette is less than 256. Wu and Witten [5] suggested the mean-split algorithm. The hyperplane they used for partition passes through the mean of the longest projected data component. Since it is simpler to compute the mean than to compute the median, the processing speed of the mean-split algorithm is faster than that of the median-cut algorithm. However, a partition plane passing through the mean instead of the median does not necessarily result in lower quantization error [6]. Wan et al. therefore proposed an elegant method called the variance-based algorithm [1] to quantize colors based on the goal of minimizing the sum of squared error. The idea of that algorithm is to sweep a cutting plane along a direction parallel to one of the reduced RGB coordinate axes, and then split the current box at the position where the sum of squared “1-D” error (defined in equation (11) of [1]) of the projected distribution in the corresponding axis is minimized. Since the algorithm is specifically designed by inspecting the sum of squared error from time to time, the algorithm can often produce a quantized image whose sum of squared “3-D” error, which is the right hand side of our Eq. (1) given at the end of this section (without the denominator), is low (but...
not minimal). Unfortunately, the computation complexity is not attractive. (The computation complexity would have been even higher if they had tried to cut along one of the coordinate axes at the position where the sum of "3-D" error is minimized.) Orchard and Bouman [7] presented a delicate algorithm for the design of a tree-structured color palette incorporating a performance criterion which reflects the subjective evaluation of image quality. Although the algorithm results in images with small artifacts, the implementation of the algorithm is not very simple, because the computation for the principal axis is required each time a node is split during the construction of the color palette. Wu [8] used the dynamic programming technique to derive a nearly optimal color palette. The basic idea of the algorithm is to simultaneously optimize K-1 cuts along the principal axis if K colors are desired. This partition strategy leads to small quantization error. However, the time complexity of the algorithm is a problem. For example, it may take about 3 min on a personal IRIS workstation to quantize a full-color image into one with 256 colors. This method is therefore not suitable for the applications in which the processing speed is of big concern. A method faster than Wu's dynamic programming technique is the center-cut algorithm [9] proposed by Joy and Xiang who made three modifications to the median-cut method. The three modifications are (1) to utilize 5-6-4 instead of 5-5-5 color reduction for each RGB pixel; (2) to partition the color box whose longest-dimension is the longest among all boxes (explained in [9]) instead of partitioning the color box whose pixel count is the largest; and (3) to cut through the center of the color box instead of bisecting the box's pixel count. Basically, the center-cut algorithm inherits the advantages of the median-cut method and improves the performance of that technique. However, the distortion of the color hue introduced by the center-cut algorithm is severe.

From the above analysis, we can see that the existing algorithms often become computationally inconvenient or yield unsatisfactory quantization error. In this paper, we develop a new simple method for color palette generation. The method can work well if it is used alone; on the other hand, it can also be used to initialize and accelerate the LBG algorithm. In the new method, we employ the radius weighted mean (RWM), a special kind of point estimate briefly mentioned in Section 2. The radius weighted mean is defined by

\[
\text{RWM} = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{P(i, j) - P'(i, j)}{P(i, j) - P'(i, j)^2} \quad (1)
\]

if the image size is \(M \times N\). Here, \(P\) and \(P'\) denote the 3-D color values of the input image and the quantized image, respectively.

The remaining part of the paper is organized as follows. The proposed RWM-cut algorithm for deriving a color palette is described in Section 2. In Section 3, the RWM-cut algorithm is first used alone. Then, the LBG algorithm with initial palette generated by different methods are discussed. In Section 4, we compare the real-time simulations obtained from various algorithms, including the RWM-cut, median-cut, mean-split, center-cut, and variance-based algorithms. Examples of applying the spatial dithering to the quantized images generated by the proposed RWM-cut algorithm are also provided there. The summary of the paper is given in Section 5.

