Color quantization

Article · January 2003

CITATIONS
34

READS
816

2 authors:

Luc Brun
French National Centre for Scientific Research
201 PUBLICATIONS **1,821** CITATIONS
SEE PROFILE

Alain Treméau
Université Jean Monnet
251 PUBLICATIONS **2,094** CITATIONS
SEE PROFILE

Color Quantization

Luc Brun
Équipe Traitement Numérique des Images -Laboratoire LERI
Université de Reims Champagne Ardenne - France
E-Mail: luc.brun@univ-reims.fr

Alain Trémeau
Équipe Ingénierie de la Vision - Laboratoire LIGIV
Université Jean Monnet de Saint-Etienne - France
E-Mail: tremeau@vision.univ-st-etienne.fr

Contents

Chapter 1

Color Quantization

1.1 Introduction

Color image quantization is used to reduce the number of colors of a digital image with a minimal visual distortion. Color quantization can also be defined as a lossy image compression operation. Until lately, quantization was used to reproduce 24 bit images on graphics hardware with a limited number of simultaneous colors (e.g 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 band width requirements in multimedia applications.

Given a color image I, let us denote by $\mathcal{C}_I$ the set of its colors and by M the cardinality of $\mathcal{C}_I$. The quantization of I into K colors, with $K < M$ (and usually $K << M$) consists in selecting a set of K *representative colors* and replacing the color of each pixel of the original image by a suitable representative color.

Since first applications of quantization were used to display full color images on low-cost color output devices, quantization algorithms had from the beginning to face to two constraints. On one hand, the quantized image must be computed at the time the image is displayed. This makes computational efficiency of critical importance. On the other hand, the visual distortion between the original image and the reproduced one has to be as small as possible. The trade off between computational times and quantized image quality is application dependent and many quantization methods have been designed according to various constraints on this trade off.

One straightforward way to obtain quantized images with low computational times consists to use a preselected set of representative colors [29, 49]. Such methods, described in Section 1.2, are referenced as *image independent quantization* methods (see also arrow (1) in Figure 1.1). Quantized images with higher quality are generally obtained by building the set of representative colors according to the color distribution of the input image. Such methods are called *Adaptative quantization* methods or *Image dependent quantization* methods (arrows (2) in Figure 1.1).

Given an input image and a set of representative colors, the mapping of each color of the original image to one representative is performed by the *Inverse colormap operation* (Figure 1.1 and Section 1.7). Since each color of the original image is mapped onto a representative color, the definition of the set of representative colors induces a partition of the image color set $\mathcal{C}_I$. The notions of color set partition and representative colors are thus closely linked and many quantization methods define first a partition of $\mathcal{C}_I$ into a set of clusters and induce from it a set of representative colors. The brute force of enumerating all possible partitions of the set $\mathcal{C}_I$ with M colors into K subsets is out of the question here, since the number of

all possible partitions [5, 75]:

$$\frac{1}{K!}\sum_{k=0}^{K}(-1)^{K-k}\left(\begin{array}{c} K \\ k \end{array}\right)k^M$$

is astronomical for even very small M and K.

The definition of a partition of $\mathcal{C}_I$ is thus achieved by different heuristics described throughout this chapter (see also [58] for an overview on Digital Color Imaging). Each heuristic is designed according to some constraints on processing times and a particular definition of the visual distortion between the input and output images. Note that this notion of visual distortion may also be application dependent.

One other family of quantization methods define first an initial set of representatives and improves it iteratively by using the partition of $\mathcal{C}_I$ induced by the set of representatives. Such methods are referenced as *Post clustering methods* and are closely linked to the tracking of function minima by iterative methods.

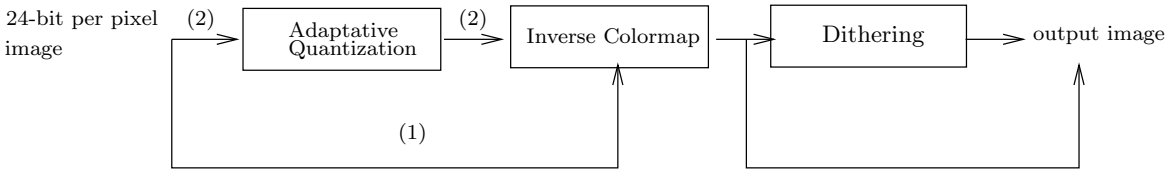


Figure 1.1: The sequence of algorithms applied to produce the output image

The optimal goal of adaptive quantization methods is thus to build a set of representative colors such that the perceived difference between the original image and the quantized one is as small as possible. The definition of relevant criteria to characterize the perceived image quality [62, 63, 65] is still an open problem. The difficulty of this problem is reinforced by our limited knowledge of the human visual system (HVS). There is thus no universal criterion available to characterize the perceived image similarities. One criteria commonly used by quantization algorithms is the minimization of the distance between each input color and its representative. Such criteria may be measured, for instance, using the total squared error (Section 1.4) which minimizes the distance within each cluster. A dual approach attempts to maximize the distance between clusters (Section 1.4.3). Note that the distance of each color to its representative is relative to the color space in which the total squared error is computed. The choice of one color space allows to take into account the characteristics of the human visual system encoded by this color space. The perception of the spatial arrangement of colors may also be encoded by weighting the distance of each color to its representative according to the local spatial arrangement of colors (Section 1.5). This idea has been recently developed by several quantization methods (Section 1.8.4) which optimize simultaneously the selection of the representative colors and their spatial arrangements.

The spatial arrangement of representative colors may also be optimized by a post-processing step named *dithering* (Section 1.8). This last step reduces visual artifacts such as false contours which lower the output image quality noticeably.

1.2 Image independent quantization methods

Digitized images are generally quantized with 8 bits of resolution for each color component R, G and B. Therefore, full color digital display systems use 24 bits to specify the color of each pixel on the screen which can thus display 2^{24} (16.8 million) colors.

The uniform quantization of a color space Ω divides it into K equal size sub-boxes and defines the set of representative colors as the sub-boxes centroids. Uniform quantization techniques differ according to the geometry of the color space in which the quantization is performed. Thus, meanwhile the RGB color space can be quantized naturally in cubical sub-boxes, the YIQ color space must be quantized in skewed rectangular sub-boxes otherwise many representative colors fall outside the YIQ color gamut[1], and hence cannot be reached by any colors in any original image (see Figure 1.2). Likewise, the $CIELab$ color space must be linearly quantized in sub-boxes with a specific shape even if the $CIELab$ color coordinates have been computed from a non-linear transformation from the RGB color space. The main problem of the uniform quantization of the $CIELab$ color space is that the exact shape of a given $CIELab$ color gamut is device dependent.

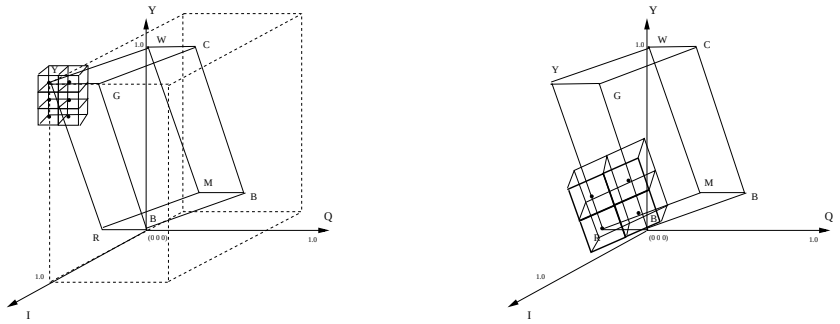


Figure 1.2: Uniform quantization of color space YIQ into rectangular sub-boxes(a) and skewed rectangular sub-boxes(b).

Image independent color quantization techniques [49] may be viewed as a generalization of uniform quantization where the chosen quantization regions are not of the same size but still independent of the image to be quantized. Such techniques are often used to divide a color space according to a criterion which varies throughout the color space. For example, Kurtz [42] divides the RGB color space in such a way that the distance between the representatives of any two adjacent regions is perceptually constant. The RGB color space being not perceptually uniform(`ref chapter??`), the size and the shape of the region associated to each representative is relative to its location in the RGB color space.

Image independent quantization algorithms place the K representative colors in the color gamut independently of their frequency of occurrence in a specific image, and of any other color characteristics computed from the studied image. These quantization methods are therefore computationally less expensive and simpler to implement than many image dependent color quantization algorithms. However, using such methods many representative colors may be assigned to locations in the color gamut where few colors of the original image reside. This is the reason why, image dependent color quantization algorithms that use representative colors based on the image to be quantized are typically preferred.

1.3 Preprocessing steps of image dependent quantization methods

Due to the complexity of color quantization, pre-processing steps are commonly employed for reducing the data processed by quantization algorithms. The major techniques used to

[1]Color gamut refers to the volume of realized colors associated with a particular device.

reduce the data are the pre-quantization which reduces the range of each coordinate in the color space and the histogram calculation which allows to manage more efficiently the image color set.

1.3.1 Pre-quantization

The "pre-quantization", in 5 bits for each color component, is commonly used by many algorithms as a basic step for the quantization process. The main purpose of this pre-quantization is to enables a trade off between quantizer complexity and image quality. Indeed, Fletcher [25] has shown that this initial step noticeably reduces the memory and computational time requirements of quantization algorithms. Other approaches have exploited this advantage further by using a "pre-quantization" in 4 bits of resolution for each color component [37], or even in 3 bits of resolution for each color component [32].

The main drawback of these approaches is that they do not take into account the non-uniformity of human visual system to perceived color differences. Indeed, while this pre-quantization may produce non-noticeable degradations in high frequency areas, it can at the same time produce visible degradations in low frequency areas or at the border of contours [61, 63]. Balasubramanian [8, 10] proposed to base the pre-quantization step on an activity measure defined from the spatial arrangement of colors (Section 1.5). However, in this case the computational cost of the pre-quantization step is no longer negligible and the additional computation time induced by this step has to be compared to the computation time of the quantization algorithm without pre-quantization.

A last approach proposed by Kurz [42] estimates the color distribution of an image and its relative importance by examining the pixels at a fixed number of random locations. Kurz proposes to use 1024 random locations for a 512×512 image. Randomizing the pixel locations avoids repeated selection of certain colors in the color set, particularly in images with periodic pattern(s).

1.3.2 Histogram calculation

The first pre-processing step of an image quantization algorithm generally consists of histogram computation of the image color set i.e., a function H is computed such that $H(R, G, B)$ is equal to the number of pixels of the image whose color is equal to (R, G, B). Using the RGB color space, one straightforward implementation of this function is to allocate a $256 \times 256 \times 256$ array. Given this array, the computation of the histogram is performed by incrementing the corresponding entry of each pixel's color. However, this method has several disadvantages: First, the size of the image being quantized is generally much smaller than the size of the full histogram indicated above and many entries of the histogram are set to 0 or very small values. This last property may induce some unwanted behavior for some algorithms. For example, the determination of the most occurring color, may "loose" important colors if they are encoded in the histogram by a set of adjacent entries with low frequencies. Secondly, this encoding requires the storage of 256^3 indexes. Encoding each entry of the array by 2 bytes, leads to a storage requirement of $32 Mbytes$. Finally, many quantization algorithms using other color spaces than RGB, the allocation of an array enclosing the color gamut of these color spaces reinforce the problems linked with the sparse property of the histogram and its storage requirements.

One partial solution to this problem proposed by Heckbert [35] consists to remove the 3 least significant bits of each component in order to store the histogram in an array of size $32 \times 32 \times 32$. This solution reduces both the memory requirements and the problems induced by the sparseness of the histogram. However, the removal of the 3 least significant bits is equivalent to a pre-quantization which may induce a visible loss of image quality when the

output image is quantized into large number of colors such as 256 (Section 1.3.1). Moreover, using a color space other than RGB, one has to define the box enclosing the reduced RGB color space. According to the color space this box may remain large beside the uniform quantization step.

One simple encoding of the histogram allocates an array whose size is equal to the number of different colors contained in the image. Each entry of this array is defined as a couple of fields encoding respectively a color contained in the image and the number of image's pixels of the corresponding color. This array may be initialized using a hash table and is widely used by tree-structured vector quantization algorithms (Section 1.4.2). Note that this last data structure is not designed to facilitate the determination of the number of image's pixels whose color is equal to a given value but rather to traverse the color space or a part of it in order to compute some of its statistical parameters such the mean or the variance.

Xiang [77] proposes an encoding of the RGB histogram based on a 2D array of lists indexed by the R and G color coordinates. During the traversal of the image each pixel with color (R, G, B) induces an update of the list of B values stored in the entry (R, G) of the array. If the component B is not present in the list, a new node is inserted with a color component storing the B value and a frequency field initialized to 1. If B is already present in the list the frequency of the corresponding node is incremented. Each list is doubly linked and sorted in ascending order according to the blue field (see Figure 1.3(a)). Therefore, if S denotes the mean size of the blue lists, the retrieval of the number of pixels whose color is equal to a given (R, G, B) triplet requires $\mathcal{O}(\frac{S}{2})$ comparisons. Balasubramanian [10] improves the search by storing in each entry of the (R, G) array a binary tree ordered according to the blue values(see Figure 1.3(b)). The computational time required to retrieve a given (R, G, B) is then reduced to $\mathcal{O}(\log(S))$.

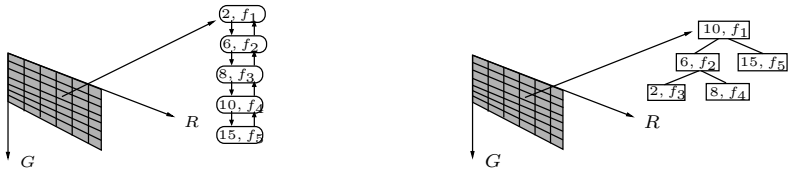


Figure 1.3: The histograms of Xiang [77] a) and Balasubramanian [10] b)

1.4 Clustering Methods

Clustering techniques perform a partition of the color space according to some criteria. Such criteria do not attempt to optimize the spatial arrangement of representatives which may be improved by a dithering algorithm applied as a correcting step (Section 1.8). The input data of such algorithms are thus the image color set $\mathcal{C}_I$ and the frequency of occurrence of each color $\mathbf{c}$ in the image: $f(\mathbf{c})$.

