A NEW TECHNIQUE FOR COLOR IMAGE QUANTIZATION

Wafae Sabbar, Abdelkrim Bekkhoucha

University Hassan II Mohammedia, FST Mohammedia, Computer sciences department
BP 146 Mohammedia 20650 Morocco

Keywords: Colour image quantization; Multi-thresholding; Highpass filtering, Lowpass filtering.

Abstract: In this paper, we introduce a new technique of color image quantization. It is carried out in two processing. In the first, we decrease the number of color using a multi-thresholding, by intervals, of the three marginal histograms of the image. In the second processing, the colors determined in the first processing are reduced by colors fusion based on the mean square error minimization. The algorithm is simple to implement and produces a high quality results.

1 INTRODUCTION

Color image quantization is an important problem in computer graphics and image processing, it is a very useful tool for segmentation, compression, presentation and transmission of images. It’s defined as an irreversible image compression technique. The main objective is to map the full color in the original image to a much smaller palette of colors in the quantized image by introducing a minimal distortion between the two images (Xiang, 1994). It is very difficult to formulate a definite solution to the image quantization problem in terms perceived image quality.

Mathematically, color image quantization can be formulated as an optimization problem:

Let \( C = \{c_i, i=1,2...N\} \) be the set of all colors in the image \( I \), \( c_i \) is a vector in one of the color spaces (Lu*v*, HSV, RGB, etc.). A quantized image \( I_Q \) is represented by a set of \( K \) colors \( C_Q = \{c_i, i=1,2...K\} \), \( K<<N \). The quantization process is therefore a mapping:

\[
q: C \rightarrow C_Q
\]

(1)

The closest neighbor principle states that each color \( c \) of the original image \( I \) is going to be mapped into the closest color \( c' \) from the colour palette \( C_Q \):

\[
q(c) = c' \quad \left\| c - c' \right\| = \min_{j=1,2...K} \left\| c - c_j' \right\|
\]

(2)

The quantization mapping defines a set of cluster \( S_k=k=1,2...K \) in the image color space \( C \):

\[
S_k = \{c \in C : \quad q(c) = c_k\}
\]

(3)

Color image quantization has been widely studied for the last fifteen year, the existing techniques of quantization can be divided into three categories (Sangwine, 1998):

Pre-clustering algorithm: Most of the proposed algorithms are based on statistical analysis of the color distribution of image pixels within the color space. The Popularity et Median Cut (Heckbert, 1982), Variance Minimization (Wan, 1990), Octree (Gervautz, 1990) and Principal Analysis Algorithm (Wu, 1992) are examples of this scheme.

Post-clustering algorithm: It involves an initial selection of a palette followed by iterative refinement of this palette using the K-Means algorithm (Linde, 1980) to minimize the Mean Square Error. Fuzzy C-mean (Lim, 1990) is an extension of the K-means algorithm. The Hierarchy Competitive Learning (HCL) (Scheunders, 1997) and Neuant (Dekker, 1994), exploiting the Kohonen Self-Organizing-Maps (Kohonen, 1989) are examples of this scheme.

Mixed algorithm: there exists a different algorithm, which combines between the two approaches precedent, for example the algorithm Split-Merge described by Brun (Brun, 2000).

In this work, we propose a new method of color image quantization witch use a multi-thresholding followed by a merge step. The next sections are organized as follows: In section 2 we describe the multi-thresholding method of a marginal histogram. In section 3 we describe our method. Experimental results are presented in section 4. Conclusions appear in section 5.
2 MULTI-THRESHOLDING METHOD OF A HISTOGRAM

The objective of multi-thresholding is to devise the histogram into a desired number of classes, every class is represented by a peak (maximum) and constituted by all values between two valleys (minimum). We first evaluate the number of peaks in the histogram, if the number of peak is more than the desired one, we must attenuate the high-frequency components and decrease the number of peak by Gaussian filtering (lowpass filtering). Otherwise, we attenuate the low-frequency components and increase the number of peaks by highpass filtering when the number of peaks in the histogram is less than the desired one (Chang, 1997).

