True Colored Image Quantization

Dr. Hisham Al-Rawi
Department of Computer Science
Amman University - Jordan

Muna Faiq
Department of Computer Science
University of Technology - Baghdad - Iraq

ABSTRACT

The problem of dealing with true colored images and palette driven images with colors that exceed the displaying capabilities of standard VGA monitors is considered. The paper is concerned with developing a quantization algorithm for properly displaying true colored images in a palette driven form with limited number of colors as per the capabilities of the displaying device. The algorithm is very useful in reducing the image file size for storing or transmission without much degradation in image quality. Results are reported and discussed.

تكميم الصور حقيقية التلوين

ملخص

تنظر الورقة في التعامل مع الصور حقيقية اللتلوين وتلك محددة الألوان التي تتجاوز ألوانها عدد الألوان القياسي في مراقب *VGA* القياسية. تهتم الورقة بتطوير خوارزمية تكميم لأغراض عرض الصور بصورة واضحة ضمن ألوان محددة تفيد عادة في تقليل حجم ملف الصورة لأغراض الإرسال أو الخزن أو العرض على شاشات محددة التلوين. النتائج معطاة ومناقشة.

تاريخ تسلم البحث: *2001/7/15* تاريخ قبول البحث: *2002/4/12*

1. Introduction

In many instances, the number of colors in an image exceeds the number of displayable colors. For example, the output from 24-bit color scanners and ray_tracing software produce what is known as *true_color* images. Which simply means that the red, green and blue (RGB) components are each eight bits wide. It is often said that the true_color image can have up to 2^{24} , or 16 million colors [1,2], but the image dimensions normally determine the upper limit; a (512*512) true_color image, for instance, cannot have more than 512^2 or (262,000) colors.

Standard VGA monitors can display images with colors selected from a 16 color palette. 256 color graphics systems are a significant improvement. Displaying true colored images using such displays may cause great degradation in image appearance and clarity. This paper deals with solving the problem of displaying true colored 24 bit images using standard (16 color) to 256 color graphics systems. It is aimed to work out quantization and palette selection algorithm to enable the display of true colored images in a palette driven form with acceptable color realism and image appearance.

2. Eight-Bit Color Basics

All 256 color workstations and PC's work pretty much the same way. The graphics hardware provides a universe of colors composed of red, green and blue primaries (RGB). varying the proportion of individual RGB primary values results in distinct colors on the screen. The range of individual values depends on the graphics devices. *UNIX* workstations typically support 256 shades of each primary color for a total of 16 million potential colors each primary is $(2^8)^3 = 2^{24}$ or 16,777,216. That's how many distinct colors the hardware is capable of, but only 256 of them can be displayed at a time. Color space is finite, it runs from black to white, or rather, from point (0,0,0) through to point (255,255,255), as can be seen in figure 1.

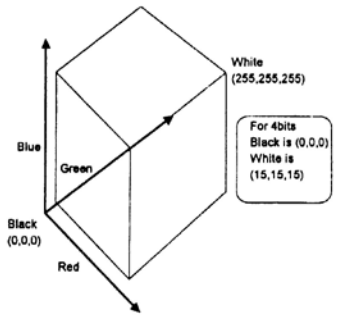

Figure 1: Color Space Cube

A 256 color VGA card supports $2^6 = 64$ shades of each primary color; putting all the primaries together gives $(2^6)^3 = 2^{18}$ or

[262,144] distinct colors. If resolution is reduced to 4-bit for each of the three color components, white is represented by the color (15,15,15) because in this 12-bit color, 15 represents a color component that is fully on, color space look like that of figure 1 yet with different scale.

3. System Description

Quantization is the process of assigning representation values to ranges of input values. In image processing, the value being quantized can be an analog or digital. *Color Image Quantization* requires selecting a set of colors to represent the color gamut of an image, and computing the mapping from color space to representative colors.

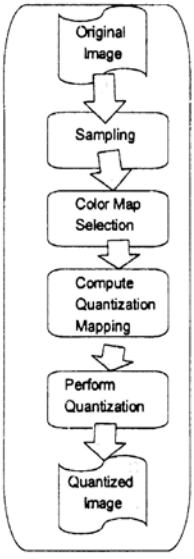

Figure 2: Colored Image Quantization System

The algorithm for color image quantization described below consists of the following four phases as in figure 2:

1- sample image to determine color distribution.
2- select colormap based on the distribution.
3-compute quantization mapping from 24-bit colors to representative colors (color in the colormap).
4-redraw the image, quantizing each pixel.

