Walailak J Sci & Tech 2005; 2(2):149-168.

A Colour Image Quantization Algorithm for Time-Constrained Applications

Wattanapong KURDTHONGMEE

Division of Computer Engineering, School of Engineering and Resources, Walailak University, Thasala, Nakhon Si Thammarat 80160, Thailand.

ABSTRACT

Many techniques have been proposed to quantize a digital colour image in order to reduce the representative number of colours to be suitable for presenting on different types of display screens. In addition, the techniques have been used to significantly reduce the amount of image data required to transfer over a communication network. Most of the published techniques are targetted for implementing on a general purpose multitasking computer with low restriction on time and resource utilizations. The drawback of these techniques relies on the fact that they cannot fulfill the requirement of some applications for real-time constraint and limited resources. In addition, most of the techniques are too complex for hardware realization. In this paper, an algorithm which is more suitable for time critical applications with an additional feature of simplicity to implement on FPGA (Field Programmable Gate Array) platforms is proposed and the details of its implementation and experimentation are presented. The dominate point of the proposed algorithm relies on the fact that it utilizes the weighted sum of the nearest distance along the axis under consideration, which is nontrivial to calculate, instead of the squared Euclidean distance to find the axis to split during. Also, the proposed algorithm has proved that by reducing the number of subspaces to be considered during the variance representative value calculation from 8 to 2 subspaces, the quality of quantized images are comparable to the previously proposed approaches. This makes it possible to further speed up the computational time of the quantization algorithm.

Key words: Image quantization - Image compression - Dynamic programming

INTRODUCTION

An RGB (red, green and blue) LED display board has been mainly used in advertising applications at tradeshows, along highways, and in stadiums and arenas. One of the critical problems of this display device is how to generate sufficiently fast frame rates to update it. In order to display animation sequences on an LED display, video data must be transferred from a computer, or a special-purpose hardware unit, over an appropriate communication network to a set of embedded controllers of the

LED display. For a moderate resolution LED display board with 640×480 pixels, an uncompressed full RGB colour image frame of size 640×480×3 = 1,228,800 bytes must be transferred with at least 30 frames per second (fps) of speed (1). These huge amounts of data can be reduced significantly by employing an appropriate image compression algorithm. To be a benefit for the system in this case and many applications similar to this, the uncompression algorithm, on the image receiver site, must be easy to implement and, at the same time, must occupy a small amount of resources. In addition, the compression algorithm must be effectively executed within a time constraint on an image transmitter site. Under the limitation of a multitasking operating system such as Windows, the image compression application is not guaranteed to be serviced within the required constraint frame rate. This comes from the nature of running an application on the multitasking environment, the operating system has many tasks, both system and user tasks, to serve. The time resolution, which is the minimum time for the operating system to switch from one task to another task in a cooperative mode of operation, depends heavily on the operating system and is user-uncontrollable. For the Windows operating system, the time resolution is limited to only 1 millisecond. This means that a moderate complex application like an image compression application with execution time of the order of 10 milliseconds running on the Windows operating system will never finish its task within one time resolution.

Many image compression algorithms have been proposed so far (2,3). The colour quantization is one of the most dominant lossy compression methods (4,5). The prominent benefit of this algorithm from the point of view of the image receiver site is the easiness to reconstruct a quantized digital image. Typically, the colour quantization attempts to find an acceptable set of palette colour that can be used to represent the original colours of a digital image (5). This lossy image compression exploits the limited ability of human perception system which is capable of distinguishing less than a thousand colours. Applying the colour quantization to RGB colour image frame of size 640×480 pixels, the resulting image of 256 colour palettes will be reduced to only 307,456 bytes which is about one-third of the original image size. This makes the colour quantization process fluently exploited in many applications especially in computer graphics and image processing.

With the benefits of the colour quantization process to the problem of image compression, the main aim of our research is to employ the process to use for speeding-up the transfer rate of image frames from a personal computer running a multitasking operating system to an LED display board in a real-time manner. And, if possible, we aim at implementing the colour quantization on a special function processor responsible for streaming frames of images on a real-time basis. To satisfy these requirements, an optimized image quantization algorithm, which occupies the CPU for as short a period as possible and is less complex to be hardware implemented, must be developed. This means that the expected colour quantization algorithm must not spend too much time on producing a compressed version of an image. In addition, the proposed algorithm should avoid relying on complex operations; i.e. floating point operations, and even an integer division.

Typically, a colour quantization algorithm contains two main parts, the first part is ***colour palette generation*** and the second part is ***pixel mapping*** (6,7). In the colour palette generation stage, the palette colours are extracted from the original image. Then, each pixel in the original image is mapped to its nearest palette colour in the pixel-mapping stage to generate the quantized version of image. Over the past decades, there are a number of approaches proposed for highest quality quantized images. While some approaches exploited excellent algorithms or theoretical issues (fuzzy logic, neural network and genetic algorithms) to their quantizers: in this paper, the proposed approach is aimed at compromizing between the acceptable image quality and video frame rate, minimization of computational cost and resource utilization of the hardware employed for realization.

The existing techniques for colour quantization can be classified into 2 groups (2). First, there is the class of splitting algorithms that divide the colour space into disjointed subspaces by consecutive splitting up of the space. From each subspace, a colour is chosen to represent the region in the colour palette. Two algorithms of this class which are regularly applied are the median-cut algorithm (8) and the variance-based algorithm (5,7). Other splitting algorithms, which incorporate techniques to take into account the human perception, were also introduced by many researchers. In general, splitting algorithms are fast. The disadvantage is that generally no global optima are obtained, because a decision made for splitting at one level cannot be further recovered.

Another class of quantization techniques performs clustering of the colour space, and cluster representatives are chosen as palette colours (2). A frequently used clustering algorithm is the *C*-means clustering algorithm. In this algorithm, an updating of the cluster representatives and an assignment of colour pixels to clusters are iteratively performed until all the required number of colour palettes have been attained. Other clustering algorithms have also been proposed. Fuzzy *C*-mean clustering, learning vector quantization, and a self-organizing map were also applied to colour quantization.

In this paper, the proposed image quantization process is modified from Kanjanawanishkul and Uyyanonvara's approach (5). The approach, which is in fact adapted from Wu's popular dynamic programming quantizer (7) to be suitable for time-constrained applications, is in the class of splitting algorithms, so it is, from now on, called KSA (Kanjanawanishkul et al's Splitting Algorithm) for short. According to Wu's and KSA quantizers, the dynamic programming was exploited to construct a bottom-up cumulative distribution. This was carried out to avoid excessive re-computation of the centroid and the variance when finding the position of the cutting plane. The cumulative distribution construction process is applied after a 3D-colour histogram is obtained. The obvious difference between Wu's and the KSA method relies on the fact that the KSA method cumulates only the 0^{th} and 1^{st} order moment distribution. The 2^{nd} order moment which is very computationaly intensive process and produces one additional data structure is not required by the algorithm.

