8.1

Image Quantization, Halftoning, and Printing

Ping Wah Wong
IDzap, LLC

1 Introduction and Printing Technologies

The rapidly decreasing cost and increasing power of modern day computing devices have contributed to the prevalent use of digital images in recent years. Image processing systems that have found applications in everyday life include digital photography, electronic publishing, on-line museum, electronic catalog shopping, home surveillance, image rich souvenirs, and so on. A typical end-to-end image processing system typically includes input devices, processing devices, and output devices. Output devices are very important in imaging systems because they allow the image data to be presented to the user in a human observable form. Popular imaging output devices include image printers, video displays, liquid crystal displays, projection devices, and many others.

In practical applications, quantities in digital images are usually quantized to 8 bits. Specifically, we use 8 bits to represent the image pixel intensity (graylevel) for monochrome (grayscale) images, and use 24 bits to represent the combined red, green and blue intensity levels for color images (i.e., 8 bits for each color plane). Hence we have 256 graylevels for monochrome images and 16 million colors for color images. It has been demonstrated in practice that such a representation is normally sufficient to be considered "continuous tone."

Many image output devices share characteristics that lead to two common problems requiring image processing solutions. The first one is that many devices are only capable of rendering a small number of output levels. Example include many printers that can only print at two levels (on–off) for each primary color, and liquid crystal displays that typically can display 16 graylevels. This means that an image must be converted from its "continuous-tone" representation to one that matches the characteristics of the printers or displays.

There are several printing technologies that are suitable for producing hardcopies of images. They include laser printing, inkjet printing, dye sublimation, thermal wax transfer, liquid toner laser transfer, offset printing, and so on. Among these, laser and inkjet printing are perhaps the most popular because of their relatively low cost, high performance, high quality, and excellent availability. Traditional laser and inkjet printers can only print at two levels for each primary color. To print a monochrome image, we first generate a halftone that consists of black and white pixels. Then we put black toners on the paper corresponding to the black pixel locations, and use the white paper background to represent the white pixels. This problem of rendering images in two levels is called *halftoning*.

In color printing, we generally use cyan (C), magenta (M), and yellow (Y) as the three primary colors. We mix dots of

these three colors to give an impression of other colors. Putting all three colors at the same pixel location produces black output, which is called *composite black*. It is observed in practice that composite black is usually a bit dull in appearance. As a result, many color printers are designed to use four toners, namely, cyan, magenta, yellow, and black (K). For these four-toner printers, we have several ways to obtain black output; we can either mix CMY together, or we can use the black toner, or we can mix CMYK together in some proportion. Choices of these are generally dependent on the specific printer design.

Recent advances in both laser and inkjet printing technology have led to the appearance of printers that can print multiple (e.g., 4, 8, or 16) levels per pixel. This opens up a new dimension of image halftoning. The corresponding problem to rendering at multiple levels is called *multitoning*.

Many low-cost displays, while being able to handle "continuous tones," have limited-sized video buffers that necessitate only a subset of the 2^{24} colors to be used. As a result we need to choose the subset of colors (called a *color palette*) to be displayed, and perform the mapping from the original image to the chosen color palette. This problem is called *color quantization*.

Image security has become an important problem in recent years. There has been significant research looking into the problems of *watermarking* and *information embedding* for halftone images. These problems are interesting both in the embedding of information, as well as in attacks to the information embedding algorithms.

Since quantization is a core component of both halftoning and color quantization, we first discuss in Section 2 the basics of scalar quantization. The design of optimum scalar quantizers is presented. In Section 3, we discuss the main approaches of halftoning, and show examples of various types of popular halftoning algorithms. The interplay between halftoning and compression is also considered. In Section 4, we discuss color quantization methods that enable images to be displayed in devices with a limited video buffer. Section 5 discusses recent advances in watermarking and information embedding in halftones.

2 Scalar Quantization

2.1 Quantization Basics

The function of *scalar quantizers* is to convert a quantity to a finite precision representation (such as converting real numbers to integers in digital computing). We can represent scalar quantization as

$$y = Q(x)$$

where $Q(\cdot)$ is the quantization function. Mathematically, a K-level scalar quantizer is specified by the $K+1$ *decision levels* $(d_0, d_1, \ldots, d_K)$ and the K *output levels* $(y_1, y_2, \ldots, y_K)$. Each region from d_{i-1} to d_i is referred to as a quantization bin. Specifically, we can specify a quantization function as

$$Q(u) = y_i \qquad \text{if } d_{i-1} \le u < d_i; \qquad i = 1, 2, \ldots K.$$

An example of the input/output characteristics of a scalar quantizer is shown in Fig. 1.

