COLOR IMAGE QUANTIZATION FOR FRAME BUFFER DISPLAY

by

Paul S. Heckbert

Submitted in Partial Fulfillment
of the Requirements for the
Degree of Bachelor of Science
at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1980

Signature of Author ______________________________
Department of Mathematics, 20 May 1980

Certified by ______________________________
Nicholas Negroponte, Thesis Supervisor
Professor of Computer Graphics

Certified by ______________________________
Gilbert Strang, Thesis Supervisor
Professor of Mathematics

Accepted by ______________________________
Chairperson, Departmental Committee on Theses

COLOR IMAGE QUANTIZATION FOR FRAME BUFFER DISPLAY

by

Paul S. Heckbert

Submitted to the Department of Mathematics on 20 May 1980 in partial fullfillment of the requirements for the degree of Bachelor of Science.

ABSTRACT

Algorithms for approximately optimal quantization of color images are discussed. The distortion measure used is the distance in RGB space. These algorithms are used to compute the color map for low-depth frame buffers in order to allow high-quality static images to be displayed. It is demonstrated that most color images can be very well displayed using only 256 or 512 colors. Thus frame buffers of only 8 or 9 bits can display images that normally require 15 bits or more per pixel.

Work reported herein was sponsored by the IBM Corporation though a general grant agreement to MIT dated July 1, 1979.

TABLE OF CONTENTS

Chapter I

INTRODUCTION

Most research in image quantization has been for the purpose of data compression in picture transmission systems. In these systems, quantization is frequently used in combination with other compression methods such as transform coding, DPCM, and coarse-sampling (see Netravali and Limb [35]). Until recently, most image processing work was on monochromatic images. The research described in this thesis addresses itself to the problem of color image display rather than image transmission or storage. To solve this problem, we borrow some monochromatic quantization methods which are discussed thoroughly in the literature.

The organization of this paper is as follows. First, some background into the architecture of frame buffers is given, and the computational problem of finding the optimal color map for an image is defined. An abstraction of the problem is derived in order to relate the work to previous research in monochromatic (1-dimensional) quantization. The history of that subject is discussed, and several algorithms for optimal and approximate quantization are given. Then we advance to 2 and 3-dimensional quantization, and discuss what research has been done in this area. Several new algorithms for approximate color quantization are given. These new algorithms represent the bulk of the research presented. Finally, improvements to the current quantization algorithms are considered.

Chapter II

FRAME BUFFERS AND COLORMAPS

Before describing the hardware that is used in color image processing, it is helpful to define what we mean by "color". For our purposes, we will define a color to be an ordered triple of color components in the Red-Green-Blue color space: (Cr, Cg, Cb). Each component of the color is an integer in the range [0,255] requiring 8 bits of memory. This is a common representation for colors in the image processing world. This is not the same as the video standard used by broadcast television in the United States. That standard, termed NTSC encoding, is described in [28].

FRAME BUFFERS

A frame buffer, or picture memory, is a computer memory organized into an m x n rectangular grid of dots, called picture elements, or pixels for short. Each pixel requires a certain number of bits, varying from 1 in bit-map displays to 24 or more in high-quality color displays. We call the number of bits per pixel the "depth" of the frame buffer.

A common size of frame buffers is m=n=512, at a depth of 8 bits: 512x512x8. A picture this size takes up 256 Kilobytes. When interfaced to a processor, it is possible to read and write into the frame buffer much like one reads and write into standard computer memories. The main difference is that this memory is scanned, usually 30 times per second, to generate a video picture which is displayed on a CRT monitor. A magnified view of the pixels in a color image is shown in figure 1. Further explanation of frame buffers, color displays, and other computer graphics hardware can be found in Newman and Sproull [36].

There are several different raster-scan frame buffer architectures commercially available at present. It is useful to distinguish between two classes of color frame buffers: let's call them "segregated" and "integrated". In segregated frame buffers, there are three independent memories for the red, green, and blue components of an image. Typically 8 bits are used per picture element (pixel). A color lookup table for each component is inserted between the picture memory and display device in order to allow contrast correction.

An integrated color frame buffer, on the other hand, stores a single color number at each pixel rather than three separate components. These color numbers (called pixel values in the remainder of the paper) are used as addresses into a single color lookup table. This "colormap" provides a level of indirection between the data in the picture memory and the actual displayed image. Also, it should be realized that the order of colors in a colormap is arbitrary.

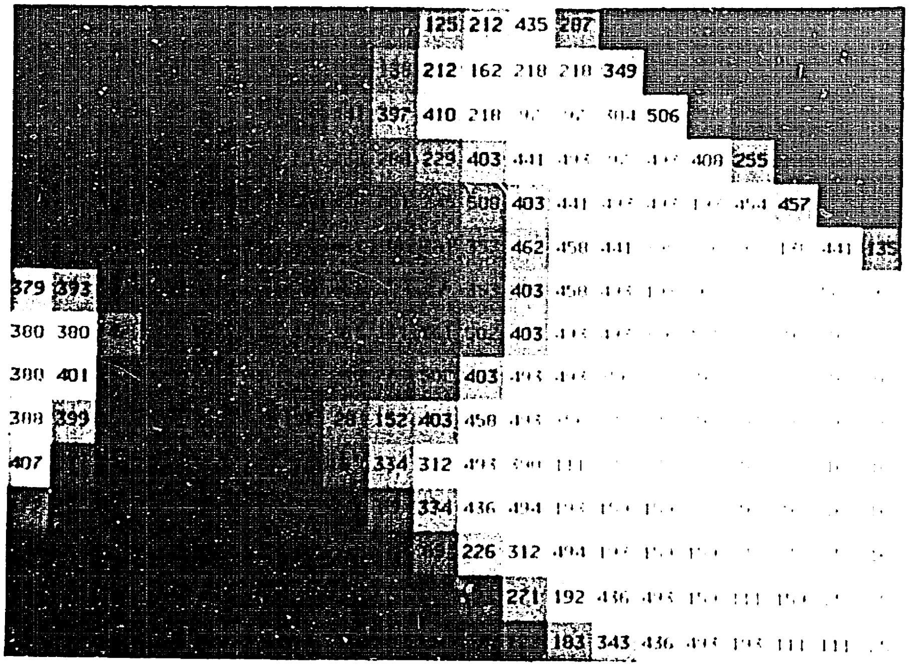


fig. 1, magnified view of pixels from a color image (pixel values are shown in blue)

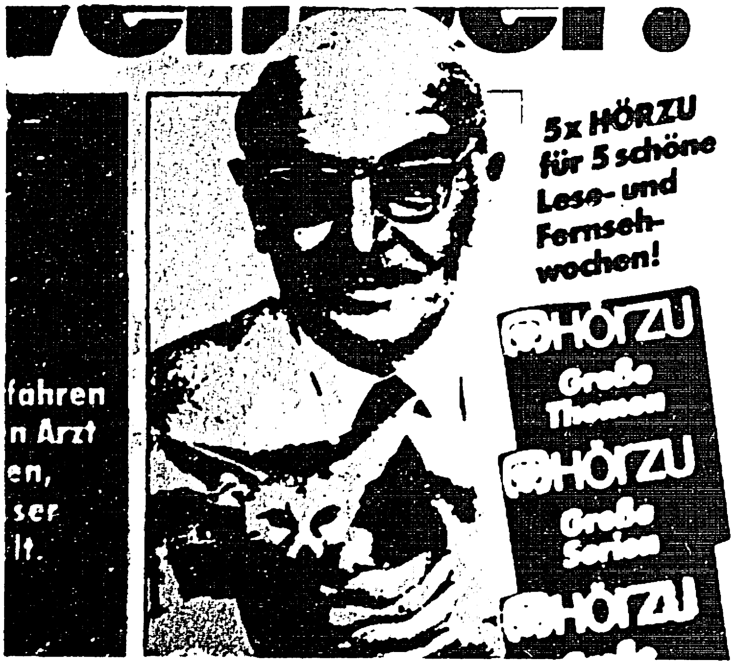


fig. 2, 2-bit quantized image showing contouring

With an integrated frame buffer of 3*n bits per pixel, one can simulate a segregated frame buffer having n bits per primary; an integrated frame buffer is more general than a segregated one. Unfortunately, the memory required for the colormap of a deep integrated frame buffer gets to be prohibitive. The colormap of a 24-bit integrated frame buffer, if one were to be built, would require at least 48 Megabytes of memory. This would make the colormap larger than the picture memory, which would require approximately 0.8 Megabytes. For this reason, one rarely finds integrated frame buffers with more than 8 or 9 bits per pixel.

Currently, segregated frame buffers are much more common than integrated systems in the image processing world, probably because of their simplicity and the higher quality of images. This may change, however. At this moment, shallow, low-resolution integrated graphic terminals such as the Apple are making their way into the home computer market. They may become much more common than the huge frame buffers one finds in universities. If this were to happen, the demand for color image quantization programs would certainly increase.

Most people feel that 6 to 8 bits per component are required for high quality grayscale images. With fewer than 100 quantization levels, the eye is distracted by the artificial edges called "contours." Figure 2 shows a 2-bit quantized image.

Note that the colormap also allows some redundancy. For example, there are many ways of displaying a totally black image. One could black out the entire colormap and pepper the picture memory with random pixel values, or one could zero the picture memory and only set slot 0 of the color map to be black (R=G=B=0).

CURRENT HARDWARE:

The algorithms outlined in chapter IV were implemented for the Ramtek 9300 Raster Display Frame Buffers at the Architecture Machine Group (AMG) at MIT. These are integrated 9-bit systems with 24-bit colors in the colormap. This implies that the colormap (also called a color matrix) is a 9x24 table: 9 bits in, 24 bits out. Colormaps for the Ramtek require 2K of memory. The resolution of these picture memories is m=640 pixels horizontally by n=480 lines vertically, or a total of N=m x n= 307,200 pixels.

Color image separations come from several sources: photographs digitized by vidicon through red, green, and blue filters, digitized video frames, and images from other labs recorded on magnetic tape. The quantization programs described later take the three 8-bit separations and quantize them down to 8 or 9.

OPTIMIZING THE COLORMAP

When an image is quantized, each of the 3-dimensional colors must be encoded into a single pixel value. At the AMG, there are 307,200 24-bit colors in the original image, each of which must be transformed into a pixel value. This is the quantizing step, which will be done once (in software) before saving the image on disk or tape. When the image is loaded into the frame buffer, the display processor in the frame buffer does the decoding for us through the colormap. This transforms our color numbers (pixel values) back into a 3-dimensional color video signal.

