SEQUENTIAL SCALAR QUANTIZATION OF COLOR IMAGES

Raja Balasubramanian, Charles A. Bouman, Jan P. Allebach

School of Electrical Engineering, Purdue University, West Lafayette, IN 47907.

Appeared In*:*

Journal of Electronic Imaging, vol. 3, no. 1, pp. 45-59, January 1994.

Corresponding Author:
Raja Balasubramanian, Xerox Webster Research Center
800 Phillips Road, Bldg 128-29E
Webster, NY 14580.
Phone: (716) 265 7838
email: raja@wrc.xerox.com

Co-authors:
Charles A. Bouman, School of Electrical Engineering,
Purdue University, West Lafayette, IN 47907.
Phone: (317) 494-0434
email: bouman@ecn.purdue.edu

Jan P. Allebach, School of Electrical Engineering,
Purdue University, West Lafayette, IN 47907.
Phone: (317) 494-3535
email: allebach@ecn.purdue.edu

ABSTRACT

We propose an efficient algorithm for color image quantization based on a new VQ technique which we call sequential scalar quantization (SSQ). The scalar components of the 3-D color vector are individually quantized in a predetermined sequence. With this technique, the color palette is designed very efficiently, while pixel mapping is performed with no computation. In order to obtain an optimal allocation of quantization levels along each color coordinate, we appeal to asymptotic quantization theory, where the number of quantization levels is assumed to be very large. We modify this theory to suit our application, where the number of quantization levels is typically small. In order to utilize the properties of the human visual system (HVS), the quantization is performed in a luminance-chrominance color space. A luminance-chrominance weighting is introduced to account for the greater sensitivity of the HVS to luminance than to chrominance errors. A spatial activity measure is also incorporated to reflect the increased sensitivity of the HVS to quantization errors in smooth image regions. The algorithm yields high quality images, and is significantly faster than existing quantization algorithms.

1. INTRODUCTION

Many low cost color output devices use 8 bit frame buffers to display their images. This means that a maximum of $2^8 = 256$ colors may be displayed simultaneously. Since most typical images contain far more than 256 colors, they cannot be displayed directly on such a device. However, these systems often retain a resolution of 8 bits per color at the input of each of the three D/A converters, and allow the user to choose a palette of 256 colors from a total of $2^{24} \approx 1.6 \times 10^7$ colors. The two-step process of selecting a palette of colors and mapping each input image pixel to a palette color is known as color image quantization. One may design either a universal palette to be used for all images, or a customized palette for each individual image. The former is clearly an inferior strategy, and will often require some postprocessing such as halftoning in order to enhance the visual quality of the image. However, it is attractive from a computational standpoint, since the palette is designed only once. We will be focusing on color quantization based on image dependent palettes. High image quality may be achieved with this method, since the palette for a given image is designed based on the color distribution of that specific image. However, since the palette needs to be redesigned for each new image, this strategy is more computationally intensive. It is therefore important that while we seek to produce high quality quantized images, we also place considerable emphasis on reducing the computational cost of the quantization algorithm, so that the processing time and memory are not objectionable to the user of any practical desktop application.

Since a color at a pixel is a triplet or 3-D vector of *R*, *G*, and *B* signals, color quantization may be viewed from the context of vector quantization (VQ). The colors in the image form a training or test set of color vectors, and the palette is the codebook of output color vectors. Several VQ based color quantization algorithms have been described. The methods proposed by Heckbert,[1] Braudaway,[2] and Gentile *et al.*[3] involve an initial selection of a palette followed by iterative refinement of this palette using the Linde, Buzo, Gray VQ algorithm.[4] Gentile *et al.*[3] proposed performing the VQ in a uniform color space. These methods yield high

quality images, but are computationally very intensive. Another class of algorithms[5, 6] uses clustering techniques to generate a palette of colors. In Dixit's scheme,[5] the image is randomly sampled and a set of color clusters is generated by a pairwise nearest neighbor (PNN) merging technique. The cluster centroids are then chosen as the palette colors. Balasubramanian and Allebach proposed an algorithm[6] based on a clustering VQ method developed by Equitz,[7] which also uses a PNN technique. Here, k-dimensional trees are utilized to perform efficient nearest neighbor searches. In addition, the algorithm uses histogramming and prequantization to decrease the size of the input data set, and a spatial activity measure is introduced to improve the visual quality of the image.

A third class of algorithms uses splitting techniques to divide the color space into smaller subregions and pick a representative palette color from each subregion. These algorithms include Heckbert's median cut technique,[1] a variance based method proposed by Wan *et al.*,[8] and a binary splitting algorithm developed by Orchard and Bouman.[9] The differences lie in the way the splitting is performed. The median cut and variance based algorithms both use splitting planes that are perpendicular to the coordinate axes. Hence, the quantization regions are rectangular polytopes in 3-D space. The binary splitting algorithm is a tree-structured VQ algorithm and is the closest to optimal (in a quantitative sense) of the three splitting techniques. In each region, the splitting plane is chosen to pass through the centroid of all the colors in that region, and is oriented to be perpendicular to the direction of maximum total squared variation. The latter is derived from the principle eigenvalue and eigenvector of the covariance matrix of the data in that region.[9] At each step, the region with the largest associated principle eigenvalue is chosen to split. This algorithm also incorporates spatial activity measures to enhance the subjective quality of the image. Balasubramanian *et al.* proposed a modified binary splitting algorithm,[10] where the binary splitting is preceded by a histogramming and prequantization step. This has the effect of reducing the computational cost of the splitting operation while preserving a high level of image quality.

In a related paper,[11] we propose a general method for quantizing vectors, which we call sequential scalar quantization (SSQ). As the name implies, the basic idea behind the technique is to sequentially quantize the scalar components of a vector, rather than to quantize the vector as a whole. Since we are quantizing scalar rather than vector variables, SSQ involves far less computation than VQ. At the same time, due to its sequential nature, SSQ shares with conventional VQ the ability to exploit the correlation and statistical dependency between scalar components of a vector. As is the case with any VQ technique, SSQ attempts to minimize a distortion measure. We appeal to asymptotic or high-rate quantization theory in order to analyze the performance of SSQ with respect to such a measure.[11] This theory provides us with closed form expressions for the distortion resulting from SSQ as a function of the quantizer design parameters, and allows us to find the optimum parameter values that minimize the distortion at each successive stage of the sequential quantization.

In this paper, we apply SSQ specifically to the color quantization problem. The asymptotic theory proves to be a very useful tool in designing color palettes even when the number of output levels is small. The color vectors are transformed to an appropriate color space prior to the sequential quantization, and the palette is designed very efficiently. Moreover, unlike existing algorithms, where the pixel mapping places a heavy demand on either computation or storage, our technique allows the pixel mapping to be performed with no computation and with moderate storage. Hence, this method is far more efficient than any of the algorithms described above. In addition, the algorithm uses properties of the human visual system in a simple way to improve the visual quality of the quantized image. The resulting image quality is comparable with or superior to that obtained from existing VQ algorithms.

This paper is organized as follows. In Sec. 2, we begin by describing the basic VQ problem. We then introduce SSQ and discuss the benefits of using SSQ over conventional VQ and independent scalar quantization. In Sec. 3, we present relevant results from asymptotic quantization theory; and in Sec. 4, we show how these results may be used to design a color

quantization algorithm. Experimental results are presented in Sec. 5; and concluding remarks are collected in Sec. 6.

2. VECTOR QUANTIZATION

We begin by introducing the terminology and basic ideas behind VQ. Throughout this paper, lower case notation will be used to denote real variables and vectors, while random variables and vectors will be written in upper case notation. Vectors will be represented by boldface notation. We denote by $p(\ )$ the probability density function of a random variable or vector, and by $P(\ )$ the probability of an event. The notation R^k will be used to denote the k-dimensional Euclidean space of real variables.

2.1 Basic Definitions

An N point k-dimensional vector quantizer Q is a function that maps an input $\boldsymbol{X}$ with a probability density function $p(\boldsymbol{x})$ to one of a finite number of output vectors $\boldsymbol{y}_1,\ldots,\boldsymbol{y}_N$. The set of output vectors $C = \{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_N\}$ is called a codebook. Such a quantizer defines a partition $S = \{S_1,\ldots,S_N\}$ of N regions in R^k, where $S_i = \{\boldsymbol{x} \in R^k : Q(\boldsymbol{x}) = \boldsymbol{y}_i\}$. For the case where $k = 1$, a quantizer is specified by a set of decision thresholds and output levels. The quantization consists of two steps: the codebook design, which involves appropriately selecting the output points $\boldsymbol{y}_1,\ldots,\boldsymbol{y}_N$; and the mapping of each input vector to one of the output points according to the rule $Q(\boldsymbol{x}) = \boldsymbol{y}_i$ if $\boldsymbol{x} \in S_i$. In practice, the vector mapping consists of an encoder which assigns to each input $\boldsymbol{x}$ a channel symbol, and a decoder which maps each channel symbol to a unique output vector in the codebook. The quantizer is designed to minimize some average distortion measure between its input $\boldsymbol{X}$ and output $Q(\boldsymbol{X})$. This implies that the quantizer chooses its output vectors in some manner that reflects the distribution $p(\boldsymbol{x})$ of the input. The distortion measure that is often used is the mean-squared error (MSE),

$$D_k = \frac{1}{k} E\left\{\|\boldsymbol{X} - Q(\boldsymbol{X})\|^2\right\}, \tag{1}$$

where $E\{\ \}$ denotes expected value with respect to the input distribution p, and $\|\ \|$ denotes Euclidean distance. The two necessary conditions for a quantizer to be optimal with respect to MSE[12] are that 1) the output vector $\boldsymbol{y}_i$ in each region S_i is chosen to be the centroid of all $\boldsymbol{x}$ in S_i; and 2) each input $\boldsymbol{x}$ is quantized to the closest output vector $\boldsymbol{y}_i$ in the codebook, *i.e.* $S_i = \{\boldsymbol{x} \in R^k : \| \boldsymbol{x} - \boldsymbol{y}_i \|^2 \leq \|\boldsymbol{x} - \boldsymbol{y}_j \|^2,\ j = 1,..., N\}$. This latter step, which involves an exhaustive distance calculation between the input vector and each output vector in the codebook, is the operation that makes VQ a computationally intensive scheme. Efficient strategies have been devised to reduce the computation required for nearest neighbor searches.[12]

The above two conditions are the basis for the iterative codebook design algorithm proposed by Linde, Buzo, and Gray.[4] Because the iterations are computationally expensive, other suboptimal but efficient techniques have been proposed such as tree-structured VQ.[12] We will use the term conventional VQ to refer to all such methods that quantize a vector as a whole entity.

