Color Image Quantization in Windows Systems with Local K-means Algorithm

Oleg Verevka
Department of Computing Science
University of Alberta
Edmonton, AB., Canada
T6G 2H1
oleg@cs.ualberta.ca

Abstract

Color image quantization is a process of representing an image with a small number of well selected colors. Commonly used color quantization algorithms take a recursive pre-clustering approach. Such techniques do not account for complex interrelationships between color clusters. They also tie the error minimization process to the rough approximation of the quantization regions.

This research deals with a color quantization problem for windows systems. In such environments it is desirable to account for already allocated colors in the shared color map. The author believes that pre-clustering algorithms cannot be easily modified to compute a palette that includes a subset of predefined colors.

This work examines the use of a recently designed local K-means algorithm. The paper describes how this post-clustering scheme can be modified for the efficient quantization within windows systems. A number of experiments suggest that it is a feasible approach able to preserve image quality even when many images should be displayed simultaneously.

1 Introduction

Color quantization is one of the most frequently used operations in computer graphics and image processing. Traditionally, quantization is used to reproduce 24 bit images on a graphics hardware with a limited number of simultaneous colors (i.e. frame buffer displays with 4 or 8 bit colormaps). Even though 24 bit graphics hardware is becoming more common, color quantization still maintains its practical value. It lessens space requirements for storage of image data and reduces transmission bandwidth requirements in multimedia applications.

Color quantization is usually defined as a *lossy image compression operation* that maps an original full color image to an image with a small color palette. The objective of computer graphics research in this area is to select an optimal palette. Such a palette will ensure minimization of a perceived difference between the original and quantized images.

Let c_i be a 3-dimensional vector in one of the color spaces (Lu^*v^*, HSV, RGB etc.). The set $C = \{c_i, i = 1, 2 \dots N\}$ is the set of all colors in the full color image I. A quantized image $\overline{I}$ is represented by a set of K colors $\overline{C} = \{\overline{c_j}, j = 1, 2 \dots K\}, K \ll N$. A quantization is a mapping:

$$q : C \rightarrow \overline{C} \tag{1}$$

It is often assumed that a color set $\overline{C}$ can contain any color of the chosen color space without any restriction.

The paper deals with a special instance of the quantization problem, where K^* colors of the palette $\overline{C}$ are predefined. The quantization algorithm is free to choose only $K - K^*$ colors. Such a problem often arises in applications running within various windows environments (e.g. X or Microsoft Windows). All applications of a such environment may share a common color map. The windowing system reserves a small number of colors K^* for its own purposes such as windows' borders, standard icons, etc. When an image has to be displayed, the windowing system allocates the image colors in the common color map thus increasing the number K^*. The image should be quantized if its palette is larger than $K - K^*$ available colors.

Clearly, the described color allocation problem can be reduced to the general quantization problem. It is possible to quantize an image to $K - K^*$ colors without taking into account the rest of K^* cells of the palette. Unfortunately, quantization to a small number of colors significantly distorts the appearance of the displayed image.

This research attempts to answer the question: *How a quantization algorithm can account for the previously allocated colors ?*

The author proposes to use a recently developed local K-means quantization algorithm. The paper describes how this algorithm relates to the other commonly used quantization techniques and discusses its advantages in the solution of the stated special quantization problem.

2 Commonly used quantization algorithms

We define quantization as a mapping (1). It is based on the *nearest neighbor principle* that maps each color c of the original image I into the closest color $\overline{c}$ from a reduced color set $\overline{C}$:

$$q(c) = \overline{c}: \quad \|c - \overline{c}\| = \min_{j=1,2\ldots K} \|c - \overline{c_j}\|, \tag{2}$$

where $\| * \|$ is one of the norms in the chosen color space.

We say that colors of the original image that have the same nearest color in the reduced color palette define a cluster of a color space. Thus the overall procedure creates a Voronoi tessellation of the space. A color mapping operation substitutes each color of the original image by a center of its cluster. The goal is to minimize a distortion error introduced by the mapping q within all clusters simultaneously.