We wish the quantized, displayed image to approximate as closely as possible the 24-bit original. Although we cannot (at the AMG) display the 24-bit original for subjective comparison with the quantized image, the "quality" of the approximation can be measured with distortion formulas. This distortion formula is a measure of the similarity of the quantized image with the original. The formula we will use is:

$$D = \sum \left(\begin{array}{l} \text{distances between corresponding pixels} \\ \text{in original and quantized images} \end{array} \right)$$

$$\text{distance}^2(CO, CQ) = \Delta r^2 + \Delta g^2 + \Delta b^2 \quad \text{where} \quad \begin{cases} \Delta r = CQ_r - CO_r \\ \Delta g = CQ_g - CO_g \\ \Delta b = CQ_b - CO_b \end{cases}$$

This is simply the distance squared between the colors of the two pixels in RGB space.

We try to minimize D by varying the colormap. In other words, we want to approximate a 24-bit-per-pixel image with an 8 or 9-bit one by minimizing the sum of the distances between corresponding pixels in the original and quantized images. This is a difficult optimization problem with no fast solution.

One dimensional (monochromatic) versions of the problem are tractable; they are discussed in the next section. In fact, there are efficient algorithms for the fast solution of the optimal quantization problem for black and white images. The problem is far from solved, however, in the color domain. As we will see, the complex topology and functional interdependence of the color cells makes the 3-d quantization problem much more difficult. Therefore, we must be content with approximations to the optimal. Several algorithms for this purpose have been investigated.

ARE WE USING THE PROPER DISTORTION MEASURE?

Image quality and image similarity is a very difficult physiological and psychological problem. We will not attempt any new discoveries in this area. Instead, we will use the simple distance formula in order to speed the quantization algorithms. A complete evaluation of image similarity would include such complex topics as texture, continuity, contrast, spatial frequency, color balance, and activity. Fortunately, one can achieve relatively good results by ignoring most of these effects and comparing only corresponding pixels in the original and quantized images. That is the approach used here.

C. F. Hall [19] and others have achieved excellent color image compression though the use of perceptual transform coding. By transforming into a perceptual space, redundant spectral and spatial data are compressed. Hall achieved good results even at 1 bit per pixel. Unfortunately, these results are not applicable to our frame buffer quantization problem, because state-of-the-art frame buffers cannot yet compute spatial transforms in real time. The best one can do within the constraints of integrated frame buffer architecture is to choose the color quantization cells based on a perceptual distortion model of color differences; no spatial effects are relevant.

What sort of color distortions can we expect when we compress images? When highly continuous and colorful images are quantized, it is not possible to match every color exactly, so we anticipate contouring effects. The contouring effect can be alleviated by the addition of color noise. Experiments with noise addition are presented in chapter IV.

We can now proceed to reformulate our image processing problem into mathematical terms. Let K be the number of colors to which we are quantizing. One could use brute force: try all $\binom{2^{24}}{K}$

possible colormaps, and choose the one with the least distortion, but this is a horrendous computation which would require eons of computer time. The natural approach is to derive our colormap from the color distribution of the image being quantized.

Some graphical representations of colormaps will be helpful. See figure 3 for a cross-sectional view of the color quantization cells of a colormap containing 16 colors. Figure 4 shows the distribution of colors for an image. It is seen that the red, green, and blue components are highly correlated in this image. Most natural scenes contain a majority of unsaturated colors - colors close to the gray intensity axis.

The distribution of colors in an image is analogous to a distribution of N points in the RGB color cube. The color cube is a discrete lattice because the colors of our original image have integer components between 0 and 255. Thus there are 256x256x256 colors in the lattice, or a total of 16 million colors. Although the number of possible colors is many times greater than the number of pixels in an image, it is highly likely that there will be two pixels with the same 24-bit color.

It is understood that we intend to generate a different colormap for each image depending on its color range. In one sentence, this is two-pass, adaptive intraframe tapered quantization.

The algorithms for color quantization described in chapter IV

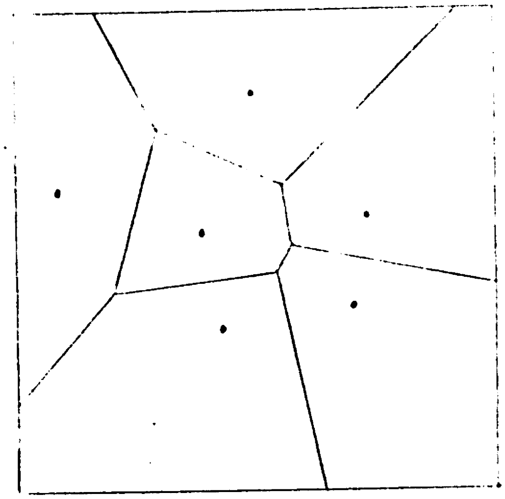


fig. 3, 2-dimensional quantization cells (the dots are the reconstruction levels)

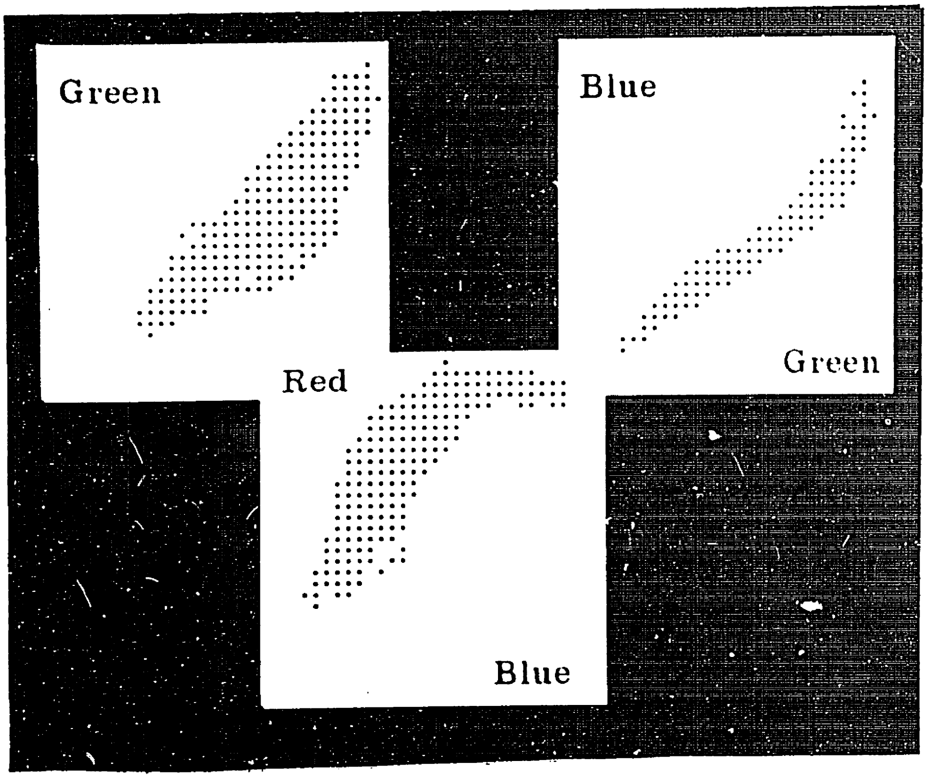


fig. 4, projections of the distribution of colors in an image will use the following four phases:

1. sample image to determine color distribution
2. select colormap based on the distribution
3. compute quantization table which maps 24-bit colors into pixel values
4. scan the image, quantizing each pixel.

Choosing the color map is the major task. For once this is done, computing the mapping table from colors to pixel values is straightforward.

To simplify the upcoming quantization algorithms, we will first define the function closest. Let C2 = closest(C1) be the number of the color in the colormap which minimizes distance(C1,C2). With this function, we can write the quantization problem as:

Choose the colormap which minimizes:

$$\text{Distortion} = \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} \text{Distance}(\,\text{color}(x,y)\,,\ \text{color}(\text{closest}(\text{pixel}(x,y$$

Several quantization algorithms will be described in detail later. But first, for practice, we will solve the simpler 1-dimensional quantization problem.

Chapter III

1-DIMENSIONAL TAPERED QUANTIZATION

One wishes to approximate a distribution of N points on the interval [0,255] with K points. The quantization cells here are called intervals, and the partitions at either end, the decision levels. The representative for each interval is called the reconstruction level, or output level.

For monochromatic images, we simply use the distribution of gray levels in the image as our point distribution. Because we are working with 8-bit graylevels, most intensities between 0 and 255 will be repeated a number of times.

In uniform quantization, the decision levels are equally spaced, and the recostruction levels are placed at the midpoint of each interval.

In non-uniform, or tapered quantization, the intervals are shortened or lengthened according to the probability of each part of the interval. In the communications field, tapered quantization is called companding, because it is achieved by compressing, then uniformly quantizing a signal to be transmitted. At the receiver, the signal is expanded to un-do the compression. The first mention of tapered quantization was made by Panter and Dite [38].

Most of the previous work on quantization has been for the purpose of analog-to-digital conversion in signal processing. In the methods described in the literature, instead of quantizing a discrete variable using frequency histograms, one quantizes over a continuous range whose statistics are given by the probability density of a random variable. Elias [12] gives an excellent history of the subject.

Max [30] was one of the first to write about the subject of optimal quantization. His method was to approximate the probability density of an analog signal with a gaussian distribution, find the optimal quantizer for the gaussian, and use that quantizer on his signal. This is slightly different from our approach; we intend to find a quantizer for each frame, rather than process a number of frames with a single quantizer.

Let the distribution be the set of points p(i) for 1<=i<=N (where N = m x n) and each p is an integer in the interval [0,255]. The quantization function y(x) has output value y[k] when x lies within the input range between x[k]+1 and x[k+1]. (x[0] = -1, x[K] = 255). If f(x) is the number of points p[i] with value x, then our goal is to minimize

$$\text{distortion} = \sum_{k} D[k] \quad \text{by varying the } x[k] \text{ and } y[k]$$

$$\text{where} \quad D[k] = \sum_{x[k]+1}^{x[k+1]} (x - y[k])^2 f(x)$$

or, if we think of p as a continuous random variable, then f(x) is the probability density of p at x. Then our sum becomes the integral:

$$D[k] = \int_{x[k]}^{x[k+1]} (x-y[k])^2 f(x)\, dx$$

We can derive a relation between y[k-1], y[k], and x[k] which reduces the number of independent variables in this optimization problem. If y[k-1] and y[k] are fixed, then what value of x[k] minimizes the sum S = D[k-1]+D[k]? The decision level should be placed midway between the output levels on either side: x[k] = (y[k-1]+y[k])/2. In addition, D[k] is minimized when y[k] is the centroid of its interval:

$$y[k] = \frac{\int_{x[k]}^{x[k+1]} x \cdot f(x)\, dx}{\int_{x[k]}^{x[k+1]} f(x)\, dx}$$

Max used a trial-and-error method to find the optimum decision and output levels. The first output level, y[0], is chosen, and all subsequent levels can be derived using (?) and (?). Many different placements for y[0] are tried until the last output level comes out to be the centroid of the interval from x[k-1] to x[k].