Choosing the colormap is the most challenging task. Once this is done, computing the mapping table from colors to pixel values is straightforward.

3.1 Phase 1: : Sampling The Original Image.

The information needed by the colormap selection algorithms of phase 2 is a histogram of colors in the original image. This is collected in one pass over the input image. To conserve

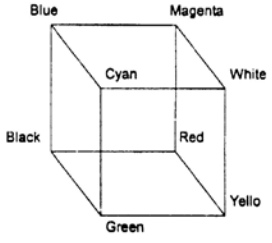

Figure 3: All Colors are inside the Color Cube

memory, a prequantization from 24 bits to 12 bits (4 bits red, 4 bits green and 4 bits blue) is suggested. In this case the color frequency histogram will be a table of length 4096. This clumping of the colors has the effect of reducing the number of different colors and increasing the frequency of each color. These properties are important to the algorithms described below.

3.2 Phase 2: Choosing A Colormap

The maximum number of possible distinct colors in a computer image is its width in pixels times its height in pixels; that is, there can be no more than one distinct color per image pixel. A [640*480] image for example, can have as many as [307,200] different colors. A quantization function should map each distinct color in the original image to one of the colors in your palette table so that when the image is displayed , it looks like the real thing

Two algorithms were developed for choosing a set of representative (colormap) based on the input distribution. These are the popularity algorithm and the median cut algorithm.

3.2.1 The Popularity Color Quantization

The popularity algorithm uses the most frequently occurring colors in the image as the palette colors. Every other color is then mapped to the popular color it is closest to. In this method image is to be scanned, building a list of all colors found in the image. Colors are to be sorted in descending order by count (ties can be broken arbitrarily) and the top 256 colors are selected for the palette map. The image is then rescaned to map image colors to palette colors[3,4].

For image colors that don't exactly match a palette color, The closest color in the palette is used. The closest color in a qualitative sense is the color in the table that is the least distinguishable from the color in the image. In quantitative terms it is the color in the table that is the shortest straight_line distance from the actual color. Remembering that all the colors are inside a 3-D cube(figure 3); then the distance between two colors is d=|c2-c1| , where c1 and c2 are two points in the color cube defined by their individual R,G and B values. In algebraic terms

$$d = \sqrt{(r2-r1)^2 + (g2-g1)^2 + (b2-b1)^2} \quad ...(1)$$

Where d is always nonnegative because of the squared terms and implying that (c2-c1)=(c1-c2). If d is 0, then the colors are the same. Figure 4 shows the flowchart of the popularity algorithm.