2. THE RWM-CUT ALGORITHM

Before giving the formal definition of the RWM-cut algorithm, which can handle data of any dimension, we first show the good partition ability of the RWM-cut. The examples are 2-D because they are easier to illustrate. The data listed in Fig. 1(a and b) are, respectively, the 2-D projection of the quite famous 87-point Chernoff Fossil data and 150-point Fisher Iris data commonly used as the test data in the field of data analysis [13]. We can see that the RWM-cut partitions the data better than the median-cut, mean-split, and center-cut do. As for the variance-based algorithm, although its partition result is competitive with that of the RWM-cut in Fig. 1, its computation speed is slower than that of the RWM-cut. The performance of the partition results are summarized in Table 1. It can be seen that the RWM-cut algorithm obtains a smaller total sum of squared error (TSSE) than the other algorithms do, and the computation speed of the RWM-cut algorithm is also not bad. Note that the TSSE (of a K-cluster partition), which is the sum of the squared Euclidean distances between each data point and its cluster mean, is defined by

\[
\text{TSSE} = \sum_{j=1}^{K} \sum_{x \in C_j} ||x - m_j||^2 \quad (2)
\]

Also note that \(x\) is the data point belonging to cluster \(C_j\), and \(m_j\) is the centroid of the cluster \(C_j\). The error represents the accumulated deviations from the
Fig. 1. The resulting clusters generated by different methods. Each row corresponds to a method; each column corresponds to a data set; each straight line \( i (i=1,2) \) corresponds to the \( i \)th partition boundary.

We discuss below the procedure that we will use to partition data. To make the reading easier, two examples are provided to illustrate the procedure. Although the examples shown here are 2-D, the reader can still get the idea how the algorithm proceeds, and then extend the idea to the 3-D case. Note that when the data are 2-D, say,

\[
S = \{(x_i, y_i)\}_{i=1}^{|S|},
\]

the centroid \( O \) is

\[
(x, y) = \left( \frac{1}{|S|} \sum_{i=1}^{|S|} x_i, \frac{1}{|S|} \sum_{i=1}^{|S|} y_i \right),
\]

whereas the RWM \( R \) (defined by Mitiche and Aggarwal [10]) is

\[
(x', y') = \frac{\sum_i w_i x_i}{\sum_i w_i}, \quad \frac{\sum_i w_i y_i}{\sum_i w_i},
\]

with

\[
w_i = \sqrt{(x_i - x)^2 + (y_i - y)^2}
\]

being the distance between the centroid and the \( i \)th data point. The boundary that we proposed to partition the 2-D data set \( S \) into two subsets is then taken to be the line through \( R \) and perpendicular to \( OR \).

**Example 1** (see Fig. 2). If the data are \{5,11\}, \{(5,5)\}, \{(14,8)\}, then the centroid is \((x, y) = (8,8)\).

Therefore, \[ w_1 = \| (5,11) - (8,8) \| = \sqrt{3^2 + 3^2} = \sqrt{18}, \]

\[ w_2 = \| (5,5) - (8,8) \| = \sqrt{3^2 + 3^2} = \sqrt{18}, \]

and \[ w_3 = \| (14,8) - (8,8) \| = 6. \]

Since \[ W = w_1 + w_2 + w_3 = 2\sqrt{18} + 6, \]

the RWM is \((x', y')\) with

\[
x' = \frac{\sqrt{18}(5) + \sqrt{18}(5) + 6(14)}{[2\sqrt{18} + 6]} = 8.73
\]

Table 1. Total sum of squared error (TSSE) and CPU time (in s) for different algorithms

<table>
<thead>
<tr>
<th>2-D Data</th>
<th>RWM-cut TSSE/time</th>
<th>Median-cut TSSE/time</th>
<th>Mean-split TSSE/time</th>
<th>Center-cut TSSE/time</th>
<th>Variance-based TSSE/time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 1(a)</td>
<td>21047/0.0007</td>
<td>45333/0.0011</td>
<td>43756/0.0003</td>
<td>43756/0.0003</td>
<td>21678/0.0018</td>
</tr>
<tr>
<td>Fig. 1(b)</td>
<td>3710/0.0006</td>
<td>9320/0.0010</td>
<td>8739/0.0002</td>
<td>8739/0.0002</td>
<td>4156/0.0015</td>
</tr>
</tbody>
</table>
After the computation, we found that \( \text{Var}(A) > \text{Var}(B) \); we therefore split class A. The RWM \( R_A \) of class A is computed and the partition boundary is the line [line 2 in Fig. 3(c)] passing through the RWM \( R_A \) and perpendicular to the line \( OA_R \). Class A is therefore split into two finer subsets, namely, classes \( A_{C} \) and \( A_{B} \). Since we already have three classes \( \{B, A_{C}, A_{B} \} \), and the expected number of classes is four, we have to split the data once more. After comparing \( \text{Var}(A_{C}) \), \( \text{Var}(A_{B}) \), and \( \text{Var}(B) \), we found that \( \text{Var}(B) \) is the largest; we therefore split B. The procedure is analogous to the one that we split A, and the resulting two smaller subsets are the \( B_{C} \) and \( B_{B} \) shown in Fig. 3(d). The final output of the system is the \( \{A_{C}, A_{B}, B_{C}, B_{B} \} \) shown in Fig. 3(e).