As anticipated in Section 1.1, quantization algorithms may be decomposed in two main families: post-clustering methods which define a set of K representatives improved by an iterative scheme and pre-clustering methods which define a partition of $\mathcal{C}_I$ into K clusters and associate one representative to each cluster. Clustering quantization methods belong to the latter family. Each cluster $(C_i)_{i\in\{1,\ldots,K\}}$ of the partition may be characterized by statistical properties such as its mean, its variance along one coordinate axis or its covariance

matrix. All these parameters may be deduced from the following quantities:

$$\begin{array}{lcl} card(C) & = & \sum_{\mathbf{c}\in C} f(\mathbf{c}) \\ M_1(C) & = & \sum_{\mathbf{c}\in C} f(\mathbf{c})\mathbf{c} \\ M_2(C) & = & \sum_{\mathbf{c}\in C} f(\mathbf{c})(\mathbf{c}_1^2, \mathbf{c}_2^2, \mathbf{c}_3^2) \\ R_2(C) & = & \sum_{\mathbf{c}\in C} f(\mathbf{c})\mathbf{c} \bullet \mathbf{c}^t \end{array} \tag{1.1}$$

where $(\mathbf{c}_i^2)_{i\in\{1,2,3\}}$ denotes the squared value of the i^{th} coordinate of the vector $\mathbf{c}$ and $f(\mathbf{c})$ denotes the number of pixels of the original image whose color is $\mathbf{c}$.

The quantities $card(C)$, $M_1(C)$ and $M_2(C)$ are respectively called the cardinal, the first and the second cumulative moments of the cluster. Note that, $card(C)$ is a real number, while $M_1(C)$ and $M_2(C)$ are $3D$ vectors. The quantity $R_2(C)$ is a 3×3 matrix whose diagonal is equal to $M_2(C)$.

The main advantage of the above quantities is that they can be efficiently updated during the merge or split operations performed by clustering quantization algorithms. For example if two clusters C_1 and C_2 must be merged, the cardinal of the merged cluster is equal to $card(C_1 \cup C_2) = card(C_1) + card(C_2)$. The same relation holds for M_1, M_2 and R_2. Conversely, if one cluster C is split into two sub clusters C_1 and C_2 and if both the statistics of C and C_1 are known, the statistics of C_2 may be deduced from the ones of C and C_1. For example, the cardinal of C_2 is defined as $card(C) - card(C_1)$. The mean, the variances and the covariance matrix of one cluster may be deduced from the cardinal, the moments and the matrix R_2 by the following formula:

$$\begin{array}{lcll} \mu & = & \frac{M_1(C)}{card(C)} & \\ var_i & = & \frac{M_2(C)_i}{card(C)} - \mu_i^2 & \forall i \in \{1,2,3\} \\ Cov & = & \frac{R_2(C)}{card(C)} - \mu.\mu^t & \end{array} \tag{1.2}$$

where var_i and μ_i denote respectively the variance of cluster C along the coordinate axis Ω_i and the i^{th} coordinate of the mean vector μ. The symbol Cov denotes the covariance matrix of the cluster.

Within the clustering quantization scheme, the covariance matrix of one cluster is used to determine the direction along which it spreads the widest. This direction named the *major axis* of the cluster is defined as the first eigenvector of the covariance matrix. Note that, the covariance matrix being real, symmetric and positive, it can be diagonalized on an orthogonal basis. Each eigenvalue of the covariance matrix is equal to the variance of the cluster along the associated eigenvector (Figure 1.4). Moreover, the information held by one eigenvector is measured by:

$$\frac{\lambda_i}{\sum_{i=1}^{3} \lambda_i} \tag{1.3}$$

where λ_i denotes the eigenvalue corresponding to eigenvector e_i.

Given a partition of $\mathcal{C}_I$ into K clusters $(C_i)_{i\in\{1,\ldots,K\}}$, clustering quantization algorithms associate one representative $\mathbf{c}_i$ with each cluster C_i. The sum of quantization errors committed when mapping pixels whose color fall in C_i to $\mathbf{c}_i$ is equal to the weighed sum of the squared distance between each color $\mathbf{c}$ in C_i and $\mathbf{c}_i$:

$$\sum_{\mathbf{c}\in C_i} f(\mathbf{c})\|\mathbf{c} - \mathbf{c}_i\|^2$$

A well known statistical result [30] states that this sum is minimal when $\mathbf{c}_i$ is equal to the mean μ_i of C_i. The resulting sum of distances is called the *squared error* of C_i and may be understood as the error committed when assimilating C_i to its mean:

$$\boldsymbol{SE}(C_i) = \sum_{\mathbf{c}\in C_i} f(\mathbf{c})\|\mathbf{c} - \mu_i\|^2 \tag{1.4}$$

a) b)

Figure 1.4: Lenna test image a) and its color set b) with the 3 eigenvectors (v_1, v_2, v_3) of its covariance matrix. The length of each vector is proportional to its eigenvalue

The squared error of one cluster C is related to the variances var_i computed along each coordinate axis $(\Omega_i)_{i\in\{1,2,3\}}$ by the following formula:

$$\boldsymbol{SE}(C) = card(C) \sum_{i=1}^{3} var_i \tag{1.5}$$

Each quantity $(card(C)var_i)_{i\in\{1,2,3\}}$ represents the contribution of the axis i to the squared error of C and is called its *marginal squared error* along axis i. Note that, if the color space is defined from the three eigenvectors of the covariance matrix, the variance along each axis is equal to the eigenvalue of the associated eigenvector. We thus obtain:

$$\boldsymbol{SE}(C) = card(C) \sum_{i=1}^{3} \lambda_i \tag{1.6}$$

where $(\lambda_i)_{i\in\{1,2,3\}}$ denotes the eigenvalues of the covariance matrix for the cluster.

The squared error of one cluster may be efficiently computed from its cardinal and the cumulative moments by the following formula(see equations 1.2 and 1.5):

$$\boldsymbol{SE}(C_i) = \sum_{j=1}^{3} M_2(C_i)_j - \frac{M_1(C_i)_j^2}{card(C_i)} \tag{1.7}$$

where $M_1(C)_j$ and $M_2(C)_j$ denotes respectively the j^{th} coordinate of vectors $M_1(C_i)$ and $M_2(C_i)$.

Using the squared error, the total quantization error induced by a partition $\mathcal{P} = \{C_1, \ldots, C_K\}$ of the image color set is measured by the *total squared error* (TSE) defined as the sum of the squared errors of the clusters defining the partition:

$$E(\mathcal{P}) = \sum_{i=1}^{K} \boldsymbol{SE}(C_i) \tag{1.8}$$

The TSE criterion is usually used in three different ways: Many papers on quantization use this criteria only *a posteriori* to demonstrate the efficiency of the proposed method. This criterion is also used by some methods to improve *a posteriori* the color palette initially designed. Finally, some authors base their quantization method on a minimization of the TSE. This latter category of methods generally provides quantized images with an higher visual

quality than the two previous ones. However, these algorithms are generally computationally intensive.

The different heuristics used to cluster colors may be decomposed into four main families: The $3 \times 1D$ splitting methods described in Section 1.4.1 use the optimal algorithms defined for one dimensional data to perform scalar quantization along each coordinate axis. The $3D$ splitting methods splits the $3D$ image color set into a set of K clusters (Section 1.4.2). A large majority of these methods split recursively the initial image color set by using a top-down scheme. Grouping methods, described in Section 1.4.3, use a bottom-up scheme by defining a set of empty clusters and aggregating each color of the image to one cluster. Finally, merge methods described in Section 1.4.4 use a mixed approach by first splitting the image color set into a set of clusters and then merging these clusters to obtain the K required clusters.

1.4.1 3x1D quantization methods

Using only one dimensional data, a partition minimizing the total squared error may be computed with a complexity $\mathcal{O}(KM)$ [72, 73] where M is the cardinality of the image color set. One first approach to take advantage of these optimal algorithms within the color quantization scheme, computes the three marginal histograms of the image color set along each of the coordinate axis. The value $h_j(r)$ of the j^{th} marginal histogram being defined as the number of pixels whose j^{th} color coordinate is equal to r. Using the three marginal histograms, a scalar quantization algorithm is applied independently on each axis of the color space. Such algorithms are referenced as *independent scalar quantization algorithms*(ISQ) [38]. However, because the ISQ uses only the marginal distribution of each scalar component, it can not take inter-data correlation into account. As a result, many representatives are wasted in regions where the input colors have zero probability of occurrence, as shown in Figure 1.5(a) on a two dimensional example.

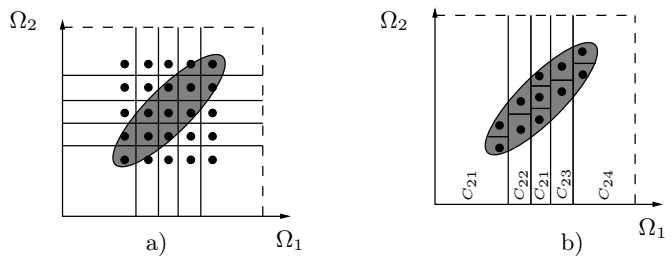


Figure 1.5: *Independent scalar quantization a) into $K = 25$ levels and Sequential scalar quantization into $K = 11$ levels.*

Another approach, proposed by Balasubramanian [9] and named *Sequential Scalar Quantization* (SSQ), consists to perform first a scalar quantization of the first coordinate axis Ω_1 into a predetermined number of levels K_1. This quantization of the first axis induces a partition $\mathcal{P}_1 = \{C_{21}, \ldots, C_{2K_1}\}$ of the image color set by planes orthogonal to Ω_1 as illustrated in Figure 1.5(b). Then the marginal histograms along Ω_2 are computed for each cluster $(C_{2i})_{i\in\{1,\ldots,K_1\}}$. A scalar quantization based on these marginal histograms is performed and splits each cluster C_{2i} into n_{2j} sub-clusters separated by planes orthogonal to Ω_2. This quantization step produces a total of K_2 clusters $\mathcal{P}_2 = \{C_{31}, \ldots, C_{3K_2}\}$. Finally, a last scalar quantization along Ω_3 is performed on the third marginal histogram of each cluster $(C_{3i})_{i\in\{1,\ldots,K_2\}}$ splitting it in n_{3j} sub-clusters. This last quantization step produces the required number K of clusters.

As claimed by the authors, this method as a lower computational cost than most of existing quantization methods. However, it raises some problems only partially solved by the authors:

first, the final total squared error is dependent of the order in which the scalar quantizations are performed (we have tacitly assumed that the Ω_j are quantized in the order Ω_1, Ω_2, Ω_3). The only means of finding the best order is to apply the quantization scheme on each of the $3! = 6$ possible orders. The authors address this problem by using the YC_rC_b color space with one luminance (Y) and two chrominance (C_r and C_b) axis. The authors perform first two scalar quantization on the chrominance plane (C_r, C_b) followed by one scalar quantization of the luminance axis Y. This strategy being based on a fixed order of the quantizations may lead to sub-optimal results. The second problem raised by this method is the determination of the number of quantization levels along each axis. In other words, what values of K_1, K_2, $(n_{2j})_{j\in\{1,\ldots,K_1\}}$ and $(n_{3j})_{j\in\{1,\ldots,K_2\}}$ should be picked to obtain the required number K of final clusters. The authors estimate these quantities by using results from the asymptotic quantization theory. This theory being valid only for very large values of K, the authors perform a preliminary quantization for some initial choice of K_1 and K_2 and correct these initial values as follows:

$$\left.\begin{array}{rcl} K_1 & = & K_1^0 \left(\frac{d_1^2}{d_2 d_3}\right)^{\frac{1}{6}} \\ K_2 & = & K_2^0 \left(\frac{d_1 d_2}{d_3}\right)^{\frac{1}{6}} \end{array}\right\} \text{ with } \forall j \in \{1,2,3\} \quad d_j = \sum_{i=1}^{K} \sum_{\mathbf{c}\in C_i} f(\mathbf{c})(\mathbf{c}^j - \mathbf{c}_i^j)^2 \tag{1.9}$$

where K_1^0 and K_0^2 are the initial choice for K_1 and K_2 and d_1, d_2 and d_3 denote the marginal total squared errors along each axis. Symbols $(\mathbf{c}^j)_{j\in\{1,2,3\}}$ and $(\mathbf{c}_i^j)_{j\in\{1,2,3\}}$ denote respectively the j^{th} coordinate of the color vector $\mathbf{c}$ and the i^{th} representative $\mathbf{c}_i$.

1.4.2 3D Splitting methods

The 3D splitting methods have been intensively explored [8, 9, 12–14, 70–72, 74, 75] since 1982 and the median cut method proposed by Heckbert [35]. These methods create a partition of the image color set into K clusters by a sequence of $K-1$ split operations.

The initial image color set is thus split by a set of planes named *cutting planes*, each plane being normal to one direction named the *cutting axis*. The location of the cutting plane along the cutting axis is named the *cutting position*. One justification for the use of planes within the 3D splitting scheme is provided by the inverse color map operation. Indeed, as mentioned in Section 1.4, given a set of representative colors $\{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$, an optimal mapping with respect to the set of representatives maps each initial color to its closest representative. This optimal mapping, induces a partition of the image color set by a 3D Voronoï diagram [60] defined by the representatives. Each cell of a 3D Voronoï diagram being delimited by a set of planes, the use of planes within the 3D splitting scheme does not induces a loss of generality. Note that, if each plane is assumed to be perpendicular to one of the coordinate axes each cluster is an hyperbox.

As mentioned in Section 1.1, the set of all possible partitions into K clusters of the initial set of M data points is too large for an exhaustive enumeration. The 3D splitting scheme must thus use a set of heuristics in order to restrict the set of possible partitions. The main heuristics used by splitting quantization algorithms may be decomposed into four steps common to all algorithms of this family:

1. selection of a splitting strategy,
2. selection of the next cluster to be split,
3. selection of the cutting axis
4. selection of the cutting position.

The remaining of this section describes the main heuristics used at each of the above steps. The different choices performed by the methods described in this section are summarized in Table 1.1.

Splitting strategy

A large majority of 3D splitting methods [8, 12, 13, 70–72, 75] split recursively the initial image color set into two sub clusters until the K final clusters are obtained. This bipartitioning strategy may be encoded by a complete binary tree and quantizers following this scheme are called *tree-structured vector quantizers* [74]. The internal nodes of this tree encode intermediate clusters and have exactly two siblings. The final clusters are encoded by the leaves of the binary tree (Figure 1.6).