The cutting plane which is used to subdivide a colour subspace of KSA's quantizer is put through the centroid of subspace under consideration *S*. The plane is perpendicular to the axis whose sum of the squared Euclidean distances between the centroid of both of *S*'s subspaces and the centroid of the subspace *S* is the greatest. The criteria which is used to select the subspace to subdivide, in each level, bases on variance comparison. That is to say the subspace to be subdivided is the one with the highest value of variance. This subspace, in theory, contains highly scattered data. This makes it reasonable to subdivide further to reduce its scatterness. It is mentioned in (5) that a direct approach for variance calculation of a group of data within a given subspace, which is used in Wu's quantizer, is very computationaly intensive task. This comes from the fact that every data within the subspace must be visited and operated upon. The "*variance representative value*" (*VRV*) is proposed as a simpler indicator of data scatterness. Suppose S , whose centroid is $\bar{x}$, is subdivided at its centroid and its subpaces are S_i where $\bigcup_{i=1}^{8} S_i = S$, $\bigcap_{i=1}^{8} S_i = \phi$. By definition, the *VRV* of subspace S is defined by the following equation:

$$VRV = \sum_{i=1}^{8} \left\| x_i - \bar{x} \right\| \qquad (1)$$

where $\|.\|$ stands for the Euclidean distance norm and x_i is the centroid of each subspace S_i .

It is obvious from Eq. (1) that instead of finding the actual variance using all data points in the subspace, the *VRV* requires only 8 points representing the centroid of each subspace to calculate. This, certainly, highly reduces the overall computational time of the quantizer.

Up to this point, it can be concluded that almost all of the proposed algorithms were targetted for software implementation on a computer and aimed at quantizing still images. The experimental results were evaluated without taking into account the nature of the multitasking operating system. That is to say, the processor time was used for evaluating and comparing the performance of algorithms. In practice, the results only make sense when neither system nor user task is run by the operating system. This can be simply summarised that even the best algorithm with the least processor time consumption, like the KSA, can not be guaranteed to generate a real-time frame rate for processing video data on a computer with a multitasking operating system. This is the reason why a further optimized quantization algorithm or its hardware implementation counterpart must be specially designed for an application with real-time requirements. In the next section, the proposed adaptations to the KSA, to make the resulting algorithm more suitable both for implementing on a personal computer and special hardware, are detailed.

MATERIALS AND METHODS

In this section, the details of the colour image quantization algorithm, focused on the KSA which is the predecessor to our proposed algorithm, are given. The details of the efficiency improvement to the critical parts of the algorithm are then described. It is noted that the improvement approaches proposed in this paper are aimed at maintaining the image quality which is indirectly measured by utilising the "*Mean Square Error (MSE)*" factor and, in addition, avoiding complex operations which are unsuitable for hardware realization.

The Algorithm for Initialising Data Structures

According to the KSA quantizer, the preprocessing stage of the algorithm is responsible for retrieving pixel data from a storage. The following four of three dimensional arrays which consist of 32×32×32 locations are initialised in this stage: *histogram*, *sumOfColourR*, *sumOfColourG*, and *sumOfColourB*. The purpose of the *histogram* array is to keep count of pixels whose colour after resolution reduction from 24-bit to 15-bit are similar. The *sumOfColourR*, *sumOfColourG*, and *sumOfColourB* arrays with similar dimensions to the *histogram* array are used to maintain the sum of colour values; red, green, and blue, respectively, of a pixel at any location. These three data structures would have been of no use, if the colour space was not reduced to 32×32×32 subspaces, or *voxel* for short. This is true, because the multiplication between the frequency and the colour value can simply be calculated by use of the product between the *histogram*'s indices and its frequency. However, this cannot be performed in the subspace reduction version of the data structures. This comes from the fact that each voxel is normally occupied by all pixels for whom the last three bits are dissimilar and the rest of the bits are alike. For example, all pixels with the following (R, G, B) colour value are stored within the same location of the voxel which is at index (23,7,9): (*10111XXX*, *00111XXX*, *01001XXX*), where *XXX* are don't care bits. It is noted for clarity that the last three bits of this pixel's colour components are removed as a result of a 3-bit right shifting operation. Both algorithms proposed by Kanjanawanishkul and Uyyanonvara and Wu (5,7) follow similar computational steps to initialise all the arrays mentioned above.

After all pixels have already been retrieved and the related data structures are initialised, the next stage of the algorithm is to find the cumulative moment of all these data structures.

The Algorithm for Performing Dynamic Distribution Calculation

As mentioned earlier the dynamic programming scheme has been applied to the problem of colour quantization. The application of this scheme helps the quantizer to avoid excessive re-computation of the centroid and the variance when finding the position of the cutting plane during the colour space subdivision process. In practice, the dynamic programming is accomplished by constructing the bottom-up cumulative distribution of all related arrays which in this case consist of the *histogram*, *sumOfColourR*, *sumOfColourG*, and *sumOfColourB*. This is performed after the completion of the three dimensional colour histogram initialisation stage which was previously detailed in section 3.1.

The KSA's Colour Space Subdivision Algorithm

After all the necessary data structures have been initialised and their cumulative distribution calculations have been performed, the KSA quantizer is now ready for the major tasks of subdividing the colour space and generating a set of predefined numbers of colour palettes. Following is the algorithm responsible for these tasks which is proposed by Kanjanawanishkul and Uyyanonvara (5) and modified here to ease understanding.

```
1:   Algorithm spacePartitioning
2:   Begin
3:       No = 0
4:       // Iterate until the required number of colours are obtained
5:       While (No < MaxColour)
6:           // Find the subspace S to subdivide in order to reduce its
             scatterness
7:           SubspaceNumber = Subspace's number with the largest
             VRV among Subspace[No-1].
8:           // Calculate the centroid of S
9:           (r, g, b)  = Centroid of Subspace[subSpaceNumber]
10:          // Calculate the squared Euclidean distance of each main
             colour of a current subspace wrt. to the
11:          // (r, g, b)
12:          SED_R = Squared Euclidean Distance of Red wrt. (r, g, b)
13:          SED_G = Squared Euclidean Distance of Green wrt. (r, g, b)
14:          SED_B = Squared Euclidean Distance of Blue wrt. (r, g, b)
15:          If Max(SED_R, SED_G, SED_B) = SED_Red Then
16:              CuttingPoint = r
17:              Axis = RED
18:          ElseIf Max(SED_R, SED_G, SED_B) = SED_Green Then
19:              CuttingPoint = g
20:              Axis = GREEN
21:          Else
22:              CuttingPoint = b
23:              Axis = BLUE
24:          EndIf
25:          Split Subspace[SubspaceNumber] at Cuttingpoint to be
             perpendicular to Axis
26:          Update the VRV of both splitted box
27:          No = No + 1
28:      End While
29:  End Algorithm
```