Such a minimization problem is at least as hard as the construction of an optimal binary decision tree [WPW90]. The latter is known to be NP-complete if the global minimal solution is sought. Taking into account the complexity of the quantization problem and the large number of pixels in an average image, previous research is concentrated on the generation of a local minimum solution.

The most popular algorithms described in the literature take one of the two possible approaches; *pre* and *post*-clustering (see [Dek94]).

2.1 Pre-clustering algorithms

Common computer graphics quantization techniques — median cut [Hec82], variance-based [WPW90] and octree algorithm [GP88], [CFM93] — use a pre-clustering approach. These algorithms recursively subdivide a color space into a set of clusters represented by boxes with faces parallel to the axis. Colors of the palette are chosen as centers of these clusters. The subdivision is meant to approximate Voronoi tessellation of the color space generated by the quantization process. As a result, the minimization process is tied not to the actual Voronoi regions but to their rough approximations.

The author believes that post-clustering algorithms are not suitable for the efficient calculation of a palette within a windows system. A recursive subdivision is not able to account for the already allocated colors in the color map.

2.2 Post-clustering algorithms

The current research suggests that a *post-clustering* scheme can be more appropriate for this special case of the quantization problem. Algorithms of this type do not rely on the approximations of the Voronoi regions. Post-clustering algorithms try to find representative colors first. These colors define the Voronoi tessellation used to quantize the image.

The process starts with $\overline{C^0}$—the initial approximation of the color palette. The colors of the original image are used to iteratively adapt the set $\overline{C^{(t)}}$ and to minimize the quantization error. The algorithm terminates when a change of the colors on the t iteration is less than a predefined value.

Post-clustering techniques have been actively studied in statistical analysis, data coding, signal processing and pattern recognition. One of the most popular computation schemes of this type is the K-means algorithm [LBG80]. It is known to converge to a local minimum. Application of the K-means quantization to color images was compared to the median-cut and variance-based methods in [WPW90]. For the test images it produced smaller average errors than pre-clustering algorithms, but was unacceptably slow.

Another post-clustering scheme is a Kohonen self-organizing map [KKL90]. It was used for color quantization in [Dek94]. This method produced quantized images of a better quality than octree and median-cut pre-clustering schemes. It requires significantly smaller amount of memory but runs much slower than its competitors.

The recently described local K-means algorithm (LKM) [VB95] can be considered as a special case of the Kohonen map. The authors found a way to speed up the iteration process to make the algorithm even faster than known pre-clustering methods.

This research is an extension of the LKM method to the quantization for windows systems.

3 Local K-means algorithm.

The local K-means algorithm is a typical post-clustering method. It is an iterative process that adapts the initial approximation of the color palette to the distribution of colors in the input image. Similar to the Kohonen maps, the basis of the LKM procedure is the competitive learning principle.

Suppose, $c^{(t)} \in C$ is an input of the t-th iteration. The distance from the input $c^{(t)}$ to all the entries in the current palette is computed and the best-matching entry $\overline{c_k}^{(t)} \in \overline{C^{(t)}}$, the "winner", is defined by the following rule:

$$\|c^{(t)} - \overline{c_k}^{(t)}\| = \min_{j=1,2\ldots K} \|c^{(t)} - \overline{c_j}^{(t)}\| \tag{3}$$

Competitive learning means that only the winner is updated to better comply with input $c^{(t)}$, that is:

$$\overline{c_j}^{(t)} = \begin{cases} \overline{c_j}^{(t-1)} + \alpha_t \|c^{(t)} - \overline{c_j}^{(t)}\| & j = k; \\ \overline{c_j}^{(t-1)} & otherwise \end{cases} \tag{4}$$

The adaptation parameter α_t is a decreasing sequence that ensures convergence of a solution to a local minimum:

- $0 < \alpha_t < 1$;
- in order to guarantee stability of the process (i.e. the algorithm does not forget information already learned) the sequence must not decrease too fast:

$$\sum_t \alpha_t \to \infty; \tag{5}$$

- the adaptation process has to be flexible to respond to a new input on every iteration, therefore the sequence does not decrease too slowly:

$$\sum_t {\alpha_t}^2 < \infty. \tag{6}$$

Verevka and Buchanan in [VB95] suggest to start the iteration process with the initial palette $\overline{C^{(0)}}$ created by random sampling of the original image. A color $\overline{c_k^{(0)}}$ is inserted into the initial palette only if it lies on a significant distance from the previously allocated colors.