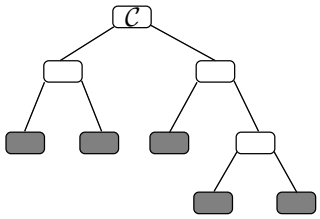


Figure 1.6: *A complete binary tree defining a partition into* 5 *clusters.*

The number of recursive splits which may be performed using the bipartitioning strategy is equal to the number of binary trees having exactly K leaves: $\frac{1}{K}\binom{2(K-1)}{K-1}$ [28]. This number is typically to large for an exhaustive enumeration, Chou et al. [21] and Lin et al. [44] create a binary tree of N leaves with $K < N << \frac{1}{K}\binom{2(K-1)}{K-1}$ and then prune the tree so as to select the K leave which induce the lowest partition error. However, according to Wu [74] this strategy induces a high overhead compared to a strategy generating only the K required clusters and does not induce a significant decrease of the total squared error.

The main drawback of the bipartitioning scheme is that each bipartitioning is performed regardless of its impact on further subdivisions performed deeper in the binary tree. This greedy local criterion may contradict the total squared error criterion which should be globally minimized. Wu [74] proposed to perform a first splitting step of the image color set by κ planes normal to the major axis of the color set. The relative position of these planes along the major axis are globally optimized using a dynamic programming scheme. The $\kappa + 1$ generated clusters are then recursively split into 2 sub-clusters using the recursive bipartitioning scheme described previously until the final number K of clusters is generated. The value of the critical parameter κ is estimated during the construction of the $\kappa + 1$ clusters. According to experiments performed by Wu, this value falls between four and eight according to the distribution of the image color set.

Cluster Selection

Using a bipartitioning scheme, each iteration of the quantization algorithm selects one leaf of the binary tree and splits it into two sub-clusters. The criterion used to select the cluster to be split varies according to the criteria minimized by the quantization algorithm.

The median cut algorithm proposed by Heckbert [35] divides the color space Ω into rectangular clusters C_k with a same cardinality. Therefore, in order to equalize the cardinality of clusters, Heckbert selects at each step the cluster with the greatest cardinality. Several splitting algorithms based on k-d trees [11] (Section 1.7.3) use an approach similar to the one

of Heckbert. These algorithms assign an approximately equal number of colors to all leaves of the tree. However, such a counterbalancing is not really adapted to image quantization. Indeed, there is no relevant justification to require that each cluster should contain a nearly equal number of colors while ignoring how these colors are distributed in the color space [70]. Using such method, a cluster with a large quantization error may not be split while a cluster containing only one color (with a high occurrence frequency) may be subdivided instead.

A large majority [12–14, 70–72, 74, 75] of methods attempt to minimize the total squared error. In order to obtain homogeneous clusters, Bouman [13] selects at each iteration the cluster whose data spread the widest along the splitting direction. Since Bouman splits clusters along their major axis it selects the cluster whose principal eigenvalue is maximal.

Wan et al. [70] proposed to split at each step the cluster with the greatest squared error. The basic idea of this strategy is to split the cluster whose contribution to the TSE is the largest. This strategy may be compared to the one of Bouman by writing the squared error into the base defined by the three eigenvectors of the covariance matrix:

$$\boldsymbol{SE}(C) = card(C) \sum_{i=1}^{3} \lambda_i$$

The heuristic of Bouman may thus be understood as an approximation of the one of Wan neglecting the possible decrease of the variance along the directions defined by the two remaining eigenvectors.

One slightly different strategy has been proposed by Wu [75]. It consists to select at each iteration the cluster whose bipartitioning yields the largest reduction of the total squared error. Given a partition of the image color set into k clusters $\mathcal{P}_k = \{C_1, \ldots, C_k\}$, the splitting of one cluster C_i into two sub clusters C_i^1, C_i^2 modifies the total squared error as follows:

$$E(\mathcal{P}_{k+1}) = E(\mathcal{P}_k) + \boldsymbol{SE}(C_i^1) + \boldsymbol{SE}(C_i^2) - \boldsymbol{SE}(C_i) \tag{1.10}$$

Note that, if both C_i^1 and C_i^2 are non-empty, the value $\boldsymbol{SE}(C_i^1) + \boldsymbol{SE}(C_i^2) - \boldsymbol{SE}(C_i)$ is strictly negative. Therefore, the total squared error is a strictly decreasing function of the number of split operations.

Wan's heuristic may thus be considered as an approximation of the heuristic of Wu neglecting the squared errors of the two generated clusters. However, using the heuristic of Wu, each cluster must be split to estimate the decrease of the total squared error. According to experiments performed by Wu [74], the difference between the two criteria in terms of final quantization errors is marginal while the increase in computational costs is two folds for the second criteria. The selection at each iteration of the cluster with the greatest squared error is thus a good compromised between the computational time and the final quantization error.

Cutting axis

Given the cluster C_i to be split one has to determine the normal and the location of the cutting plane. Since the decrease of the total squared error after the split of C_i into C_i^1 and C_i^2 is equal to $\boldsymbol{SE}(C_i^1) + \boldsymbol{SE}(C_i^2) - \boldsymbol{SE}(C_i)$, the optimal cutting plane is the one which minimizes $\boldsymbol{SE}(C_i^1) + \boldsymbol{SE}(C_i^2)$. However, the freedom to place such a plane is still too enormous and we need to fix the orientation of this plane to make the search feasible. Since the cutting of the cluster mainly decreases the variance along the cutting axis, one common heuristic consists to select the cutting axis among the directions with a high variance (equation 1.5).

Heckbert [35] approximates the variance by enclosing the cluster into a rectangular box. The two sides of this box normal to the coordinate axis Ω_i enclose the color vectors with minimum (resp. maximum) coordinates along Ω_i. The spread of the data along each coordinate axis is then estimated by the length of the box along this axis and the box is cut perpendiculary to its longest axis.

Braquelaire et al. [14] compute the variance along each axis and split each cluster along its coordinate axis with the greatest variance. This heuristic decreases the variance along the coordinate axis which brings the major contribution to the squared error. One computational advantage of this heuristic is that the data set being split along three fixed directions(the coordinate axis) the image color set may be pre-processed to optimize the determination of the cutting plane [14]. However, the coordinate axis with the greater variance is generally not the axis along which the data set spread the widest. This direction is provided by the major axis of the cluster and the relation between the squared error of the cluster and the three eigenvalues of the covariance matrix is provided by equation 1.6. Note that, if λ_1 denotes the principal eigenvalue of the major axis we have: $\forall i \in \{1,2,3\} \quad \lambda_1 \geq var_i$. Therefore, a greater decrease of the total squared error may be expected by cutting the cluster along its major axis. This last heuristic is used by Wan [70, 71], Wu [74, 75] and Bouman [12, 13].

Cutting position

Given a selected cluster C and one cutting axis defined by a unit vector A, let us denote by m and M the minimal and maximal projections of C on the cutting axis. Since the cutting plane is normal to A, its projection on the cutting axis is equal to one real number t. Any value of $t \in]m, M[$ induces a bipartition of C into two sub clusters C_t^1 and C_t^2 as illustrated in Figure 1.7. Note that, if we assume that the cutting plane belongs to C_t^2, we have: $card(C_m^1) = 0$ and $card(C_m^2) = card(C)$ while $card(C_{M+\epsilon}^1) = card(C)$ and $card(C_{M+\epsilon}^2) = 0$ where ϵ is any positive real number. More precisely, the function δ defined by:

$$\delta(t) = \frac{card(C_t^1)}{card(C)}$$

is an increasing function of t from $[m, M]$ to $[0, 1[$.

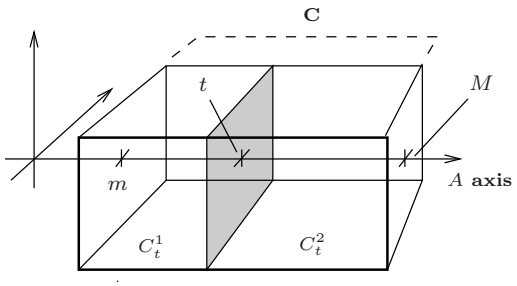


Figure 1.7: *Splitting of one cluster along axis A*

The value of t such that $\delta(t) = \frac{1}{2}$ corresponds to a split of the initial cluster into two sub clusters with a same cardinality. The median cut algorithm of Heckbert [35] defines a partition into K clusters with an equal cardinality. Therefore, given the selected cluster and the cutting axis, Heckbert sweep the cutting plane along the cutting axis from $t = m$ until $\delta(t) = \frac{1}{2}$.

Bouman [12, 13] selects as value of t the projection of the mean on the cutting axis: $\mu.A$, where μ is the mean of C. This strategy avoids the sweeping of the cutting plane along the cutting axis. However, the validity of this choice is demonstrated only for large clusters with Gaussian distribution.

Wan et al. [70] maximize the decrease of the marginal squared error along the major axis (Section 1.4 and equation 1.5). The optimal value t_{opt} maximizing the decrease of the

marginal squared error along the major axis is defined by [70]:

$$t_{opt} = \arg \max_{t \in [m,M]} \left[var - \left(\delta(t) var_t^1 + (1 - \delta(t)) var_t^2 \right) \right]$$

where var_t^1 and var_t^2 denote respectively the variance of C_t^1 and C_t^2 along the major axis of C. The above formula uses both parameters from clusters C_t^1 and C_t^2. This last property induces both an update of the parameters of C_t^1 and C_t^2 when sweeping the cutting plane from $t = m$ to $t = M$. Wan et al [70, 72] proved that the cutting position is also provided by:

$$t_{opt} = \arg \max_{t \in [m,M]} \left[\frac{\delta(t)}{1 - \delta(t)} \left((\mu - \mu_t^1) \bullet A \right)^2 \right] \tag{1.11}$$

where μ and μ_t^1 denote respectively the mean of C and C_t^1.

The main advantage of this last formula is that t_{opt} is now only a function of the parameters of C_t^1 which may be incrementally updated when sweeping the plane along the major axis of C. Moreover, using theoretical results from scalar quantization [72] Wan et al. show that the interval $[m, M]$ may be restricted to $[\frac{m+\mu \bullet A}{2}, \frac{M+\mu \bullet A}{2}]$. Note that this last interval encloses the cutting position $\mu \bullet A$ selected by Bouman. Nevertheless, Wan et al. maximize only the decrease of the marginal squared error along the major axis of the cluster while according to equation 1.6 the squared error is equal to the sum of the marginal squared errors along the three eigenvectors.

Wu [74, 75] maximizes the decrease of the total squared error (equation 1.10) defined by: $\boldsymbol{SE}(C_t^1) + \boldsymbol{SE}(C_t^2) - \boldsymbol{SE}(C)$. Wu proved that this last formula is minimal for a value of t_{opt} defined by:

$$t_{opt} = \arg \max_{t \in [m,M]} \frac{\|M_1(C_t^1)\|^2}{card(C_t^1)} + \frac{\|M_1(C) - M_1(C_t^1)\|^2}{card(C) - card(C_t^1)} \tag{1.12}$$

This last expression uses only the cardinality and the first moment of cluster C_t^1 which can be efficiently updated when sweeping the cutting plane. However, the uses of the squared value of the first moment may lead to rounding errors. Brun [14, 16] has shown that the value t_{opt} maximizing the decrease of the partition error may also be defined as:

$$t_{opt} = \arg \max_{t \in [m,M]} g(t) \text{ with } g(t) = \frac{\delta(t)}{1 - \delta(t)} \left(\mu - \mu_t^1 \right)^2 \tag{1.13}$$

This new formulation may be considered as an extension of formula 1.11 to the 3D case. It allows to manipulate smaller quantities than formula (1.12) and thus avoids rounding errors induced by this last formula.

Brun has also shown that the function $g(t)$ is bounded above by one parabola U and below by an hyperbola L (see Figure 1.8). The parabola U allows one to stop the search of the optimal value t_{opt} as soon as $\delta(t) \geq \delta_{max}$ as indicated in Figure 1.8. According to experiments performed by Brun the mean reduction of the interval $[m, M]$ with a quantization into 16 colors is about 10%. Note that, since the functions U and L reach their maximum when δ is equal to $\frac{1}{2}$, the maximum of $g(t)$ is reached for a value of t such that $\delta(t)$ is about $\frac{1}{2}$. This value $\delta = \frac{1}{2}$ corresponds to the median position of the splitting plane selected by the Heckbert's median cut algorithm.

1.4.3 Grouping methods

Grouping techniques generate the K representative colors during one traversal of the image. Using such a scheme, each representative is defined according to the representatives already selected and the current color set. Indeed, the color set used at each iteration of the algorithm may be restricted to the already scanned colors. Alternative techniques [31] define K representatives by scanning the K first colors of the image and update this set during the traversal of the image.

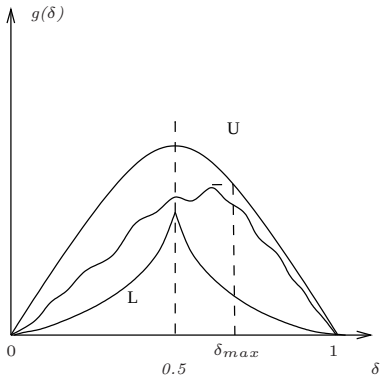


Figure 1.8: Evolution of the total squared error induced by the split of one cluster by a plane as a function of the cardinality of one of the two clusters.

Table 1.1: Different strategies used by 5 tree-structured vector quantizers.

		1	2	3	4	5
Cluster selection	Greatest squared error		⋆		⋆	⋆
	Greatest eigenvalue			⋆		
	Greatest cardinality	⋆				
Cutting axis	Longest Coordinate axis	⋆				
	Coordinate axis with greatest variance					⋆
	Major axis		⋆	⋆	⋆	
Cutting position	Median cut	⋆				
	Marginal squared error minimization		⋆			
	TSE minimization				⋆	⋆
	Pass through the mean			⋆		

(1)	:	Heckbert [35]
(2)	:	Wan and Wong[71]
(3)	:	Bouman and Orchard[12, 13]
(4)	:	Wu [75]
(5)	:	Braquelaire and Brun [14]

The merge and box algorithm.

This algorithm proposed by Fletcher [25] is based on an iterative process which is repeated until the boxes with the K representative colors are formed. This algorithm iterates the following steps: each distinct color of the color space is placed into a sub-box of length side equal to one. If the input color can be approximated, with an error no greater than half the length of box's diagonal, by one of the sub-boxes already generated this color is included in the corresponding sub-box. Otherwise, a new sub-box of side one is added to the list. To obtain a list which do not exceed K sub-boxes, a merging process is used when the number of sub-boxes is equal to $K+1$ to gather the pair of sub-boxes which must to be merged into a single sub-box. The merging pair is the one which minimize the largest side of the merged sub-box.

According to Fletcher [25], the heuristic used by the merger generates roughly cubical sub-boxes with a similar size. Using a Manhattan distance between corners would result in elongated sub-boxes, causing larger quantization errors. This algorithm may be parallelized on a SIMD parallel processor. Moreover, its calculating time may be further reduced by using a spatial sampling to approximate accurately the image color set.