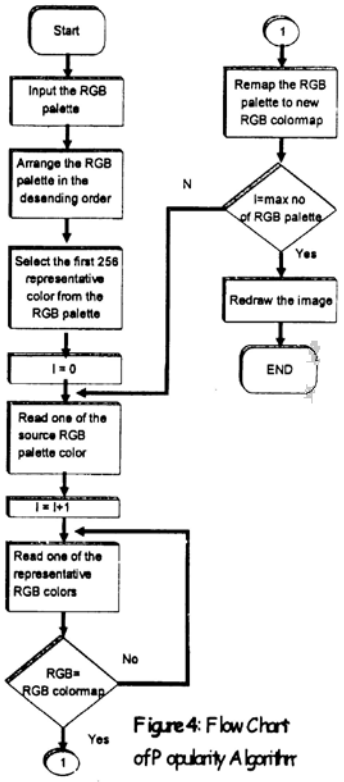

Figure 4: Flow Chart of Popularity Algorithm

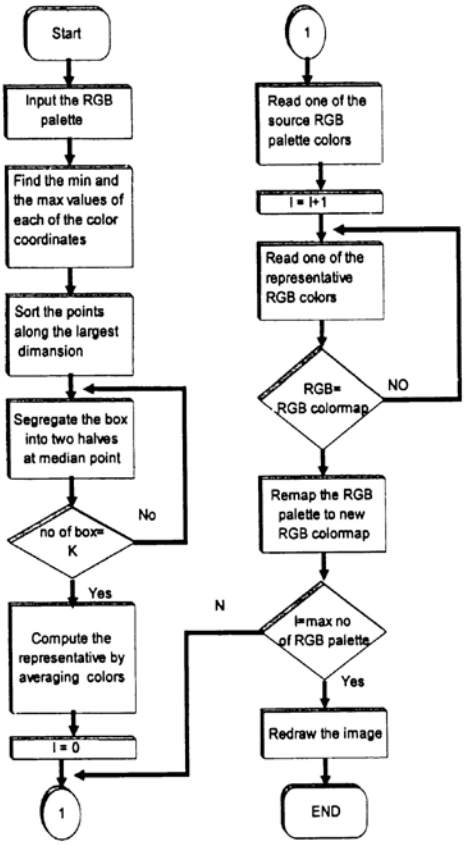

Figure 5: Flow Chart Of The Median Cut Algorithm

3.2.2 The Median-Cut Color Quantization

The aim of median-cut algorithm is to have each of the 256 output colors represent the same number of pixels in the input image. The starting point is the RGB cube that corresponds to the whole image, around which a tight_fitting cube is placed. The cube is then split at the median of the longest axis. This ensures that about the same number of colors is assigned to each of the new cubes. The procedure is recursively applied to the two new cubes until the required number (say 256 cubes) are generated. The centroide (average values) of the cubes become the 256 output colors [5].

The time-consuming part of the median cut quantization is finding the median along an axis of a cube. Algorithms exist for finding the median of a set of numbers with a run-time complexity of O(N). The following method was used. First the numbers are sorted in ascending order and their sum is computed. Then starting at the smallest number a running sum is computed. When the running sum is equal to half the total, then that is the median. The C standard-library function qsort (based on the Quicksort algorithm with its average run-time complexity of $O(N \log_2 N)$ was used to do the sort. However, the algorithm has a potentially serious pitfall. When it is called to sort data that is already in order, the run-time complexity is $O(N^2)$. The difference between an algorithm with complexity $O(N \log_2 N)$ and one with $O(N^2)$ is dramatic when N is large. Another problem with Quicksort is that it is often implemented as a recursive routine, and can rapidly deplete the program stack when a large number of data points must be sorted. A nonrecursive implementation needs much less auxiliary storage. Thus, a good implementation of Quicksort is normally nonrecursive, and it randomizes the input data some what to provide for the case were the data is already in sorted order.

3.3 Phase 3: Mapping Colors To Nearest Neighbors In Colormap.

Given an input distribution c and a set of representative y , D(distance) is minimized when q maps a point to its nearest representative. This operation is sometime called a "nearest neighbor query" or "inverse colormap", since it maps colors into pixel values. By evaluating this function for each color in the original image, and saving this information in a table, one can speed up phase 4 significantly. The alternative is to evaluate p once per pixel. The former will be faster if the number of different colors in the original image is smaller than the number of pixels in the image. For computing the function p one can do an

a- Exhaustive Search :

The straightforward way to compute *p(x)* is to test all representative and choose the one which minimize the distance. This method is slow, that why in this paper *p(x)* was calculated using what can be called a locally sorted search.

b- Locally Sorted Search

A database is created consisting of an N*N*N lattice of cubical cells each containing a sorted list of representative. Each cell's list should include all representative which are the nearest neighbors of some point in that cell. Each list entry contains two variables: a representative number and its distance from the nearest point in the cell. Distance is defined to be zero for representative inside the cell. To create the list, each representative's distance is computed from the cell, put these in the list, and then sort the list by distance key. A given representative can occur in several lists.

3.4 Phase 4 : Quantizing And Redrawing The image

To quantize the image, simply each pixel of the original image is passed through the quantization mapping table created during phase 3, and the pixel values is written into a frame buffer. The image is then redrawn using only k colors.

4. Testing Results

The developed quantization (software) system was used to quantize several true colored images into palette driven images of different number of color palettes. Among these images we report here a true colored image of the size of 192x144 pixels shown in figure 6a. Figure 6 shows the system output using the popularity quantization. 8, 7, 6 and 5 bits palette driven images obtained from the 24 bit true colored image of figure 6a are shown in the figure. Image histograms are shown in figure 7 for the 256, 128, 64, 32 colors palette driven images obtained from the system output.

In comparison with the images obtained using the median cut quantization one may not notice any difference for 8 or 7 bit images as can be seen from the cube images shown in figure 8. However the palettes and the histograms are little bit different as can be concluded from comparing figure 7 with figure 9 that shows the histogram of the images obtained using median cut quantization.

Difference in image appearance using both popularity and median cut quantization start to be distinguishable (for the viewer) when obtaining 6 bits or less images. This difference can be seen very clear in the 5 bits images of figures 6e and 8e. The image obtained using median cut quantization has better color appearance than that obtained using the popularity quantization.