Another approach is that used by Bruce [8], and rediscovered by the author. It is a dynamic programming algorithm which takes advantage of the independence of different quantization intervals. Dynamic programming is a process by which a complex, multivariate decision is broken down into a sequence of decisions; it is called a multistage decision process [5], [6], [31].

Netravali and Saigal [32] also devised an algorithm for optimal 1-dimensional quantization. Theirs is based on methods for finding fixed points in mappings.

FINDING THE OPTIMAL QUANTIZATION BY DYNAMIC PROGRAMMING

Because we are limiting ourselves to one distortion formula - the squared distance in RGB space - our algorithm here is slighty simpler than the general one described by Bruce. If the decisions levels x[k] are known, the y[k] can be derived by the centroid formula given above. Therefore, to find a K-interval quantization of a probability density function p, all we need do is find the K-1 decision levels x[1], x[2], ..., x[K-1].

The Algorithm:

We will consider one decision level at a time, advancing from left to right (small k to large), accumulating information about the quality of different placements of x[k]. Given a placement of x[k] from 0 to 254, we want to find the location of x[k-1] which minimizes the distortion "so far": the sum D[0] + D[1] + D[2] + ... + D[k-1]. This is facilitated by accumulating these partial sums of "left-distortions" in a table. We generate a two-dimensional table containing, for all k and x*, the minimum left-distortion possible, given that x[k] = x*. In addition, we will need a "backpointer" table which holds the position of x[k-1] which minimized the error to the left of x[k].

```
    LDIST(k,x*) = min sum D[j]  - given that x[k]=x*
         over all possible positions for x[k-1], x[k-2], ..., x[1]
```

```
    PREV(k,x*) = position of x[k-1] which minimized LDIST(k,x*)
```

Generating this table level by level is similar to generating Pascal's triangle row by row. They are both Dynamic Programming methods because they avoid recursion by keeping a table of accumulated results. The big win is that LDIST can be computed at each level without a lot of recomputation:

```
    LDIST(k,x*) = min D[k-1] + LDIST(k-1,x') over all possible x'
```

So when k=K-1 is reached, the backpointers are traced to find the K reconstruction levels.

This is far superior to the brute-force method. It can find the optimal 16 level quantization for a black and white image in one or two minutes. Figure 6 shows a comparison between a 16-level (4-bit) uniform quantization of "Pamela" and a tapered quantization, also with 16 levels. Figure 7 shows the

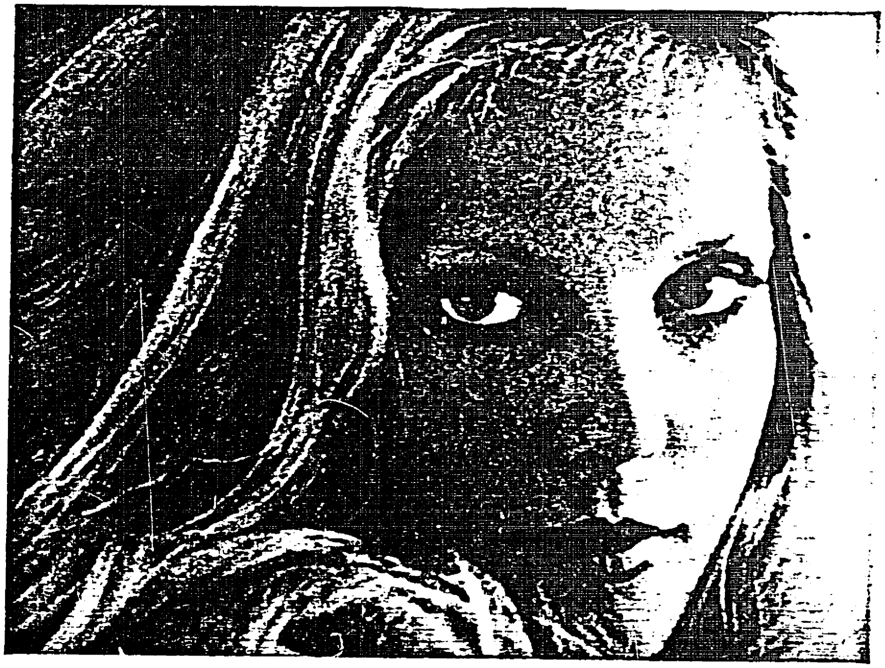


fig. 5, 8-bit original image

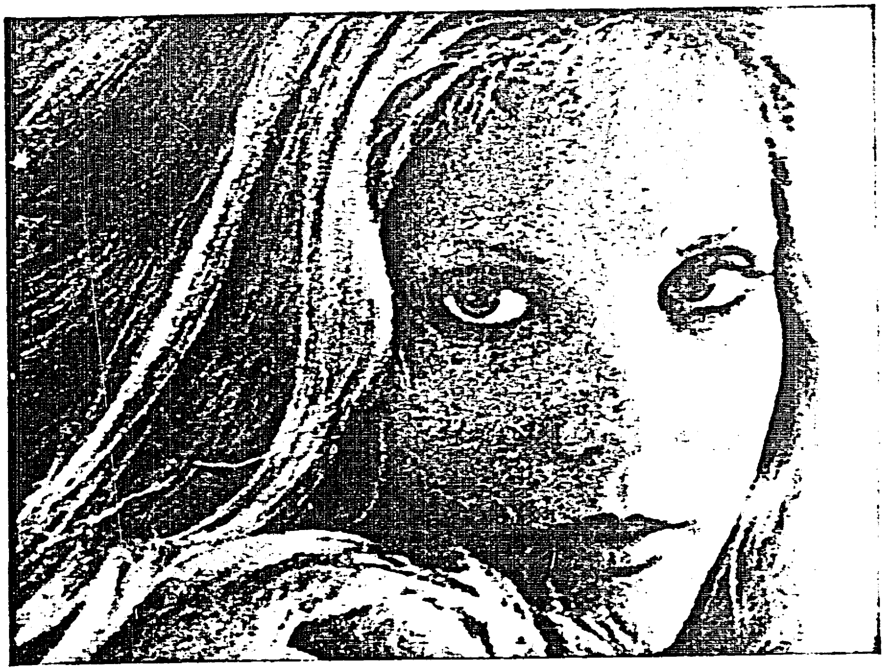


fig. 6a, 4-bit uniform quantization

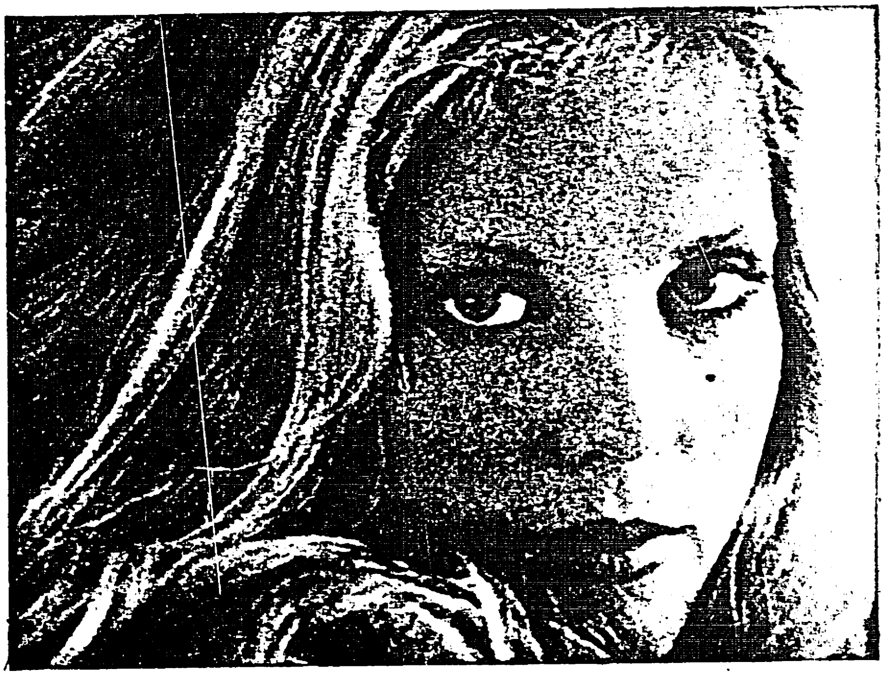


fig. 6b, 4-bit tapered quantization

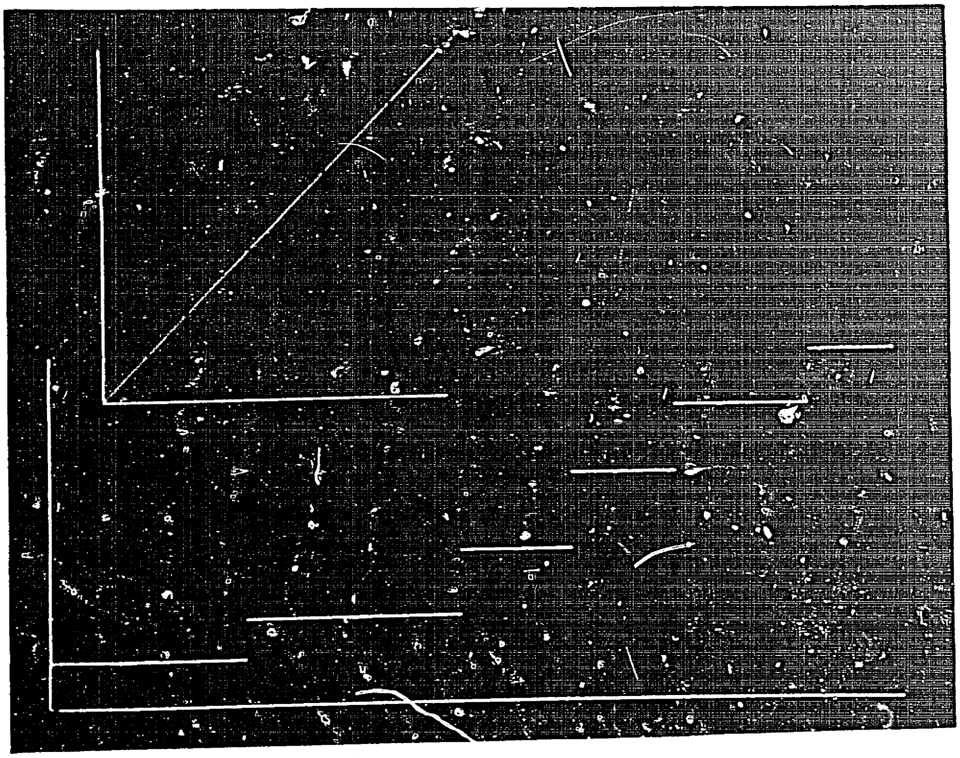


fig. 7, tapered quantization
top graph: quantization function y(x)
bottom graph: function p(i)

quantization function, and another plot of the inverse of the probability distribution, p(i).