Gervautz [31] proposed a similar approach based on a hierarchical decomposition of the RGB cube by an octree. Using this decomposition each cube of side 1 used by Fetcher is encoded as a leaf of the octree. Gervautz builds the first K leaves of the octree by scanning the first K colors of the image. Each leaf of the tree represents thus one cluster of the RGB color space. If an additional color belongs to one of these clusters, the mean color of the associated leaf is updated and the structure of the octree remains unchanged. Otherwise, a new leaf is created from the merger of its children in order to keep only K leave. The set of K representative colors is then defined from the centroid of the clusters associated to each leaf of the octree.

The max-min algorithm.

The max-min algorithm first proposed by Houle and Dubois[37] iterates the following steps(Figure 1.9):

1. a single color c_1 is first selected from the original image. It could be the most frequently occurring one.

2. A new representative color c_k is then computed, c_k is the un-selected color whose minimum distance to any of the representative colors thus far selected is maximum, i.e., c_k must verify :

$$\min_{k'=1,\dots k-1} \|c_k - c_{k'}\|^2 \;\geq\; \min_{k'=1,\dots k-1} \|c - c_{k'}\|^2 \quad \forall \mathbf{c} \in \{\Omega^I - \Omega^Q\}$$

   where symbols Ω^I and Ω^Q denote respectively the image color set and the set of representatives already selected.

   Step 2 of the max-min algorithm is iterated until K representative colors are selected.

A similar approach has been proposed by Xiang [76] who initializes the first color c_1 arbitrary. Note that, an arbitrary selection of the first color avoids the computation of the image's histogram. This quantization scheme has also been investigated by Trémeau et al.[64] to extend it to a vector quantization process based on the nearest color query principle.

According to Houle and Dubois the set of representatives generated by this algorithm is uniformly distributed and extends right to the boundary of the image color set. This distribution is well suited when a pseudo-random noise vector is added to the color value of the pixel to be quantized, i.e., when we try to minimize errors of quantization by dithering

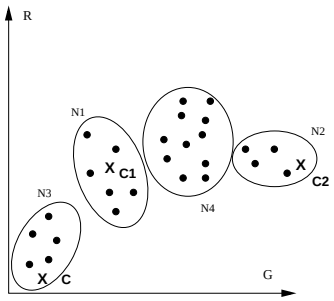


Figure 1.9: The max-min algorithm. On this example, the algorithm first select the most occurring color c_1 and then c_2 and c according to the distance to the previously selected colors.

the quantized image (Section 1.8). In this last case, the max-min algorithm give better depiction of low contrasts details but at the expense of granularity due to the pseudo-random quantization.

1.4.4 Merge methods

Methods described in Section 1.4.2 split recursively the image color set $\mathcal{C}_I$ until the required number K of clusters is obtained. Such methods obtain generally quantized images with an high visual quality but has to traverse each color $\log_2(K)$ times. On the other hand the grouping methods described in section 1.4.3 traverse each pixel of the original image to map its color to one cluster. Therefore, such methods traverse the original image only once. However, the function used to map each color to one cluster must have a very low computational cost in order to achieve overall low processing times. This constraint forbids the use of complex heuristics in the mapping function.

Merge methods may be understood as intermediate methods between pure split or pure grouping methods. Such methods perform a first splitting step in order to define a set of $N > K$ clusters and then perform a sequence of merges to obtain the K final clusters. The first splitting step has to reduce the number of data points while keeping the main properties of the image color set. Based on this reduced set of data, the merge algorithm may apply heuristics with an higher computational complexity than the ones designed within the grouping quantization framework. The set of clusters together with their adjacency relationships may be thought as a graph where each vertex encodes one cluster and each edge an adjacency relation between two clusters. Then the merge of two clusters is equivalent to the contraction of the edge encoding their adjacency and the removal of any multiple edges between adjacent clusters. The value of each edge encodes the distance between two clusters. After the merge of two clusters, the value of the edges incident to the newly created cluster has to be updated. Using the TSE (equation 1.8) the value of each edge is defined as the increases of the total squared error induced by the merge of the two adjacent clusters.

If $\mathcal{P}$ and $\mathcal{P}'$ denote respectively the partition of the image color set before and after the merge of two clusters C_k and $C_{k'}$ we have: $\mathcal{P}' = (\mathcal{P} - \{C_k, C_{k'}\}) \cup \{C_k \cup C_{k'}\}$ and the total squared error of $\mathcal{P}'$, $E(\mathcal{P}')$ may be deduced from $E(\mathcal{P})$ by:

$$E\left(\mathcal{P}'\right) = E(\mathcal{P}) + \frac{card(C_k)\ .\ card(C_{k'})}{card(C_k) + card(C_{k'})} \parallel \mu_k - \mu_{k'} \parallel^2 \tag{1.14}$$

The value of an edge encoding an adjacency relationship between clusters C_k and $C_{k'}$ is

thus equal to:

$$\Delta_{k,k'} = \frac{card(C_k)\ .\ card(C_{k'})}{card(C_k) + card(C_{k'})} \parallel \mu_k - \mu_{k'} \parallel^2 \tag{1.15}$$

Note that one is effectively minimizing a weighted Euclidean distance between the two centroids μ_k and $\mu_{k'}$ of the two clusters C_k and $C_{k'}$. This is the reason why such methods are often referenced as pairwise nearest neighbor clustering [10, 23].

The main computational cost of merge methods comes from the merger which has to traverse all pairs of adjacent clusters at each step. Two approaches have been proposed and sometimes combined [10] to reduce this computational cost: The reduction of the number of initial clusters and the use of heuristics in the merge strategy.

Brun and Mokhtari [17] first perform a uniform quantization of the RGB cube into N clusters and then create a complete graph from the set of clusters. This graph is reduced by merging at each step the two closest clusters according to equation 1.14. Experiments performed by Brun et al. show that if K is lower than 16, very low values of N such as 200 or 300 are sufficient to obtain high quality quantized images.

Dixit [22] reduces the initial number of clusters by a sub sampling of the image at random locations. According to Dixit, as few as 1024 random locations on a 512×512 image are sufficient to obtain a good estimate of the distribution of the image color set. The set of clusters is then sorted in ascending order according to the cardinality of each cluster. The sorted table of clusters is then traversed from the clusters with lowest cardinality and each current cluster is paired with the closest remaining one according to equation 1.14. These two clusters are then excluded from the merge. Therefore, no cluster is allowed to be paired with more than one cluster and the set of clusters is reduced by a factor two at each iteration. Note that this fixed decimation ratio between two successive iterations may lead to the merger of clusters which are far apart.

Xiang and Joy [77] create the initial set of clusters by using a uniform quantization of the RGB color space into N clusters. Xiang reduces the N initial clusters to K final clusters by a sequence of $N - K$ merges. At each step, the merge criterion selects two clusters such that the newly created cluster is enclosed into a minimal bounding box. Xiang thus estimates the homogeneity of one cluster by the size of its bounding box. Using an initial uniform quantization into $2 \times 1 \times 4$ boxes, each cluster has approximately the same luminance in the YIQ-NTSC system. Therefore, the size of a bounding box may be understood as the shift in luminance within the associated cluster and Xiang algorithm mainly attempts to minimize the shift in luminance within the quantized image. Such shifts may produce significant visual artifacts in the ray-traced images used by Xiang.

Equitz [23] reduces the computational cost of the merger by using a $k - d$ tree (see Section 1.7.3) which decomposes the set of initial clusters into a set of regions each composed of 8 clusters. The complete graph defined by Brun [17] is thus decomposed into a set of non-connected complete sub-graphs, each composed of 8 vertices. Then, Equitz finds the two closest clusters in each sub-graph according to equation 1.14 and merges a fixed fraction of them (such as 50%). After each iteration, the decomposition of the initial graph into regions is also updated in order to obtain roughly equal size sub-graphs. Note that this merge strategy neglects adjacencies between vertices belonging to different sub-graphs. Equitz's heuristic has been improved by Balasubramanian and Allebach [10] who first perform a prequantization step described in Section 1.5. Balasubramanian also weights the distance between two clusters by an activity measure defined from the initial image (Section 1.5).

1.4.5 Popularity methods

This family of methods, introduced by Heckbert [35], uses the histogram of the image color set to define the set of representatives as the K most occurring colors in the image. This simple

and efficient algorithm may however perform poorly on images with a wide range of colors. Moreover it often neglects color in sparse regions of the color set. Likewise, this algorithm performs poorly when asked to quantize to a small number of colors (say < 50) [35].

This algorithm has been modified and optimized by Braudaway [15] in 1986. Braudaway encodes the image histogram by a uniform partition of the color space Ω into $L \times L \times L$ sub-boxes Ω_k of equal size N/L (see Section 1.3.2 and Figure 1.10). In order to prevent the next representative color from being chosen too close from the previous one, a reduction function is applied to the histogram after the selection of each representative. The cardinal $card(\Omega_k)$(equation 1.1) of each sub-box Ω_k is then reduced by a factor $(1 - e^{-\alpha\ r^2})$ where r is the distance between Ω_k and the previously selected color.

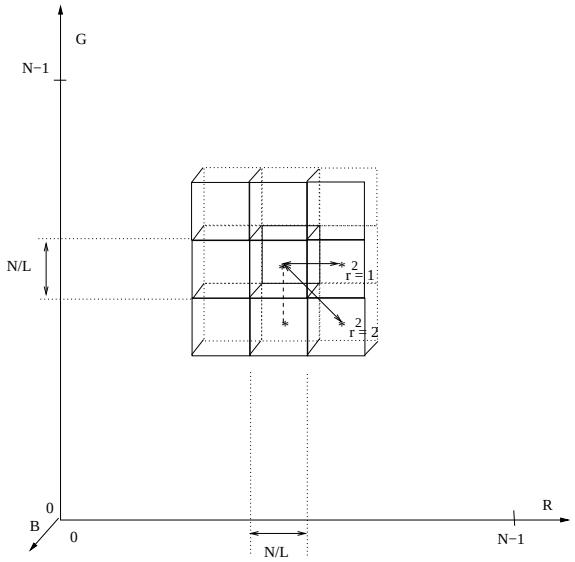


Figure 1.10: The reduction factor applied by Braudaway's algorithm [15]. The color space is enclosed in a $N \times N \times N$ box and subdivided into L^3 sub-boxes. The symbol r denotes the distance of each sub-box to the center one.

The degree of histogram reduction for a fixed value of r is controlled by α. In the investigation done by Braudaway, α was chosen so that the histogram was reduced by a factor of $\frac{1}{4}$ at a distance $r = \frac{N}{4}$ between the studied sub-box Ω_k and the previous selected color.

1.5 Quantization algorithms based on weighted errors

We saw in Section 1.4.2 several methods based on a minimization of the total squared error. The use of the total squared error within the quantization framework relies on the assumption that a partition of the image color space into homogeneous clusters produces an output image visually closed to the original. This assumption is partially justified by the following equation:

$$E(\mathcal{C}) = \sum_{i=1}^{K} \boldsymbol{SE}(C_i) = \sum_{(i,j)\in\{1,\ldots,m\}\times\{1,\ldots,n\}} \|I(i,j) - I'(i,j)\|^2 \tag{1.16}$$

where $m \times n$ denotes the size of the images and $I(i,j)$, $I'(i,j)$ denote respectively the color of the pixel (i,j) in the original and quantized images.

The total squared error may thus be understood as the sum of the squared distances between the pixels of the original image and the quantized one. Despite this interesting

property the total squared error should not be considered as a visual distance between the original and quantized images. This is confirmed by the experiments displayed in Figure 1.11. The $55,510$ different colors of image 1.11(a) are reduced to 8 colors by a quantization algorithm [17]. The quantized image is displayed in image 1.11(b). Image 1.11(c) is obtained from image 1.11(b) with an additional dithering step performed by the Flyod-Steinberg [27] algorithm (Section 1.8). Although image 1.11(c) seems visually closer from image 1.11(a) than image 1.11(b), the total squared error of image 1.11(c) computed thanks to equation 1.16 is nearly twice bigger than the one of image 1.11(b). This surprising result is due to the different criteria used by the dithering and quantization algorithms. The dithering algorithm attempts to optimize the local arrangement of representatives while the quantization algorithm minimizes the global homogeneity of clusters.

(a) (b) (c)

Figure 1.11: The original image (a), the quantized one with 8 colors (b) and (b) improved with a dithering algorithm (c)

One method to take into account the local properties of the image during the quantization step is to replace the frequency function f in the squared error's equation(equation 1.4) by a weighted frequency function W, where $W(c)$ is defined as a sum of local attributes computed on each pixel with color c. Note that, from this point of view, the usual squared error may be understood as a special case of weighted squared error where the weight of each color is equal to the number of pixels mapped to it. Given an initial color space $\mathcal{C}$ partitioned into K clusters $\{C_1, \ldots, C_K\}$, the weighted TSE is thus defined by:

$$E(\mathcal{C}) = \sum_{i=1}^{K} \sum_{c \in C_i} W(c) \|c - \mu_i\|^2 \tag{1.17}$$

One prominent artifact of a limited palette size is false contouring. False contours occur when a smoothly varying region is mapped into a small number of colors from the color map. Instead of displaying slow variations of colors across the region, smaller regions with a constant color are displayed with abrupt changes across region boundaries. These abrupt color changes are perceived as contour in the quantized image. Bouman [12, 13] estimates the size of uniformly quantized regions by using the variations of the luminance defined in the NTSC system:

$$y(p) = [0.300, 0.586, 0.115] I(p)^t \tag{1.18}$$

where the color of pixel p, $I(p)$, is defined in the RGB color space.

Bouman shown that within a region with a linear variation of the luminance with slope $\bigtriangledown y$, the ratio between the distance of each pixel's color to its representative and the local gradient of the luminance is proportional to the width of the uniformly mapped regions. More precisely, we have:

$$\sum_{p \in R_n} \frac{\|I(p) - c_n\|^2}{\|\bigtriangledown y(p)\|^2} \approx |R_n| \frac{\Delta^2}{12}$$

where R_n denotes the set of pixels p whose color $I(p)$ is mapped to c_n. The cardinality of this set is denoted by $|R_n|$. The symbol Δ denotes the width of an uniformly quantized regions in the direction of the luminance gradient $\bigtriangledown y$.