In the course of iterations the algorithm uses a series of input sets constructed by sampling the image in decreasing step sizes. If the step sizes are prime numbers then the input sets do not intersect too much.

The experiments in [VB95] show that the iteration process usually stops after accessing at most 15% of all pixels in the image.

3.1 The adaptation of the LKM algorithm to quantization for windows systems

The local K-means algorithm can be easily adapted to solve the quantization problem for windows environments outlined before.

Suppose, the common color map contains K^* colors. Then these fixed colors can be inserted into the initial approximation of the palette $\overline{C^{(0)}}$. All the other $K - K^*$ entries are chosen.

The LKM adaptation procedure (4) is modified as follows:

$$\overline{c_j}^{(t)} = \begin{cases} \overline{c_j}^{(t-1)} + \alpha_t \|c^{(t)} - \overline{c_j}^{(t)}\| & j = k, k > K^*; \\ \overline{c_j}^{(t-1)} & otherwise \end{cases} \tag{7}$$

This modification ensures that the fixed K^* colors of the shared color map will be effectively complemented by the other $K - K^*$ newly chosen colors. All the colors in the new common palette will take equal part in the approximation of the original image, as if there was no restriction of the windowing system.

4 Fast nearest neighbor search.

Iterative post-clustering techniques were usually considered to be too slow for color image quantization. The reason lies in the cost of the nearest neighbor search that is required on every iteration of these algorithms.

In order to speed up the search Freidman et. al. proposed to use $k-d$ trees [FBF77]. In his software Poskanzer implements the search using various hash functions (see in [Pos91]). Unfortunately these techniques cannot be used in the framework of such iterative procedures as local K-means. Positions of possible representative colors $\overline{c_j}$ are constantly changing, therefore a $k-d$ tree or a hash table must be adjusted after every iteration.

It is possible to optimize the nearest neighbor search at the expense of its accuracy. For example, the Euclidean L_2 norm in (3) can be substituted by computationally less expensive L_1 norm (see [Dek94]). Unfortunately, the nearest neighbor determined by L_1 norm is not necessarily the nearest neighbor in L_2 norm.

The authors of the local K-means algorithm proposed to use the L_α norm [CCW92] as an approximation of the Euclidean norm. For a vector $x \in R^n$ the L_α norm is defined as follows:

$$\begin{aligned} \|x\|_\alpha &= (1-\alpha)\|x\|_\infty + \alpha\|x\|_1 \\ &= (1-\alpha)\sum_{i=1}^{n} |x_i| + \alpha \max_i |x_i|. \end{aligned} \tag{8}$$

A number of experiments showed that an $L_{\alpha=1/2}$ is a good approximation of the Euclidean norm. In comparison with the L_1 norm, the nearest neighbor search with $L_{\alpha=1/2}$ results in about three times less misclassification. Moreover, it runs almost four times faster than an algorithm based on the Euclidean norm.

The computing time of the search can be further reduced using the following considerations [Hod88]:

- *Calculation of a partial sum.*
  Before each addition in the norm calculation (8) a partial sum Σ_p is compared with the current minimum distance Σ_{min}. The norm calculation terminates if $\Sigma_p > \Sigma_{min}$
- *Sorting on one coordinate.*
  The palette colors are sorted using one of the coordinates. Suppose, that the first coordinate is chosen. The search selects palette entries in the increasing first-coordinate distance order starting with the closest color. This process terminates when the first coordinate distance between the next palette entry and the input is larger than the current minimum Σ_{min}.