RELATION TO FUNCTION APPROXIMATION

Plotting the distribution this way shows graphically that the one-dimensional quantization problem is related to the approximation of certain functions of one variable. That relation can be explored mathematically:

If $F(x) = \int f(x)dx$ and $i = F(x)$ then $x = p(i) = F^{-1}(i)$

Both the probability distribution and its inverse function are non-decreasing. Approximation of p(i) by a staircase function having K steps is equivalent to the quantization formulation. This staircase function is a stepwise-constant function r(i) defined by:

r(i) = y[k] if x[k]+1 <= p(i) <= x[k+1]

or (equivalently) if t[k]+1 <= i <= t[k+1]

where t[k] = F(x[k]) and x[k] = p(t[k])

Minimizing the D in the quantization problem corresponds to the least-squares best-fit, here in the function approximation problem.

$$D = \sum_{k=0}^{K-1} E[k] = \int (p(i) - r(i))^2 di$$

$$\text{Let } E[k] = \int_{t[k]}^{t[k+1]} (p(i) - y[k])^2 \, di$$

Why is this an equivalent problem? We can transform Max's distortion formula into the one above:

$$D[k] = \int_{x[k]}^{x[k+1]} (x - y[k])^2 \, f(x) \, dx$$

$$\text{and } f(x) = \frac{dF}{dx} \Rightarrow f(x)dx = dF = di$$

$$\text{and also } x = x[k] \Rightarrow i = F(x[k]) = t[k]$$

$$\text{so } D[k] = \int_{t[k]}^{t[k+1]} (x - y[k])^2 \, di$$

$$\text{but } x = p(i), \text{ so } D[k] = E[k].$$

So we see that the monochromatic color-selection problem is related to several others:

the analog signal tapered-quantization problem

the approximation of a random variable, or a point distribution

the approximation of non-decreasing functions with staircase functions

Chapter IV

3D TAPERED QUANTIZATION

HISTORY OF COLOR QUANTIZATION

Color quantization is usually done by treating the three color coordinates independently. Although the three color components can be decorrelated by transforming the color space to YIQ or Lab or some other color space (see [28] and [41]), independent quantization in these spaces is inefficient because much of their space lies outside the RGB color cube [24]. In any event, color transforms are of little use in quantization for display; their proper place is in image compression systems. An excellent review of the subject is Limb [28]. Multidimensional quantization is discussed by In Der Smitten [23], Stenger [47], and Zador[51].

RELATION TO SPHERE-PACKING

Color quantization of a uniformly-distributed set of 3-d points is analogous to close-packing of spheres into a cube. Coxeter [11] describes empirical experiments to determine the highest density packing of spheres. Cubic close-packing is the densest known, but just as in multi-dimensional quantization, there are a number of unanswered questions in this field.

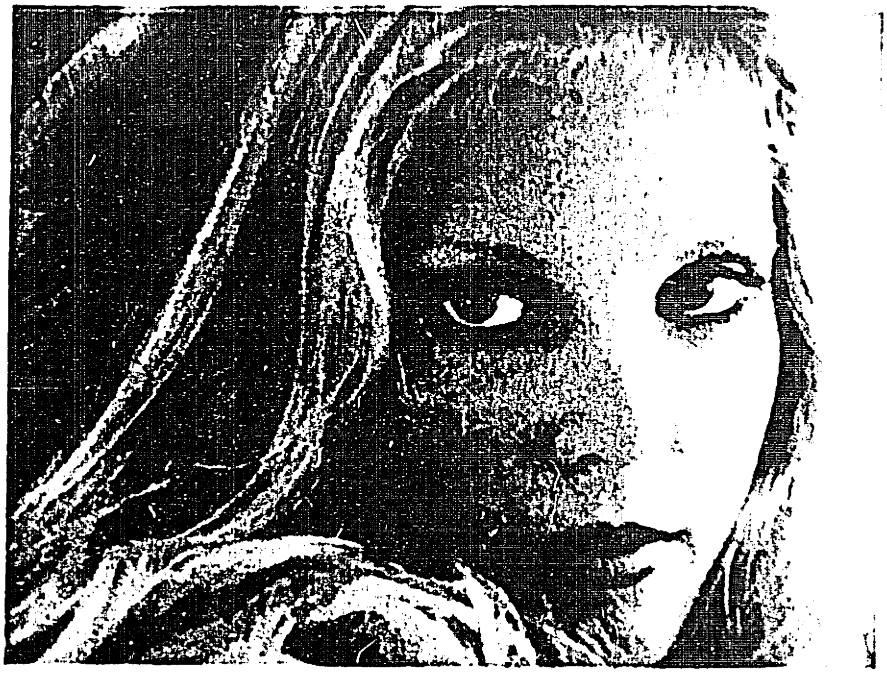


fig. 8, 9-bit uniform quantization
of color (3 bits per primary)

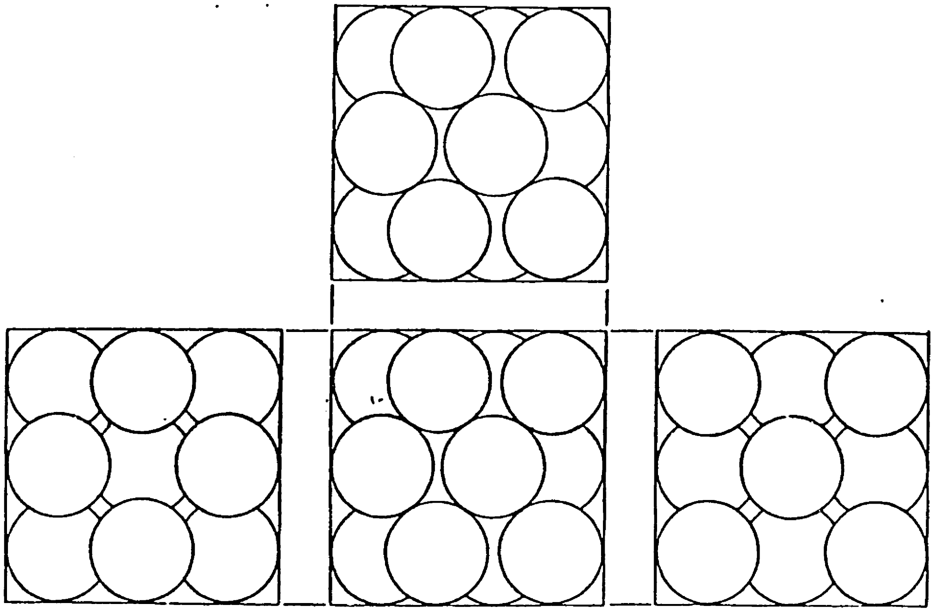
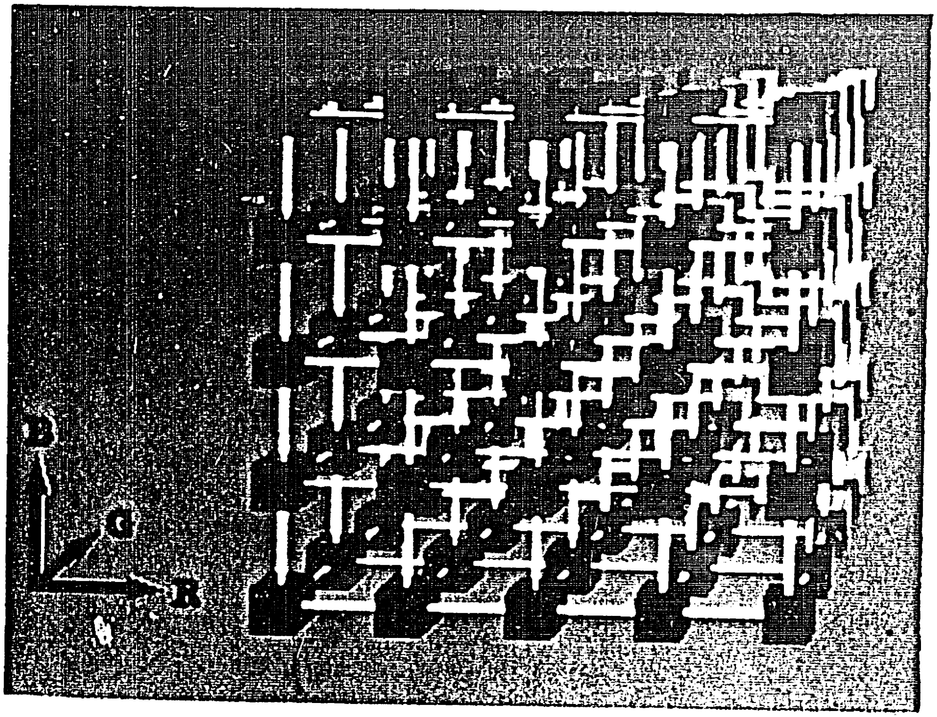
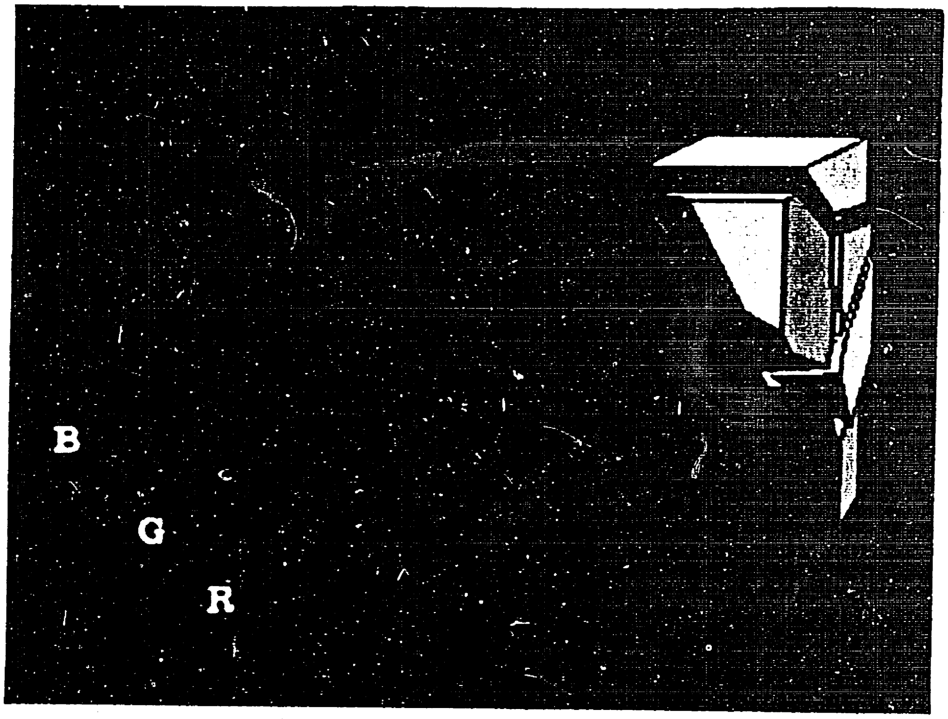


fig. 9, packing 18 spheres into
a cube (from Goldberg)



Perspective view of 125 color cells in a uniform quantization lattice. (struts added for clarity)



Exploded view of 16 non-uniform quantization cells in the RGB cube.