From the above algorithm, it can clearly be seen that the algorithm iterates until the original subspace has already been subdivided into *MaxColour* number of non-intersecting subspaces. During each iteration, at line 7 it first searches for the subspace whose data is highly scattered among the rest of the subspaces available so far. This subspace, *Subspace[SubspaceNumber*], is then subdivided further with the aim to reduce its scatterness. The centroid of the *Subspace[SubspaceNumber*] are then calculated at line 9 by using the following equations (5):

$$\begin{aligned} SumOfX_{(r0,g0,b0)\rightarrow(r1,g1,b1)} = {} & SumOfColou\ rX(r1,g1,b1) - SumOfColou\ rX(r0,g1,b1) - \\ & SumOfColou\ rX(r1,g0,b1) - SumOfColou\ rX(r1,g1,b0) - \\ & SumOfColou\ rX(r0,g0,b0) + SumOfColou\ rX(r0,g0,b1) + \\ & SumOfColou\ rX(r0,g1,b0) + SumOfColou\ rX(r1,g0,b0) \end{aligned} \tag{2.1}$$

$$\begin{aligned} Freq_{(r0,g0,b0)\rightarrow(r1,g1,b1)} = {} & Histogram(r1,g1,b1) - Histogram(r0,g1,b1) - \\ & Histogram(r1,g0,b1) - Histogram(r1,g1,b0) - \\ & Histogram(r0,g0,b0) + Histogram(r0,g0,b1) + \\ & Histogram(r0,g1,b0) + Histogram(r1,g0,b0) \end{aligned} \quad (2.2)$$

$$CentroidOfSubspaceInXAxis = {SumOfX}\Big/{Freq} \quad (2.3)$$

It is noted that the term *SumOfX* is used in Eq. (2) to represent the *SumOfR*, *SumOfG*, and *SumOfB* values. By utilizing the dynamic programming, it is obvious that each colour component of the centroid can be quickly calculated. That is to say, the algorithm only retrieves the precalculated values of the *SumOfColourX*s from the appropriate data structures and uses those values to substitute into Eq. (2.1) to solve for the *SumOfX* values. In addition, the cumulative frequency of pixels within subspaces between (r_0, g_0, b_0) and (r_1, g_1, b_1) can also be retrieved directly from the *Histogram* array and used to produce the *CentroidOfSubspaceInXAxis* result in Eq. (2.3).

It is mentioned by Kanjanawanishkul and Uyyanonvara in (5) that the key difference between the KSA quantizer and the previously published algorithms in the class is how to position the cutting plane which gives rise to the optimal result. While Heckbert's median-cut algorithm (8) simply places the cutting plane at the location which separates S into 2 subspaces S' and S'' whose number of pixels are equal. In constrast, to improve image quality and reduce the algorithmic computation time, the KSA quantizer utilises the cutting plane normal to the colour axis with the highest value of an appropriate indicator and passing through the centroid of S. The squared Euclidean distance (*SED*) is proposed to be used as such an indicator.

With respect to the KSA's *spacePartitioning* algorithm at line 12-14, the *SED_X* of all colour components with respect to the centroid of *Subspace[SubspaceNumber*] are calculated by following this equation:

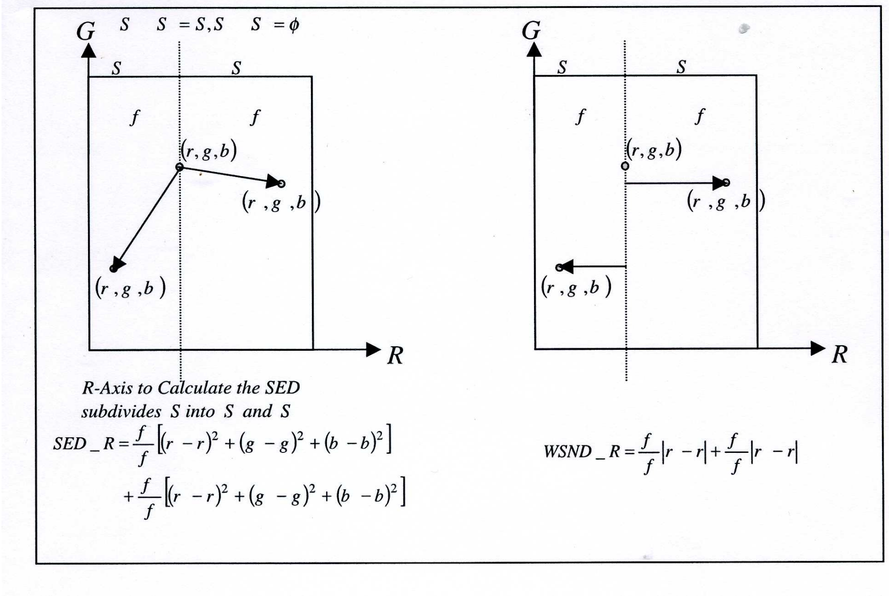
$$SED\_X = \frac{f'}{f'+f''}\left[(r'-r)^2+(g'-g)^2+(b'-b)^2\right]+\frac{f''}{f'+f''}\left[(r''-r)^2+(g''-g)^2+(b''-b)^2\right] \quad (3)$$

where (r',g',b') and (r'',g'',b'') are the centroids of two subspaces, S' and S'', belonging to the *Subspace*[*SubspaceNumber]*, S, subdivided at the centroid of S which is at (r,g,b). f' and f'' are the numbers of pixels within those two subspaces.

The next step of the algorithm in line 15-24 is to determine which axis the subspace S actually needs to be subdivided. Under the criteria given by Kanjanawanishkul and Uyyanonvara (5) the axis, *Axis*, to be split is the axis whose *SED* is the largest among the three colour axes. Having obtained this information, *S* is subdivided along the *Axis* at the *CuttingPoint*. This produces two subspaces whose *VRVs*, described by Eq. (1), are further updated. Finally, the subspace S is removed from the *Subspace*-list and both newly created subspaces are, in turn, inserted into the list which results in 1 additional subspace to the list (at line 27).

The Proposed Adaptations to the Critical Parts of the KSA's Colour Space Subdivision

In this section, the detail of the adaptations to the critical parts of the KSA quantizer, in order to improve its efficiency and make it suitable for hardware implementation, are presented. First of all, consider Eq. (3) closely, it can be summarised that the KSA quantizer actually utilises the weighted sum of *SED* between the centroids of S' and S'' to S as a factor to determine the colour axis to subdivide at each level. **Figure 1(a)** illustrates the geometric interpretation of Eq. (3) when a red colour axis is being considered. Although, the equation gives rise to the correct indicator at the cost of reducing computational cost comparing to the previously published techniques, it still requires a fairly high computational cost to perform in software. Additionally, from the hardware implementation's point of view, this computational stage produces a fairly high complex hardware unit. In our proposed algorithm, the computational cost is further reduced while the image quality is maintained as clearly indicated in the experimental results (given in Section 4). Consider **Figure 1(b)**, it can be seen that the weighted sum of the *SED*s within the red-green colour plane can be replaced by the weighted sum of the one dimensional distances between S' to S and S to S'' along the red colour axis. The weighted sum of these two distances, which are represented by dotted lines within the Figure, can be simply calculated by use of the first equation within the following group:


G
S S = S,S S = ϕ
S
S
f
f
(r,g,b)
(r ,g ,b)
(r ,g ,b)
R
R-Axis to Calculate the SED subdivides S into S and S
SED_R = f/f [(r − r)² + (g − g)² + (b − b)²] + f/f [(r − r)² + (g − g)² + (b − b)²]
G
S
S
f
f
(r,g,b)
(r ,g ,b)
(r ,g ,b)
R
WSND_R = f/f |r − r| + f/f |r − r|

(a) **(b)**

Figure 1. (a) The geometric interpretation of the *SED* when a red colour axis is being considered and (b) the geometric definition of weighted sum of the nearest distances from both centroids S' and S'' to the Line which passes through the centroid of S and is perpendicular to the i colour axis.

$$\begin{aligned} WSND\_R &= f'\left|r'-r\right|+f''\left|r''-r\right| \\ WSND\_G &= f'\left|g'-g\right|+f''\left|g''-g\right| \\ WSND\_B &= f'\left|b''-b\right|+f''\left|b''-b\right| \end{aligned} \tag{4}$$

where $WSND\_X$ is the weighted sum of the nearest (normal) distances from both centroids S' and S'' to the line which passes through the centroid of S and is perpendicular to the i colour axis. (r,g,b), (r',g',b'), and (r'',g'',b'') are the centroids of S, S' and S'', respectively. Further, f' and f'' are the frequency of pixels whose values are within S' and S'', respectively. It is noted that to be correct both f' and f'' must be divided by $f'+f''$. However, in practice it is needless to do so because $f'+f''=f$, the overall frequency of pixels within S, which is constant in every colour axis when subspace S is being considered.