2.2 Scalar Quantization of Vectors

Another simple but suboptimal method of quantizing a vector $\boldsymbol{X} = [X_1,...,X_k]^t$ is to quantize each of its individual scalar components X_i, $1 \leq i \leq k$. This may be done either independently or in a sequential fashion.

2.2.1 Independent Scalar Quantization (ISQ)

This is the conventional method of scalar quantization. A codebook C_i of scalar outputs is designed independently for each scalar component X_i, $1 \leq i \leq k$, according to its marginal distribution $p(x_i)$. The final codebook is a k-fold Cartesian product of the k scalar codebooks, and is therefore known as a product code. A 2-D example of this scheme is shown in Fig. 1a for a 25 point quantizer. The symbols **x** denotes the output vectors, which are taken to be the centroids within each region. Vector mapping may be accomplished by independently encoding each X_i to a channel symbol through a set of k lookup tables (LUT's). This is depicted in Fig. 2a for $k = 3$. The outputs of the k LUT's are independently decoded to scalar outputs $Y_1,..,Y_k$, which constitute

the output vector $\boldsymbol{Y} = [Y_1,...,Y_k]^t = Q(\boldsymbol{X})$. Since the codebook design only involves quantization of scalar variables, and the encoding operation only entails indexing into LUT's, ISQ involves far less computation than conventional VQ. However, because ISQ uses only the marginal distribution of each scalar component, it cannot take inter-data correlations into account. As a result, many output points are wasted in regions where the input has zero probability of occurrence, as shown in Fig. 1a.

2.2.2 Sequential Scalar Quantization (SSQ)

With this approach, the first scalar X_1 is quantized to some predetermined number of levels N_1 based on its marginal distribution $p(x_1)$, as with the ISQ scheme. Each subsequent X_i, $2 \le i \le k$, is then quantized based on a set of conditional distributions of X_i within regions formed from the quantization of the scalars $X_1,...,X_{i-1}$. A 2-D example of SSQ is shown in Fig. 1b for the same input distribution. In this example, first X_1 is quantized to $N_1 = 5$ levels. This results in the columns B_{2j}, $1 \le j \le 5$, in R^2. Next, we quantize X_2 but confine the quantization to the columns B_{2j} formed from the quantization of X_1. Each B_{2j} is quantized to n_{2j} levels along x_2. This results in a 2-D quantizer with $N_2 = 16$ output points in R^2. The encoding of input vectors may be performed through a sequential or multistage LUT, as shown in Fig. 2b for $k = 3$. The input to the first LUT is X_1. The output symbol b_{i-1} of the $(i-1)$th LUT $2 \le i \le k$, is then fed to the input of the i-th LUT along with the i-th scalar component X_i. Finally, the output symbol b_k of the last encoder is decoded to one of the output vectors in the codebook C.

As is the case with ISQ, the codebook design in SSQ involves only scalar quantization, and the encoding operation entails no computation. Hence, there is a significant computational advantage to be gained by using SSQ rather than conventional VQ. On the other hand, unlike ISQ, the SSQ technique uses both marginal and conditional distributions of the scalar components, and can therefore exploit inter-data correlations. This is seen in our 2-D example, as SSQ places all its output points only within the region of support of the joint distribution of the input, thus offering performance that is superior to that of ISQ. We choose SSQ for our color

quantization application, because it combines the computational advantage of ISQ and the performance advantage of VQ.

Due to the relevance to our color quantization application, we will assume henceforth that the input is a 3-dimensional random vector $\boldsymbol{X} = [X_1, X_2, X_3]^t$. We can summarize the basic steps involved in designing the codebook for an N point SSQ as follows:

1) Quantize the first component X_1 to some predetermined number of levels N_1 according to its marginal distribution $p(x_1)$. This results in N_1 columnar regions B_{2j}, $1 \le j \le N_1$, in R^2. (Here B_{ij} refers to the j-th quantization region in R^i.)

2) Quantize X_2 within each B_{2j}, $1 \le j \le N_1$, to n_{2j} levels according to its conditional distribution $p(x_2 \mid B_{2j})$. This results in a total of N_2 rectangular quantization regions in R^2, and therefore a set of N_2 columnar regions B_{3j}, $1 \le j \le N_2$, in R^3.

3) Quantize X_3 within each B_{3j}, $1 \le j \le N_2$, to n_{3j} levels according to its conditional distribution $p(x_3 \mid B_{3j})$. This results in the desired $N_3 = N$ quantization regions in R^3.

4) Pick the centroid of each of the N regions as the output codeword for that region.

Note that we have the following two constraints:

$$N_1 \le N_2 \le N_3 = N; \tag{2a}$$

$$\sum_{j=1}^{N_{i-1}} n_{ij} = N_i \tag{2b}$$

Several issues need to be addressed in this scheme. First of all, for a fixed number of overall quantization levels, how do we allocate the relative number of quantization levels along each x_i? In other words, referring to the algorithm above, for fixed N_3, what values do we pick for N_1, N_2, n_{2j}, $1 \le j \le N_1$, and n_{3j}, $1 \le j \le N_2$? This is analogous to the bit allocation problem encountered in ISQ.[13] Secondly, we need to determine the best order in which to quantize the X_i's (we have tacitly assumed that the X_i's are quantized in the order $X_1, X_2, ..., X_k$). Finally, we

must specify a scalar quantization scheme, *i.e.* a method of placing a fixed number of decision and reconstruction levels along a scalar dimension according to either the marginal or conditional distribution within some region. In the next section, we use asymptotic quantization theory[11] to choose these design variables so as to optimize SSQ with respect to the MSE distortion measure. Although MSE is often not an accurate measure of the subjectively perceived error, we will use it as a starting point to design the algorithm, as it is a mathematically tractable measure. Later, we will also discuss a simple visually weighted MSE criterion.

3. RESULTS FROM ASYMPTOTIC SSQ THEORY

The fundamental assumptions in this theory are that the number of output quantization levels is large (or equivalently, the quantization cells are small) and that the distribution $p(\boldsymbol{x})$ of the input vector $\boldsymbol{X}$ is a relatively smooth function of $\boldsymbol{x}$.[11] In this section, we present only the results necessary for our application; the reader is referred to Ref. 11 for a complete derivation of the theory. In Sec. 4, we will show how these results can be used in a practical application, namely color image quantization, even when the number of quantization levels is relatively small. For notational simplicity, we will assume for now that X_1 is quantized first, followed by X_2, and finally X_3. Our main objective is to pick the SSQ design parameters described in Sec. 2.2.2 so as to minimize the 3-D MSE subject to the sequential structure.

We first turn to the more basic question of what quantization method we should use to locate the output levels and decision thresholds along a scalar dimension. Let X be a random variable with a probability distribution $p(x)$ for which we wish to design an N point scalar quantizer so as to minimize the resulting 1-D MSE. For large N, the quantizer may be characterized by a function $\lambda(x)$ known as the quantizer density function[12] (qdf) which specifies the relative spacing of quantization levels in the neighborhood of x. That is, $N\lambda(x)dx$ is the number of quantization levels within the interval $[x, x+dx]$. Note that by definition, $\lambda(x)$ integrates to 1. It has been shown[12] that the function λ that minimizes the 1-D MSE is given by

$$\lambda(x) = \frac{p(x)^{1/3}}{\int p(x)^{1/3} dx}. \tag{3}$$

Similarly, if we wish to design a quantizer for a scalar input X according to its conditional distribution $p(x \mid B)$ within some region B, then the optimal quantizer spacing is given by

$$\lambda_B(x) = \frac{p(x \mid B)^{1/3}}{\int p(x \mid B)^{1/3} dx}. \tag{4}$$

We will assume that the scalar quantization scheme used in SSQ is one that achieves the optimal spacing given in (3) and (4).

We now return to the problem of designing an N point 3-D SSQ as described in Steps 1 - 4 in Sec. 2.2.2 to minimize the overall 3-D MSE. Since the MSE in (1) is a separable distortion measure, we can write it as a sum of individual MSE's along each of the three coordinates

$$\begin{aligned} D_3 &= \frac{1}{3} E\left\{\|\boldsymbol{X} - \boldsymbol{Y}\|^2\right\} \\ &= \frac{1}{3}\left[E\{(X_1 - Y_1)^2\} + E\{(X_2 - Y_2)^2\} + E\{(X_3 - Y_3)^2\}\right] \\ &= d_1 + d_2 + d_3 \ , \end{aligned} \tag{5}$$

where $\boldsymbol{Y} = Q(\boldsymbol{X})$. If we make the asymptotic assumption that the number of quantization levels along each dimension is very large, then it can be shown[11] that the MSE may be approximated by a function of the form:

$$D_3 = \frac{1}{N_1^{\,2}}\alpha + \frac{N_1^{\,2}}{N_2^{\,2}}\beta + \frac{N_2^{\,2}}{N^2}\eta \ , \tag{6a}$$

where N_1 and N_2 are the relative allocations of quantization levels described in Sec. 2.2.3, and α, β, and η are constants that depend on the statistics of the input vector $\boldsymbol{X}$

$$\alpha = \frac{1}{24}\left\|p(x_1)\right\|_{1/3} \ ;$$

$$\beta = \frac{\left\|\left\|p(x_2 \mid x_1)\right\|_{1/3} p(x_1)^{5/3}\right\|_{1/3}}{\left[\int p(x_1)^{1/3} dx_1\right]^2} \ ; \tag{6b}$$

$$\eta = \frac{\left\|\left\|p(x_3 \mid x_1, x_2)\right\|_{1/3} p(x_1, x_2)^{5/3} p(x_1)^{4/9}\right\|_{1/3}}{36\left\{\int\int p(x_1, x_2)^{1/3} p(x_1)^{2/9} dx_1 dx_2\right\}^2} \ .$$

Here we have used the notation $\|p(x_i)\|_m \equiv \left[\int p(x_i)^m dx_i\right]^{1/m}$.