Therefore, in order to minimize the size of the uniformly mapped regions (estimated by Δ), Bouman uses a weighting inversely proportional to the luminance gradient of the image. For each color c of the original image, the weighting $W(c)$ is defined by:

$$W(c) = \sum_{p \in I | I(p)=c} \left(\frac{1}{h \star \min\{\|\bigtriangledown y(p)\|, 16\} + 2} \right)^2 \tag{1.19}$$

where the constants 16 and 2 bound the dynamic range of the gradient, while the function h is a smoothing kernel the convolution of which allows to estimate the mean gradient over the regions.

Note that any scale of the luminance component in the original image by a factor s scales the weighting factor $W(sc)$ by $\frac{1}{s^2}$ while the squared distance between c and its representative is scaled by a factor s^2. Therefore, the factors $W(c)\|c - c_i\|^2$ and the weighted TSE (see equation 1.17) are insensitive to any global scaling of the luminance. This obeys Weber's law [55], which suggests that any measure of subjective image quality should be invariant to absolute intensity scaling.

Bouman integrates the weighting defined by equation 1.19 into a tree structured vector quantizer (Section 1.4.2 and Table 1.1). Using the weighted TSE (equation 1.17), the weighted moments of a cluster C are simply defined by substituting $f(c)$ by $W(c)$ in equations 1.1. The weighted mean and covariance of the cluster are then defined from the weighted moments using equation 1.2. Bouman uses the weighted mean μ_i^w and covariance matrix Cov_i^w of each cluster C_i to adapt his tree-structured vector quantizer as follows:

1. Select the cluster C_m whose weighted covariance matrix has the greatest eigenvalue. Let us denote by e_m the eigenvector associated with this eigenvalue.

2. Split the selected cluster by a plane orthogonal to e_m and passing through μ_m^w.

Note that since the introduction of weights modifies the distribution of the initial image color set, it modifies all key steps of Bouman's recursive split algorithm. Moreover, the weights introduced by Bouman may be easily adapted to other recursive splitting schemes. For example, a weighted minimization of the TSE may be achieved by computing the weighted squared error using the weighted moments (see equations 1.19 and 1.7) and selecting at each step the cluster whose weighted squared error is maximal.

One other characteristic of the Human Visual System is its greater sensitivity to quantization errors in smooth areas than in busy regions of the image. Balasubramanian and Allebach [8, 10] define a color activity measure on each pixel by:

$$\alpha_k = \frac{1}{64} \sum_{p \in P_k} \|c_p - \overline{c_k}\|$$

where P_k is a 8×8 block centered around pixel k. The symbols c_p and $\overline{c_k}$ denote respectively the color of pixel p and the mean color of P_k.

These values are then accumulated on colors by associating to each color the minimal color activity measure of the color over all spatial blocks:

$$\tilde{\alpha_c} = \min_{k \in K_c} \alpha_k \tag{1.20}$$

where K_c denotes the set of blocks in which color c occurs.

Since luminance component has the greatest variations, the gradient of the luminance is also used as an alternative [9] or to complement [10] the measure of the spatial masking. Experiments show that the masking of quantization noise by image activity at a pixel depends not only on the luminance gradient at the pixel but also on gradients at the neighboring pixels. Based on these experiments, Ballasubramanian [9, 10] computes the gradient $\bigtriangledown y$ of the luminance defined in the NTSC system (equation 1.18) on each pixel. Then, the luminance activity α_k^l of each 8×8 block k is defined as an average value of its gradient. The luminance activity of each color is then defined as the average value of the block's gradient in which this color occurs:

$$\tilde{\alpha_c^l} = \frac{1}{|K_c|} \sum_{k \in K_c} \alpha_k^l \tag{1.21}$$

where K_c denotes the set of blocks in which color c occurs.

The color or luminance activity may then be used to weight the colors during the quantization step. Balasubramanian chose $W(c) = \frac{1}{\tilde{\alpha_c}}$ and $W(c) = \frac{1}{\tilde{\alpha_c}^2}$ respectively in [9] and [10]. In both cases, the importance of colors with low activity measure is reinforced by the weighting to the detriment of colors with high activity measure whose quantization errors are less readily perceived by the human visual system.

Balasubramanian uses the color and luminance activity measures within his quantization algorithm based on a merge scheme [10](see Section 1.4.4). The prequantization step performed by his method is based on the activity measure defined by equation 1.20. Colors are then categorized as belonging to low, medium or high activity classes according to two thresholds t_1 and t_2 determined experimentally. The luminance activity measure (equation 1.21) is used in complement to color activity to decompose the set of colors with a low color activity measure in two subsets corresponding to low and high luminance activity. This decomposition is based on an additional threshold g. Based on this color classification, the image color set is decomposed into a set of cubes with length $l = 2$, 4 or 8, all colors of one cube belonging to a same class (see Figure 1.12). Therefore, the merge process tends to avoid the merger of clusters associated with low activity measures, while colors associated to high activities are already grouped into larger clusters.

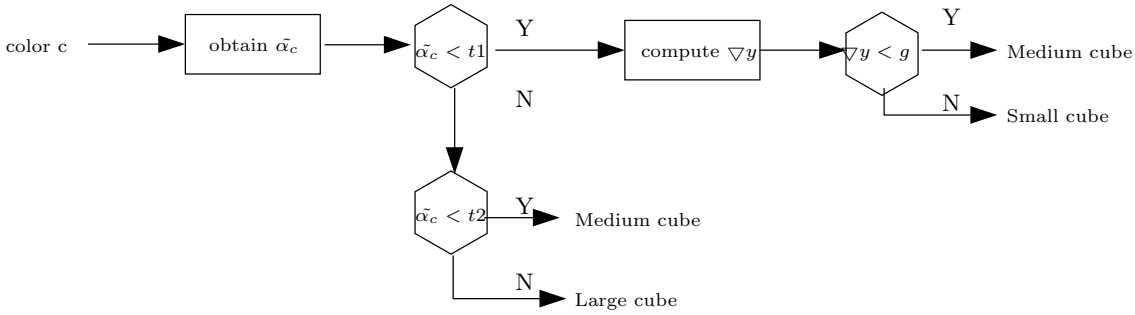


Figure 1.12: *Block diagram of prequantization scheme.*

Balasubramanian uses also color activity measures to weight the distance between clusters. Using the same scheme than the aggregation of the activity measure on colors (see equation 1.20) Balasubramanian defines the color activity of one cluster as the minimal activity of its colors. Therefore, given two clusters C_i and C_j with color activities $\hat{\alpha_{C_i}}$ and $\hat{\alpha_{C_j}}$, the color activity of the merged cluster $\hat{\alpha}_{C_i \cup C_j}$ is equal to $\min(\hat{\alpha}_{C_i}, \hat{\alpha}_{C_j})$. The weight $\omega_{i,j}$ of two clusters is then defined as:

$$\omega_{i,j} = \frac{1}{\hat{\alpha}^2_{C_i \cup C_j}}$$

In order to attach large weights to distance between clusters with small activities and vice versa Balasubramanian defines the weighted distance between clusters C_i and C_j as $\omega_{i,j}\Delta_{i,j}$ where $\Delta_{i,j}$ is the increase of the total squared error induced by the merge of clusters C_i and C_j (equation 1.14). Using such weights, clusters with high activity measure are more likely to be merged before low activity clusters.

Balasubramanian and Bouman combined their methods [8] in order to integrate the color weights defined by Balasubramanian within the tree-structured vector quantizer defined by Bouman. This method defines the aggregate weight of each cluster as the average weight of its colors. Since the cluster's weights are more meaningful for small and close clusters, Balasubramanian et al. perform $\frac{2}{3}K$ un-weighted binary split (Section 1.4.2 and Table 1.1). Then the $\frac{1}{3}K$ remaining splits are performed by splitting at each step the cluster whose weighted eigenvalue $\omega_k\lambda_k$ is maximal. The symbols ω_k and λ_k denote respectively the weight and the maximal eigenvalue of cluster C_k.

1.6 Post clustering methods

Quantization strategies presented so far use a *pre-clustering* scheme which computes the set of representatives only once. Another strategy consists to define an initial set of representatives and to improve it iteratively. Such a quantization scheme is called a *post-clustering* strategy. While the pre-clustering strategy is commonly used, the post-clustering one is less popular. Indeed, the use of a post-clustering strategy induces generally three main drawbacks: First, the number of iterations being not bounded, quantization algorithms using a post-clustering scheme may be computationally intensive. A much bigger disadvantage generally is the loss of the structure induced by the original strategy, which often makes the mapping step very computation intensive. Lastly, the iterative improvement of the set of representatives generally converges to the local minimum the closest from the initial solution. Therefore, such algorithms are often trapped into local minima of the optimized criterion. However, several heuristics, presented in this section have been proposed to improve the computational efficiency of post-clustering algorithms. Moreover, using a post-clustering scheme, informations such as the interrelationships between neighboring clusters or the spatial distribution of colors may be naturally integrated into quantization algorithms. Finally, one common heuristic consists to combine the pre and post-quantization schemes as follows: First, an initial set of representatives closed from the optimal solution is determined by a pre-quantization algorithm. This initial set of representative is then moved to the (hopefully) global optimum by a quantization algorithm based on a post-clustering scheme.

1.6.1 The LBG and k-means algorithms

Let us first introduce the LBG algorithm. The design of a locally optimal vector quantizers was both investigated by Llyod in 1982 [47], and by Linde et al. in 1980 [46]. The initialization step defines one set of representatives. This could be the K most occurring colors or any set of colors chosen arbitrary. The iteration begins by assigning each input color to one representative according to some distortion measure. These assignments define a partition of the image color set into clusters, each cluster being associated to one representative. The set of representatives is then modified to minimize the errors relative to input colors. This two step process is iterated until no significant change occurs within the set of representatives between two successives iterations.

Several quantization processes, such as those proposed by Heckbert [35] or Braudaway [15], use the LBG algorithm to improve the set of representatives obtained from their quantizers. However, since the LBG algorithm converges only to a local minimum, this step often yields

only slight improvements [58].

Several investigations have been performed to improve the performance of this algorithm, such as the one by Goldberg [32], who extends this algorithm to a high resolution solution which selects a set of representative vectors from input vectors. Likewise, Feng [24] has extended this algorithm to a vector quantizer that exploits the statistical redundancy between the neighboring blocks to reduce the bit rate required by the algorithm.

More significant improvements can be obtained thanks to other heuristics that attempt to select the palette through a sequential splitting process while reducing the TSE at each step. Several investigations such that those of Orchard and Bouman [13] or the one of Balasubramanian [8] propose various splitting procedures and different selection criteria to reduce the TSE.

The k-means algorithms belong to the same category of post-clustering techniques. These algorithms are based on an iterative process which is repeated until it converges to a local minimum solution, e.g. until the cluster centers of the K generated clusters do not change from one iteration to the other. The number of iterations required by the algorithm depends on the distribution of the color points, the number of requested clusters, the size of the color space and the choice of initial cluster centers [70]. Consequently, for a large clustering problem, the computation can be very time consuming. Recently, faster clustering approaches have been proposed. Among them, some are commonly used in color image quantization [56], even if these approaches have been originally developed to color image segmentation applications such as the C-means [19], the fuzzy C-means [43] and the genetic C-means [56] clustering algorithms or the hierarchical merging approach [7].

1.6.2 The NeuQuant Neural-Net image quantization algorithm

The NeuQuant Neural-Net image quantization algorithm has been developed to improve the common Median Cut algorithm (Section 1.4.2). This algorithm operates using a one-dimensional self-organizing Kohonen Neural Network, typically with 256 neurons, which self-organizes through learning to match the distribution of colors in an input image. Taking the position in RGB-space of each neuron gives a high-quality color map in which adjacent colors are similar.

By adjusting a sampling factor, the network can either produce extremely high-quality images slowly, or produce good images in reasonable times. With a sampling factor of 1, the entire image is used in the learning phase, while with a factor of e.g. 10, a pseudo-random subset of 1/10 of the pixels are used in the learning phase. A sampling factor of 10 gives a substantial speed-up, with a small quality penalty. Careful coding and a novel indexing scheme are used to make the algorithm efficient. This confounds the frequent observation that Kohonen Neural Networks are necessarily slow.

1.6.3 The local k-means algorithm.

The local K-means algorithm (LKM), which has been introduced by Verevka and Buchanan [68, 69] is an iterative post-clustering technique that approximates an optimal palette using multiple subsets of image points. This method is based on a combination of a K-means quantization process and a self-organizing map (or Kohonen neural network). The aim of this method is to minimize simultaneously both the TSE and the standard deviation of squared error of pixels $\sigma = \sqrt{\sum_{\mathbf{c}\in C}(\|\mathbf{c} - q(\mathbf{c})\| - E(\mathcal{C}))^2}/card(C)$. Meanwhile small values of the TSE guarantee that a quantization process accurately represents colors of the original image, the minimization of the standard deviation preserves variations of colors in the quantized image. The main limitation of this method comes from the two measures used which treat each pixel independently. Consequently the spatial correlation among colors is not taken into

account. The main advantage of this method is its ability to select a palette without making any assumptions about the boundaries of color clusters.

1.7 Mapping methods

The inverse colormap operation is the process which maps an image into a limited set of representative colors. These representatives may be defined by a quantization algorithm or imposed by the default colormap of the output device. In order to minimize the visual distortion between the input image and the output one, inverse colormap algorithms map each color c of the input image to its nearest representative $\boldsymbol{Q}(c)$. The function $\boldsymbol{Q}$ may be defined by :

$$\begin{array}{rcl} \mathcal{C} & \rightarrow & \{c_1, \ldots, c_K\} \\ c & \mapsto & \boldsymbol{Q}(c) = \arg\min_{z \in \{c_1, \ldots, c_K\}} \|z - c\| \end{array} \tag{1.22}$$

where $\mathcal{C}$ represents the set of colors of the input image, and $\{c_1, \ldots, c_K\}$ the set of representative colors. The symbol $\|z - c\|$ denotes the Euclidean norm of the 3D vector $z - c$ computed in a given color space.

The value $\boldsymbol{Q}$(c) for a given color c may be computed thanks to an exhaustive search of the minimum of $\|z - c\|$ for all representative colors. Given a 256×256 image and a colormap of 256 colors, this trivial algorithm requires more than sixteen million distance computations (see equation 1.22). Thus, despite its simplicity and the fact that this method provides an exact solution, it is not practical for large images or interactive applications. Several methods described below have thus been designed to optimize the search of the closest representative. The complexity of the main algorithms is resumed in Table 1.2.