Having seen how 2-D data can be partitioned using RWM, we give below the 3-D RWM-cut algorithm used to construct a color palette. Let \( S = \{(r_i, g_i, b_i) | i = 1, 2, \ldots, |S| \} \) be the given 3-D data set to be partitioned. The centroid \( (r, g, b) \) of \( S \) is evaluated by

\[
\begin{align*}
r & = \frac{1}{|S|} \sum_{i=1}^{|S|} r_i, \\
g & = \frac{1}{|S|} \sum_{i=1}^{|S|} g_i, \\
b & = \frac{1}{|S|} \sum_{i=1}^{|S|} b_i,
\end{align*}
\]

The RWM \((r', g', b')\) of \( S \) is defined by extending the 2-D definition to 3-D case:

\[
\begin{align*}
r' & = \frac{1}{W} \sum_{i=1}^{|S|} r_i w_i, \\
g' & = \frac{1}{W} \sum_{i=1}^{|S|} g_i w_i, \\
b' & = \frac{1}{W} \sum_{i=1}^{|S|} b_i w_i,
\end{align*}
\]

where

\[
W = \sum_{i=1}^{|S|} w_i = \sum_{i=1}^{|S|} \sqrt{(r_i - r)^2 + (g_i - g)^2 + (b_i - b)^2}.
\]

Note that \( w_i \) is the distance from the centroid \((r, g, b)\) to the \( i \)th point \((r_i, g_i, b_i)\). Once the RWM \( R = (r', g', b') \) and the centroid \( O = (r, g, b) \) are known, the decision boundary proposed here to split
Fig. 3. A 2-D example showing how the RWM-cut algorithm splits a data set into four classes. The input is (a), and the final output is (e). Each "+" represents a centroid, whereas each "×" represents an RWM point.
the given set $S$ into two smaller subsets is then taken to be the plane passing through the RWM $R$ and perpendicular to the line $OR$. The technique partitions a data set into two subsets, and we can repeat the technique $K - 1$ times to obtain $K$ mutually disjoint subsets. The method proceeds in a hierarchically divisive manner. The details of the method are described in the following algorithm.

**Algorithm.** The design of a color palette by the 3-D RWM-cut.

**Input:** a data set $S = \{(r_i, g_i, b_i) | i = 1, 2, \ldots, |S|\}$ and a predetermined value $K$.

**Output:** a $K$-color palette $C = \{(r_j, g_j, b_j) | 1 \leq j \leq K\}$.

**Method:**

1. **Step 0:** Initially, let cell $c_1$ be the entire data set $S$ and go to Step 2.
2. **Step 1:** Choose the cell $c_i$ with the largest variation for further partition.
3. **Step 2:** Compute the centroid $O = (r, g, b)$ of the cell $c_i$.
4. **Step 3:** Compute the RWM $R = (r', g', b')$ of the cell $c_i$.
5. **Step 4:** Partition cell $c_i$ into two smaller cells. The partition boundary is the plane passing through $R$ and perpendicular to the line $OR$. (When $O = R$, although this case seldom occurs, just take the partition boundary to be the plane passing through $R$ and perpendicular to the coordinate axis with the largest variance.)
6. **Step 5:** Repeat Steps 1-4 until the number of cells reaches $K$.
7. **Step 6:** Collect the centroids of the $K$ cells. These centroids form the $K$ desired representative colors. Therefore, a $K$-color palette $C$ has been produced.
8. **Step 7:** Stop.

3. EXPERIMENTAL RESULTS

Three full-color $512 \times 512$ images, namely, Lena, Peppers, and Painting, are shown in Fig. 4. Each RGB pixel of these three input images is represented by 24 bits, 8 bits per component. All tests were implemented on a Sun SPARC 10 workstation. Our simulations were performed with the following two aims: (i) to know the MSE and execution time when the RWM-cut algorithm is used alone, and (ii) to show that the RWM-cut algorithm can be used to generate a good initial palette for the LBG algorithm. In (ii), we also list the experimental results when the LBG algorithm is equipped with other kinds of initial palettes, such as the palettes generated by the median-cut, uniform quantization, and mean-split algorithms.