Making use of the $WSND\_X$ to represent the *SED_X* has some minor limitations as illustrated in **Figure 2**. While **Figure 2(a)** confirms that as long as the magnitude of the *SED_R* is changed, so does its $WSND\_R$. In constrast, when the magnitude of the *SED_R* is constant but its direction is changed, the $WSND\_R$ reflects an incorrect result as clearly shown in **Figure 2(b)**. It is, however, needed to experimentally confirm that with respect to our domain such a case does not occur frequently to cause the severe final result to a quantized image. Experiments were, therefore, carried out to evaluate the error resulting from this kind of deficiency. In the experiments, the frequency of a conflict decision to split subcubes between using *SED_X* and $WSND\_X$ are recorded. The inputs to the experiments consisted of 100 randomly generated images of size 512×512 with 256 required final subcubes (which is equivalent to 256 palette colours). We have found that, on average, only 1.7 percent of 256 subcubes have conflict decision. The minimum and maximum percentages of inconsistency are 0.4 and 4.0, respectively. Obviously, this confirms that, without taking the perceived images and the MSE factor into account, more than 95 percent of subcubes are split correctly, with respect to the *SED_X*, even by utilising a simple calculation $WSND\_X$-factor.

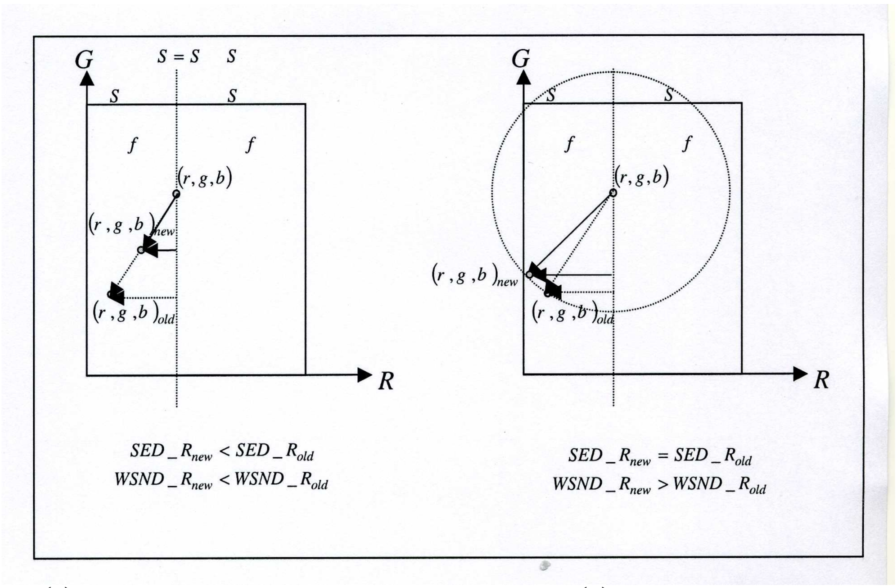
Due to the lower complexity of Eq. (4) and the positive confirmation from the experiment described earlier, the $WSND\_X$ are used instead of the *SED_X* in the algorithm proposed in this paper. It is noted that the rules for deciding the colour axis to be split are still preserved with the newly proposed factor.

Another part of the KSA quantizer which consumes much of the computational time is at line 25. The algorithm spends time on updating the *VRV*, Eq. (1), of the two subspaces resulting from the subdivision stage described earlier. The computational steps for each subspace S^*'s *VRV* calculation consist of finding the centroid of S^* and using this centroid to produce 8 non-intersecting subspaces belonging to S^*. The centroids of the 8 subspaces are then calculated and its values are substituted into Eq. (1) to produce the required *VRV*. The first approach to improve the efficiency of the quantizer is to use the square of the Euclidean distance norm instead of the Euclidean distance norm. This reduces the computational time and, at the same time, makes it possible to realize the algorithm by means of hardware implementation since there is no need to perform a complex square root calculation within Eq. (1).

Splitting S^* at its centroid to produce 8 non-intersecting subspaces in order to find the *VRV* of S^* seems to make sense as mentioned and experimentally confirmed by Kanjanawanishkul et al in (5). From our experimental results, we have found that even splitting S^* into only 2 or 4 non-intersecting subspaces also results in an output image with comparable error and acceptable image quality at the cost of further reducing computational time of the overall algorithm. With the reduction in the number of subspaces into 2 and 4, the Eq. (1) can now be rewritten:

$$\begin{aligned} VRV_{2P} &= \sum_{i=1}^{2} \left\| x_i - \bar{x} \right\|^2, \\ VRV_{4P} &= \sum_{i=1}^{4} \left\| x_i - \bar{x} \right\|^2, \end{aligned} \tag{5}$$

for 2 and 4 subspaces, respectively. It is noted that the square of Euclidean distance norm in Eq. (5) means that the squared root of each term, within the summation, is not required to be taken.



(a) **(b)**

Figure 2. The relationships between the *SED* of S'-subspace and its *WSND* when (a) only magnitude of *SED* is changed and (b) only the angle between *SED*-vector and the red colour axis is changed.

At this point, we have already given the details of the adaptations to the KSA quantizer in order to reduce its computation time while preserving the image quality (to be illustrated in section 4). In addition, the proposed algorithm is more suitable for hardware implementation because it removes all hardware unrealizable parts from the algorithm.

The Pixel Mapping Stage

In this stage of an image quantization algorithm, the nearest neighbour in three-dimensional discrete colour space of a pixel *P* from the original image is searched and its colour value is used to represent *P* in the quantized version of the image. This process is iterated for all *P*s belonging to the original image. Both the KSA and our proposed quantizers utilise a similar algorithm as proposed by Wu (7). The algorithm is the "*centroid mapping algorithm*" which is very fast. During operation, it maps every pixel within a voxel under consideration to the voxel's centroid. The algorithm can effectively be used when the number of colours in the original image is smaller than the number of pixels in the image. As the proposed algorithm quantizes the original 24-bit colour into 15-bit, the overall number of colours are under 32,768. This makes it possible to employ the algorithm in this context.

In the next section, the implementation details of the proposed algorithm are given. This is, then, followed by the comparison of the experimental results obtained from different versions of quantizers in the same class in terms of both the image quality and the execution time. Finally, discussions are given.