2.1 Lowpass filtering

Given a marginal histogram \( h(x) \) of an image, the lowpass filtered \( L(x, \sigma) \) of \( h(x) \) can be obtained using the convolution of \( h(x) \) with a Gaussian function \( g(x, \sigma) \):

\[
L(x, \sigma) = \int_{-\infty}^{\infty} h(\mu) \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right) d\mu
\]

(4)

Where \( \sigma \) denotes the spread parameter of \( g \) and \( \mu \) is a dummy variable. Local maxima (peaks) and minima (valley) of \( h(x) \) can be found from the first partial derivate \( L_x(x, \sigma) \) of \( L(x, \sigma) \). The peaks of \( L(x, \sigma) \) are zero crossings in \( L_x(x, \sigma) \) whose signs change from plus to minus. However, valleys of \( L(x, \sigma) \) are zero crossing in \( L_x(x, \sigma) \) whose signs change from minus to plus.

Let \( P(\sigma) \) denote the number of peaks in the filtered histogram, \( P(\sigma) \) is function of the spread parameter \( \sigma \) and decrease monotonically while \( \sigma \) increase, we must find a good \( \sigma \) to obtain the desired number \( N_{\text{class}} \) of class (peak). In order to resolving the problem, we apply a Dichotomy method, when the search stops, we obtained an interval of approached value. Let \( \sigma^* \) be the upper bound of the interval, we have two case: If \( P(\sigma^*) < N_{\text{class}} \) we stop research. If \( P(\sigma^*) > N_{\text{class}} \), we must merge \( P(x, \sigma^*) - N_{\text{class}} \) peak that gives a weak variance.

2.2 Highpass filtering

When the histogram has a smaller number of peaks than the desired number of classes, we apply a highpass filtering. The highpass used here is

\[
h(x, s) = \exp\left(-\frac{x^2}{s^2}\right)
\]

(5)

Where the parameter \( s \) determines the bandwidth of the stop band in the highpass filter. The highpass filtered histogram \( H(x, s) \) of \( h(x) \) can be easily computed in the frequency domain. We first use a discrete Fourier Transform (DFT) to transform the histogram \( h(x) \) to frequency domain and then apply the highpass filter to attenuate its low frequency components. Finally, the inverse Discrete Fourier Transform (IDFT) is used to obtain the highpass filtered histogram.

3 QUANTIZATION METHOD WITH THE MULTI-THRESHOLDING BY INTERVAL

Our method first uses a multi-thresholding, by intervals, of the three marginal histograms \( h_{\text{red}}, h_{\text{green}} \) and \( h_{\text{blue}} \) of the image. In the second processing, the colors determined in the first step are reduced by colors fusion based on the mean square error minimization.

3.1 Multi-thresholding by interval

Each marginal histogram \( h_{\text{red}}, h_{\text{green}} \) and \( h_{\text{blue}} \) of image can be cut out in several intervals presented by modes. The number of thresholds to be determined in each interval must be proportional to the quantity of information of this one. We evaluates the number of peaks in each interval, if the number of peak is more than the desired one, we decrease the number of peak by lowpass filtering. Otherwise, we increase the number of peaks by highpass filtering when the number of peaks in the interval is less than the desired one.

3.2 Fusion of colour

Let \( C = \{c_0, c_1, \ldots, c_N\} \) the \( N \) produced colour by the first processing, so we have \( N \) cluster \( P = \{C_0, C_1, \ldots, C_N\} \). If \( K \) is the required number to quantify our image, we must merging \( N-K \) cluster in \( N-K \) iteration. The merge of two cluster \( C_i \) and \( C_j \) create a new partition \( P' \) with a square error \( E(P') \) (Brun,2000):

\[
E(P') = E(P) + \frac{|C_i| |C_j|}{|C_i| + |C_j|} \left[ \mu(C_i) - \mu(C_j) \right]^2
\]

(6)
\[ E(P) = \sum_{i=1}^{E} \sum_{j=i}^{E} f(c) \left\| \mathbf{c} - \mathbf{c}_i \right\| \]

and
\[ \mu(C) = \frac{M_1(C)}{|C|} \]
\[ M_1(C) = \sum_{c \in C} f(c) \cdot c \]  

where \( f(c) \) is the number of pixels with color \( c \) in the image and \( \mu(C) \) is the means color for the cluster \( C \).