For our application, we will not concern ourselves too much with the expressions for α, β, and η. In fact, as we will see shortly, the algorithm does not even compute these quantities, but rather estimates them from the image data. Our main interest is in the functional form of (6a), which allows us to minimize the MSE with respect to N_1 and N_2, yielding the optimal values

$$N_1 = N^{1/3}\left(\frac{\alpha^2}{\beta\eta}\right)^{1/6} \quad ; \quad N_2 = N^{2/3}\left(\frac{\alpha\beta}{\eta^2}\right)^{1/6} . \tag{7}$$

In deriving (6), it may also be shown[11] that the optimal allocation of the number of quantization levels n_{ij} among the B_{ij}, $1 \le j \le N_{i-1}$, is given by

$$n_{ij} = N_i \left(\frac{r_j}{\sum_{m=1}^{N_{i-1}} r_m} \right); \qquad i = 2, 3; \quad j = 1,..., N_{i-1} \ , \tag{8}$$

where $r_j = P(B_{ij})^{1/3} \int p(x_i \,|B_{ij})^{1/3} dx_i$; N_1 and N_2 are as given in (7); and $N_3 = N$. Here, $P(B_{ij})$ is the probability that an input vector $\boldsymbol{x}$ falls in the region B_{ij}. In general, (8) will yield real-valued n_{ij}. In Sec. 4, we will address the problem of picking integer-valued n_{ij} that satisfy the constraint (2b).

Finally, we turn to the issue of the order in which to quantize the scalar components of the vector $\boldsymbol{X}$. There is no simple way to predict the optimal ordering except to evaluate (6) with the optimal allocations given in (7) and (8) for every possible ordering, and pick the ordering for which D_3 is minimum. In the next section, we will show how an appropriate choice of the color space for quantization can be used to restrict the number of possible orderings.

4. APPLICATION TO COLOR IMAGE QUANTIZATION

In the preceding sections, we have described an algorithm for the sequential scalar quantization of a 3-D vector. In this section, we focus specifically on how SSQ may be used for

color image quantization. For this application, the input to the quantizer is the set of image colors, and the codebook of output vectors is the color palette. First, we need to specify the color coordinate system that we are going to use to define a 3-D color vector.

4.1 Choice of Color Space for Quantization

A color vector is normally represented by a triplet $X = [R,G,B]^t$ of red, green, and blue primary signals, with each signal being represented by 8 bits of precision (*i.e.* an integer in the range [0, 255]). Suppose R_L, G_L, and B_L are values of three color primaries that are equal to (or linearly related to) the measured luminance of the primaries. We will call these primaries linear color primaries. These primaries cannot be displayed directly because most monitors exhibit a nonlinear relationship between the input value of the color signal and the corresponding output luminance. The nonlinearity for a particular primary may be approximated by a power law relationship:

$$C_L = C^{\gamma}, \qquad C = R, G, B \tag{9}$$

where C is the input value of the primary, C_L is the luminance of the primary, and γ is a value that usually falls between 2 and 3, depending on the monitor. (More sophisticated models may be used to characterize the nonlinearity.[3]) In order to account for this nonlinearity, users typically perform a transformation on the linear color primaries of the form $R = R_L^{1/\gamma}$; $G = G_L^{1/\gamma}$; $B = B_L^{1/\gamma}$. The signals R, G, B are called gamma-corrected signals and are ready to be displayed on the given monitor. Many device-independent color spaces are also characterized by gamma-corrected coordinates. One such color space is the SMPTE RGB color space, which was proposed as a television standard.[14] The value of γ for this color space is 2.2. All the color images used in this work are assumed to be in SMPTE gamma-corrected RGB coordinates.

It is well known that the human visual system (HVS) perceives a color stimulus in terms of luminance and chrominance attributes, rather than in terms of R, G, and B values. Hence, we propose transforming the image to a luminance-chrominance space prior to performing the quantization. To this end, we pick the YC_rC_b opponent color space[14] which is related to the

SMPTE RGB space by a simple linear transformation. Assuming that *R*, *G*, and *B* occupy the range 0-255, the transformation is given by:

$$Y = 0.299\,R + 0.587\,G + 0.114\,B \tag{10}$$
$$C_r = 0.713\,(R - Y) + 128$$
$$C_b = 0.564\,(B - Y) + 128.$$

where the *Y*, C_r, and C_b values have been scaled to also occupy the range 0-255. From now on, 3-D color vectors will be assumed to be in YC_rC_b coordinates. Since YC_rC_b is a linear transformation of a gamma-corrected RGB space, it is also a gamma-corrected space. As can be seen from (10), *Y* is the gamma-corrected luminance component representing achromatic colors, the C_r coordinate describes the red-green variation of the color, and the C_b coordinate describes the yellow-blue variation of the color. The conversion to YC_rC_b is one of the key factors that enables us to achieve high image quality. After the palette is generated, the palette colors are converted back to SMPTE RGB through the inverse of (10), and then transformed to the RGB space of the output device. The latter transformation involves a conversion from SMPTE RGB to linear coordinates; a linear transformation from the SMPTE phosphor primaries to those of the monitor; and gamma correction to account for the monitor's nonlinearity. The reader is referred to the paper by Gentile *et al.*[3] for details of this procedure. The entire calibration process is summarized in Fig. 3. We note that the transformation T_1^{-1} and all subsequent transformations are only performed on the palette colors.

4.2. Preprocessing

The SSQ design requires information about the distribution of the image color vectors. Since we are designing a customized palette for a given image, the 3-D color histogram of the image contains all the necessary information about the input distribution. To this end, the image is scanned, and a 3-D histogram is generated that keeps track of each distinct color and its frequency of occurrence.

4.2.1 Prequantization

Since the HVS is far more sensitive to the luminance coordinate than to chrominance, we maintain the full 8-bit resolution along luminance, and drop the least significant bit of each of the chrominance coordinates while generating the histogram. This prequantization has the effect of significantly reducing the number of image colors without noticeably degrading the perceived quality of the image. Note that we are effectively scaling or dividing each chrominance coordinate by a factor of 2. The quantization is performed on the scaled coordinate system, and the inverse scaling is performed at the end on the palette colors. We emphasize here that the chrominance scaling factor is introduced only to reduce the number of image colors. For the purpose of SSQ analysis and design, we will implicitly account for this factor so that all 3 components are equally weighted in the distortion measure (5).

4.2.2 Efficient Histogramming

Storing a color histogram at a resolution of 22 bits (8 bits for luminance and 7 bits for each of the chrominance components) would require a prohibitive amount of memory ($2^{22} \approx 4$ million histogram elements). We use a technique developed previously[10] that employs a 2-D array to store 8 bits of the luminance component and 7 bits of the C_r chrominance component by direct access, while using a linear search strategy to store 7 bits of the C_b coordinate. Since the searches are performed on small linear lists, histogram access and updates are performed efficiently. Finally, the linear lists are reformatted as a single linked list that keeps each distinct color and its count. The memory requirements are $2^{15} \approx$ 32K elements for the 2-D array, and N_c elements for the linked list, where N_c is the number of distinct image colors.[10] The reader is referred to Ref. 10 for a detailed description of the histogram data structures.

4.2.3. Subsampling of the Image

Often, pixels within a small neighborhood have similar or identical color values. Hence, rather than considering every pixel during histogram generation, we can look at a subset of the image pixels and obtain a histogram that still represents the color distribution reasonably well.

To this end, we subsample the image by a factor of 2 in each direction when building the histogram. This subsampling greatly reduces the overall computational requirement of the algorithm.

4.3 Palette Design Using SSQ

After the histogram is generated, we have a list of N_c distinct colors $\boldsymbol{x}_j$, and their frequencies of occurrence f_j, $j = 1,..., N_c$. In contrast to our earlier discussion of SSQ, where we had a continuous input random vector $\mathbf{X}$ with distribution p($\mathbf{x}$), we now have a discrete random color vector with discrete probabilities $p(\boldsymbol{x}_j) = f_j/N_c$ which we shall assume to be completely specified by the image histogram. Although we speak of integrals and statistical expectation, these are in fact implemented as summations and sample averages, respectively. This discretization does not in any way alter the basic SSQ algorithm. The color palette may be designed using the steps outlined in Sec. 2.2.2 and the formulas (7), (8). However, we need to exercise caution in using the asymptotic results for our application, because the assumption that the number of output vectors is very large no longer holds true when we are designing palettes that are of sizes on the order of 256. We discuss below how the theory may be used in a meaningful way to implement the algorithm for small palettes.