1.7.1 Improvements of the trivial inverse colormap method

The improvements of the trivial inverse colormap algorithm are numerous: Poskanzer [52] proposed improving the search by using a hash table which allows one to avoid the search of the nearest representative for any color already encountered. However, this optimization remains inefficient for images with a large set of different colors such as outdoor scenes. One other approach consists to approximate the L_2 Euclidean norm by a less expensive distance metric. Chaudhuri et al. [20] proposed the L_α norm as an approximation of the Euclidean distance. The L_α norm of a color $\mathbf{c}$ being defined by:

$$\begin{array}{rcl} \|\mathbf{c}\|_\alpha & = & (1-\alpha)\|\mathbf{c}\|_1 + \alpha\|\mathbf{c}\|_\infty \\ & = & (1-\alpha)\sum_{j=1}^{3} |\mathbf{c}^j| + \alpha \max_{i \in \{1,2,3\}} |\mathbf{c}^j| \end{array}$$

According to experiments performed by Verevka [68] the $L_{\frac{1}{2}}$ norm speeds up significantly the search without introducing a noticeably loss in output image quality.

The search can be further reduced by using the following considerations [36]:

- Partial sum: Using the square of the L_2 distance, the L_1 norm or the $L_{\frac{1}{2}}$ norm the distance between the input color and one representative is defined as a sum of three terms. The partial sum should be compared to the current minimal distance before each new addition. The distance calculation terminates is the partial sum is greater than the current minimal distance.

- Sorting on one coordinate: Let us assume that the representatives are sorted along one coordinate axis (e.g. the first one). Then the search starts with the representative whose first coordinate is the closest from the input color and continue in the increasing first coordinate distance order. The process terminates when the first coordinate distance

between the next representative and the input color is greater than the current minimal distance. Note that one should use an axis with a large variance to accelerate the termination of the search.

- Nearest neighbor distance: Given a representative $\mathbf{c}_j$ with $j \in \{1, \ldots, K\}$ its closest representative $\mathbf{c}_{q(j)}$ is defined by:

$$q(j) = \arg \min_{k \in \{1,\ldots,K\}, k \neq j} d(\mathbf{c}_j, \mathbf{c}_k)$$

Let us suppose that the current representative $\mathbf{c}_i$ traversed by the inverse colormap algorithm is the current closest representative from the input color $\mathbf{c}$. We additionally suppose that the current minimal distance $\Sigma_{min} = d(\mathbf{c}, \mathbf{c}_i)$ is less than one half of $d(\mathbf{c}_i, \mathbf{c}_{q(i)})$. We have thus:

$$\left\{ \begin{array}{lcl} \Sigma_{min} & = & d(\mathbf{c}, \mathbf{c}_i) \\ \Sigma_{min} & \leq & \frac{1}{2} d(\mathbf{c}_i, \mathbf{c}_{q(i)}) \end{array} \right.$$

Then, given any other representative color $\mathbf{c}_k$:

$$\begin{array}{lcll} & 2\Sigma_{min} & \leq & d(\mathbf{c}_i, \mathbf{c}_{q(i)}) \qquad \leq d(\mathbf{c}_i, \mathbf{c}_k) \leq d(\mathbf{c}_i, \mathbf{c}) + d(\mathbf{c}, \mathbf{c}_k) \\ \Rightarrow & 2\Sigma_{min} & \leq & \Sigma_{min} + d(\mathbf{c}, \mathbf{c}_k) \\ \Rightarrow & \Sigma_{min} & \leq & d(\mathbf{c}, \mathbf{c}_k) \end{array}$$

The inverse colormap algorithm should then terminate since no representative may be closest from $\mathbf{c}$ than $\mathbf{c}_i$.

1.7.2 Inverse colormap algorithms devoted to a specific quantization method

As mentioned in Section 1.1 (see also Figure 1.1) any quantization algorithm requires an inverse colormap step. The set of representatives defined by quantization algorithms is built in order to minimize a quantization error defined within some color space (e.g. $CIELuv$ or YIQ). On the other hand, inverse colormap algorithms map each color to its nearest representative using an Euclidean norm also defined within a given color space. Therefore, the inverse colormap algorithm should use the same color space than the quantization algorithm in order to be consistent with the criterion used to design the set of representatives.

The assignment of each color to its nearest representative by inverse colormap algorithms induces a partition of the image color set by a three-dimensional Voronoï diagram defined by the representatives. The partition defined by pre-clustering quantization methods generally does not constitute a Voronoï diagram but may be used as a close approximation of it if the partition has a low total squared error. Moreover, using the partition defined by quantization algorithms, data structures generated by these algorithms may be used to reduce the computation cost of the mapping function. For example, using a tree-structured vector quantizer (Section 1.4.2), the binary tree generated by the quantization algorithm may be used to retrieve the representative of an input color with approximately $\mathcal{O}(\log_2(K))$ operations.

1.7.3 Inverse colormap operations using $k-d$ trees

Inverse colormap methods may also be studded within the more general nearest neighbor framework. Friedman [11] proposed an algorithm based on an optimized $k-d$ tree to determine the m nearest neighbors of a given color. Given a set of colors, the $k-d$ tree is built by a recursive bipartitioning scheme (Section 1.4.2), each cluster being split along its coordinate axis with the greatest variance in order to have an equal number of colors in both sub clusters.

The recursive split algorithm terminates when each leaf of the binary tree generated by the algorithm contains less than a given number b of colors.

The search procedure initializes a list of the m closest colors encountered so far . Whenever a color is examined and found closer than the most distant member of this list, the list is updated. If the node under investigation is not terminal, the recursive procedure is called for the sub cluster enclosing the query color. When control returns, a test is made to determine if it is necessary to consider the other sub cluster. This test checks if the box delimiting the sub cluster overlaps the ball centered at the query color with radius equal to the distance to the m^{th} closest color so far encountered. This test is referred to as the *bounds overlap-ball* test. If the bounds overlap-ball test fails none of the colors on the other sub cluster can be among the m closest neighbor of the query color. If the bounds do overlap the ball, then the colors of the other sub cluster should be examined and the procedure is called recursively on this sub cluster.

The above algorithm may be applied to the inverse colormap framework, by setting $m = 1$ and using the set of representative as the initial set of colors encoded by the $k - d$ tree. According to Friedman, the closest representative is determined in $\mathcal{O}(\log_2(K))$ operations.

1.7.4 The locally sorted search algorithm

Heckbert's locally sorted search algorithm [35] uses the same basic idea as the bounds overlap-ball test but instead of performing a recursive decomposition of the color set, Heckbert defines a uniform decomposition of the RGB cube into a lattice of N cubical cells. Each cell of this lattice contains the list of representatives which could be the nearest representative of a color inside the cell. Each cell's list is defined by computing the distance r between the representative closest from the center of the cell and the farthest corner of the cell (see Figure 1.13). This distance gives an upper bound on the distance of any color in the cell to its nearest representative. Therefore, any representative which have a distance to the cell greater than r may be rejected from the list.

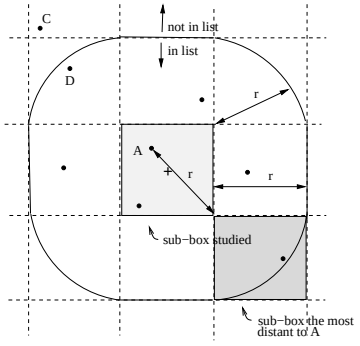


Figure 1.13: The locally sorted search algorithm. The representative A is the closest from the center of the studded sub-box. The distance from A to the farthest corner of the sub-box defines the distance r. The representative D belongs to the list associated to this sub-box while C is r units away from the sub-box and is thus rejected from the list.

Given an input color **c**, Heckbert's method determines the cell which contains **c** and traverses its associated list in order to determine its closest representative. The complexity of this method depends on the mean size of representative list which is determined by the number, the distribution of representative colors and by the number N of cells composing the lattice. If we denote by L the mean size of the representative lists, the complexity of

the preprocessing step which defines the lattice of N cells and sort the representative lists is equal to $\mathcal{O}(NK + NL\log(L))$ [35]. Experiments performed by Heckbert show that the number of distance computations required to determine the representative $\boldsymbol{Q}(\mathrm{c})$ of a color c may be decreased by 23 for 256 representative colors and 512 cells. Nevertheless, with a fixed number N of cells, this method remains approximately linear with respect to the number K of representatives. Moreover, the optimal number N of cells is difficult to estimate.

Locally sorted search shows the greatest advantage over exhaustive search when K is large and when the colors in the input image have a wide distribution [35]. In these cases the pre-processing time to create the database is overshadowed by the savings in search time.

Some variants of this algorithm such as those of Houle and Dubois [37] or Goldberg [32] have been proposed.

1.7.5 Inverse colormap operation using a 3D Voronoï diagram

Thomas's method [60] computes the values of the function $\boldsymbol{Q}$ thanks to a 3D discrete Voronoï diagram defined by the representative colors of the colormap. A discrete Voronoï diagram defined by p points is a partition of a discrete image into a set of p cells, all pixels of one cell being closer from one representative than the others. One advantage of discrete Voronoï diagrams beside real ones is that they can be computed thanks to incremental methods which initialize the n-dimensional image to be partitioned. Thus, using a discrete Voronoï diagram, the cell enclosing a given pixel is retrieved by reading its index in the n-dimensional image.

Using Thomas method, the Voronoï diagram is encoded thanks to a 3D array of integers. Each entry of this array represents a color and contains the index of its nearest representative. The main advantage of Thomas's method is that once the 3D array encoding the 3D Voronoï diagram has been computed the representative of any input color may be retrieved without any computation. This method is thus quite adapted to map any color to the default colormap of a screen. However, it has several drawbacks: First, this algorithm computes the representative of each displayable color. Thus, it involves many useless computations when applied to a few images. Secondly, using the RGB color space, the computation of the 3D Voronoï diagram requires the storage of 256^3 indexes. In order to reduce the number of initialized values and the amount of required memory, Thomas removes the three least significant bits of each R, G and B component. The 3D Voronoï diagram is then computed on a $32 \times 32 \times 32$ cube. This heuristic decreases significantly the amount of required memory but introduces several quantization errors (Section 1.2). Moreover, many quantization methods [14, 70, 74] use color spaces such as $CIELuv$, $CIELab$ [59] or YC_rC_b [10] which are more adapted to human vision than the RGB one. In this case, the inverse colormap algorithm has to use the same color space (Section 1.7.2). Using Thomas's method and a different color space than the RGB one, we have to allocate and initialize a 3D image which encloses the transformation of the RGB space. This constraint reinforces the problems linked to the number of data which must be allocated and initialized.

1.7.6 Inverse colormap operation by a 2D Voronoï diagram

The basic idea of this method proposed by Brun [18] approximates the 3D image color set by one of its 2D projection in order to perform the inverse colormap operation with a 2D Voronoï diagram instead of a 3D one. Experiments performed by Ohta [48] show that up to 99% of the information (see equation1.3, Section 1.4) of an image color set is contained in the plane defined by the first two eigenvectors of the covariance matrix.

Given an image color set, Brun computes the first two eigenvectors of the covariance matrix (equation 1.2) and defines a projection operator p onto the plane P_{princ} defined by

these two eigenvectors. The 3D Voronoï diagram associated to the colormap $\{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$ is then approximated by a 2D diagram V_I defined by the sites $\{p(\mathbf{c}_1), \ldots, p(\mathbf{c}_K)\}$.

The mapping of a color $\mathbf{c}$ to the representative whose associated cell in V_I encloses $p(c)$ induces two successive approximations: First, the 3D distances are approximated by 2D ones. This first approximation neglects the variations of the data set along the third eigenvector. Secondly, the use of a discrete Voronoï diagram involves rounding of each projected color $p(\mathbf{c})$ to the nearest pixel in V_I.

Due to these successive approximations this first mapping function often fails to map the input colors to their closest representatives. However, these approximations cause minor errors on the distance computations, so that the errors often affect the indices of adjacent Voronoï cells. The correction of the approximations requires thus to compute the neighborhood of each Voronoï cell. This neighborhood may be encoded by using the dual of the Voronoï diagram V_I named a Delaunay graph [6] and denoted by D_I. The data structures V_I and D_I are then combined in the following way:

Given an input color $\mathbf{c}$, Brun reads in V_I the index $V_I[p(c)]$ of the cell enclosing $p(c)$. The set $D_I[V_I[p(c)]]$ containing $V_I[p(c)]$ and the index of the cells adjacent to it is then read from the Delaunay graph D_I. The color $\mathbf{c}$ is then mapped to its closest representative among $D_I[V_I[p(c)]]$:

$$\boldsymbol{Q}(c) = \arg \min_{i \in D_I[V_I[p(c)]]} \|\mathbf{c} - \mathbf{c}_i\|^2 \tag{1.23}$$

where $\mathbf{c}_i$ denotes a representative color of the colormap $\{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$.

Note that equation 1.23 uses 3D distances, the 2D Voronoï diagram being only used to restrict the number of distance computations. Therefore, this method is similar to the one of Heckbert (Section 1.7.4) which performs a partition of the color space and associates a list of representative to each cell of the partition. The complexity of this inverse colormap method is determined by the mean size of lists $D_I[V_I[p(c)]]$ (see equation 1.23) which is related to the mean number of neighbors of a 2D Voronoï cell. It can be shown [53] that this number is bounded by 6. Therefore, once the diagrams V_I and D_I are computed this method requires less than 7 distance computations per pixel. The overall complexity of the method is bounded by $\mathcal{O}(9|I| + |V_I|)$ where $|I|$ and $|V_I|$ denote respectively the size of the input image and the array V_I.

Table 1.2: Complexity of the main inverse colormap methods. Symbols $|I|$ and K denote respectively the size of the input image and the number of representatives. Symbols L and N are defined in Section 1.7.4 while symbol $|V_I|$ is defined in Section 1.7.6.

		Complexity of	
Algorithm	Exact computation	inverse colormap step	pre-processing step
Trivial method	Yes	$\mathcal{O}(K\|I\|)$	
$k-d$ tree	Yes	$\mathcal{O}(\|I\| \log(K))$	$\mathcal{O}(K \log(K))$
Locally sorted search	Yes	$\mathcal{O}(L\|I\|)$	$\mathcal{O}(NK + NL \log(L))$
3D Voronoï diagram	No	$\mathcal{O}(\|I\|)$	$\mathcal{O}(2^{15} \log(K))$
2D Voronoï diagram	No	$\mathcal{O}(7\|I\|)$	$\mathcal{O}(2\|I\| + \|V_I\|)$

1.8 Dithering methods

Digital Halftoning was until recently commonly considered as a process of displaying continuous tone images on bi-level output devices (e.g. printers) [34, 39, 41, 50]. In order to extend these methods to color images, several generalizations have been proposed.