5. Conclusion

The developed color quantization system proved to be producing good appearance colored palette driven images with unnoticeable color differences from the original true colored image. This is believed to be completely true and image independent using either of the quantization methods considered if quantization is from infinite colors to 256

colors or even to 128. However, color quantization errors may be large enough to be noticeable if less colors are used. Results reported in this paper showed acceptable image color appearance for 8, 7, 6 and 5 bit images using the median cut quantization.

The popularity quantization might miss small but visually significant areas of colors if non of them are popular enough. It may work well for many images, but it performs poorly on those with wide range of colors or small number of colors!. Although, the median cut quantization is better than the popularity; yet in many cases the differences are visually insignificant

Depending on the image, the quantization errors may be obvious or invisible. Image with high spatial frequencies will show quantization errors much less than pictures with large, smoothly shaded areas. However, it is expected that some of the images will have acceptable clearness using the popularity quantization and others may require to be quantized using the median cut. In both cases color correction some times may be required but can not be done due color limitation. The authors are currently working of developing color image enhancement techniques to help improve the color appearance of quantized images.

References

1- M. Luse, A. Binstock and J. Rex ; "*Bitmapped Graphics Programming in C++.*" Wesley 1993.

2 - *S . Rimmer ;"Super charged Bitmapped Graphics " , Windcrest /Mc Graw Hill, 1992* .

3- P. Hecbert; "Color Image Quantization for Frame Buffer Display." *computer graphics* vol 16 no 3 pp 297-307, July 1982.

4 - D. Clark; " The popularity Algorithm. " *Dr.Dobb's Journal* , pp 121-127 July 1995.

5- A. Kruger; "Midian_Cut Color Quantization ". *Dr.Dobb's journal* , pp 46-53 September 1994.

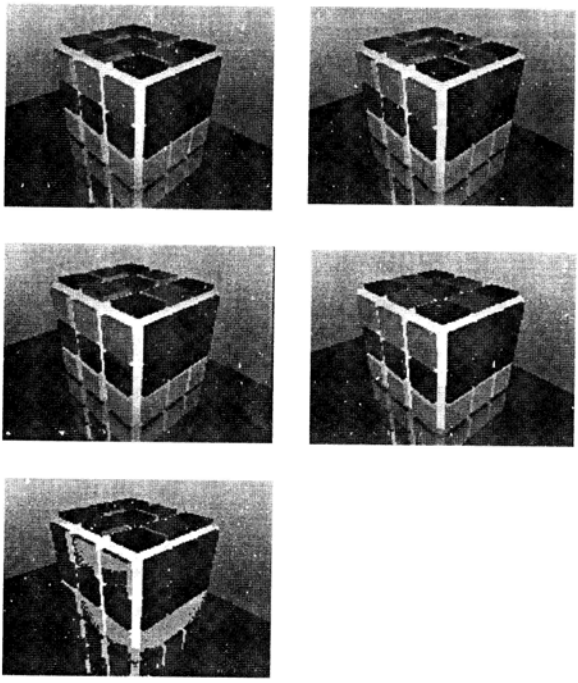

Figure 6: Popularity Quantization
(a) 24 bits True colored Image.
(b) 8 bits 256 colors palette driven image.
(c) 7 bits 128 colors palette driven image.
(d) 6 bits 64 colors palette driven image.
(e) 5 bits 32 colors palette driven image.

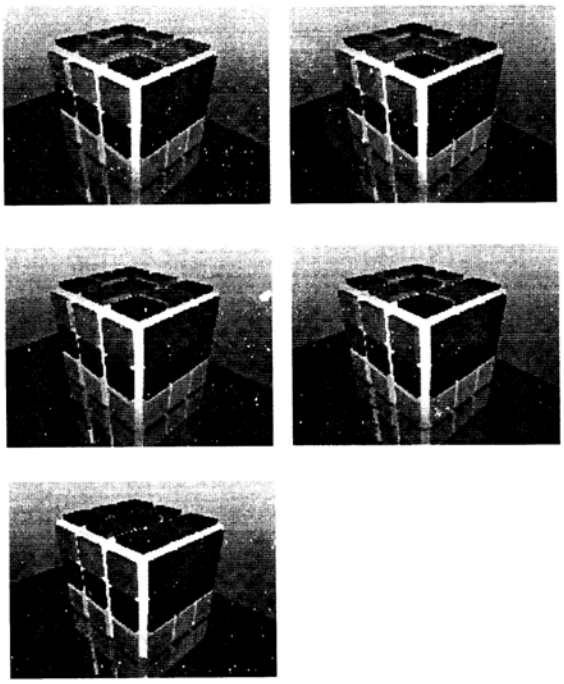

Figure 8: Median_Cut Quantization
(a) 24 bits True colored Image.
(b) 8 bits 256 colors palette driven image.
(c) 7 bits 128 colors palette driven image.
(d) 6 bits 64 colors palette driven image.
(e) 5 bits 32 colors palette driven image.