The RWM-cut algorithm was first applied to quantize the three input images to 256 colors. The 256-color quantized images generated by the RWM-cut algorithm are depicted in Fig. 5. The MSE and execution time produced by the RWM-cut algorithm, and by the LBG algorithm with a random initialization, are shown in Table 2. Note that there is nothing to be initialized in the RWM-cut algorithm, while the LBG algorithm will need to choose the initial palette carefully. We then tried to improve the performance of the LBG algorithm by using a better initial palette. The four initial palettes used were the palettes generated by the RWM-cut, median-cut, uniform quantization, and mean-split algorithms, respectively. The results are given in Table 3. The LBG algorithm is stopped when no obvious improvement is made between two consecutive iterations. The threshold for stopping the LBG algorithm is the same for these four approaches. We can see that the RWM-cut is the best among these four approaches.

4. REAL-TIME APPLICATIONS

For real-time applications, many authors used a pre-processing to truncate the 24 bit input image to a 15 bit image, and then apply their color quantization methods to this partially quantized image. In the following, we will also use this pre-processing. The 24 bit image is reduced to a 15 bit image first, with 5 bits for each of the RGB components (that is to
say, we discard the three least significant bits for each 8 bit component). A (modified) 1-D RWM-cut algorithm is then hierarchically applied to the coordinate component with the largest variance. (Using 1-D RWM-cut is faster than using 3-D RWM-cut.)

Below we illustrate how to partition a 3-D data set into two subsets using a 1-D RWM-cut algorithm. First, inspect the variance in each of the three coordinate components \{R,G,B\}. Without the loss of generality, let \( H = \{ h_i \}_{i=1,2,\ldots,|H|} \) be the 1-D component with the largest variance. The centroid \( \bar{h} \) of \( H \) is then defined by

\[
\bar{h} = \frac{1}{|H|} \sum_{i=1}^{|H|} h_i. \tag{12}
\]

Analogously, the RWM \( h' \) of \( H \) is defined by

\[
h' = \left( \sum_{i=1}^{|H|} w_i h_i \right) / \left( \sum_{i=1}^{|H|} w_i \right) \tag{13}
\]

with \( w_i = |h_i - \bar{h}| \) for all \( i \). The decision boundary used to split the 3-D data set into two subsets is then taken to be the plane intersecting the component axis at \( h' \), and perpendicular to the component axis. The procedure is repeated \( K-1 \) times until the desired \( K \) subsets are obtained. The detail is omitted.

According to [9], the 5-6-4 bit-allocation [shown in Figs 6(b), 7(b), and 8(b)] was used as the 15 bit version for the center-cut algorithm. As for the remaining algorithms, the 5-5-5 bit-allocation [shown in Figs 6(a), 7(a), and 8(a)] was used. The 256-color Lena generated by different algorithms are shown in Fig. 6. Similarly, the 64-color Peppers and Painting produced by different algorithms are shown in Figs 7 and 8, respectively. Some wrong blue dirty spots appear on the hat and inside the mirror in Fig. 6(d) (the median-cut algorithm), while a long (but not too obvious) red scar appears on the right cheek of Lena in Fig. 6(g) (the variance-based algorithm). On the other hand, although the perceived quality of Fig. 6(f) (the center-cut algorithm) is better than those of Fig. 6(c) (the RWM-cut algorithm) and Fig. 6(e) (the mean-split algorithm), the color hue of the quantized image is somewhat too yellow in Fig. 6(f), as is compared with Fig. 4(a). (In fact, the color hue is still not quite right in Fig. 7(f) where the red peppers and the white reflection are too yellow [cf. Fig. 4(b)].) In the right part of Fig. 8(d) (the median-cut algorithm) and Fig. 8(e) (the mean-split algorithm), some orange-color spots appear on the two red plants.