RESULTS AND DISCUSSION

The proposed amendments to the critical parts of the KSA quantizer described in this paper were implemented on a Pentium® Centrino notebook with the following specifications: processor speed 1.4 GHz, memory 496 MB, and Windows XP operating system. The C programming language was selected for coding the previously proposed algorithms, Wu's and KSA, and the algorithm proposed in this paper. Microsoft Visual C++ version 6.0 was the tool of choice for performing software development during the course of our research. Following are five different versions of quantizers which were developed and used for comparing the efficiency improvement:

- Wu's quantizer (*WU*),
- KSA's quantizer (*KSA*),
- Our proposed quantizer which utilises the *WSND* as an indicator for selecting the axis to subdivide colour spaces, does not take square root of the squared Euclidean norm within Eq. (1), and calculates the *VRV* by use of:
  - 8 subspaces within S^* (*WCQ8P*),
  - 4 subspaces within S^* (*WCQ4P*) and,
  - 2 subspaces within S^* (*WCQ2P*).

During execution each version of these quantizers, the processor's execution times were started recording immediately after the image had already been loaded into an image buffer and ended after the pixel mapping phase had already been performed. It is noted that for all of the implemented quantizers, the centroid mapping approach was used to map the original image pixels to their nearest palette colour and produce the quantized version of the image.

Another important factor in which we are interested from the proposed algorithms is the quality of the quantized images. Actually, the evaluation of this factor is still an unsolved problem in the image processing research community. That is to say, there is still no good objective criterion available for measuring the perceived image similarity. However, the simplest and most widely used criteria is the MSE which can simply be computed by use of the following equation:

$$MSE = \frac{\sum_{x=1}^{M}\sum_{y=1}^{N}\left\|I(x,y) - I'(x,y)\right\|^2}{MN} \tag{6}$$

where $I(x,y)$ and $I'(x,y)$ are the pixel's intensity at (x,y) taken from the original and the quantized images, respectively. M and N are the height and width of the image and $\|.\|$ stands for the Euclidean distance norm. It is noted that *MSE* measures the average amount of difference between pixels of the original and quantized images. A small *MSE* indicates that the quantized image closely resembles the original one.

To compare the efficiency improvement and the outcome image quality of the quantizer, five standard test images of 512×512 pixels in 24-bit TIFF format were used to conduct the experimentations. The images were: Lena, Sailboat on Lake, Airplane, Pepper and Baboon. For each quantizer, the processor's execution times and *MSE* were collected from five required palette sizes of 16, 32, 64, 128 and 256 colours. The experimental results for all of these configurations are shown in Table 1, the images produced by the quantizers are shown in **Figure 3-8**, and a comparison between execution times and MSEs among different test images for a fixed palette colours of 256 is shown in **Figure 9**.

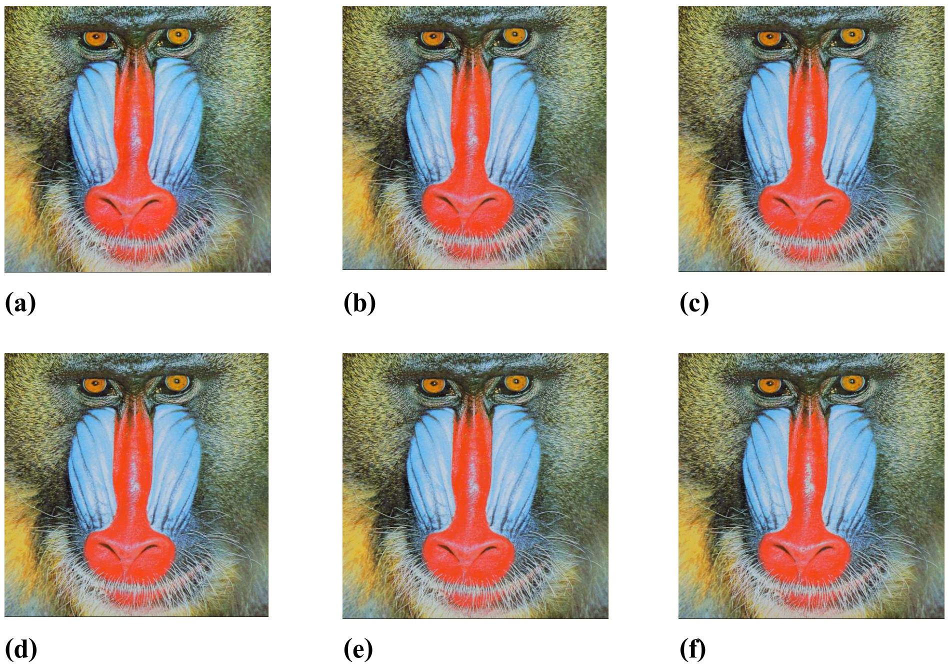


Figure 3. (a) original baboon image and its ouantized versions of 256 palette colours by: (b) Wu's quantizer, (c) *KSA* quantizer, (d) our proposed *WCQ8P*, (e) our proposed *WCQ4P* and, (f) our proposed *WCQ2P*.

From the experimental results, it is obvious that as far as the execution time is concerned, the proposed algorithms outperform all previously published colour quantization algorithms within the same class. The quantizer which utilises *WSND* indicator and employs only 2 subspaces to calculate the *VRV* without taking the square root of the Euclidean norm, *WCQ2P*, is the fastest one, approximately 2.5 times faster than the KSA quantizer and 4 times faster than Wu's quantizer. When the quality of the images is considered indirectly by means of *MSE*, it is obvious that, apart from Wu's quantizer, all quantizers have approximately equal degree of *MSE*. That is to say, the *MSE*s of the *KCQ*, *WCQ*, *WCQ4P*, and *WCQ2P* quantizers are roughly within $\left[Mean_{MSE} - SD_{MSE}, Mean_{MSE} + SD_{MSE}\right]$ for every set of the test images.

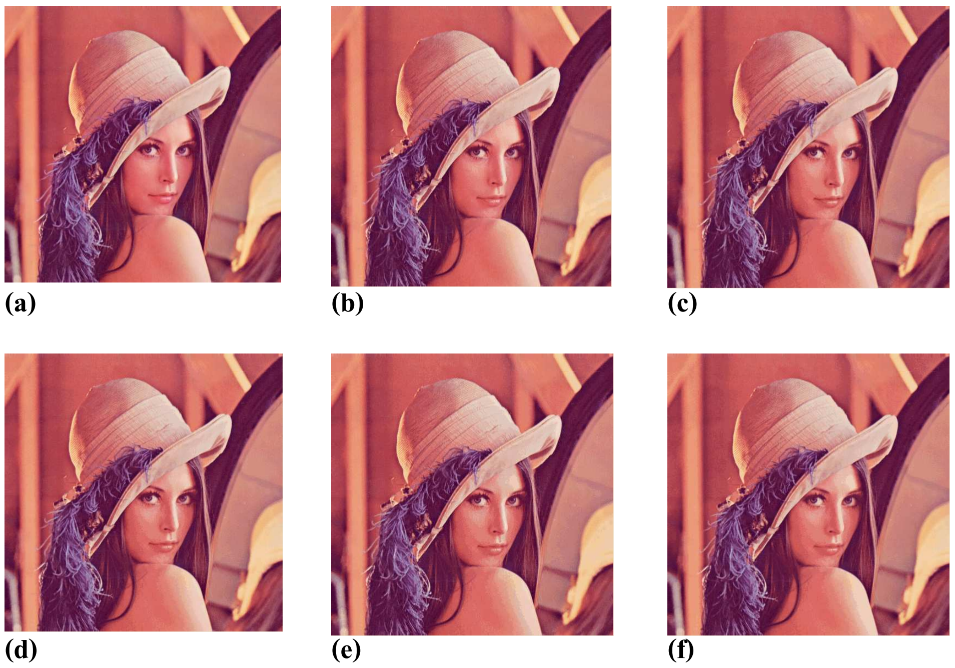


Figure 4. (a) Original lena image and its quantized versions of 256 palette colours by: (b) Wu's quantizer, (c) *KSA* quantizer, (d) our proposed *WCQ8P*, (e) our proposed *WCQ4P* and, (f) our proposed *WCQ2P*.