4.3.1 Design of a Scalar Quantizer According to λ(x)

Although the concept of a quantizer density function $\lambda(x)$ is meaningful only in an asymptotic sense, it can still be used as a guiding rule to design a scalar quantizer for small N. Consider an input X whose distribution $p(x)$ has support in the finite interval $[A,B]$. The problem at hand is to design an N point quantizer Q whose output points are spaced according to the optimal function $\lambda(x)$ given in (3). Recall that a scalar quantizer is defined by a set of decision boundaries $\{z_i, 0 \le i \le N\}$, with $A = z_1 < z_2 < \cdots < z_N = B$; and a set of output points $\{y_i, 1 \le i \le N\}$. An input x is quantized according to the rule $Q(x) = y_i$ if $z_{i-1} \le x < z_i$. By definition of λ, the number of quantization levels N_{ab} in an interval $[a,b] \subset [A,B]$ is given by:

$$N_{ab} = N\int_a^b \lambda(x)dx \quad . \tag{11}$$

Using (11) and the fact that the number of quantization levels in the interval $[A, z_i]$ must be equal to i, we have:

$$\frac{i}{N} = \int_{A}^{z_i} \lambda(x)dx \ , \qquad 1 \le i \le N\text{-}1. \tag{12}$$

Eqn (12) gives us a simple rule to position the z_i's. Namely, the z_i's must be placed such that the areas under the function $\lambda(x)$ within the intervals $[z_{i-1}, z_i]$, $1 \le i \le N$, are all equal to $1/N$. Having fixed the decision boundaries, an obvious choice for the output points y_i are the centroids within each interval $[z_{i-1}, z_i)$. We may also design a quantizer according to its conditional distribution $p(x \mid B)$ within some region B by simply replacing $\lambda(x)$ by $\lambda_B(x)$ in (12), where $\lambda_B(x)$ is given by (4).

Note that with the aforementioned scheme, we are only attempting to equalize the area under $\lambda(x)$, and there is no control over the variation of the data within an interval. Figure 4a illustrates a potential problem with such a scheme. We have a wide interval $[z_0, z_1)$, and all input values in this interval are mapped onto the centroid y_1. The distance between input values in the small peak on the far left and the output value is thus very large. In an image, such an effect manifests itself as isolated spots with visually objectionable color errors. We correct for this in the following manner. After quantizing along a coordinate according to $\lambda(x)$, we identify each region where there is a large variation between input and output value, and perform a split through the midpoint of the data in that region, as shown in Fig. 4b. This corrective operation effectively removes the spot-like artifacts. Note that each splitting operation will increase the total number of quantization regions by one. If the quantization is along the last component Y, the splitting may result in the desired palette size N being exceeded. To avoid this situation, we design a palette of size $N' < N$, and then perform $N\text{-}N'$ splits in the most "problematic" regions. In our implementation we have chosen $N' = 0.95N$.

4.3.2 Optimal Allocation of Quantization Levels

The optimum choices for N_1 and N_2 given in (7) are based on the assumption that the MSE is well modeled by (6a) and (6b). This approximation, which is indeed valid in the asymptotic case, need not hold true when N_1, N_2, and $N_3 = N$ are relatively small. However, interestingly enough, we have observed that in practice, the basic functional form in (6a) is a good model for many natural images. It is only the constants α, β, and η that are not always well predicted by the input statistics in (6b) when N becomes small. In other words, if we were to plot the experimental MSE incurred by SSQ as a function of N_1 and N_2 for a given color image, this curve is reasonably well modeled by (6a) for some values of α, β, and η which may be very different from those obtained using the formulas (6b). We therefore need to somehow estimate these constants so as to obtain a reliable prediction for the MSE curve. We do this by performing a preliminary quantization on the image for some initial choice of N_1 and N_2. We follow steps 1 - 4 outlined in Sec. 2.2.2 using the scalar quantization procedure described in the preceding subsection and (8) to obtain the n_{ij}'s. We then compute the experimental MSE's d_1, d_2, and d_3 along the three coordinates:

$$d_i = \frac{1}{3}\sum_{l=1}^{N}\sum_{\boldsymbol{x}_j \in S_l}\left[x_{ij} - y_{il}\right]^2 \;\; ; \;\; i = 1, 2, 3, \tag{13}$$

where $\boldsymbol{x}_j = [x_{1j}, x_{2j}, x_{3j}]^t$ is the j-th input color, and S_l is the l-th quantization region with output color $\boldsymbol{y}_l = [y_{1l}, y_{2l}, y_{3l}]^t$. The i-th experimental MSE term in (13) is equated with the corresponding i-th theoretical MSE term in (6a) to yield empirical values α_e, β_e, η_e

$$\alpha_e = N_1^2 d_1 \; ; \quad \beta_e = \frac{N_2^2}{N_1^2} d_2 \; ; \qquad \eta_e = \frac{N^2}{N_2^2} d_3 \; . \tag{14}$$

Using these empirically derived constants in (6a), we obtain a reliable prediction of the MSE as a function of N_1 and N_2. It follows that the optimal N_1 and N_2 are given by (7) with α, β, and η being replaced by α_e, β_e, and η_e. We now perform the final N point quantization on the image by repeating steps 1 - 4 with these optimum allocations.

Figure 5 illustrates this process graphically for a 2-D example with chrominance data. In this case, $N_2 = 16$ is fixed, and we desire the optimum N_1 that minimizes the 2-D MSE $D_2 = d_1 + d_2$. The experimental MSE incurred for each N_1 is shown as the solid curve in Fig. 5. We

pick an initial value of $N_1 = 5$. The process described above is equivalent to finding a theoretical curve of the form (6a) (with $\eta = 0$) that coincides with the experimental curve at $N_1 = 5$. This theoretical curve is shown as the dashed plot in Fig. 5. The value of N_1 that minimizes the theoretical curve is then chosen as the predicted optimum allocation. In this case, the predicted minimum is at $N_1 = 4$, which also happens to be the location of the true experimental minimum.

4.3.3 Order of Quantization

Asymptotic theory tells us that for some useful cases such as a Gaussian input distribution, the performance of SSQ does not depend on the order in which the scalar components are quantized.[11] However, for our practical application, we do find that the order can have an effect on image quality. As was mentioned previously, the only means of finding the optimal order would be to perform the preliminary quantization described above for each of the 3! = 6 possible orderings and pick that which results in the minimum 3-D MSE. However, the purpose of deriving a computationally efficient quantization algorithm would be defeated if we have to execute it 6 times on one image! For our application, we observe that only 4 out of the 6 choices make intuitive sense: either we quantize the two chrominance components, followed by luminance; or vice versa. We pick the former option, as this roughly conforms to the intuitively appealing idea of assigning hues to the various objects in an image through chrominance quantization, and then providing the necessary shading to each hue patch or object through luminance quantization. This limits us to two possible orders: C_r-C_b-Y or C_b-C_r-Y. For each order, we do the following: (1) perform the preliminary quantization described in the preceding subsection using some initial N_1 and N_2; (2) estimate α_e, β_e, and η_e according to Eqn. (14); (3) determine the optimal N_1 and N_2 from Eqn. (7); and (4) use these optimal allocations in (6a) to predict the minimum MSE for that order. We pick the order of quantization that yields the smaller predicted minimum MSE and perform sequential quantization as described by steps (1) - (4) in Sec. 2.2.2.

4.3.4 Other Implementation Issues

a) Integer Representation of n_{ij} . The formula (8) yields real-valued n_{ij} which satisfy the constraint (2b). In practice, of course, these quantities must be integers since they represent numbers of quantization levels. For each j = 1,..., N_{i-1}, we replace n_{ij} in (8) by its nearest integer. In doing so, however, the constraint (2b) may be violated. Suppose the sum of the n_{ij} is greater than N_i. We must decrease the value of one or more of the n_{ij} until (2b) is satisfied. Since a reduction in the number of quantization levels n_{ij} in region B_{ij} will increase the MSE along x_i in that region, we must alter the n_{ij} in a manner such that the overall error increases minimally. We use the asymptotic theory[11] to predict the MSE e_{ij} in B_{ij}

$$e_{ij} = \frac{1}{n_{ij}^2} \left\| p(x_i \mid B_{ij}) \right\|_{1/3} \quad . \tag{15}$$

We must multiply e_{ij} by $P(B_{ij})$ in order to obtain the contribution of the error in B_{ij} to the overall MSE along x_i . The increase in MSE or the cost δ_{ij} resulting from decrementing n_{ij} by 1 is then given by

$$\delta_{ij} = \left[\frac{1}{(n_{ij} - 1)^2} - \frac{1}{n_{ij}^{\ 2}} \right] \left\| p(x_i \mid B_{ij}) \right\|_{1/3} p(B_{ij}) \ . \tag{16}$$

Starting with the region with the smallest δ_{ij} and proceeding through the regions B_{ij} in order of increasing cost, we decrement n_{ij} until (2b) is satisfied. (Regions with n_{ij} = 1 are left unaffected.) Conversely, suppose the sum of the integers n_{ij} is smaller than N_i. Using an analogous argument, we obtain the decrease in MSE or the gain δ_{ij} resulting from incrementing n_{ij} by 1 as

$$\delta_{ij} = \left[\frac{1}{n_{ij}^{\ 2}} - \frac{1}{(n_{ij} + 1)^2} \right] \left\| p(x_i \mid B_{ij}) \right\|_{1/3} p(B_{ij}) \ . \tag{17}$$

We now proceed through the regions in order of decreasing δ_{ij} and increment each n_{ij} by 1 until (2b) is satisfied.

b) Initial Choices of N_1 *and* N_2 *for Preliminary Quantization.* These were based purely on empirical observation. For several test images, we noted the values of N_1 and N_2 that minimized the experimental MSE. The average of these values were taken to be initial choices for the preliminary quantization. For N = 256, these initial values were N_1 = 18 and N_2 = 30.

c) Preliminary Quantization in Chrominance. Instead of performing preliminary quantization along all 3 coordinates, as was described in Sec. 4.3.2, in our implementation, we initially quantize only the two chrominance coordinates to obtain d_1 and d_2. We then use asymptotic theory to predict the MSE d_3 along Y without actually performing the quantization along this coordinate. The rationale for doing this is that from our observations, the theory tends to predict the error along luminance far better than it does the error along chrominance. This is largely because the number of quantization levels along Y tends to be relatively large, thus more closely satisfying the asymptotic assumption. The overall MSE along Y can be estimated by

$$d_3 = \frac{1}{3}\sum_{j=1}^{N_2} e_{3j} P(B_{3j}) \ , \tag{18}$$

where e_{3j} is as given in (15). The advantage with this strategy is that (18) involves less computation than it would to actually quantize along luminance and empirically determine d_3.