In this work we have developed a new method for color image quantization in two processing: we use a multi-thresholding, by intervals, of the three colors space for a fair comparison, we also use the RGB space. The Table 1 shows the distortions produced by the quantization methods with 16 and 256 output colors. We remark that our method gives the lowest distortion for different images, the difference is great for a quantization with 16 output colors. Small values of distortion guarantee that a quantization process accurately represents colors of the original image. We notice also that the image loses its quality from a quantization in 16 colours or inferior.

### Table 1:
The distortions produced by our algorithm, SM, Median cut and Octree.

<table>
<thead>
<tr>
<th>Nb of color</th>
<th>Lena</th>
<th>Mandrill</th>
<th>Monarch</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>27.58</td>
<td>31.53</td>
<td>10.55</td>
</tr>
<tr>
<td>256</td>
<td>24.65</td>
<td>25.88</td>
<td>10.58</td>
</tr>
<tr>
<td>SM</td>
<td>28.08</td>
<td>33.02</td>
<td>23.02</td>
</tr>
<tr>
<td>Median cut</td>
<td>19.06</td>
<td>9.46</td>
<td>8.16</td>
</tr>
</tbody>
</table>

In order to illustrate the results of our proposed quantization method, we choose different images tests very widely used in the domain of the image processing: “Lena”(Figure 1), ”Mandrill” (Figure 2) and “Monarch” (Figure 3), its coded in 24 bit. We compare the new method with popular quantization algorithm found in public domain image processing software: Median cut, Octree and SM(Split and Merge). These implementation worked in RGB colors space for a fair comparison, we also use the RGB space.

\[ \Delta(i, j) = \frac{|C_i|}{|C_i| + |C_j|} \left\| \mu(C_i) - \mu(C_j) \right\|^2 \]  

If we encode the “CAG” by an half-matrix \( G \), the two nodes \( n_i \) and \( n_j \) which have to be merged are defined by:
\[ (n_1, n_2) = \text{ArgMin}_{i \neq j \neq n_1 \neq n_2}(\forall i \in [1, ..., N]) G[i][j] \]

In order to improve the search of the nodes which have to be merged, we use an array \( \text{Min} \) defined by:
\[ \text{Min}[i] = \text{ArgMin}_{i \neq j \neq n_1 \neq n_2}(\forall i \in [1, ..., N]) G[i][j] \]

The two merged nodes \( (n_i, n_j) \) are then defined by:
\[ n_1 = \text{ArgMin}_{i \neq j \neq n_1 \neq n_2}(\forall i \in [1, ..., N]) G[i][\text{Min}[i]] \]
\[ n_2 = \text{Min}[n_i] \]  

Using the array \( \text{Min} \), the computation of nodes \( n_i \) and \( n_j \) is performed with \( O(N) \) tests.

If we invalidate nodes \( n_i \), the update of this half matrix requires to invalidate line and column \( n_i \) and to update line and column \( n_j \). The invalidation of lines and column \( n_i \) being performed in constant time, this last point requires \( O(N) \) operations. Moreover, we also have to update the value \( \text{Min}[j] \) if \( \text{Min}[j] \) is equal to \( n_j \) or if the update of line and column \( n_i \) has changed the minimum value of one line. In the best case where no update occurs, we have to perform \( O(N-n_i-1) \) tests. In the worst case, all values \( \text{Min}[i] \) for \( i \) greater than \( n_i \) have to be updated. In this last case, the update of the array \( \text{Min} \) require \( O(N \times n_i) \) operations.

The merger needs at each node the centroid and the cardinal of the corresponding clusters. By using equation (8), the mean can deduced from the first cumulative moment \( M_1 \) and the cardinal. Moreover, as all cluster are disjoints, the first moment and the cardinal are linear operators:

\[ M_1(C_1 \cup C_2) = M_1(C_1) + M_1(C_2) \]

As the mean can be deduced from the first moment and the cardinal, we only store these two data in each cluster.

### 4 EXPERIMENTS

### 5 CONCLUSION

In our merger create a graph called the Cluster Adjacency Graph “CAG” (Brun,2000). The “CAG” is mesh-like and has few edges that cross over each other. Furthermore, the variance in the length of the edges between adjacent nodes is small.