- *Nearest neighbor distance.*
  The search for the nearest color should terminate when Σ_{min} is less than the half of the distance from the current palette color to its closest palette neighbor.

These shortcuts where tested using the standard test image "Lenna". The experiments in [VB95] showed that the above considerations allowed to bring the performance of the color mapping algorithm close to that of k-d trees.

The optimization of the nearest neighbor search made the local K-means algorithm significantly faster than common pre-clustering techniques (see Table 1).

Table 1: Execution time in seconds for color map generation (image "Lenna")

Algorithm	16 colors	256 colors
Median-cut	1.73	1.92
Octree	0.84	1.39
Kohonen SOM		41.43
Local K-means	0.31	0.61

5 Quantization errors

The goal of the quantization process is to make the perceived difference between the original image and its quantized representation as small as possible. It is difficult to formulate a definite solution to the image quantization problem in terms of *perceived* image quality. In fact, there is no good objective criterion available for measuring the perceived image similarity.

In the color quantization literature it is common to use image dependent distortion measures (see [Hec82], [WPW90], [Wu92]). Let an image I be an array of M pixels (x, y), then $c_{(x,y)}$ is a color of each image pixel. The average quantization error per pixel can be defined as follows:

$$\epsilon_{q(C,I)} = \frac{1}{M} \sum_{(x,y)\in I} \|c_{(x,y)} - q(c_{(x,y)})\| \tag{9}$$

where $\| * \|_2$ is the Euclidean L_2-norm. Wu in [Wu92] recommends to use CIE Lu^*v* space where the Euclidean norm can reasonably approximate a perceived color difference.

Even though an average distortion measures can give a reasonable estimate of the perceived image differences, it also can be very misleading. It is often the case that significant image information is carried by very "rare" colors (e.g. specular highlights). A small $\epsilon_{q(C,I)}$ value does not guarantee that these "rare" colors of the original are well represented in a quantized image.

In the course of experiments Wan et. al. [WPW90] found that an image with a small average quantization error can be significantly distorted. Since the human visual system is more sensitives to color differences than to absolute color values, the measure of quantization errors should characterize the preservation of color variations.

Verevka and Buchanan [VB95] proposed the use of a combination of the the average distortion per color:

$$\epsilon_{q(C)} = \frac{1}{N} \sum_{i=1}^{N} \|c_i - q(c_i)\|, \tag{10}$$

and the standard deviation of the distortion per pixel:

$$\sigma = \sqrt{\frac{\sum_{(x,y)\in I} (\|c_{(x,y)} - q(c_{(x,y)})\| - \epsilon_{q(C,I)})}{M}}. \tag{11}$$

The goal of the quantization process is to minimize both values simultaneously. A color mapping with small deviation of errors σ introduces approximately equal color distortion to every pixel. Thus such an

Table 2: Quantization errors for 16 colors palette for image "Kiss" (Painting by Gustav Klimt)

Method	Max	$\epsilon_{q(C)}$	$\epsilon_{q(C,I)}$	σ
Median cut	161	29.8	20.77	14.71
Variance based	146	25.4	17.63	11.57
Octree	133	26.4	19.41	13.65
Local K-means	102	20.4	26.44	9.49

algorithm is able to preserve variations of colors and to minimize the perceived difference between the original and quantized images.

The local K-means algorithm was able to generate color maps with small values of the proposed quantization errors (Table 2). Images quantized using the LKM chosen palettes seemed to reproduce a full chromatic range of colors of the original. The perceived quantization fidelity was more apparent for images with high dynamic ranges and very uneven distribution of colors.

6 Experiments with quantization for windows systems

In order to show the application of the local K-means algorithm for quantization in a windows system a number of experiments performed. These tests intended to simulate commonly used techniques in color allocation and mapping (e.g. Mosaic for X Windows).