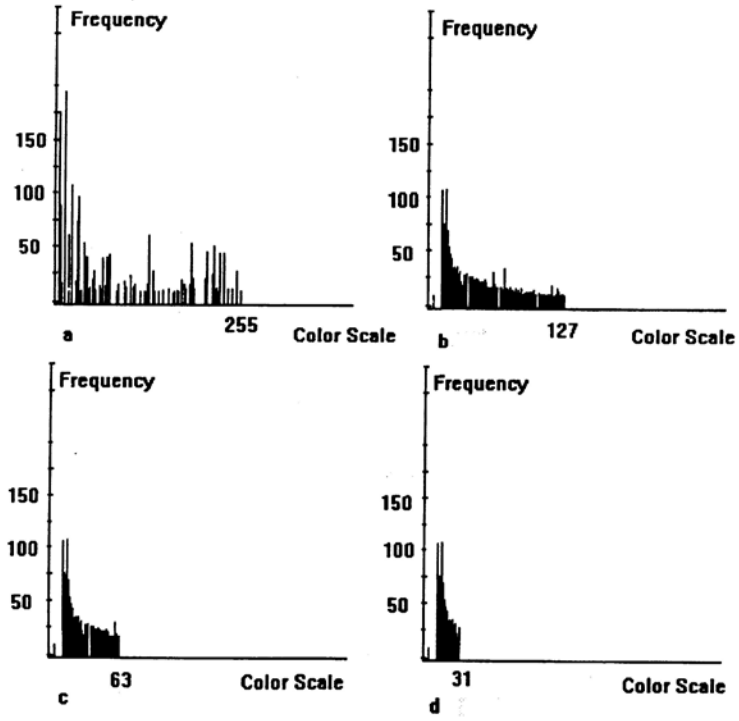

Figure 7: (a) Histogram of the image of figure 6b.
(b) Histogram of the image of figure 6c.
(c) Histogram of the image of figure 6d.
(d) Histogram of the image of figure 6e.

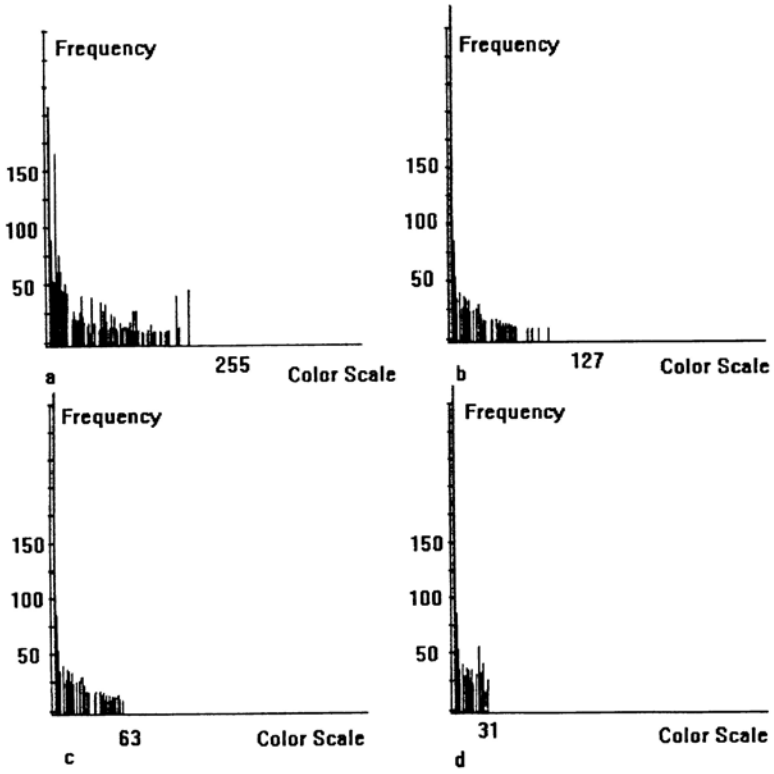

Figure 9: (a) Histogram of the image of figure 8b.
(b) Histogram of the image of figure 8c.
(c) Histogram of the image of figure 8d.
(d) Histogram of the image of figure 8e.