TAPERED QUANTIZATION

When the quantization problem is generalized to 2 or 3 dimensions, it becomes much more difficult. The reason for this is the increased interdependency of quantization cells. While in the one-dimensional case all intervals are determined by the two decision levels at either end, in the two and three-dimensional cases the quantization cells can be polygons (and polyhedra) with any number of sides. Because each quantization cell contacts a number of others (3 or more), and the topology of quantization cells is variable, we can no longer represent a quantization by a list of decision levels x[k].

Our only recourse is to derive the quantization decision levels from the quantization outputs, in contrast to the algorithm given previously for the 1-d case, which first found the decision levels, and then computed the reconstruction levels.

We should formalize the multi-dimensional quantization problem. If our points are d-dimensional, then let

P[i]=(x1[i],x2[i],...,xd[i])   for 1<=i<=N

be the list of (unordered) points. We wish to choose the colormap (a set of points Y[k]=(y1[k],...,yd[k]) which minimizes the distortion measure D:

$$D = \sum_{i} distance\,(\,P[i],\ y[closest\,(P[i])]\,)$$

where (as before) closest(P) is the k which minimizes the distance between P and Y[k]. k is the cell number, or pixel value.

The locus of points for which closest(P)=k is the quantization cell having output k. An example is shown in figure 3.

Although it has not been proven, it seems certain that there is no efficient (polynomial time) algorithm for the solution of the optimal multidimensional quantization problem. All further discussion is limited to approximations to optimal quantization.

ALGORITHMS FOR 3-D QUANTIZATION:

The Popularity Algorithm:

The popularity algorithm was invented by Tom Boyle and Andy Lippman at the AMG in the Summer of 1978. In this scheme one first quantizes the original colors down to 5 bits per primary. (This reduces the total number of bits per color from 24 to 15, thus allowing us to fit one color in two bytes of the computer for storage.) This clumping, or grouping of the colors has the effect of reducing the number of different colors, and increasing the frequency of each color. The assumption is that a good choice for the colormap can be made by finding the densest regions in the color cube. The popularity algorithm chooses the K clumps with the greatest popularity as its colormap.

Boyle's program, which was called the "Color Maker", was rewritten by the author in July 1979. The new version is called IMAGE. IMAGE has been coded in assembler, thoroughly debugged, and is now in regular use at the AMG. It can complete all four phases of the process in 2.5 minutes. More implementation details for IMAGE are given in the appendix. Several examples of its quantized images are displayed in figures 10-15.

TWO PROBLEMS AND THEIR SOLUTIONS:

Boyle's MAKER program used a different approach when mapping those colors which were not among the lucky few to make it into the colormap. Instead of using a function like closest which finds the nearest color spectrally, it would choose the pixel's neighbor spatially.

With this algorithm, color searching was done while the image was being redrawn (during phase 4), and there was no phase 3 at all. MAKER would examine each pixel to determine if its color was among the chosen. If not, the program would simply write the value of the pixel to its left (which it had just computed). This method did not work well, as you can see from the image below. Boyle frequently got streaks near vertical edges in his images (see fig. 16).

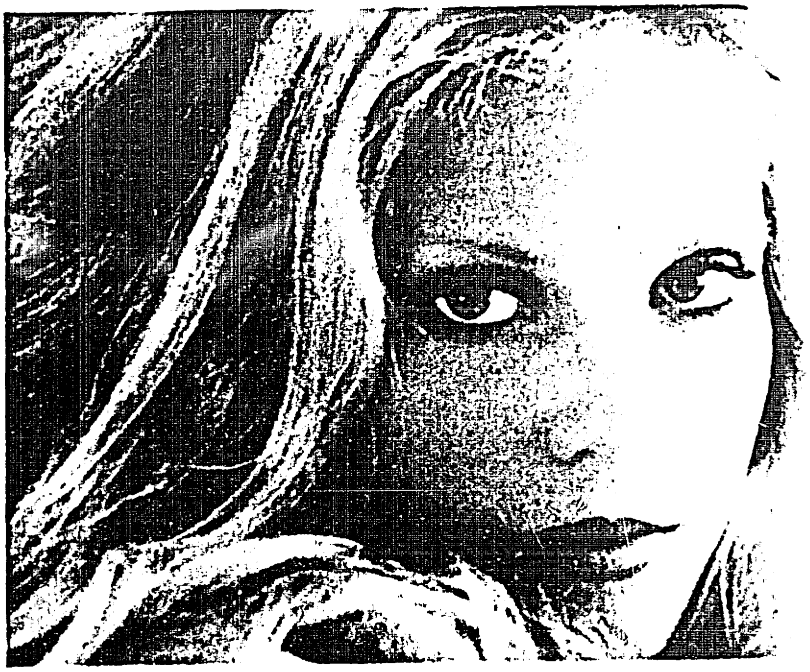


fig. 10, 9-bit "Pamela"
(made with IMAGE)

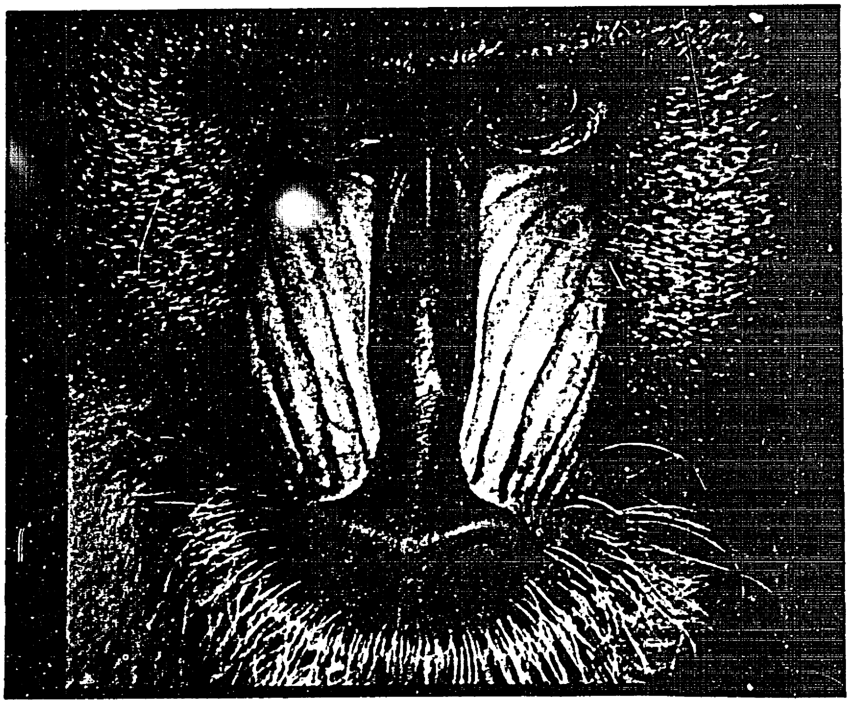


fig. 11, 8-bit "Mandrill"
(made with IMAGE)

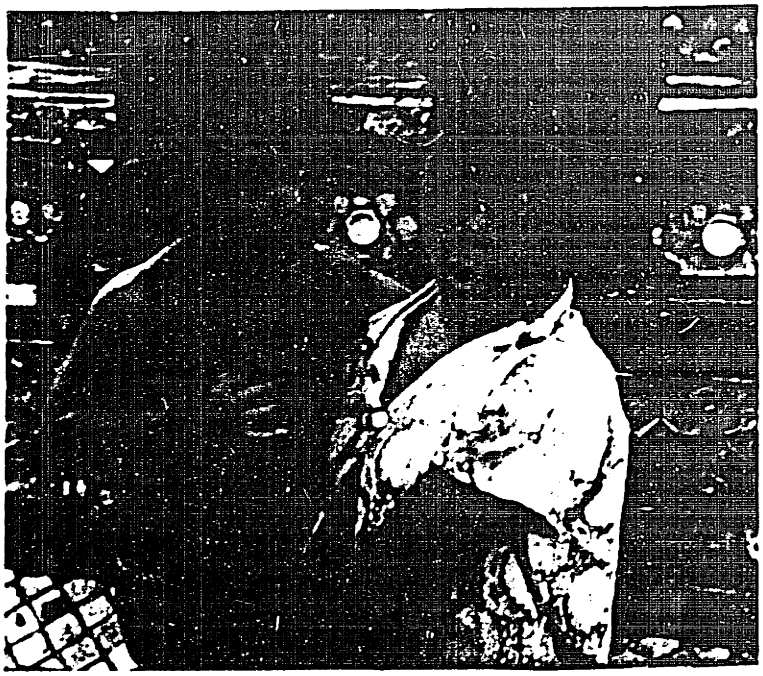


fig. 12, 9-bit "Brothers"
(made with IMAGE)