4.4 Incorporation of Human Visual System

So far, we have described an algorithm that attempts to minimize an objective error metric. Although such a metric provides us with a basis for palette design that is mathematically tractable, it often does not accurately reflect the perceived error in an image. In this section, we describe two ways to incorporate the human visual system (HVS) into the palette design.

4.4.1 Visually Weighted MSE

In general, the HVS is more sensitive to errors in luminance than in chrominance. This is particularly true in color quantization, where the most visible artifact is usually contouring in luminance. Since we are performing the quantization in a luminance-chrominance space, we may weight the MSE in a simple way to account for this property. Having elected to quantize the luminance component last, let us assume for this discussion that $\boldsymbol{X} = [X_1, X_2, X_3]^t = [C_r, C_b, Y]^t$. Consider the weighted squared error metric

$$D_3^w = d_1 + d_2 + \mathrm{K} d_3, \tag{19}$$

where as before d_i is the MSE along the i-th coordinate, i = 1,2,3, and K ≥ 1. We are effectively making the squared error along x_3 (luminance) K times more significant than the squared error along x_1 and x_2 (chrominance). The optimal quantization allocations N_1 and N_2 for the weighted MSE may easily be shown to be[11]

$$N_1 = \frac{N^{1/3}}{K^{1/6}}\left(\frac{\alpha^2}{\beta\eta}\right)^{1/6} \quad ; \qquad N_2 = \frac{N^{2/3}}{K^{1/3}}\left(\frac{\alpha\beta}{\eta^2}\right)^{1/6} \quad . \tag{20}$$

Note that the greater the value of the luminance weight K, the smaller the allocations N_1 and N_2 of quantization levels to the chrominance components. In our experiments, a value of K = 4 has been seen to reduce objectionable luminance contouring artifacts in several images. Sometimes, this improvement is at the expense of slightly increased chrominance errors, which are generally less objectionable.

4.4.1 Spatial Activity Measure

It is well known that the HVS is more sensitive to quantization errors in smooth rather than in busy regions of the image. This phenomenon, sometimes known as spatial masking, has been incorporated into palette design.[6, 9, 10] We employ a spatial activity measure similar to one used earlier.[10] We have found that such a measure often improves the subjective quality when the palette size is very small ($N \leq 128$). Define a gradient measure $|\nabla_{mn}| = |Y_{m,n} - Y_{m,n-1}| + |Y_{m,n} - Y_{m-1,n}|$ that reflects the variation of luminance Y at each pixel location (m,n). In order to reduce the effect of noise on the gradient measure, we divide the image into 8 × 8 blocks, and for each block l, we compute the average α_l of gradients $|\nabla_{mn}|$ over all pixels in l. We then assign a block weighting function ω_l that reflects the subjective visibility of quantization errors in that block. From our discussion above, we would like this function to be inversely proportional to the activity α_l in the block. Hence, we choose $\omega_l = 1/\alpha_l$. (In the actual implementation, we limit the dynamic range of the denominator.[10]) Each image color $\boldsymbol{x}$ is then assigned a subjective weight $\omega_{\boldsymbol{x}}$ to be the average of block weights over all blocks in which that color appears.[10] Note that if this color appears mainly in smooth spatial regions of the image, it will have a large subjective weight associated with it.

Since the weighting is based purely on luminance variation, we incorporate it into the SSQ palette design at the luminance quantization step. When allocating the number of quantization levels n_{3j} along luminance within each region B_{3j} according to (8), we simply replace r_j by $r_j' = \omega_j' r_j$ where ω_j' is the average of the subjective weights $\omega_{\boldsymbol{x}}$ of all colors $\boldsymbol{x}$ in B_{3j}. If a particular region B_{3j} contains colors from mainly smooth spatial areas in the image, it will have a larger weight ω_j' and therefore a larger quantization allocation n_{3j} associated with it. Hence, we achieve the desired effect of assigning more quantization levels to colors from smoothly varying image regions.

4.5 Pixel Mapping

Having designed a palette, the remaining step is to map each input pixel in the image to a color in the palette. For a given palette, the optimal mapping is the palette color that is closest to the input color in the color space. In general, finding the closest palette color is a computationally intensive operation, as it involves an exhaustive search through the palette for each input color. Techniques such as binary splitting provide a structure to the palette that enables a fast, albeit suboptimal, mapping between input and palette colors. Since any splitting technique breaks up the color space into N quantization regions, it is sufficient to locate the quantization region to which a given input color belongs, and map the input color to the centroid of that region. Note that this is not necessarily the optimal mapping, as shown by the example in Fig. 1b. The vector $\mathbf{C}$ would be mapped onto the output vector $\mathbf{Q_1}$, although the output vector that is closest to it is actually $\mathbf{Q_2}$. However, this suboptimal strategy is satisfactory for most applications. The binary splitting algorithm is implemented with a binary tree structure, so that a search for a quantization cell can be performed in logarithmic time (an average of $\log_2 N$ searches are needed to locate a quantization cell for a given input color[9]).

In the SSQ technique, each input color is mapped to a palette color very efficiently through the use of the three stages of LUT's shown in Fig. 2b. This offers a significant

advantage over any of the other quantization algorithms that have been reported, with the exception of independent product codes which result in inferior image quality. The LUT's are actually built during the palette design. The i-th stage of LUT's, i = 1,2,3, is generated immediately after quantization along the i-th coordinate x_i. Pixel mapping proceeds as follows. Referring to Fig. 2b, at the first stage, the X_1 value of the input color is passed to the X_1 LUT which locates the quantization region to which the color belongs from the first N_1 point quantization along X_1. Each of the N_1 regions contains an X_2 LUT which uses the X_2 value of the input pixel to locate the quantization region to which the color belongs from the second set of quantizations along X_2. Finally, each of the N_2 quantization cells in the X_1-X_2 plane contains an X_3 LUT that uses the X_3 value of the input pixel to locate an index for the quantization region formed from the final set of quantizations along X_3. Hence, the entire mapping operation consists only of indexing into LUT's, and involves no computation. Once we have obtained an index m for the input pixel, it is a simple matter to decode it to the m-th palette color using a colormap. In practice, the LUT's may easily be implemented with low cost hardware, thus allowing the pixel mapping operation to be performed in real time. Figure 6 summarizes the entire SSQ algorithm in a block diagram.

4.6 Complexity Considerations

4.6.1. Time Complexity

Let N_p be the number of pixels in the image, $N_s \leq N_p$ be the number of pixels from the subsampled image, N_c be the number of distinct colors in the image, and N be the palette size. We are assuming that the images have already been transformed to the YC_rC_b color space, and do not include that computation here. The preprocessing (*i.e.* histogram generation) involves $O(N_s)$ operations, each operation entailing a few additions and comparisons.[6] If the spatial activity measure is included, the computation of ω_1 is of order $O(N_p)$. The sequential quantization involves two passes through the histogram for each coordinate: one pass during the preliminary quantization to obtain the optimum allocations N_1 and N_2, and one pass during the final quantization. This amounts to 6 passes through the histogram, which is of size N_c. Hence,

the sequential quantization involves O($6N_c$) operations. The dominant computation here is in the raising of each marginal and conditional distribution to the 1/3rd power to obtain the qdf λ. Finally, the pixel mapping is of order O(N_p), where the operations only entail indexing through LUT's. Table 1 summarizes the computational requirements of the various steps of the algorithm.

4.6.2. Space Complexity

The 2-D histogram array requires just over 32 Kbytes of memory, while the size of the linked list that keeps all the distinct colors is image dependent and is typically on the order of 600 Kbytes (in comparison with 16 Mbytes required for a full 3-D array). The memory requirement for the LUT's depends on the relative quantization allocations N_1, and N_2. We need 1 LUT for the first stage of Fig. 2b, and N_1 and N_2 LUT's for the second and third stages respectively. Each individual LUT has 256 entries, and therefore requires 256 bytes. The overall memory needed for the multistage LUT is $256(1+N_1+N_2)$ bytes. Taking a fairly typical example, if $N_1 = 5$ and $N_2 = 20$, the LUT's require approximately 6.6 Kbytes of storage. Clearly the histogram data structure demands by far the greatest storage.

5. EXPERIMENTAL RESULTS

All algorithms were run on a SUN SPARCstation 2, and the images were displayed on a RasterOps Sony Trinitron video monitor. The calibration procedure described in Sec. 4.1 was used to transform original and quantized RGB images to the RGB space of the monitor. All images contain 512×512 pixels, and were subsampled by a factor of 2 in each direction for the palette design. We incorporated the spatial activity measure only for palettes whose size was 128 or less.

5.1 Qualitative Performance

In this section, unless otherwise mentioned, a luminance-chrominance weighting of K = 4 was used for palette design. Figures 7 (a)-(d) compare four original images with images

quantized to 256 colors using the SSQ algorithm. The image "lenna" consists of many smooth variations of colors from a small region of the color space, while "peppers" contains smooth variations of highly saturated colors from very different parts of the color gamut. The third image "picnic" consists of a mixture of busy regions, texture from the grass, and smooth variations in the sky. The final image "balloon" is a difficult image to quantize transparently, as it contains many different hues; and for each hue, there are many fine gradations in luminance.