It was assumed that the windows system is able to display $K = 256$ simultaneous colors and reserves for its own use $K^* = 8$ cells. A simulated windows application generates a small palette with 32 colors for each displayed image. These small palettes are added to the shared color map of the windows system, thus decreasing a number of free cells by 32. In order to display an image a quantization process can either use the entire color map or just a previously generated palette.

In the course of the experiment three images were quantized to 32 colors, so the shared color map contained $K^* = 104$ entries. Quantization results for the third image are presented in Table 3. The original version of the local K-means algorithm was used to find a palette. The image was displayed using all 104 shared colors. Even though the quantization algorithm generated the palette independently from the entire color map, the numbers in Table 3 indicate that utilization of all available colors reduces quantization errors.

On the second stage of the experiment the local K-means procedure generated a 32 color palette taking into account already allocated 72 entries of the shared map. This allowed to improve the display of the image.

7 Conclusions

This work was an attempt to develop an algorithm that solves a quantization problem within windows systems. Even though this problem can be reduced to the general image quantization problem, it is desirable to choose a palette taking into account the already allocated colors.

Table 3: Quantization errors for local K-means algorithm applied in windowing systems, image "Kiss"

Colors	Max	$\epsilon_{q(C)}$	$\epsilon_{q(C,I)}$	σ
Original LKM				
32	85	17.8	17.8	6.52
104	72	15.2	17.0	6.02
Windows LKM				
104	72	14.0	15.1	5.52

This work explores the advantages of the post-clustering of the image quantization techniques. The author suggests that a recently described local K-means algorithm can be easily modified for application in windows environment. A number of preliminary experiments indicate that this is a feasible approach. The modified LKM procedure was able complement the set of previously allocated colors by a small palette. This resulted in reduction of the overall quantization error for the tested images. Moreover, an application of this algorithm tends to create shared color maps that do not contain similar colors. Such maps allow the display of more images without significant quantization distortion.

The above conclusions are derived from a series of simulations. In order to prove their correctness an implementation of a complete windows application is necessary.

References

[CCW92] C. A. Chaudhuri, W. T. Chen, and J. S. Wang. An modified metric to compute distance. *Pattern Recognition*, 7(25):667–677, 1992.

[CFM93] P. Coltelli, G. Faconti, and F. Marfori. On the application of quantization and dithering techniques to history of arts. *EUROGRAPHICS*, 12:351–362, 1993.

[Dek94] Anthony H. Dekker. Kohonen neural networks for optimal colour quantization. *Network: Computation in Neural Systems*, 5:351–367, 1994.

[FBF77] J. H. Friedman, J. L. Bently, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, September 1977.

[GP88] M. Gervautz and W. Purgathofer. A simple method for color quantization: Octree quantization. In *New Trends in Computer Graphics*, pages 219–231. Springer-Verlag, New York, NY, 1988.

[Hec82] Paul Heckbert. Color image quantization for frame buffer display. *ACM, Computer Graphics*, pages 297–304, 1982.

[Hod88] Michael H. Hodgson. Reducing the computation requirements of the minimum stance classifier. *Remote Sencing of Environment*, 25:117–128, 1988.

[KKL90] Jari A. Kangas, Teuvo A. Kohonen, and Jorma T. Laaksonen. Variants of self-organizing maps. *IEEE Transactions on Neural Networks*, 1(1):93–99, 1990.

[LBG80] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communication*, 28(4):84–95, January 1980.

[Pos91] Jef Poskanzer. `PPMQUANT`, 1991. PBM+ Image Processing Software Package.

[VB95] Oleg Verevka, John Buchanan. Local K-means algorithm for color image quantization. To appear in Proceedings of Graphics Interface 95, Quebec City, Canada, 1995.

[WPW90] S. J. Wan, P Prusinkewicz, and S. K. M. Wong. Variance-based color image quantization for frame buffer display. *Color Research and Application*, 15(1):52–58, February 1990.

[Wu92] Xiofan Wu. Color quntization by dynamic programming and principal analysis. *ACM Transactions on Graphics*, 11(4):348–372, October 1992.