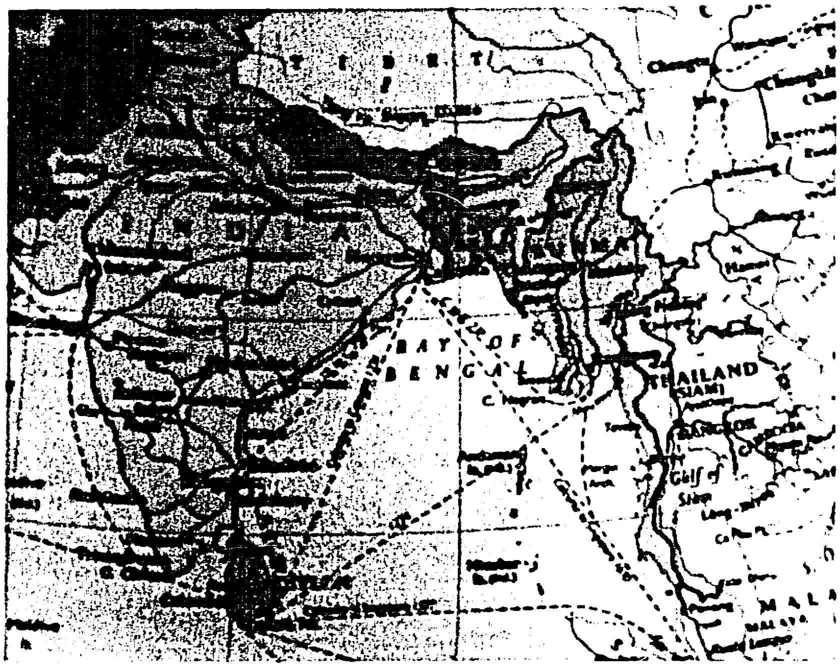


fig. 13, 9-bit "Far_East"
(made with IMAGE)

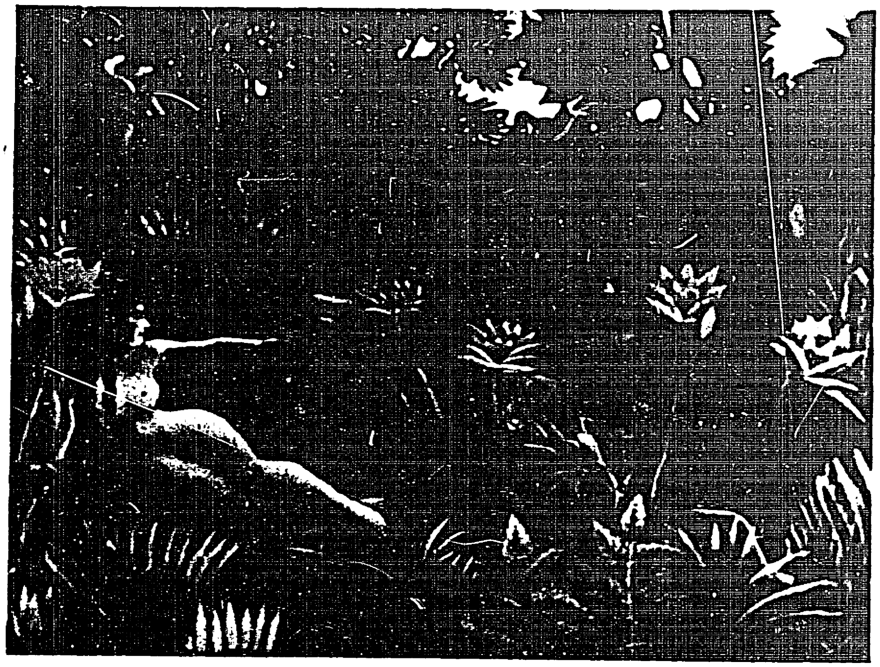


fig. 14, 9-bit "Rousseau"
(made with IMAGE)

Colorplate 62 Franz Marc *Blue Horses* 1911
Oil on canvas, 40¼ x 71¼ Walker Art Center Minneapolis

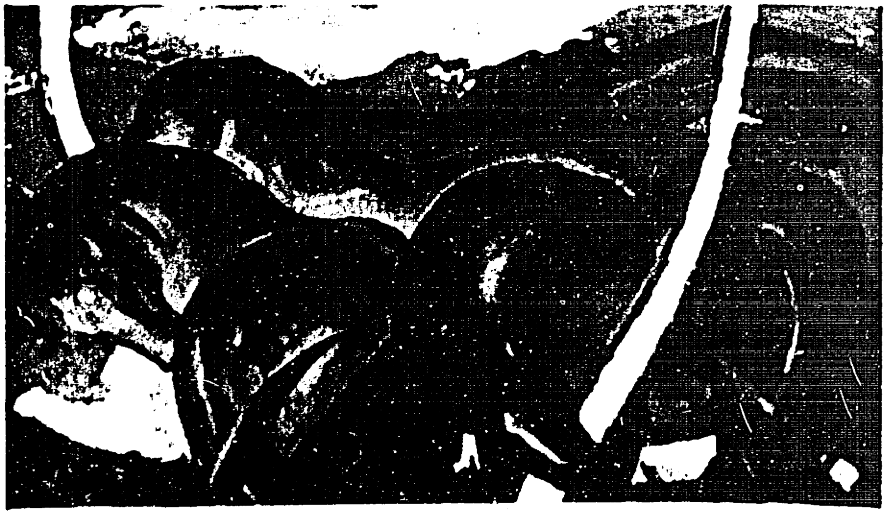


fig. 15, 9-bit "Marc2"
(made with IMAGE)

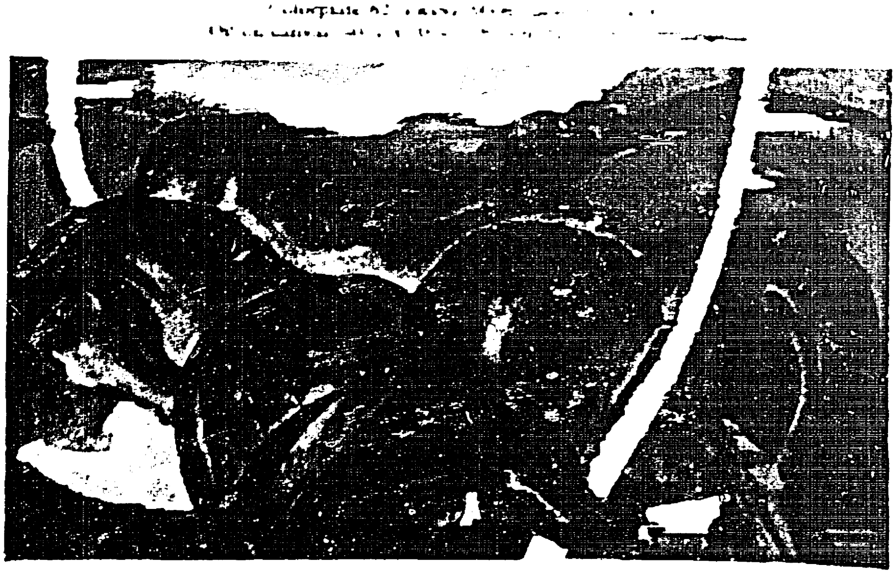


fig. 16, 8-bit "Marc2" showing streaks (made with MAKER)

To prevent these defects, MAKER was modified to use the closest function.

IS HUMAN INTERACTION NECESSARY?

There are several cases in which IMAGE will neglect the colors of high-interest areas of an image, such as eyeballs. To remedy this, a method for manually weighting the sampling phase was developed. The DOODLE program allows a user, employing a digitizing tablet, to point to areas of the image he wants to emphasize. These points will be multiply counted in the frequency table, guaranteeing better representation in the colormap. Generally this feature is not needed, but occasionally it is very effective.

MORE PROBLEMS WITH THE POPULARITY ALGORITHM:

It was found that IMAGE performs poorly when extreme compression is desired (quantization to 6 or fewer bits). The shortcomings of the popularity algorithm arise because it frequently neglects sparse, but remote, regions of the color cube. The populous regions are always well-represented, at the expense of less-popular regions which do not have large clump-counts.

It is the grays and low-saturation colors which are popular and over-represented, and the highly-saturated primaries which are neglected in the colormaps chosen by the popularity algorithm.

One can draw a useful analogy to a political quantization problem: the selection of members to the legislative branch of government. The popularity algorithm is like the House of Representatives - there are few representatives from the sparse regions (eg. Alaska and Arkansas). The Senate, on the other hand, is like uniform quantization. It provides a better representation of the Unites States because its makeup is makeup is determined by the spatial extent of states. We will borrow this idea of emphasising the spatial over the popular in the Median Cut algorithm.

The Median Cut Algorithm

This algorithm was invented by the author in December, 1979. It was undertaken as an alternative to the popularity algorithm. The median cut algorithm repeatedly subdivides the color cube into smaller and smaller rectangular boxes.

It starts by finding the minimum and maximum red, green, and blue among all N colors (N usually 307200).

Iteration step: Split a box.
For each box, the minimum and maximum value of each component is found. The largest of these three determines the dominant

dimension - the dimension along which we would want to split the box. The box with the largest dominant dimension is found, and that box is split in two. The split point is the median point - the plane which divides the box into two halves so that equal numbers of colors are on each side. The colors within the box are segregated into two groups depending on which half of the box they fall.

The above step is repeated until the desired number of boxes is generated. Then the representative for that cell (the quantization output) is computed by averaging the colors contained. The list of recontruction levels is our colormap.

The CUT program was implemented by taking IMAGE and replacing step 2 (the colormap selection phase), with the median cut algorithm described above. An example of its output is figure 17.