Considering the capability of the human visual system to distinguish different drops rapidly for high spatial frequencies, digital halftoning has been exploited to enhance *a posteriori* quantization processes by adding complementary perceived colors. Indeed, the human visual system may create additional perceived colors by averaging colors in the neighborhood of each pixel. Thus, halftoning trades-off high spatial resolution in favor of increased color resolution. Several investigations [33] have shown that excellent image quality with color palettes of very small size (as few as four colors) could be obtained when digital halftoning is used. A variety of halftoning techniques have been proposed to reach this objective, such as error diffusion techniques, dithering techniques or model-based halftoning techniques.

The common problem of these approaches is that they consider each color component as an individual gray scale image, consequently, due to the correlation between the color components, color shifts and false textural contours appear on the resulting image [4, 58, 67]. This problem is all the more noticeable when the number of colors that form a palette is very small. That is the reason why, some investigations proposed to develop vector error diffusion techniques in order to reduce effects induced by the correlation between color components.

In order to optimize the performance of dithering techniques, some adaptive dithering techniques [4, 66] have been also investigated. The key idea of these techniques is to compensate quantization errors by using an image-adaptive cost-function. More recently, others approaches combined dithering and quantization processes or designed quantizers according to the requirements of dithering algorithms.

1.8.1 The Error diffusion methods

The object of error diffusion is to quantize the image in such a way that the average value of the quantized image in a local region is the same as the average value of the original one in the same local region. In error diffusion, any errors produced in the quantization process in a pixel are propagated to neighboring pixels to be negated subsequently (see Figure 1.14), so that in a small group of neighboring pixels the average color is more accurate.

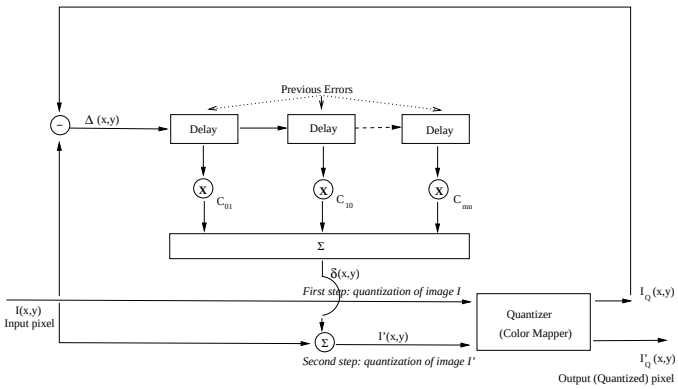
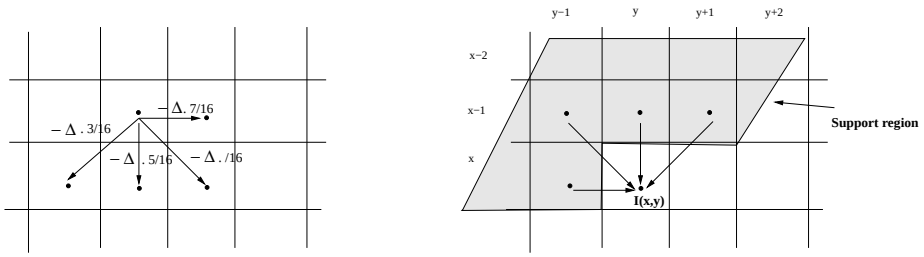


Figure 1.14: The dithering algorithm. Input pixels are dithered by the weighted sum of the previous quantization errors before being quantized for the second time.

Let $I'(x, y)$ be the pixel value resulting of the pseudo-random process, and $\Delta(x, y) =$



a) b)

Figure 1.15: The Flyod-Steinberg algorithm. The propagation of the quantization error to neighboring pixels (a) and the contribution of neighboring pixels to the noise vector (b)

$I(x,y) - I_Q(x,y)$ be the quantization error at pixel location (x,y), then $I'(x,y)$ can be described by :

$$I'(x,y) = I(x,y) \ + \ \delta(x,y)$$

with

$$\delta(x,y) = \sum_{i=0}^{m}\sum_{j=0}^{n} C_{ij} \ \Delta(x-i, y-j)$$

which represents a two-dimensional spatial filter (see Figure 1.15), such that:

$$C_{00} = 0 \text{ and } \sum_{i=0}^{m}\sum_{j=0}^{n} C_{ij} = 1$$

where the indexes i and j define the neighborhood over which quantization errors are accumulated according to coefficients C_{ij}.

The constraints on C_{ij} ensures that locally the quantization error average to a zero value. Since the objective of error diffusion is to preserve the average value of the image over local regions, a unity-gain low pass finite impulse response (FIR) filter is used to distribute the error. Since the eye is less sensitive to high spatial frequencies, the image resulting from an error diffusion process typically appears closer from the original image than those obtained with ordered dither [58](Section 1.8.2).

This algorithm has been originally proposed by Flyod and Steinberg [27] in 1975. Several variants have been proposed, such as those proposed by Fletcher [27], Heckbert [35], Bouman [12], Dixit [22], or Akarun [3]. Several improvements have also been proposed to speed the process, such as the parallel implementation proposed by Fletcher [25], or the fuzzy technique proposed by Akarun [1]. According to Fletcher [22], this optional pixel correction function is usually necessary when the number of output colors is very small, e.g. 32 or smaller.

Several investigations have shown that the error diffusion algorithm presents some major drawbacks. For example, according to Fletcher [27] or Bouman [12], colormap selection algorithm may be altered by the dithering process when input colors lie outside the convex hull of the representative colors. It is thus necessary to guarantee that all colors in the original image may be generated by taking a linear combination of colors in the colormap. Likewise, according to Knuth [41], an other major drawback of the error diffusion algorithm is its inherent serial property, e.g., the value $I'(x,y)$ depends in all pixels of input data $I(x,y)$. Furthermore, it sometimes puts "ghosts" into the picture. Even if the ghosting problem can be ameliorated by choosing the coefficients C_{ij} so that their sum be less than 1; the ghosts cannot be exorcised completely in this way. Finally, according to Bouman [12],

when the error diffusion algorithm is used in conjunction with an unstructured color palette, it requires an exhaustive search of the closest representative (Section 1.7). Although some basic improvements of the exhaustive search method (Section 1.7.1) this method remains computationally intensive.

1.8.2 The Ordered dither methods

The ordered dithering methods add a non-random patterns $\delta(i,j)$ to blocks of pixels $I(x,y)$ before the quantization step. The quantized image $I'(x,y)$ is thus deduced from the original one by :

$$I'(x,y) = I(x,y) \ + \ \delta(x \ mod \ n, y \ mod \ n)$$

where $\delta(i,j)$ is a n by n matrix of dithering values. The matrix δ is designed to have a 0 average value and an energy spectrum with a minimal energy at low spatial frequencies [41]. The amplitude of the pattern is determined so that any area of constant color value is quantized into a variety of nearby color values, thereby breaking up regions of false contouring. This reduces the correlated error patterns that are characteristic of areas of false contouring.

In contrast with random dithering processes, the basic idea of ordered dither techniques is to use a finite, deterministic, and localized threshold function to decrease the correlation of quantization errors. Likewise, in contrast with random dithering processes, ordered dither techniques are image-adaptative, e.g. these techniques exploit the local characteristics of the image to obtain improvements over a constant filter.

Several investigations have shown that the ordered dither algorithm presents some major drawbacks. According to Bouman [12], the application of ordered dithering is complicated by the non-uniform spread of colors within the palette. As the distance between nearby colors may vary significantly across the color space, it may be impossible to determine the amplitude which will sufficiently dither all areas of constant color value in the image without adding noticeable noise to other areas. Bouman et al [12] have thus proposed some adjustments to improve this technique. Nevertheless, it has been shown that no ordered dither method guarantees that after quantization the error spectrum remains concentrated at high spatial frequencies.

According to Kim et al. [40], ordered dither methods have the disadvantage of producing a binary-recursive and computerized texture that is unfortunately unsuitable. Meanwhile the Flyod-Steinberg method has the disadvantage to produce intrusive and snake-like patterns that are also unsuitable. They thus proposed the dot diffusion method which avoids both unsuitable previous properties but combines the sharpness of the Flyod-Steinberg method and the parallelism of the order dither method. According to Kim et al. dot diffusion methods have the desired property but tend to blur the image in the edge regions through the diffusion of the quantization error. This last problem is solved by an activity-weighted dot diffusion method which divide the dithering process into two sub-processes, one dependent on the object frequencies and one independent.

Another approach of ordered dithering technique has been proposed by Lin et al. [45] to enhance performance of compression schemes such as joint bilevel group (JPIG) techniques. The interest of this approach, based on a pixel interleaving (i.e. grouping pixels with similar dithering thresholds) strategy, is that it could be also extended to quantization scheme.

1.8.3 Vector dither methods

The major drawback of scalar diffusion techniques is that the color components are analysed independently while they are highly correlated. Indeed the natural ordering used by scalar quantizers is lost within the vector quantizers framework. Imposing an ordering, or organization, on a vector quantizer requires careful considerations. Rather than defining an

ordering on vectors, some investigations proposed a two-stage approach. In the first stage the original image is quantized by a scalar process and an error propagation mask is applied to compensate the quantization errors. In the second stage, the color palette is ordered and the resulting image is once again quantized by a vector process based on the ordered palette. Finally, an error propagation mask is used one more time to compensate the quantization errors. This approach has been used by Goldschneider et al. [33] to propose an embedded multilevel error diffusion technique.

Another strategy of vector error diffusion has been proposed by Akarun et al. [4]. In this paper, authors proposed two new adaptive diffusion techniques respectively based on vector and scalar diffusion, to correct the disturbing effects of error diffusion strategies. The proposed adaptation criterion is based on the statistics of the quantization error obtained from the dithered image. Rather than considering each color component as an individual gray scale image to obtain a scalar diffusion, Akarun et al. [4] proposed to use either the Karhunen-Loeve color coordinate system or a pseudo-Karhunen-Loeve system which provides better results than the two others scalar diffusion techniques.

1.8.4 Joint quantization and dithering methods

Instead of performing *a posteriori* the dithering process to eliminate contouring effects or other disturbing effects, some investigations combined the dithering and quantization processes. Since the error diffusion step changes pixel values after the quantization step, the set of representative designed by the quantizer is obviously not optimal in regards to the corrected image.

A first attempt to design the set of representatives based on a dithering process has been proposed by Akarun et al. [2]. Akarun's method combines a tree-structured vector quantizer (Section 1.4.2) and a dithering process based on a controlled displacement of cluster centers. The key idea is to obtain a wider colorspace in order to obtain more quantization colors in the color palette, and hence eliminate color impulses and false contours.

Another strategy consists to process a joint optimization of the selection of the representatives and their locations in order to benefit from the spatial averaging performed by the Human Visual System. Indeed as mentioned in Sections 1.5 and 1.8, one notable characteristic of the Human Visual System is the rapid drop of its capabilities to distinguish different colors for high spatial frequencies. This phenomenon creates additional perceived colors by a visual local averaging of the representatives. This effect is widely used by dithering techniques but can not be readily incorporated in usual quantization algorithms since the set of representatives is known only after the quantization step.

A joint optimisation of the selection of the representatives and their locations has been proposed by Scheunders et al. [57]. This method combines a quantization process based on a competitive learning clustering technique and a dithering process based on a simple local error diffusion technique. To take the error diffusion process into account, an objective error function is firstly computed and next minimized thanks to a competitive learning scheme which alternatively updates the palette colors in the color space and the color pixels in the color image.

Other authors [26, 54] use a model of the spatial averaging performed by the Human Visual System within the joint quantization scheme. This spatial averaging may be modeled by 3 low pass filters [26, 51] $(W_k)_{k\in\{1,2,3\}}$. Let us denote by $S_{i,k}$ the spatial support of filter W_k at position i. Given an image $I = (I_i)_{i\in\{1,\ldots,N\}}$ the k^{th} component of the perceived image $\tilde{I}$ is given by:

$$\forall k \in \{1,2,3\}\ \forall i \in \{1,\ldots,N\} \quad \tilde{I}_k(i) = \sum_{j\in S_{i,k}} W_k(j) I(j)_k$$

Given a set of representatives $\{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$, the location of each representative may be encoded by a $N \times K$ Boolean matrix such that $M(i,j)$ equals 1 if the pixel I_i is mapped to $\mathbf{c}_j$ and 0 otherwise. Using the $K \times 3$ vector R of representatives such that $R(i,k)$ is the k^{th} coordinate of $\mathbf{c}_i$, the output image I_Q is defined as:

$$\forall i \in \{1, \ldots, N\} \quad I_Q(i) = M(i,.)R$$

where $M(i,.)$ denotes the i^{th} line of matrix M.

The perceived output image $\tilde{I_Q}$ is then given by:

$$\forall k \in \{1,2,3\}\ \forall i \in \{1, \ldots, N\} \quad \tilde{I_Q}(i)_k = \sum_{j \in S_{i,k}} W_k(j) M(j,.) R(.,k)$$

where $R(.,k)$ denotes the k^{th} column of matrix R. Note that $I_Q(i)_k = M(i,.)R(.,k)$

The distance between the perceived input and output images may then be defined as the squared Euclidean distance between perceived colors:

$$\begin{aligned} H(M,R) &= \sum_{k=1}^{3} \sum_{i=1}^{N} \left(\tilde{I}(i)_k - \tilde{I_Q}(i)_k \right)^2 \\ &= \sum_{k=1}^{3} \sum_{i=1}^{N} \left(\sum_{j \in S_{i,k}} W_k(j) I(j)_k - \sum_{j \in S_{i,k}} W_k(j) M(j,.) R(.,k) \right)^2 \end{aligned}$$

The minimization of the error $H(M,R)$ requires thus to set both the matrices M and R. A minimization of $H(M,R)$ using a fixed value of R has been proposed by several halftoning methods [26, 51]. However, a joint mimization of M and R has been proposed only recently by Puzicha [54] who uses an alternating minimization scheme: 1) Minimize $H(M,R)$ with respect to M keeping R fixed. 2) minimize $H(M,R)$ with respect to R leaving M fixed. This two step strategy is iterated until convergence. Each optimization step is performed by an annealing method whose convergence is accelerated by a hierarchical decomposition of the image.

1.9 Conclusion and perspectives

This chapter has surveyed the current research on color quantization. Meanwhile significant gains in image quality may be obtained by researches on processing algorithms and display techniques, considerable improvements should result from researches on color metrics designed for complex image scenes instead of the CIE metrics which are based on large uniform areas. That is, only few papers account for the perceptual aspects of color within quantization algorithms. Two strategies have nevertheless been investigated to extend the use of perceptual aspects within quantization algorithms.