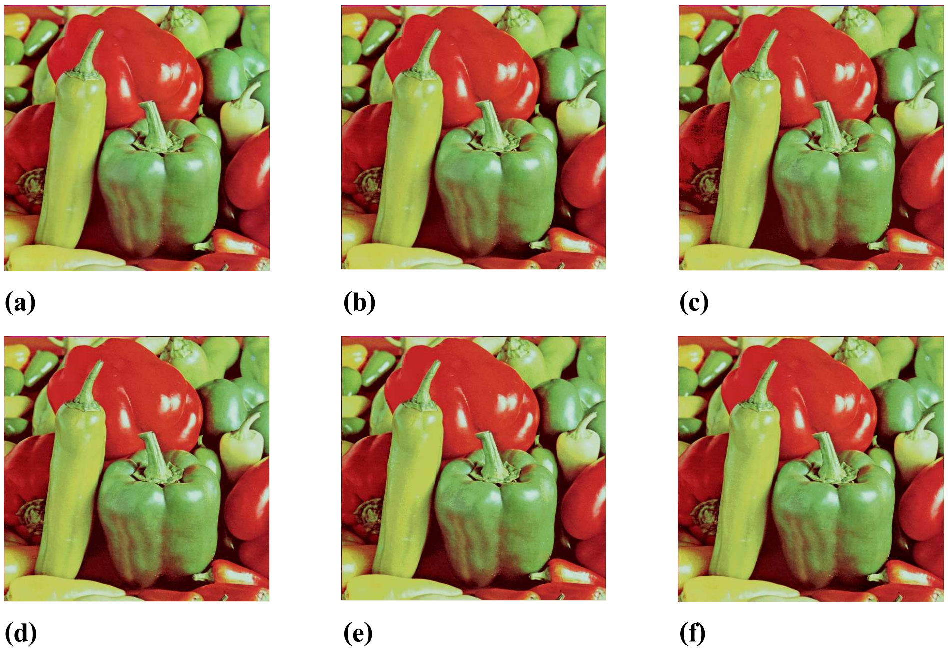


Figure 5. (a) Original pepper image and its quantized versions of 256 palette colours by: (b) Wu's quantizer, (c) *KSA* quantizer, (d) our proposed *WCQ8P*, (e) our proposed *WCQ4P* and, (f) our proposed *WCQ2P*.

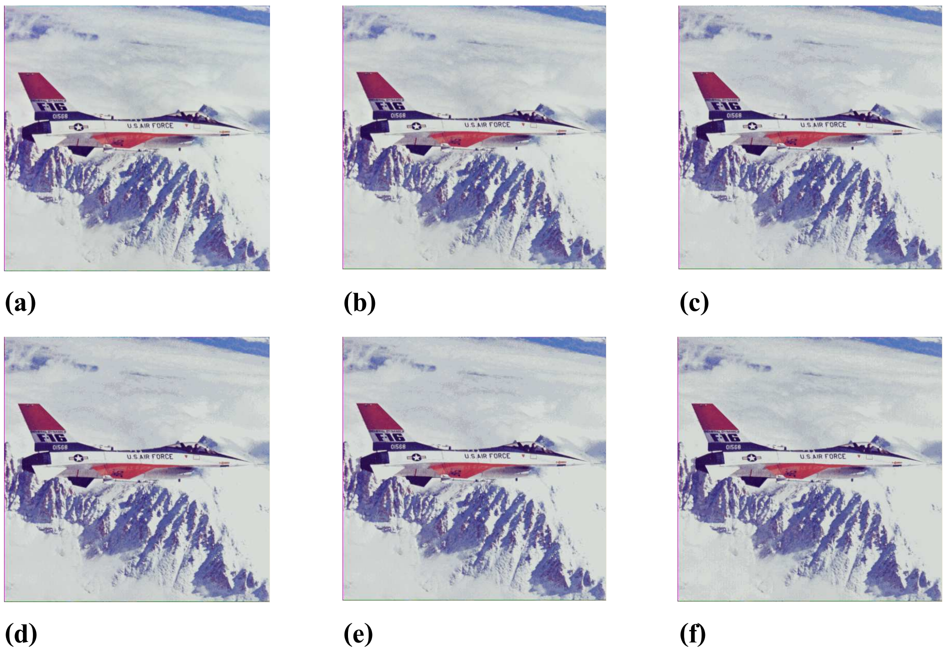


Figure 6. (a) Original airplane image and its quantized versions of 256 palette colours by: (b) Wu's quantizer, (c) *KSA* quantizer, (d) our proposed *WCQ8P*, (e) our proposed *WCQ4P* and, (f) our proposed *WCQ2P*.

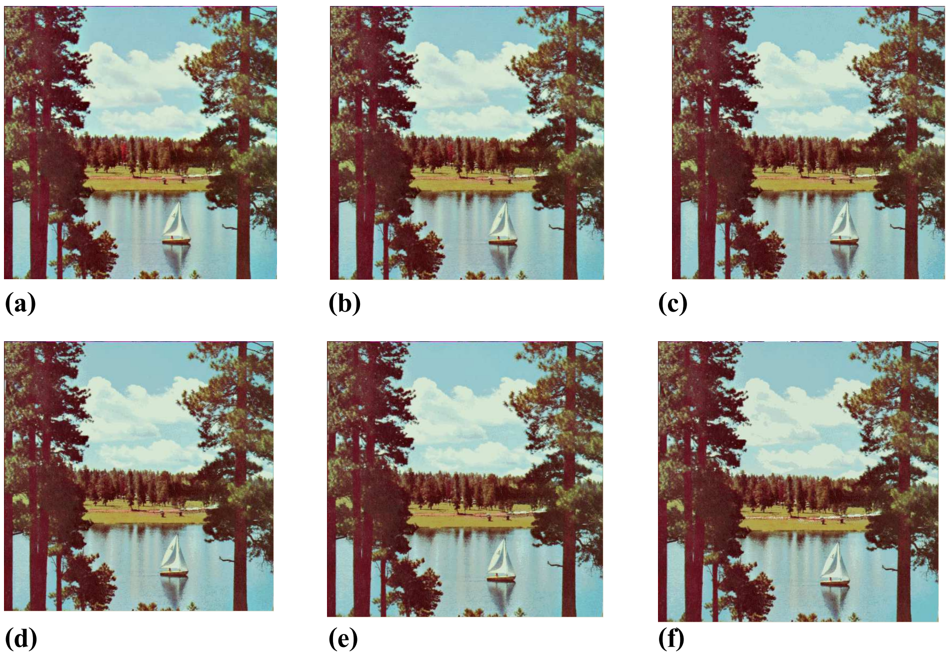


Figure 7. (a) Original sailboat on lake image and its quantized versions of 256 palette colours by: (b) Wu's quantizer, (c) *KSA* quantizer, (d) our proposed *WCQ8P*, (e) *WCQ4P* and, (f) *WCQ2P*.