Figure 8 compares results of the median cut algorithm[1] and the fast binary splitting algorithm,[10] referred to as binary splitting with prequantization (PQBS). We choose these two algorithms for comparison with our technique because the former is a widely used standard algorithm; and to our knowledge, the latter offers the best performance in terms of image quality and computational complexity among algorithms that have been previously reported in the literature. Looking at Fig. 7(d) and Fig. 8, we see that the median cut algorithm performs poorly, while SSQ and PQBS perform comparably. Upon close inspection, it can be seen that SSQ with the luminance weighting (K = 4) has reduced much of the contouring artifacts in the balloons that remain visible even with the PQBS technique.

Figure 9 shows the effect of the corrective splitting operation described at the end of Sec. 4.3.1. For the purpose of illustration, a 2-D C_rC_b histogram of an image, shown in grayscale, is displayed along with the quantization regions formed by SSQ and the region centroids (shown by white dots). With the histogram on the left, we see that in the lowermost region, there is a large distance between the colors in the two clusters and the output centroid. The corrective scheme described in Sec. 4.3.1 detects this problem and splits this region into two regions along the horizontal (C_r) axis, as shown in the histogram on the right. The splitting plane passes through the mid point of the input values in that region. We then represent each of the two new regions with their respective centroids; and the error between input and output colors is greatly reduced.

Figure 10 shows the effect of the luminance-chrominance weight K. Four images are shown with increasing values of K. We see that for K = 0.5, there are not enough quantization levels allocated to luminance, and the resulting contouring is objectionable. On the other hand, for K = 8, there is an excessive allocation to luminance and an insufficient number of quantization levels along chrominance, which results in visible hue shifts in the face. Intermediate values of K = 1 and K = 4 yield better results. A close inspection of these two cases indicates that weighting luminance more than chrominance (*i.e.* K = 4) yields a visually more pleasing image with less contouring on the face and balloons. We have observed that K = 4 yields satisfactory results for a variety of images in our database. Figure 11 shows the effect of the spatial activity weighting on an image quantized to 128 colors. As is seen in the image on the right, the weighting allocates more colors to the smooth regions in the sky, hence reducing the contouring seen in the image on the left. We have observed that this weighting can produce a noticeable improvement only when the palette is very small (of size ≤ 128).

5.2 Quantitative Performance

We have designed an algorithm that strives to minimize the MSE. As noted earlier, however, this error metric is often a poor measure of visual quality. Hence, rather than presenting elaborate numerical error results that are visually irrelevant, we discuss briefly the general MSE performance of SSQ, PQBS, and the median cut algorithm. We compared the MSE's in both RGB and YC_rC_b color spaces for several images, and found that (a) the PQBS algorithm consistently yields a slightly lower MSE than SSQ, and (b) the median cut algorithm yields a much larger MSE than either PQBS or SSQ. It is not surprising that PQBS outperforms SSQ with respect to MSE, because from a VQ point of view, the former allows arbitrary shapes of quantization regions in 3-D space; whereas the latter is constrained to rectangular quantization regions. In contradiction to this, however, SSQ often yields better visual quality than PQBS because of the greater weight given to luminance variations. It is of no surprise that the median

cut algorithm performs poorly under the MSE criterion, as it is not explicitly designed to minimize this error measure.

5.3 Computational Performance

Table 1 shows a breakdown of the computation times involved in the various steps of SSQ. The palette design takes the most time, as it involves 6 passes through the histogram for the preliminary and final quantization, and these passes involve raising histogram values to fractional powers. Although the preprocessing and pixel mapping entail passes through the entire image, these steps involve simple operations such as additions and indexing, and as such, take up only a minor portion of the overall execution time. Table 2 compares the execution times of SSQ, PQBS, and the median cut algorithm for the four test images. As is seen here, SSQ is by far the fastest algorithm of the three. On average, SSQ requires only 21% of the time taken by PQBS and 24% of the time taken by the median cut algorithm.

6. CONCLUSIONS

We have proposed an efficient algorithm for color image quantization, wherein each scalar component of the color vector is quantized sequentially. Since quantization is performed on scalar variables, the palette is designed very efficiently. In addition, due to the sequential structure of the palette, pixel mapping is performed using LUT's, and thus involves no computation. We use asymptotic quantization theory to optimize the performance of SSQ with respect to MSE. A luminance-chrominance weighting and spatial activity measure are incorporated into the palette design in order to account for some basic properties of the HVS. SSQ achieves high visual quality, while incurring significantly fewer computations than existing quantization algorithms. Halftoning techniques such as dithering and error diffusion may be readily incorporated into this algorithm in order to increase the perceived number of colors. However, one must be careful in employing such techniques, since spatial resolution is sacrificed for tonal resolution, and visible texture artifacts may result.

7. ACKNOWLEDGMENTS

This work was partially supported by an Eastman Kodak Company Fellowship (R. B.) and an NEC Faculty Fellowship (C. A. B.). The authors wish to acknowledge helpful conversations with J. P. DiVincenzo and H. C. Lee, both of Eastman Kodak Company, on the subject of color image quantization. They also would like to thank H. C. Lee for the use of his median-cut algorithm software. Finally, the authors are grateful to Eastman Kodak Company for the use of their images "balloon" and "picnic" in the experiments.

8. REFERENCES

1. P. Heckbert, "Color image quantization for frame buffer display," *Computer Graphics*, vol. 16, no. 3, pp. 297-307, July 1982.

2. G. Braudaway, "A procedure for optimum choice of a small number of colors from a large color palette for color imaging," *Electronic Imaging '87*, San Francisco, CA, 1987.

3. R. S. Gentile, J. P. Allebach, and E. Walowit, "Quantization of color images based on uniform color spaces," *Journal of Imaging Technology*, vol. 16, no. 1, pp. 12-21, Feb. 1990.

4. Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun*., vol. COM-28, pp. 84-95, Jan. 1980.

5. Sudhir S. Dixit, "Quantization of color images for display/printing on limited color output devices," *Comput. & Graphics*, vol. 15, no. 4, pp. 561-567, 1991.

6. R. Balasubramanian and J. P. Allebach, "A new approach to palette selection for color images," *Journal of Imaging Technology*, vol. 17, no. 6, pp. 284-290, Dec. 1991.

7. W. H. Equitz, "A new vector quantization clustering algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, no. 10, pp. 1568-1575, Oct. 1989.

8. S. J. Wan, P. Prusinkiewicz, and S. K. M. Wong, "Variance-based color image quantization for Frame Buffer Display," *COLOR Research and Applications*, vol. 15, no. 1, pp. 52-58, Feb. 1990.

9. M. T. Orchard and C. A. Bouman , "Color quantization of images," *IEEE Trans. Signal Processing*, vol. 39, no. 12, pp. 2677-2690, Dec. 1991.

10. R. Balasubramanian, C. A. Bouman, and J. P. Allebach, "Color image quantization using a fast binary splitting technique," to be submitted to *Journal. Optical Soc. of America - A.*

11. R. Balasubramanian, C. A. Bouman, and J. P. Allebach, "Sequential scalar quantization of vectors: theory and applications," to be submitted to *IEEE Trans. Image Processing*.

12. A. Gersho, R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, 1991.

13. J. Makhoul, S. Roucos, and H. Gish, "Vector quantization in speech coding," *Proc. IEEE*, 73, 1551-1588 (1985).

14. James M. Kasson, and Wil Plouffe, "An analysis of selected computer interchange color spaces," submitted to *Computer Graphics*.

Table 1 Breakdown of computational complexity and execution times for various steps in the SSQ algorithm applied to the image "lenna".

Operation	Computational Complexity	ExecutionTime†
3-D Histogram	$O(N_s)$††	0.4 sec
Sequential Palette Design	$O(6N_c)$*	1.2
Pixel Mapping via Multistage LUT	$O(N_p)$ **	0.3