The first strategy consists to use either a perceptually uniform or a luminance/chrominance color space. The main characteristic of a uniform color spaces is that the perceived difference between two colors is proportional to the Euclidean distance between their representations in the uniform color space. A number of such spaces such as the $CIELAB$ color space which achieves this objective to varying degrees have been used in color image quantization [29]. Most of uniform color spaces are also based on a luminance/chrominance separation. Some investigations [29, 38, 54] studded the influence of the color space on the quality of the quantized images. However, none of these study exhibited one color space having definitive advantages within the quantization framework.

The second strategy consists to use spatio-color parameters which enable either to compensate *a posteriori* the main visible degradations between the original image and the quantized one or to minimize *a priori* the number of visible degradations which may appear on the

quantized image in regards to the original image. The strategy which minimizes *a priori* the spatio-color distortions should obviously be the best one. However, quantization algorithms using this strategy are generally computationally intensive. Moreover the design of analytic measures taking into account all visual distortions of a quantized image is still an open problem.

Image context playing an important role in human color vision, the use of an uniform color space is not sufficient to quantify perceptual color distance. Likewise, the relative positions of different colors in the image plane greatly influence our color interpretation. In order to handle this characteristic of the human visual system, several approaches such as those described in Section 1.5 have integrated a color activity criterion into the quantization scheme. Unfortunately, such approaches remain highly heuristic, and far from offering a mathematical model for context dependency of colors, to be integrated into as an objective function to a quantization process [40, 74]. In this context, color appearance models may appear promising. However, most of them are fairly complex and their suitability for color imaging remains to be comprehensively evaluated [58]. The development of simple models, more adapted to complex image scenes is a promising challenge for color imaging applications [62].

Bibliography

[1] L. Akarun, D. Ozdemir, and E. Alpaydin. Fuzzy error diffusion of color images. In *IEEE Processing*, pages 46–49, 1997.

[2] L. Akarun, D. Ozdemir, and O. Yalcin. Joint quantization and dithering of color images. In *Proceedings of IEEE*, pages 557–560, 1996. ICIP'96.

[3] L. Akarun, Y. Yardimci, and A. E. Cetin. Adaptive methods for dithering color images. In *Proceedings of IEEE*, pages 125–128, 1995.

[4] L. Akarun, Y. Yardlmcl, and A. E. Cetin. Adaptive methods for dithering color images. *IEEE Transactions on Image Processing*, 6(7):950–955, July 1997.

[5] M. E. Anderberg. *Cluster analysis for applications.* Academic Press, New York, 1973.

[6] F. Aurenhammer. Voronoi diagrams: a survey of fundamental geometric data structure. *ACM Computing surveys*, 33(3):345–405, 1991.

[7] R. Balasubramaian and J. Allebach. A New Approach to Palette Selection for Color Images. *Journal of Imaging Technology*, 17(6):284–290, December 1991.

[8] R. Balasubramaian, J. Allebach, and C. A. Bouman. Color-Image Quantization with Use of a Fast Binary Splitting Technique. *Journal of the Optical Society of America*, 11(11):2777–2786, November 1994.

[9] R. Balasubramaian, C. A. Bouman, and J. Allebach. Sequential Scalar Quantization of Color Images. *Journal of Electronic Imaging*, 3(1):45–59, 1994.

[10] R. Balasubramanian and J. Allebach. A New Approach to Palette Selection for Color Images. *Human Vision, Visual Processing, and Digital Display III (1991)*, SPIE 1453:58–69, 1991.

[11] J. L. Bentley, J. H. Friedman, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Software*, 3:209–226, September 1977.

[12] C. Bouman and M. Orchard. Color Image Display with a Limited Palette Size. *Visual Communications and Image Processing IV (1989)*, SPIE 1199:522–533, 1989.

[13] C. Bouman and M. Orchard. Color Quantization of Images. *IEEE Transactions on Signal Processing*, 39(12):2677–2690, December 1991.

[14] J. P. Braquelaire and L. Brun. Comparison and optimization of methods of color image quantization. *IEEE Transactions on Image Processing*, 6(7):1048–1052, july 1997.

[15] G. Braudaway. In *A Procedure for Optimum Choice of a Small Number of Colors from a Large Color Palette for Color Imaging*, pages 75–79, November 1986.

[16] L. Brun. *Segmentation d'images couleur à base Topologique*. PhD thesis, Université Bordeaux I, 351 cours de la Libération 33405 Talence, December 1996.

[17] L. Brun and M. Mokhtari. Two high speed color quantization algorithms. In cépaduès éditions, editor, *Proceedings of CGIP'2000*, pages 116–121, Saint Etienne, October 2000.

[18] L. Brun and C. Secroun. A fast algorithm for inverse color map computation. *Computer Graphics Forum*, 17(4):263–271, DECEMBER 1998. email me to obtain the article.

[19] M. Celenk. A color clustering technique for image segmentation. *Computer Vision, Graphics, and Image Processing*, 52:145–170, 1990.

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

[21] P. A. Chou, L. T., and G. R. M. Optimal pruning with applications to tree-structured source coding and modeling. *IEEE Trans. Inf. Theory*, 2:299–315, 1989.

[22] S. S. Dixit. Quantization of Color Images for Display/Printed on Limited Color Output Devices. *Computers and Graphics*, 15(4):561–568, 1991.

[23] W. H. Equitz. A new vector quantization clustering algorithm. *IEEE transactions on acoustics, speech, and signal processing*, 37(10):1568–1575, october 1989.

[24] Y. S. Feng and N. M. Nasrabadi. Dynamic address-vector quantization of rgb colour images. *IEEE Proceedings*, 138(4):225–???, August 1991.

[25] P. Fletcher. A SIMD Parallel Colour Quantization Algorithm. *Computers & Graphics*, 15(3):365–373, 1991.

[26] T. J. Flohr, B. W. Kolpatzik, R. Balasubramanian, D. A. Carrara, C. A. Bouman, and J. P. Allebach. Model Based Color Image Quantization. *Human Vision, Visual Processing, and Digital Display IV (1993)*, SPIE 1913:270–281, 1993.

[27] R. W. Flyod and L. Steinberg. An adptative algorithm for spatial gray scale. In SID, editor, *Int. Symp. Dig. Tech. Papers*, volume 36, 1975.

[28] C. Froidevaux, M.-C. Gaudel, and M. Soria. *Types de données et algorithmes*. Mc Graw-Hill, 1990.

[29] R. Gentile, J. Allebach, and E. Walowit. Quantization of Color Images Based on Uniform Color Spaces. *Journal of Imaging Technology*, 16(1):11–21, February 1990.

[30] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Norwel, 1991.

[31] M. Gervautz and W. Purgathofer. A Simple Method for Color Quantization: Octree Quantization. In N. Magnenat-Thalmann and D. Thalmann, editors, *New Trends in Computer Graphics*, pages 219–231. Springer-Verlag, New York, NY, 1988.

[32] N. Goldberg. Colour Image Quantization for High Resolution Graphics Display. *Image and Vision Computing*, 9(1):303–312, 1991.

[33] J. R. Goldschneider, E. A. Riskin, and P. W. Wong. Embedded mutilevel error diffusion. *IEEE Transactions on Image Processing*, 6(7):956–964, July 1997.

[34] J. Gomes and L. Velho. *Image processing for computer graphics*, chapter Digital halftoning. Springer Verlag, 1997.

[35] P. S. Heckbert. Color Image Quantization for Frame Buffer Display. *ACM Computer Graphics (ACM SIGGRAPH '82 Proceedings)*, 16(3):297–307, 1982.

[36] M. H. Hodgson. Reducing the computation requirements of the minimum distance classifier. *Remote sencing of Environnement*, 25:117–128, 1988.

[37] G. Houle and E. Dubois. Quantization of Color Images for Display on Graphics Terminals. *Proc. IEEE Global Telecommunications Conference (GLOBE_COM '86)*, pages 284–297, December 1986.

[38] A. K. Jain and W. K. Pratt. Color Image Quantization. *National Telecommunications Conference 1972 Record*, December 1972.

[39] J. F. Jarvis, C. N. Judice, and W. H. Ninke. A survey of techniques for the display of continuous tone pictures on bilevel displays. *Computer Graphics and Image Processing*, 4:13–40, 1976.

[40] K. M. Kim, C. S. Lee, E. J. Lee, and Y. H. Ha. Color image quantization and dithering method based on visual system characteristics. *Journal of Imaging Science and technology*, 40(6):502–509, Nov. 1996.

[41] D. E. Knuth. Digital halftones by dots diffusion. *ACM Transactions on Graphics*, 6(4):245–273, 1987.

[42] B. J. Kurz. Optimal Color Quantization for Color Displays. *IEEE Computer Vision and Pattern Recognition Proceedings*, pages 217–224, January 1983.

[43] Y. W. Lim and S. U. Lee. On the color image segmentation algorithm based on the thresholding and the fuzzy c-means techniques. *Pattern Recognition*, 23(9):935–952, 1990.

[44] J. Lin, J. Storer, and M. Cohn. On the complexity of the optimal tree pruning for source coding. In *Proc. of data compression conference*, pages 63–72. IEEE computer society Press Los Angeles, 1991.

[45] Y. Lin, Y. Wang, and T. H. Fan. Compaction of ordered dithered images with arithmetic coding. *IEEE Transaction on Image Processing*, 10(5):797–802, may 2001.

[46] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Trans. Commun.*, 1:84–95, january 1980. COM-28.

[47] S. P. Llyod. Least squares quantization. *IEEE Trans.*, pages 129–137, 1982. IT-28.

[48] Y.-I. Ohta, T. Kanade, and T. Sakai. Color information for region segmentation. *Computer Vision, Graphics, and Image Processing*, 13:222–241, 1980.

[49] A. W. Paeth. Mapping RGB Triples Onto Four Bits. In A. S. Glassner, editor, *Graphics Gems*, pages 233–245, 718. Academic Press Professional, Cambridge, MA, 1990.

[50] T. Pappas. Model-based halftoning of color images. *IEEE Transactions on Image Processing*, 6(7):1014–1024, July 1997.

[51] T. Pappas. Model based halftoning of color images. *IEEE Trans. Image Processing*, 6:1014–1024, July 1997.

[52] J. Poskanzer. Pbm+ image processing software package, 1991.

[53] F. P. Preparata and M. Shamos. *Computational Geometry*. Springer-Verlag, New York, 1985, 1985.

[54] J. Puzicha, M. Held, J. Ketterer, J. M. Buhmann, and D. W. Fellner. On spatial quantization of color images. *IEEE Transaction on Image Processing*, 9(4):666–682, April 2000.

[55] B. Rogowitz. The human visual system: A guide for display technologist. In *Proc. of the SID*, volume 24, pages 235–252, 1983.

[56] P. Scheunders. A genetic approach towards optimal color image quantization. In *IEEE Proceedings*, pages 1031–1034, 1996.

[57] P. Scheunders and S. D. Backer. Joint quantization and error diffusion of color images using competitive learning. In *IEEE Proceedings*, volume 1, pages 811–814, 1997. ICIP-97.

[58] G. Sharma and H. J. Trussell. Digital color imaging. *IEEE Transactions on Image Processing*, 6(7):901–932, 1997.

[59] J. Tajima. Uniform color scale applications to computer graphics. *Computer Vision, Graphics, and Image Processing*, 21(3):305–325, march 1983.

[60] S. W. Thomas. Efficient Inverse Color Map Computation. In J. Arvo, editor, *Graphics Gems II*, pages 116–125, 528–535. Academic Press Professional, Cambridge, MA, 1991.

[61] A. Trémeau. *Analyse d'images couleurs : du pixel à la scène*. PhD thesis, Université Jean Monnet, 1998. Habilitation à Diriger des Recherches.

[62] A. Trémeau. Color contrast parameters for analysing image differences. In Colour and I. Institute, editors, *CIS Proceedings*, pages 11–23, Derby, April 2000. Univ. of Derby. Color Image Science 2000 Conference.

[63] A. Trémeau, M. Calonnier, and B. Laget. Color quantization errors in terms of perceived image quality. In *Proceedings of the Int. Conf. on acoustics, speech and signal processing*, volume 5, pages 93–96, Adelaide, Apr. 1994. IEEE. ICASSP'94.

[64] A. Trémeau, C. Charrier, and H. Cherifi. A vector quantization algorithm based on the nearest neighbor of the furthest color. In *International Conference on Image Processing*, volume 3, pages 682–685, Santa Barbara, Oct. 1997. IEEE. ICIP'97.

[65] A. Trémeau, E. Dinet, and E. Favier. Measurement and display of color image differences based on visual attention. *Journal of Imaging Science and Technology*, 40(6):522–534, Nov. 1996.

[66] A. Trémeau and B. Laget. Color dithering based on error diffusion and subjective measures of image quality. In *IASTED International Conference on Signal and Image Processing*, pages 359–362, Las Palmas, Feb. 1998.

[67] O. Verevka. Digital halftoning. from Web Site http://web.cs.ualberta.ca/ oleg/dithering.html, 1995.

[68] O. Verevka and J. Buchanan. Local K-Means Algorithm for Color Image Quantization. *Graphics/Vision Interface '95*, May 1995.

[69] O. A. Verevka and J. W. Buchanan. Local k-means algorithm for color image quantization. In *Proceedings of GI 95*, Quebec Canada, 1995.

[70] S. Wan, S. Wong, and P. Prusinkiewicz. An Algorithm for Multidimensional Data Clustering. *ACM Transactions on Mathematical Software*, 14(2):153–162, 1988.

[71] S. J. Wan, P. Prusinkiewicz, and S. K. M. Wong. Variance-Based Color Image Quantization for Frame Buffer Display. *Color Research and Application*, 15(1):52–58, 1990.

[72] S. Wong, S. Wan, and P. Prusinkiewicz. Monochrome image quantization. In *Canadian conference on electrical and computer engineering*, pages 17–20, September 1989.

[73] X. Wu. Optimal quantization by matrix searching. *Journal of Algorithms*, 12(4):663–673, December 1991.

[74] X. Wu. Color Quantization by Dynamic Programming and Principal Analysis. *ACM Transactions on Graphics*, 11(4):348–372, October 1992.

[75] X. Wu and K. Zhang. A better tree-structured vector quantizer. In *Proceedings of the IEEE Data Compression Conference*, pages 392–401. IEEE Computer Society Press, 1991.

[76] Z. Xiang. Color image quantization by minimizing the maximum intercluster distance. *ACM Transactions on Graphics*, 16(3), July 1997.

[77] Z. Xiang and G. Joy. Color Image Quantization by Agglomerative Clustering. *IEEE Computer Graphics and Applications*, 14(3):44–48, May 1994.

1.10 List of Figures