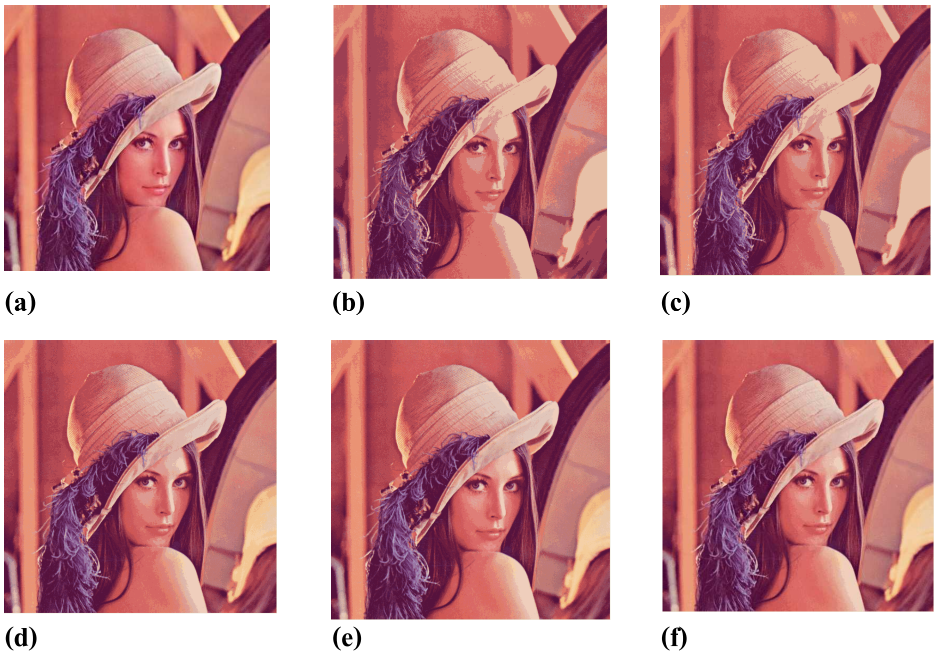


Figure 8. (a) Original lena image and its quantized by the *WCQ2P* quantizer versions of different palette colours: (b) 16, (c) 32, (d) 64, (e) 128 and, (f) 256.

From the experimentation results, we can clearly conclude that our proposed algorithm for colour image quantization has a superior performance in terms of execution time compared to all of its predecessors at the cost of comparable image quality. Above all, many complex operations found in the previously published algorithm have been substituted by simpler counterparts. This makes the algorithm more suitable to be realized in the hardware form on FPGA platforms.

CONCLUSION AND FUTURE WORK

In this paper, techniques to quantize a digital colour image in order to reduce the representation number of colours to be suitable for different type of displays were surveyed. The disadvantages of the techniques for highly restricted resource utilization and real-time constrained applications were also described. With such disadvantages in mind, we proposed a superior quantization algorithm in terms of computational time and suitability for hardware realization. The algorithm employs a simple geometry to get rid of complex operations found in previously published papers. The proposed algorithm has proved to outperform the previously proposed quantizers in the class in terms of execution time with comparable image quality. With the benefits of simple operations within the algorithm, it opens an opportunity to transform the algorithm into hardware-based implementation.

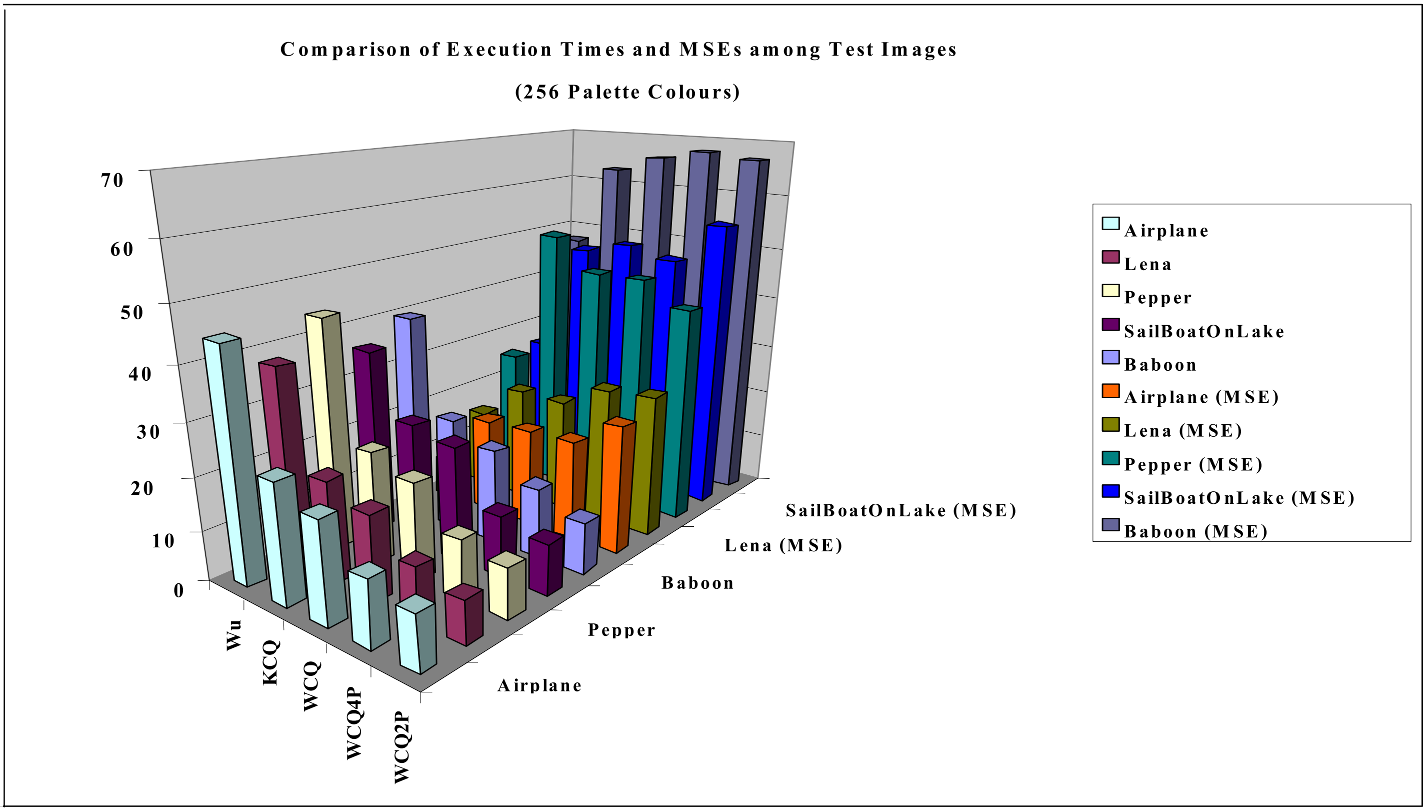
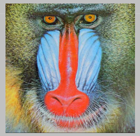
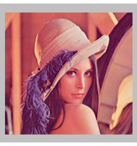
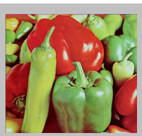


Figure 9. A comparison of execution times and MSEs among different test images for a fixed palette colours of 256.