†Time taken to quantize a 512 × 512 image to 256 colors on a Sun SPARCstation 2

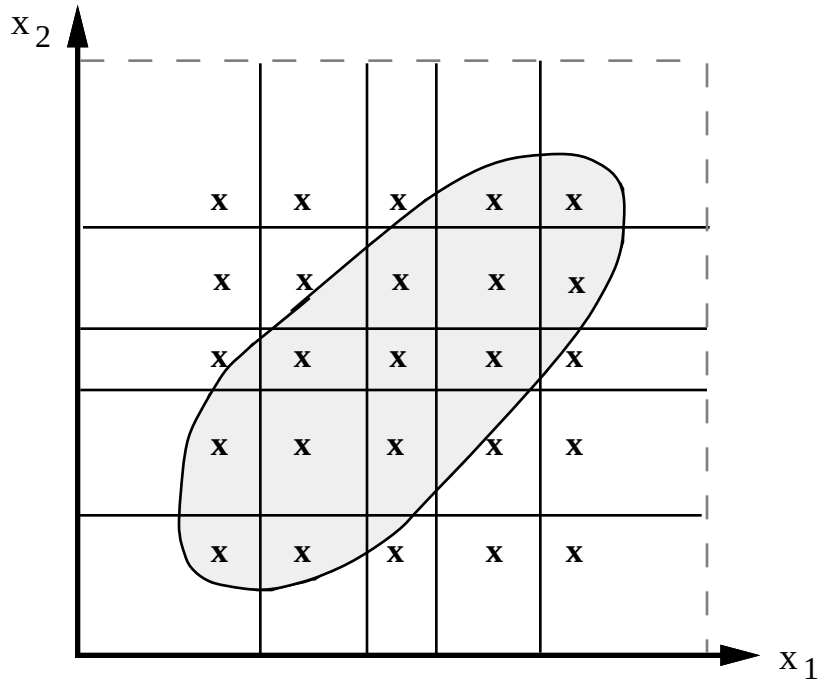
††Number of pixels in subsampled image

*Number of distinct image colors

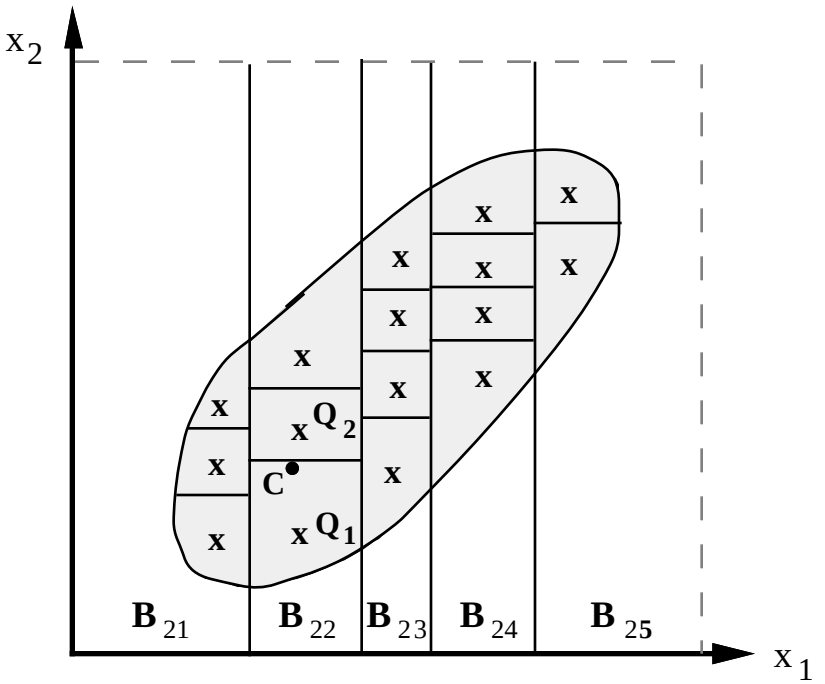
**Number of pixels in original image

Table 2 Comparison of execution times for 256 color quantization using sequential scalar quantization (SSQ), binary splitting with prequantization (PQBS), and the median cut algorithm.

Image	ExecutionTime (sec)		
	SSQ	PQBS	Median Cut
Lenna	1.9	9.3	7.0
Peppers	2.4	12.6	9.8
Balloon	1.9	9.1	7.3
Picnic	2.0	10.0	11.8
Average	2.05	10.25	8.98

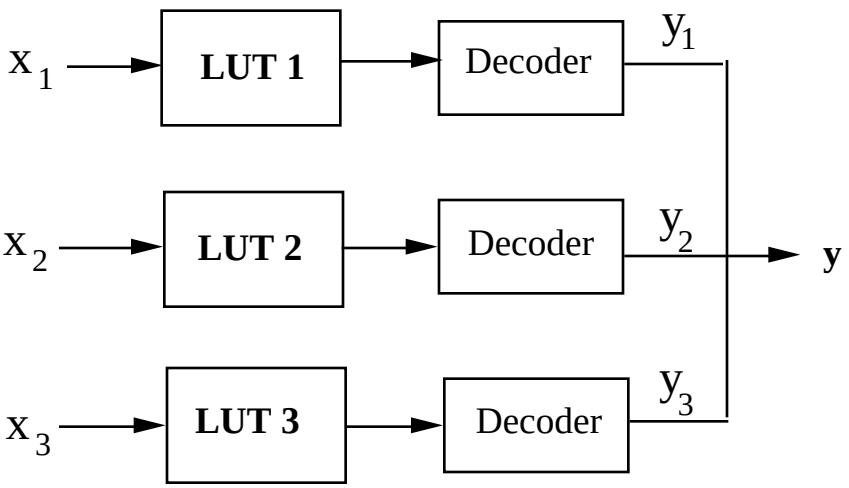


(a)

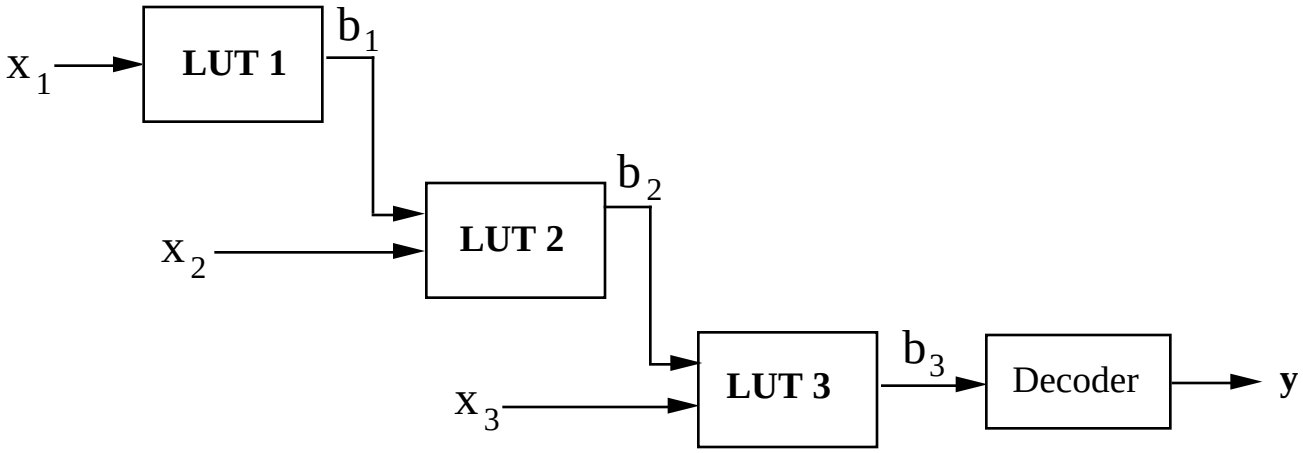


(b)

Fig. 1 2-D example of (a) independent scalar quantization for N = 25; (b) sequential scalar quantization for N = 16.



(a)



(b)

Fig. 2 Encoder-decoder operation for (a) independent and (b) sequential scalar quantization.

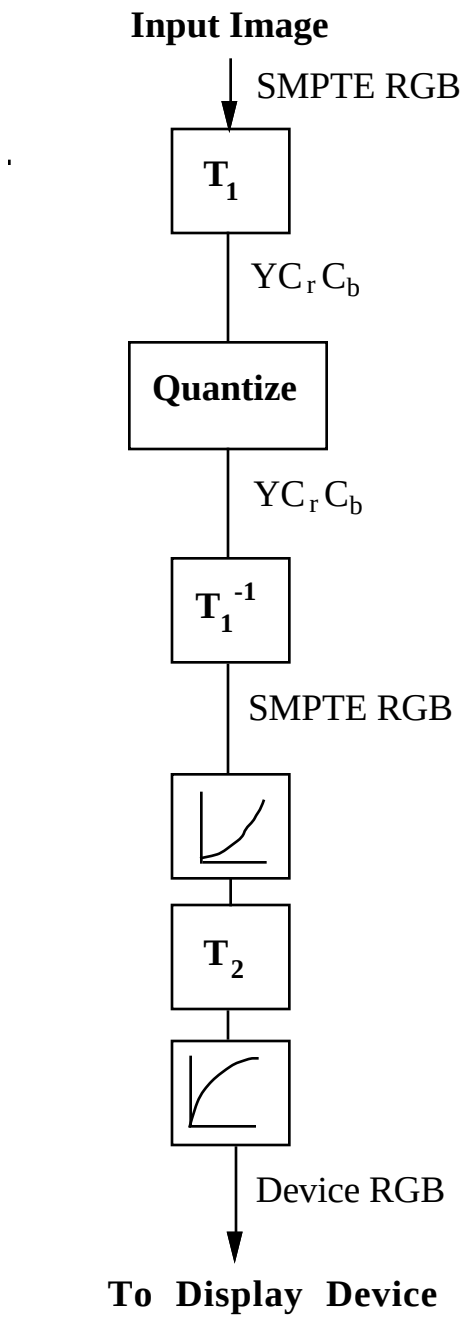


Fig. 3 Steps involved in quantization and display of a color image.