The Variance Bisection Algorithm:

This algorithm is a variation on the Median Cut algorithm. There is only one change: instead of splitting at the median point, we split spatially. The program considers all possible split planes perpendicular to the dominant dimension, and for each finds the variance of colors in the left and right halves of the box. The split point is chosen in order to minimize that sum. This

Colorplate 62 Franz Marc Blue Horses
Oil on canvas 40¼ x 70⅞ Walker Art Center Minneapolis

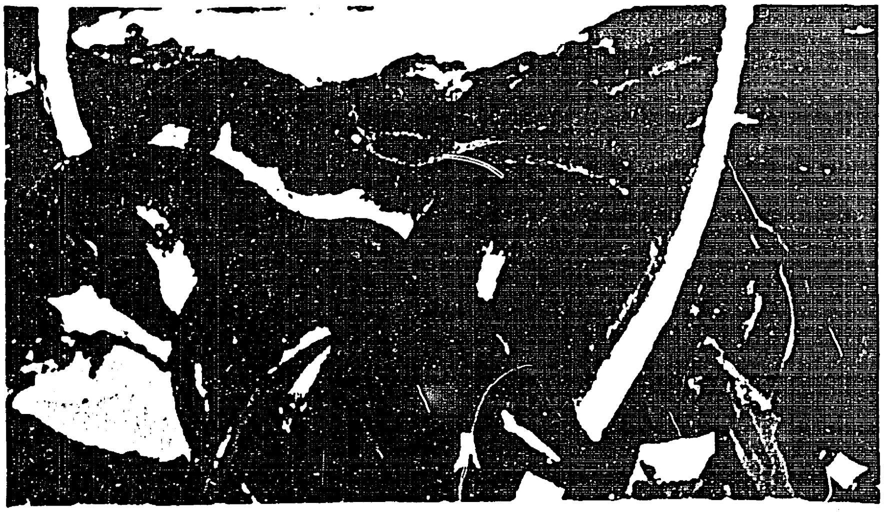


fig. 17, 4-bit "Marc2"
(made with CUT)

Colorplate 62 Franz Marc Blue Horses
Oil on canvas 40¼ x 70⅞ Walker Art Center Minneapolis

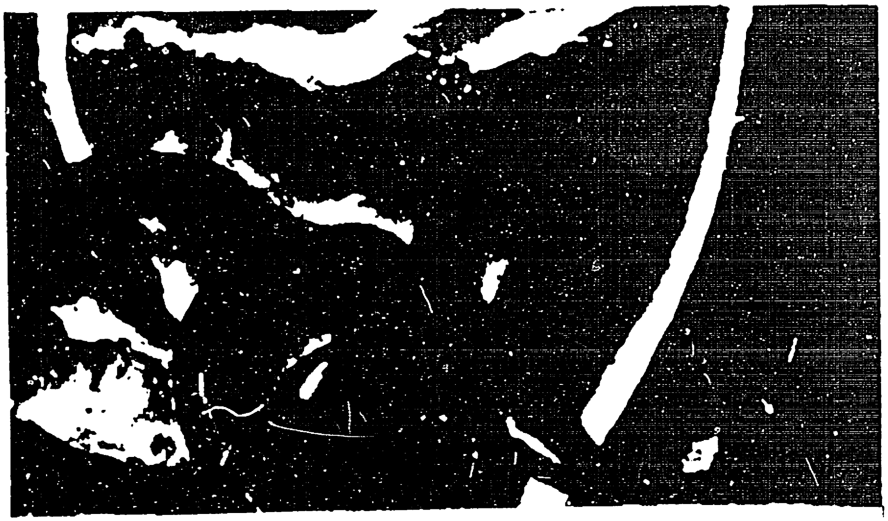


fig. 18, 4-bit "Marc2"
(made with IMAGE)

should tend to minimize the mean squared error much better than the median cut method. The variance bisection method has not been implemented yet.

THE ADDITION OF NOISE

As mentioned before, the addition of noise to a severely quantized image can help remove quantization distortion. This was tried for monochromatic images by Goodall [16] and Roberts [42]. They found that black and white images can be quantized to 3 or 4 bits with good results. Huang et al. [20] point out that this noise addition succeeds because the eye objects more to structured noise than unstructured noise. The lesson is that it is best to transform quantization noise into random noise.

Two examples of the addition of color noise are shown in figures 19 and 20.

Colorplate 62 Franz Marc *Blue Horses* 1911
Oil on canvas, 40¼ x 70¾ Walker Art Center Minneapolis

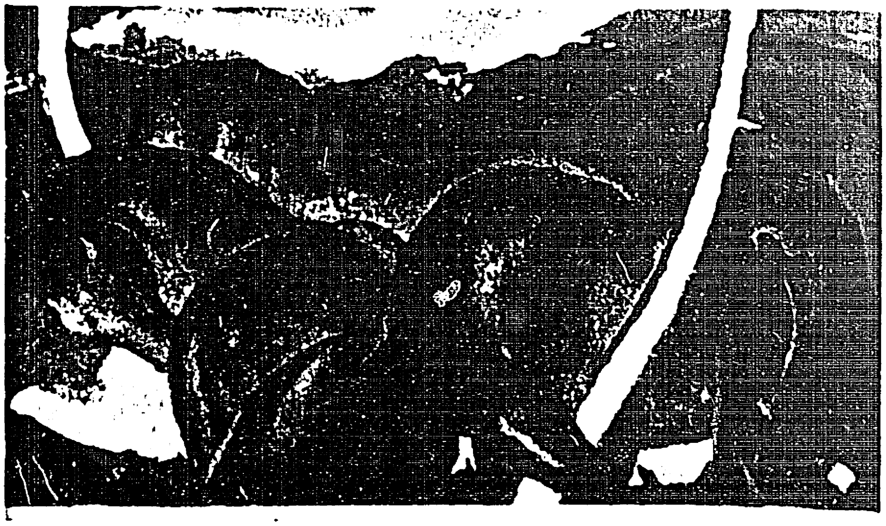


fig. 19, 4-bit "Marc2" with noise added before quantization

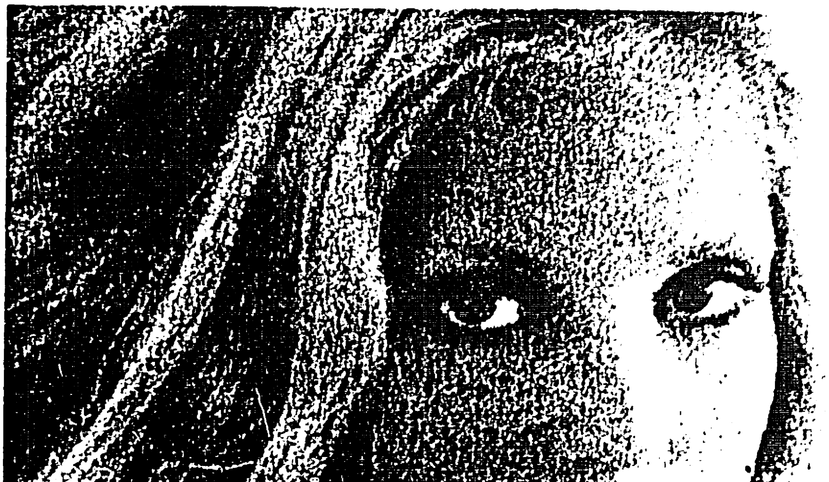


fig. 20, 2-bit (only 4 colors) "Pamela" with noise added before quantization

Chapter V

CONCLUSIONS AND IDEAS FOR FURTHER STUDY

IMPROVEMENTS TO THE DISTORTION FORMULA:

The error criterion used in all of the research described above is a squared error formula in RGB space:

$$distortion = \Delta r^2 + \Delta g^2 + \Delta b^2$$

This formula is not ideal, since it does not conform to the subjective sensitivity of viewers. The ideal distortion measure would be based on the perceptual properties of human viewers, since it is they who ultimately use the images [47].

MacAdam (see [28]) has carried out extensive physiological tests to determine the perceptual differences between colors. Ideally, a distance measure, or color metric, should be defined using his empirical measurements. This would provide a more accurate model of the sensitivity of the eye and brain to different color ranges. Unfortunately, the formula for such a model would be spatially variant, unlike the simple distance formula used in our model. Thus the distance between two colors would be a function of six variables:

$$distance(c_1, c_2) = f(r_1, g_1, b_1, r_2, g_2, b_2)$$

so we would no longer be working in a Euclidean space. This makes computation much more difficult.

If a complete perceptual distortion model is not feasible, then what improvements our simple the distance formula could be made? One improvement on our formula would be the addition of coefficients to weight the three components differentially:

$$distance = ar \cdot \Delta r^2 + ag \cdot \Delta g^2 + ab \cdot \Delta b^2$$

Limb et al. [28] indicate that the proper coefficients here should be approximately: ar = 1.0, ag = 1.2, ab = 0.8. Another improvement could be achieved by using the logarithm of intensity rather than the intensity itself, since humans are more sensitive to intensity differences at low intensities than at high ones (the Weber-Fechner law - see [9] and [25]).

GENERAL CONCLUSIONS

We found that the architecture of integrated frame buffers forces certain restrictions on any attempt to display color images. One is naturally led to the non-separable multidimensional quantization problem. Although the optimal solution of this problem is computationally intractable, there are approximate techniques which allow high-quality color quantization to be done efficiently. Using one of the algorithms described, it is possible to display a full-color image using only 256 colors, thus tripling memory efficiency.

It is conjectured that the use of random noise could improve efficiency even further.

ACKNOWLEDGEMENTS

I would like to thank Professor Nicholas Negroponte and Andrew Lippman of the Architecture Machine Group for their support. They have provided me with abundant resources and freedom over the last three and a half years. Among fellow students working at the AMG, the most deserving of credit was Tom Boyle, who introduced me to the Color Image Quantization problem a year ago.

Other members of the AMG were important to this research as well. Dan Franzblau and Walter Bender were constant critics of my images, as was Professor Ron MacNeil of the Visible Language Workshop. Paul Shelman, Tom Brigham, and David Backer were my principal image users - they kept me busy documenting and debugging my programs. The theoretical formulation of the problem was helped by Paul Trevithick and Professor Gilbert Strang. Jean-Pierre Tanaka was invaluable as a proofreader.

Paul Pangaro, formerly at the AMG, deserves mention as well. It was he who convinced me to write this thesis. I must also thank UROP, the Undergraduate Research Opportunities Program at MIT, for introducing me to the Architecture Machine Group in 1976.

Appendix

THE IMAGE PROGRAM

PHASE 1:

Sample the 5-bit separations of the image, accumulating the frequency counts of each of the possible 2^15=32768 colors.

Because of the word-size on our computers (Interdata 7/32's), the frequency counts cannot exceed 32767, so only a fraction (usually 1/20) of the pixels in an image are actually counted in the frequency table. The sampling pattern used is a random sprinkling. There is also a method of manually choosing the sampling points which was described in chapter IV.

Another hardware-related problem is the huge size of the frequency table. There is no room for a 64K frequency table in the processor on which IMAGE is run, so we must compact it. On most images, only 2000 to 5000 of the 32768 possible colors appear. This means that we can store the data using less memory by creating a linear list of colors and their frequencies. One long list was not used, however, because of the long search time to find a color. To optimize search time, a hash table was used.

The low 3 bits of each 5-bit primary are concatenated into a 9-bit hash key. This key is our index into 512 short lists.

Because the low 3 bits of each component are used instead of the 3 most significant bits, we are more likely to get a uniformly-filled hash table, and the average search time will be minimized.

Each hash element requires 6 bytes: 2 bytes for the 15-bit color, 2 for the frequency count, and a 2-byte offset to the next hash element in the hash bucket.