<table>
<thead>
<tr>
<th>24 bit images</th>
<th>No. of colors</th>
<th>RWM-cut MSE/time (s)</th>
<th>LBG algorithm MSE/time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>16</td>
<td>15.16/18.25</td>
<td>20.39/359.32</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>11.08/21.83</td>
<td>18.10/708.10</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>8.81/25.75</td>
<td>11.70/1223.98</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>6.85/29.57</td>
<td>8.20/1948.64</td>
</tr>
<tr>
<td>Painting</td>
<td>16</td>
<td>15.75/18.78</td>
<td>19.39/368.69</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>12.12/27.72</td>
<td>12.20/743.30</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>10.76/32.75</td>
<td>11.00/1192.84</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>9.28/39.90</td>
<td>10.79/1952.49</td>
</tr>
</tbody>
</table>
Table 3. MSE and execution time (in s) for the LBG algorithm with initial palette generated by different algorithms.

<table>
<thead>
<tr>
<th>24 bit images</th>
<th>No. of colors</th>
<th>RWM-cut MSE/time</th>
<th>Median-cut MSE/time</th>
<th>Uniform quantization MSE/time</th>
<th>Mean-split MSE/time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>16</td>
<td>13.10/156.81</td>
<td>13.18/275.07</td>
<td>13.42/747.15</td>
<td>13.20/272.49</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>9.83/400.15</td>
<td>9.80/490.72</td>
<td>11.07/1769.66</td>
<td>9.91/843.55</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>7.74/878.56</td>
<td>7.80/1403.44</td>
<td>10.83/2916.78</td>
<td>7.78/1052.77</td>
</tr>
<tr>
<td>Painting</td>
<td>16</td>
<td>13.35/64.18</td>
<td>13.36/64.68</td>
<td>17.64/606.99</td>
<td>13.37/129.16</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>10.29/126.39</td>
<td>10.49/189.98</td>
<td>12.59/753.84</td>
<td>10.67/126.41</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>7.43/495.61</td>
<td>7.45/990.06</td>
<td>12.32/2124.22</td>
<td>7.45/620.43</td>
</tr>
</tbody>
</table>

Fig. 6. The 256-color quantized images generated by using different algorithms on the 15 bit versions of Lena. (a) The (5-5-5) 15 bit version, (b) the (5-6-4) 15 bit version, (c) the RWM-cut, (d) median-cut, (e) mean-split, (f) center-cut, and (g) variance-based algorithms.
Fig. 7. The 64-color quantized images generated by using different algorithms on the 15 bit versions of Peppers. (a) the (5-5-5) 15 bit version, (b) the (5-6-4) 15 bit version, (c) the RWM-cut, (d) median-cut, (e) mean-split, (f) center-cut, and (g) variance based algorithms.

while the purple color of the third wave in the sea becomes wrong. (For the mean-split algorithm, the third wave even becomes broken or thinner, and the color of the pink palm is changed.) In Fig. 8(g) (the variance-based algorithm), white dots become too obvious on the pink palm and the third (purple) wave. As for Fig. 8(f) (the center-cut algorithm), many colors are wrong (the colors of the pink palm, pink tree, green land, orange island, sea, waves, etc. are all wrong). The MSE (distortion from the 24 bit images) and execution time generated by different algorithms for different numbers of representative colors are provided in Table 4. It is observed that the MSE for the RWM-cut algorithm is quite often the smallest among these five quantization algorithms, and the execution time for the RWM-cut algorithm is similar to the one for the mean-split algorithm.
C.-Y. Yang and J.-C. Lin

Fig. 8. The 64-color quantized images generated by using different algorithms on the 15 bit versions of 
Painting. (a) The (5-5-5) 15 bit version, (b) the (5-6-4) 15 bit version, (c) the RWM-cut, (d) median-cut, (e) 
mean-split, (f) center-cut, and (g) variance-based algorithms.

It is well-known that the subjective quality of 
the quantized images can be improved by applying a 
digital halftoning (or spatial dithering) technique. 
Existing spatial dithering techniques include error 
diffusion [14, 15], ordered dithering [16, 17], and dot 
diffusion [18]. Since contouring effects can be 
considerably alleviated by error diffusion, we used 
it in our experiments to improve the quality of the

(but shorter than those for the remaining three 
algorithms). We therefore think that the proposed 
RWM-cut algorithm is a method with small 
quantization error and competitive processing 
speed. Also note that a very simple method, the 
so-called uniform quantization method, is not listed 
here because that method was observed to give 
false contours in most of the experiments.
Table 4. MSE and execution time (in s) for different algorithms\(^1\) when the 15 bit version is used in each input image