Table 1. Experimental Results: processor execution times and mean squared errors, obtainable from difference quantizers operated upon the set of standard test images: baboon, lena, pepper, airplane, sailboat on lake.

Test Image	Palette Colours	WCQ2P		WCQ4P		WCQ8P		KCQ		WU	
		MSE	Time (mS)	MSE	Time (mS)	MSE	Time (mS)	MSE	Time (mS)	MSE	Time (mS)
	16	135.7	5	141.6	9	141.5	12	142.7	18	136.5	33
	32	127.9	8	125.5	10	123.0	14	125.5	18	118.0	35
	64	106.2	9	106.0	10	104.1	15	104.8	19	93.2	37
	128	92.8	10	83.7	13	83.0	16	84.2	20	68.5	38
	256	67.0	10	67.7	14	66.1	18	62.9	22	47.5	40
	16	94.0	7	100.0	9	100.4	12	103.0	17	84.6	31
	32	74.5	8	71.4	9	73.3	15	71.3	19	58.4	33
	64	49.0	8	52.4	10	48.7	16	50.5	18	37.4	35
	128	36.7	8	38.6	10	32.8	16	33.1	19	24.2	36
	256	27.9	8	26.9	11	22.4	17	22.2	20	15.3	38
	16	109.7	9	106.5	9	115.8	17	115.3	20	109.7	33
	32	90.9	9	96.3	10	104.4	17	101.5	20	84.7	35
	64	76.2	9	79.2	10	87.0	18	84.9	21	58.7	36
	128	60.4	10	57.6	11	61.0	18	63.8	22	39.9	36
	256	42.1	10	46.4	12	46.2	19	52.1	22	25.4	44
	16	61.5	9	73.9	10	75.7	18	77.4	20	46.5	31
	32	58.4	9	71.6	10	71.8	19	71.0	22	27.6	31
	64	43.3	9	44.0	11	44.3	18	43.2	23	18.7	33
	128	30.8	10	39.4	13	28.4	19	26.9	23	11.9	36
	256	25.1	11	19.6	13	19.0	20	18.8	23	7.9	44
	16	105.9	8	104.6	11	106.5	18	109.8	21	92.6	31
	32	98.0	8	96.7	11	97.5	19	95.4	22	75.6	32
	64	82.8	9	90.5	12	83.1	19	74.8	24	53.8	33
	128	67.7	9	65.1	12	62.6	21	56.8	24	38.6	34
	256	56.0	10	48.4	12	50.1	22	48.0	24	25.9	36

ACKNOWLEDGEMENTS

This work has been supported by Thailand's National Electronics and Computer Technology Center (NECTEC) under the "Real-Time True Colour Imaging LED dot matrix Sign Board System" project. The author would also like to thank the reviewers for their comments on the earlier versions of this paper.

REFERENCES

1) Kurdthongmee W. Design and Implementation of an FPGA-Based Multiple-Colour LED Display Board. *J of Microprocessors and Microsystems 2004; 29(7): 327-36.*
2) Scheunders P. A Comparison of Clustering Algorithms Applied to Colour Image Quantization. Pattern Recognition Letters, 18 (1997), p. 1379-84.
3) Sirisathikula Y Auwatanamongkola S Uyyanonvara B. Colour Image Quantization Using Adjacent Colours' Line Segment. *Pattern Recognition Letters 2004; 25(9): 1025-43.*
4) Cheng SC Tang CK. A Fast and Novel Technique for Colour Quantization Using Reduction of Colour Space Dimensionality. *Pattern Recognition Letters 2001; 22: 845-56.*
5) Kanjanawanishkul K Uyyanonvara B. Novel Fast Colour Reduction Algorithm for Time-Constrained Applications. *J of Visual Communication and Image Reconstruction, 2005; 16(3): p.311-32.*
6) Hsieh IS Fan KC. An Adaptive Clustering Algorithm for Colour Quantization. *Pattern Recognition Letters 2000; 21: 337-46.*
7) Wu X. Colour Quantization by Dynamic Programming and Principle Analysis. *ACM Transaction on Graphics, 1992; 11: 348-72.*
8) Heckbert P. Colour Image Quantization for Frame Buffer Displays. *J of Computer and Graphics 1982; 16(3): 297-307.*

บทคัดย่อ

วัฒนพงศ์ เกิดทองมี

อัลกอริทึมในการลดจำนวนสีของภาพดิจิตอลที่เหมาะกับงานที่มีข้อจำกัดด้านเวลา

มีหลากหลายอัลกอริทึมที่นักวิจัยในวงการการประมวลผลภาพดิจิตอลได้นำเสนอเพื่อใช้ในการลดจำนวนสีในภาพถ่ายดิจิตอลเพื่อให้เหมาะกับการนำเสนอบนจอแสดงผลหลายรูปแบบที่มีใช้งานอยู่ในปัจจุบัน อัลกอริทึมเหล่านี้ยังสามารถนำมาใช้เพื่อลดขนาดของไฟล์ของภาพถ่ายดิจิตอลอันส่งผลให้ภาพถ่ายนั้นๆ สามารถส่งผ่านทางเครือข่ายคอมพิวเตอร์ได้ด้วยความเร็วสูง ข้อจำกัดของอัลกอริทึมที่ใช้อยู่คือ ส่วนใหญ่จะถูกออกแบบเพื่อใช้งานบนคอมพิวเตอร์ที่แทบไม่มีข้อจำกัดด้านเวลาและทรัพยากรของระบบ อีกทั้งอัลกอริทึมเหล่านั้นยังยากต่อการนำไปจัดสร้างเป็นฮาร์ดแวร์คอมพิวเตอร์ ในบทความนี้ ผู้วิจัยจะได้นำเสนอรายละเอียดของอัลกอริทึมที่เหมาะสมกับการนำไปสร้างเป็นฮาร์ดแวร์คอมพิวเตอร์ที่สามารถลดจำนวนสีของภาพแบบเรียลไทม์ได้ จุดเด่นของอัลกอริทึมที่นำเสนอคือ ผู้วิจัยได้ลดความยุ่งยากและใช้เวลามากในขั้นตอนของการตัดแบ่งสเปซของการสร้างตารางสีจากภาพถ่ายดิจิตอล นอกจากนั้นอัลกอริทึมยังได้ลดจำนวนของสเปซที่ใช้ในการคำนวณค่าความแปรปรวนจากเดิมที่ใช้ 8 จุดเหลือเพียง 4 และ 2 จุด โดยผลของภาพที่ได้ยังมีความผิดพลาดอยู่ในช่วงที่ยอมรับได้ ด้วยระยะเวลาในการประมวลผลที่สั้นลง

สำนักวิชาวิศวกรรมศาสตร์และทรัพยากร มหาวิทยาลัยวลัยลักษณ์ อำเภอท่าศาลา จังหวัดนครศรีธรรมราช 80160