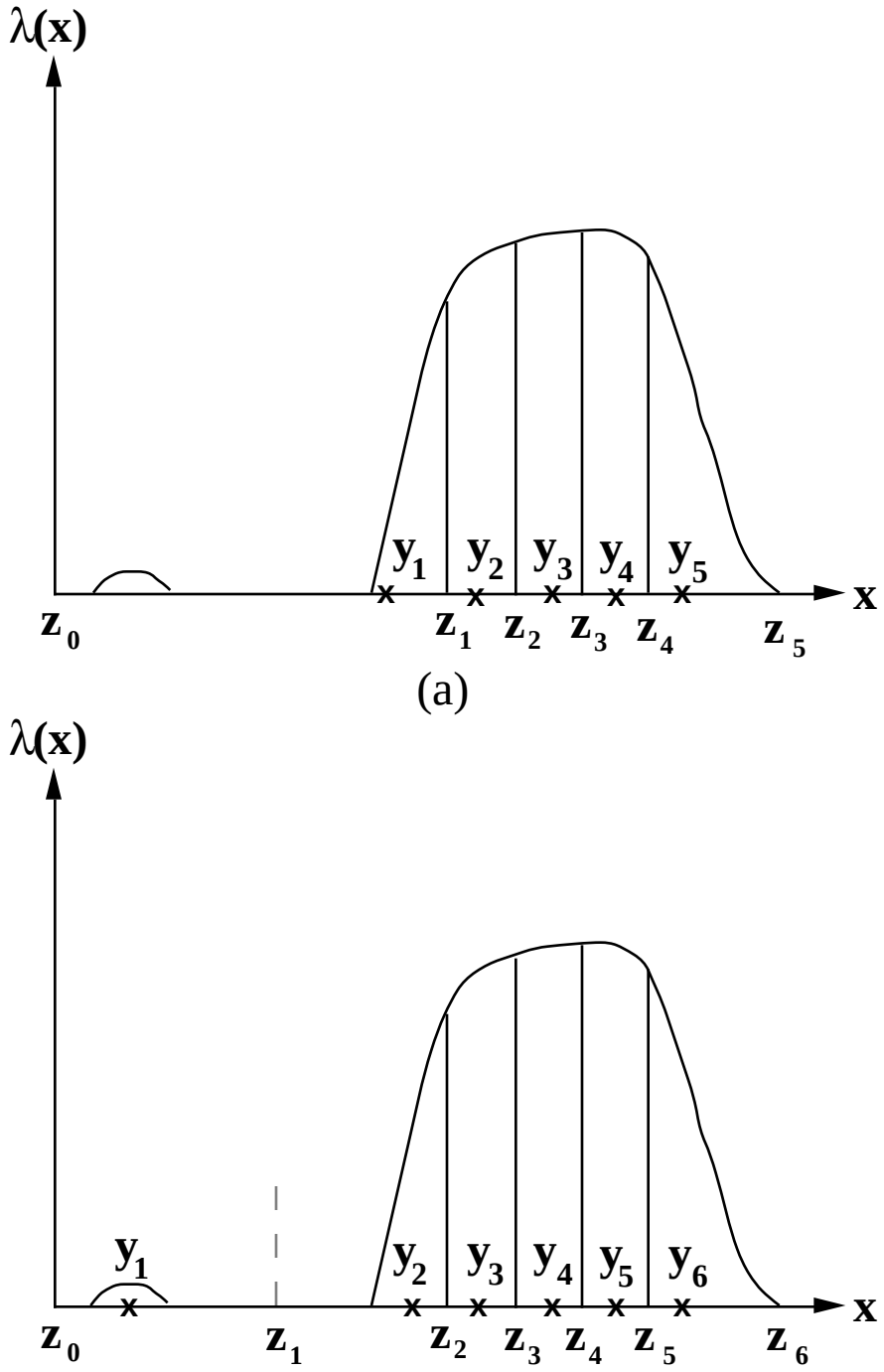


Fig. 4 Example to show quantization of a scalar x according to $\lambda(x)$ (a) prior to corrective splitting and (b) after corrective splitting.

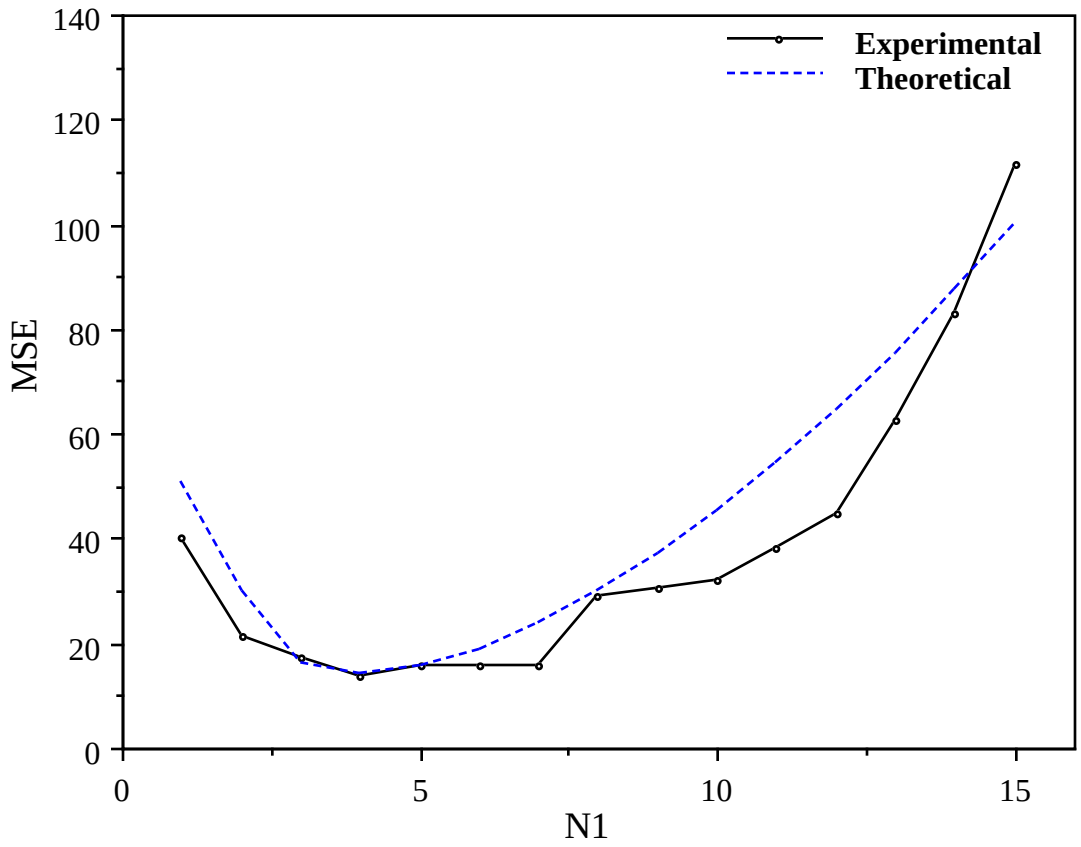


Fig. 5 Plot showing theoretical fit to an experimental curve of MSE vs N_1 for image "balloon".

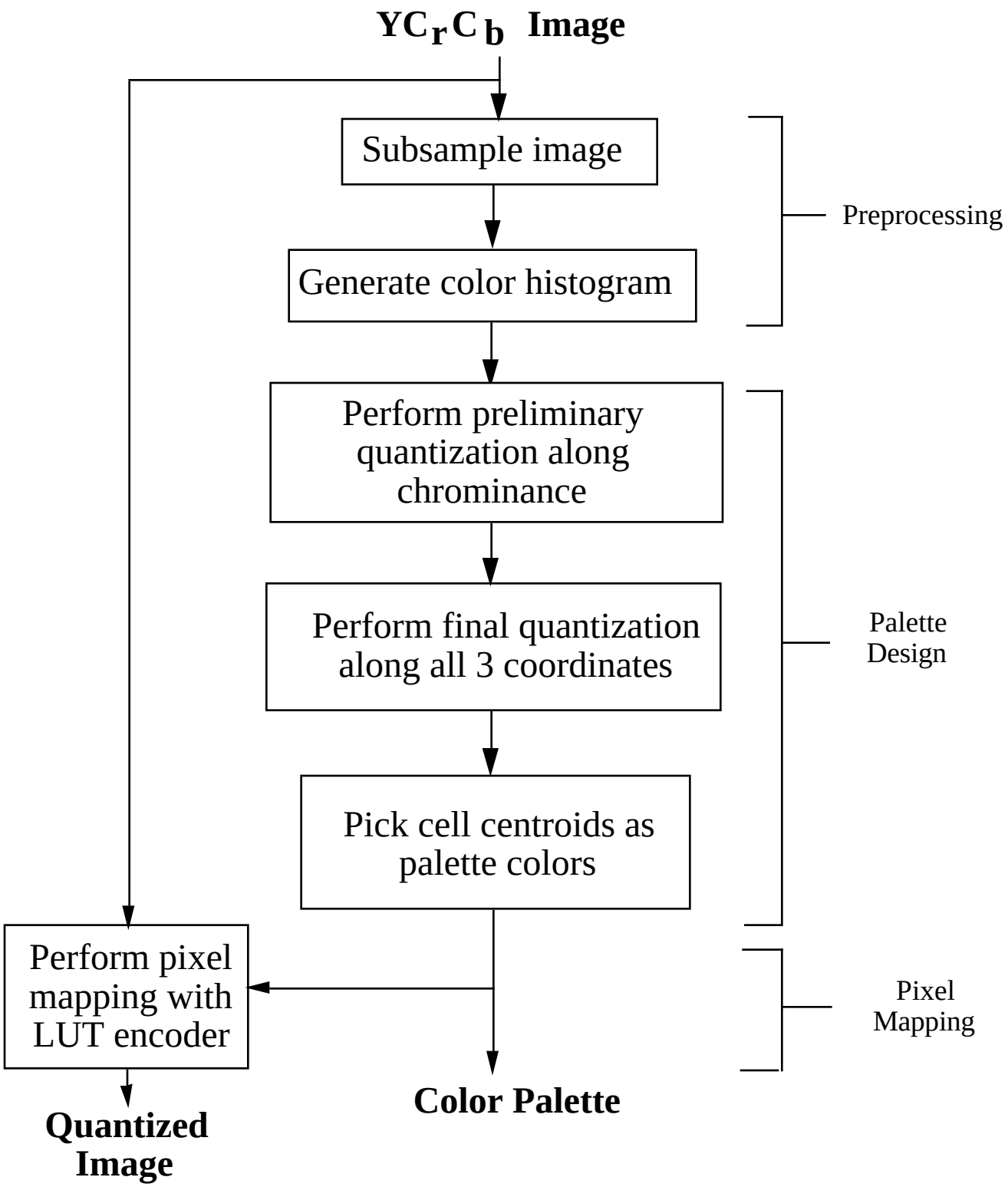


Fig. 6 Block diagram summarizing SSQ algorithm.

Fig. 7(a)

Fig. 7(b)

Fig. 7(c)

Fig. 7(d)

Fig. 7 Original image (left) and image quantized to 256 colors using SSQ (right): (a) Lenna; (b) Peppers; (c) Picnic; (d) Balloon.

Fig. 8 Image quantized to 256 colors using median cut (left) and PQBS (right).

Fig. 9 Example of a 2-D histogram of an image before (left) and after (right) corrective splitting.

Fig. 10 Effect of luminance-chrominance weighting. Top row: k = 0.5, 1; bottom row: k = 4, 8.

Fig. 11 Image quantized to 128 colors using SSQ without (left) and with (right) a spatial activity weighting.