The nodes of our graph are the cluster to be merged and each edge shared by two nodes \( n_1 \) and \( n_j \) respectively associated to the cluster \( C_i \) and \( C_j \) is valuated by \( \Delta(i, j) \):

The two merged nodes \( (n_i, n_j) \) are then defined by:

\[ \Delta(i, j) = \frac{|C_i|}{|C_i| + |C_j|} \left\| \mu(C_i) - \mu(C_j) \right\|^2 \]  

If we encode the “CAG” by an half-matrix \( G \), the two nodes \( n_1 \) and \( n_j \) which have to be merged are defined by:

\[ (n_1, n_2) = \text{ArgMin}_{i \neq j \neq n_1 \neq n_2}(\forall i \in [1, ..., N]) G[i][j] \]

In order to improve the search of the nodes which have to be merged, we use an array \( \text{Min} \) defined by:

\[ \text{Min}[i] = \text{ArgMin}_{i \neq j \neq n_1 \neq n_2}(\forall i \in [1, ..., N]) G[i][j] \]

The two merged nodes \( (n_i, n_j) \) are then defined by:

\[ n_1 = \text{ArgMin}_{i \neq j \neq n_1 \neq n_2}(\forall i \in [1, ..., N]) G[i][\text{Min}[i]] \]
\[ n_2 = \text{Min}[n_i] \]

Using the array \( \text{Min} \), the computation of nodes \( n_1 \) and \( n_j \) is performed with \( O(N) \) tests.

If we invalidate nodes \( n_i \), the update of this half matrix requires to invalidate line and column \( n_i \) and to update line and column \( n_j \). The invalidation of lines and column \( n_i \) being performed in constant time, this last point requires \( O(N) \) operations. Moreover, we also have to update the value \( \text{Min}[j] \) if \( \text{Min}[j] \) is equal to \( n_j \) or if the update of line and column \( n_i \) has changed the minimum value of one line. In the best case where no update occurs, we have to perform \( O(N-n_i-1) \) tests. In the worst case, all values \( \text{Min}[i] \) for \( i \) greater than \( n_i \) have to be updated. In this last case, the update of the array \( \text{Min} \) require \( O(N \times n_i) \) operations.

The merger needs at each node the centroid and the cardinal of the corresponding clusters. By using equation (8), the mean can deduced from the first cumulative moment \( M_1 \) and the cardinal. Moreover, as all cluster are disjoints, the first moment and the cardinal are linear operators:

\[ M_1(C_1 \cup C_2) = M_1(C_1) + M_1(C_2) \]

As the mean can be deduced from the first moment and the cardinal, we only store these two data in each cluster.

### 4 EXPERIMENTS

In order to illustrate the results of our proposed quantization method, we choose different images tests very widely used in the domain of the image processing: “Lena”(Figure 1), “Mandrill” (Figure 2) and “Monarch” (Figure 3), its coded in 24 bit. We compare the new method with popular quantization algorithm found in public domain image processing software: Median cut, Octree and SM (Split and Merge). These implementation worked in RGB colors space for a fair comparison, we also use the RGB space. The Table 1 shows the distortions produced by the quantization methods with 16 and 256 output colors. We remark that our method gives the lowest distortion for different images, the difference is great for a quantization with 16 output colors. Small values of distortion guarantee that a quantization process accurately represents colors of the original image. We notice also that the image loses its quality from a quantization in 16 colours or inferior.

<table>
<thead>
<tr>
<th>Nb of color</th>
<th>Lena</th>
<th>Mandrill</th>
<th>Monarch</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>27.58</td>
<td>31.53</td>
<td>10.55</td>
</tr>
<tr>
<td>256</td>
<td>24.65</td>
<td>25.88</td>
<td>10.58</td>
</tr>
<tr>
<td>SM</td>
<td>28.08</td>
<td>33.02</td>
<td>23.02</td>
</tr>
<tr>
<td>Median cut</td>
<td>19.06</td>
<td>9.46</td>
<td>8.16</td>
</tr>
</tbody>
</table>

### 5 CONCLUSION

In this work we have developed a new method for color image quantization in two processing: we use a multi-thresholding, by intervals, of the three marginal histograms of image in the space of color RGB to reduce and recover the interesting color. In order to find the required number of colors to quantize the image, we merge the colors given a minimal means square error using a “CAG”. The method is simple, efficient and gives good results on various test images.
REFERENCES