PHASE 2:

This is the important phase. It is where the frequency table is used to choose the colormap. IMAGE uses the popularity algorithm. That is done, as we mentioned, by finding the K most frequent 15-bit colors - usually K=512 or 256. After the colormap is chosen, it is loaded into the hardware color lookup table of the frame buffer, for use in phase 4.

PHASE 3:

During this phase we compute the mapping from colors to pixel values. For each of the colors which were present during phase 1 (there were something like 2000 or 5000 of them), we use the closest function to find the nearest of the K colors in the colormap. In other words, we quantize.

IMPLEMENTING CLOSEST

The program for fast mapping of colors to pixel values is worth describing. The straightforward way to implement closest is to search through the K colors in the colormap and find the one which minimizes the distance from color P to Y[k]. But because closest is a critical, low-level subroutine in IMAGE, it was optimized.

The subroutine for this (called BEST), also uses hash tables, though differently organized than those for the sampling phase. The hash buckets for BEST are organized into 7x7x7=343 lists, each containing the colors from a 32x32x32 cubical region of the color space. Since neighboring regions overlap by 16 units, each color can be in as many as 8 lists. This huge hashed color table is built, after the colormap is computed in phase 2, and all subsequent calls to the closest BEST use this table for fast lockup.

Given any color in RGB color space, its closest match is computed by finding which of the 343 cubes has its center closest to our input color. Then the BEST program searches through the short list of colors for that cube. For a colormap containing K=512 colors, the average list length is 12 colors. The distance between the input color and each of these colors is computed, and the color number (pixel value) of the closest is returned. If the list is empty (this happens in sparse regions of the color cube - usually the saturated colors) BEST does an exhaustive search through all K colors.

The author must confess that this algorithm does not always return the closest color. But if the closest color is within a sphere of radius 8 or outside a sphere of radius 24, the correct answer is guaranteed. In this mid-range, it is sometimes off by a factor of 3. In practice, no effects of its imperfection have been noticed.

PHASE 4:

During the previous phase the sampling hash table is filled so that along with each 15-bit color is found the new pixel value. Now we scan the 15-bit original image, looking up each color in the hash table to find its quantized pixel value. When this pixel value is sent to the frame buffer, the image is painted on the display. This completes the IMAGE algorithm.

REFERENCES

[1] Algazi, V. R., "Useful Approximations to Optimum Quantization," IEEE Trans. on Communication Technology, Vol. COM-14, No. 3, June 1966, p. 297.

[2] Andrews, H. C., Computer Techniques in Image Processing, New York: Academic Press, 1968.

[3] Andrews, H. C., A. G. Tescher, and R. P. Kruger, "Image Processing by Digital Computer," IEEE Spectrum, Vol. 9, No. 7, July 1972, p. 20.

[4] Andrews, H. C., "Digital Image Processing," Spectrum, Vol. 16, No. 4, April 1979, p. 38.

[5] Bellman, R., Dynamic Programming, Princeton: Princeton University Press, 1957.

[6] Bellman, R., and S. E. Dreyfus, Applied Dynamic Programming, Princeton: Princeton University Press, 1962.

[7] Berger, T., "Optimum Quantizers and Permutation Codes," IEEE Trans. on Information Theory, Vol. IT-18, No. 6, Nov 1972, p. 759.

[8] Bruce, J. D., Optimum Quantization, MIT R.L.E. Technical Report #429, 1965.

[9] Camana, P., "Video-Bandwidth Compression: a Study in Tradeoffs," IEEE Spectrum, Vol. 16, No. 6, June 1979.

[10] Chen, W., and C. H. Smith, "Adaptive Coding of Monochrome and Color Images," IEEE Trans. on Communications, Vol. COM-25, No. 11, Nov. 1977, p. 1285.

[11] Coxeter, H. S. M., Introduction to Geometry, New York: John Wiley and Sons, 1961.

[12] Elias, P., "Bounds on Performance of Optimum Quantizers," IEEE Trans. on Information Theory, Vol. IT-16, No. 2, March 1970.

[13] Frei, W., "Quantization of Pictorial Color Information; Nonlinear Transforms," Proc. IEEE, Vol. 61, April 1973, p. 465.

[14] Gish, H., and J. N. Pierce, "Asymptotically Efficient Quantizing," IEEE Trans. on Information Theory, Vol. IT-14, No. 5, Sept 1968, p. 676.

[15] Goldberg, M., "On the Densest Packing of Equal Spheres in a Cube", Mathematics Magazine, Vol. 44, Sept.-Oct. 1971,

p. 199.

[16] Goodall, W. M., "Television by Pulse Code Modulation," Bell System Technical Journal, Jan. 1951, p. 33.

[17] Gronemann, U. F., Coding Color Pictures, MIT R.L.E. Technical Report #422, 1964.

[18] Habibi, A., "Survey of Adaptive Image Coding Techniques," IEEE Trans. on Communications, Vol. COM-25, No. 11, Nov. 1977, p. 1275.

[19] Hall, C. F., "Digital Image Compression in a Perceptual Space," Ph.D. dissertation, USCIPI Report #790, University of Southern California, Feb. 1978.

[20] Huang, T. S., O. J. Tretiak, B. T. Prasada, and Y. Yamaguchi, "Design Considerations in PCM Transmission of Low-Resolution Monochrome Still Pictures," Proc. IEEE, Vol. 55, No. 3, Mar. 1967, p. 331.

[21] Huang, T. S., W. Schreiber, and O. J. Tretiak, "Image Processing," Proc. IEEE, Vol. 59, No. 11, Nov. 1971, p. 1586.

[22] Huang, T. S., Picture Processing and Digital Filtering, Springer Verlag, 1975.

[23] In Der Smitten, F. J., "Data-Reducing Source Encoding of Color Picture Signals Based on Chromaticity Classes," Nachrichtentech. Z., Vol. 27, 1974, p. 176.

[24] Jain, A. K., and W. K. Pratt, "Color Image Quantization," National Telecommunications Conference 1972 Record, IEEE Pub. No. 72, CHO 601-5-NTC, Dec. 1972.

[25] Kretz, F., "Subjectively Optimal Quantization of Pictures," IEEE Trans. on Communications, Vol. COM-23, No. 11, Nov. 1975, p. 1288.

[26] Limb, J. O., "Design of Dither Waveforms for Quantized Visual Signals," Bell System Technical Journal, Vol. 48, Sept. 1969, p. 2555.

[27] Limb, J. O., C. B. Rubinstein, and K. A. Walsh, "Digital Coding of Color Picturephone Signals by Element-Differential Quantization," IEEE Trans. on Communication Technology, Vol. COM-19, No. 6, Dec 1971, p. 992.

[28] Limb, J. O., C. B. Rubinstein, and J. E. Thompson, "Digital Coding of Color Video Signals - A Review," IEEE Trans. on Communications, Vol. COM-25, No. 11, Nov. 1977, p. 1349.

[29] Lippel, B., and M. Kurland,. "The Effect of Dither on Luminance Quantization of Pictures," IEEE Trans. on Communications, Vol. 6, 1971, p. 879.

[30] Max, J., "Quantizing for Minimum Distortion", IRE Trans. on Information Theory, Vol. IT-6, Mar. 1960, p. 7.

[31] Nemhauser, G. L., Introduction to Dynamic Programming, New York: John Wiley and Sons, 1966.

[32] Netravali, A. N., and R. Saigal, "An Algorithm for Optimum Quantization," Bell System Technical Journal, Vol. 55, No. 9, Nov. 1976, p. 1423.

[33] Netravali, A. N., and B. Prasada, "Adaptive Quantization of Picture Signals using Spatial Masking," Proc. IEEE, Vol. 65, No. 4, April 1977, p. 536.

[34] Netravali, A. N., and C. B. Rubinstein, "Quantization of Color Signals," Proc. IEEE, Vol. 65, No. 8, Aug. 1977, p. 1177.

[35] Netravali, A. N., and J. O. Limb, "Picture Coding: A Review," Proc. IEEE, Vol. 68, No. 3, Mar. 1980, p. 366.

[36] Newman, W. M., and R. F. Sproull, Principles of Interactive

Computer Graphics, New York: MacGraw-Hill, 1979.

[37] Oppenheim, A., Some Applications of Digital Signal Processing, Englewood Cliffs, New Jersey: Prentice-Hall, 1978.

[38] Panter, P. F., and W. Dite, "Quantizing Distortion in Pulse-Count Modulation with Nonuniform Spacing of Levels," Proc. IRE, Vol. 39, No. 1, Jan 1951, p. 44.

[39] Pearson, J. J., "Adaptive, Hybrid, and Multi-Threshold CAQ Algorithms," Advances in Image Processing Techniques, Proc. of the Society for Photo Optical Instrumentation Engineers, Vol. 87, Aug, 1972, p. 19.

[40] Peterson, J. L., J. R. Bitner, and J. H. Howard, "The Selection of Optimal Tab Settings," Communications of the ACM, Vol. 21, No. 12, Dec. 1978, p. 1004.

[41] Pratt, W. K., Digital Image Processing, New York: John Wiley and Sons, 1978.

[42] Roberts, L. G., "Picture Coding Using Pseudo-Random Noise," IRE Trans. on Information Theory, Vol. IT-8, Feb. 1962, p. 145.

[43] Roe, G. M., "Quantizing for Minimum Distortion," IEEE

Trans. on Information Theory, (Correspondence), Vol. IT-10, Oct. 1964, p. 384.

[44] Rosenfeld, A., Picture Processing by Computer, New York: Academic Press, 1969.

[45] Rosenfeld, A., and A. Kak, Digital Picture Processing, New York: Academic Press, 1976, p. 98.

[46] Schutzenberger, M. P., "On the Quantization of Finite-Dimensional Messages," Information and Control, Vol. 1, 1958, p. 153.

[47] Stenger, L., "Quantization of TV Chrominance Signals Considering the Visibility of Small Color Differences," IEEE Trans. on Communications, Vol. COM-25, No. 11, Nov. 1977, p. 1393.

[48] Stoffel, J. C., "Halftone Pictorial Encoding," Applications of Digital Image Processing, Proc. of the International Optical Computing Conference, Vol. 119, 1977, p. 56.

[49] Thompson, J. E., and J. J. Sparkes, "A Pseudo-Random Quantizer for Television Signals," Proc. IEEE, Vol. 55, No. 3, Mar. 1967, p. 353.

[50] Wintz, P., "Transform Picture Coding," Proc. IEEE, July 1972, p. 809.

[51] Zador, P., "Development and Evaluation of Procedures for Quantizing Multivariate Distributions," Ph.D. dissertation, Stanford, 1964.