<table>
<thead>
<tr>
<th>15 bit images</th>
<th>No. of colors</th>
<th>RWM-cut MSE/time</th>
<th>Median-cut MSE/time</th>
<th>Mean-split MSE/time</th>
<th>Center-cut(^1) MSE/time</th>
<th>Variance-based MSE/time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>32</td>
<td>16.39/1.32</td>
<td>20.55/1.48</td>
<td>17.05/1.32</td>
<td>21.20/1.47</td>
<td>17.38/1.68</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>15.00/1.43</td>
<td>17.68/1.61</td>
<td>15.58/1.43</td>
<td>20.15/1.57</td>
<td>15.74/1.80</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>13.85/1.57</td>
<td>16.19/1.78</td>
<td>14.02/1.56</td>
<td>18.92/1.76</td>
<td>14.86/2.01</td>
</tr>
<tr>
<td>Peppers</td>
<td>32</td>
<td>18.99/1.35</td>
<td>22.30/1.50</td>
<td>21.01/1.34</td>
<td>25.07/1.48</td>
<td>20.85/1.70</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>17.21/1.50</td>
<td>19.51/1.62</td>
<td>18.47/1.46</td>
<td>21.47/1.61</td>
<td>17.97/1.92</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>15.29/1.57</td>
<td>17.59/1.79</td>
<td>15.92/1.57</td>
<td>19.48/1.76</td>
<td>16.17/2.12</td>
</tr>
<tr>
<td>Painting</td>
<td>32</td>
<td>16.89/1.33</td>
<td>17.81/1.49</td>
<td>17.84/1.33</td>
<td>24.13/1.48</td>
<td>18.74/1.69</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>15.21/1.45</td>
<td>17.42/1.63</td>
<td>15.76/1.44</td>
<td>23.76/1.63</td>
<td>17.06/1.92</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>13.68/1.57</td>
<td>16.06/1.82</td>
<td>13.59/1.57</td>
<td>22.07/1.80</td>
<td>15.64/2.02</td>
</tr>
</tbody>
</table>

\(^1\)The LBG was not used.

\(^\d\)Although a post-processing, which uses the “24 bit” pixel values of the original image to modify the color palette, can be used to improve the color hue for the center-cut algorithm; the total computation time will make the algorithm less competitive in real-time application. Therefore, there is no post-processing throughout the paper presented here.

\(^4\)Some other control policies can be used to alleviate the high MSE, but false contours will then appear in the images, and the major advantage of using the center-cut will then disappear.

quantized images. The main steps of the dithering technique are listed below:

```plaintext
for i := 1 to M do
    for j := 1 to N do begin
        e := |P[i, j] - P'[i, j]|;
        P[i, j + 1] := P[i, j + 1] + e \times 7/16;
        P[i + 1, j - 1] := P[i + 1, j - 1] + e \times 3/16;
        P[i + 1, j] := P[i + 1, j] + e \times 5/16;
        P[i + 1, j + 1] := P[i + 1, j + 1] + e \times 1/16;
    end for
end for.
```

In the above, both M and N are 512; P, P', and e are the 3-D color values of the input image, the quantized image, and the difference between these two images, respectively. The 32-color dithered images of Lena and Painting are shown in Fig. 9. As expected, with dithering technique, a 32-color quantization is also not bad.

5. SUMMARY

In this paper, we have proposed a new simple method that uses the radius weighted mean (RWM) to generate color palettes. Experiments show that the RWM-cut algorithm is feasible and visually acceptable. When used alone, the method is more reliable and much faster than the LBG method if random initialization is used in the LBG algorithm. The method can also be regarded as a good initial palette generator for the LBG method. This was confirmed by our experiments, which compared the MSE produced by the LBG algorithm when the initial palette was generated by the RWM-cut algorithm and by other well-known methods. Besides the proposed 3-D RWM-cut algorithm, we had also proposed a simplified 1-D RWM-cut algorithm to quantize colors for real-time applications. The quantization error is small and the processing speed is competitive. In other words, a visually acceptable result can be obtained by the proposed RWM-cut algorithm in a reasonable amount of time.

Fig. 9. The 32-color dithered images generated by using the RWM-cut algorithm on the 15 bit version of (a) Lena and (b) Painting.
Acknowledgement—This work was supported by the National Science Council, Republic of China, under grant NSC85-2213-E009-111.

REFERENCES


5. X. Wu and I. H. Witten, A fast k-means type clustering algorithm. Technique Report, Department of Computer Science, University of Calgary, Calgary, Canada (1985).