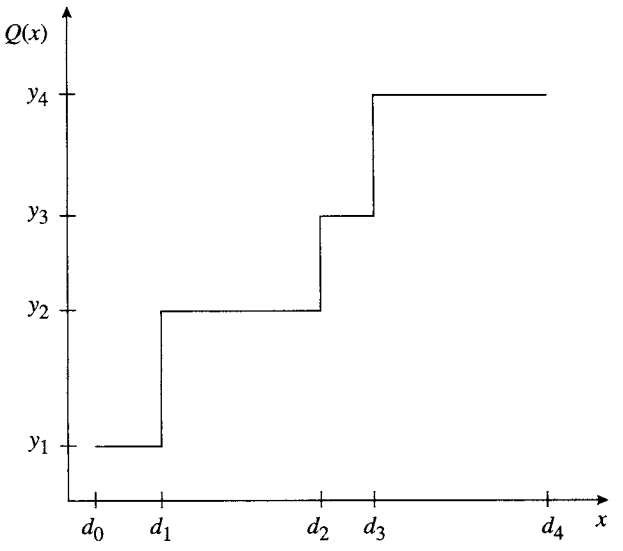


FIGURE 1 Input/output characteristics of a scalar quantizer.

Define the quantizer error $e(u)$ by

$$e(u) = Q(u) - u.$$

This quantity is of crucial importance in describing the characteristics of a scalar quantizer as well as in many design problems associated with quantizers. Let $p_X(x)$ be the probability density function of a random variable X. We can write the expected nth-power (the nth order moment) of the quantization error as

$$\begin{aligned} M_n &= E(X^n) \\ &= \int (e(x))^n p_X(x)\, dx \\ &= \sum_{i=1}^{K} \int_{d_{i-1}}^{d_i} (y_i - x)^n p_X(x)\, dx \end{aligned} \tag{1}$$

Of particular importance in practice is the second order moment, which is usually called the mean squared quantizer error (MSQE). The MSQE is often used in the design of optimum quantizers. In the next section, we will examine such a design.

2.2 Optimum Quantization

The problem of finding an optimum scalar quantizer was first solved by Lloyd in the mid-1950s [1]. Since Lloyd did not publish his results until 1982, the first optimum scalar quantizer design in the open literature was published by Max [2]. Hence, optimum scalar quantizers are usually referred to as Lloyd–Max quantizers.

If X has dynamic range (α, β), we set $d_0 = \alpha$ and $d_K = \beta$ in order to be able to generate an output for all possible values that X can take on. To determine the optimum quantizer, we first differentiate (1) for $n=2$ with respect to d_i $(i = 1, 2, \ldots, K-1)$ to obtain

$$(y_i - x_i)^2 - (y_{i+1} - x_i)^2 = 0,$$

which is equivalent to

$$x_i = \frac{y_i + y_{i+1}}{2} \qquad i = 1, 2, \ldots, K-1. \tag{2}$$

Equation (2) is a necessary condition for optimum scalar quantizers. It says that any decision level between two consecutive quantization bins is the mid-point of the two adjacent output levels. This is usually referred to as the *mid-point condition.*

Second, we differentiate (1) for $n=2$ with respect to y_i $(i = 1, 2, \ldots, K)$ to obtain

$$\int_{d_{i-1}}^{d_i} (y_i - x) p_X(x)\, dx = 0, \qquad i = 1, 2, \ldots, K,$$

which gives

$$y_i = \frac{\displaystyle\int_{d_{i-1}}^{d_i} x p_X(x)\, dx}{\displaystyle\int_{d_{i-1}}^{d_i} p_X(x)\, dx} = E[X | d_{i-1} \le X < d_i] \qquad i = 1, 2, \ldots, K. \tag{3}$$

This is another necessary condition for optimum scalar quantizers. It says that the optimum output level at any quantization bin should be the conditional expectation (centroid) of X within that bin. This is the *centroid condition* for optimum quantizers.

The Lloyd algorithm for optimum quantizer design uses both the mid-point condition and the centroid condition. Essentially one starts with a set of quantizer boundaries, and use the centroid condition to find the optimum output levels based on the quantizer boundaries. From the new output levels, one uses the mid-point condition to determine a new set of quantizer boundaries. These two steps are repeated until convergence is achieved. The nice feature about the Lloyd algorithm is that it works either if one is given a probability density function, or if one is given a set of samples in a signal. In the latter case, one replaces probabilistic averages (expect values) by sample averages. Furthermore, the Lloyd algorithm always converges in practice. The Lloyd algorithm has also been extended for designing vector quantizers [3]. For further details on quantization, the reader is referred to the text [4].

2.3 Distortion Criteria and Image Quantization

There are three common problems that require quantization of either the image pixels values or some transformations of the pixels. They are halftoning, color quantization, and image compression. Image compression will be treated in a different chapter of this book. Here we only consider the problem of image quantization in the context of halftoning and color quantization.

Generally, we formulate the problem as follows: Given an image $f(n_1, n_2)$ of size N_1 by N_2, where each pixel takes on values in the range $\{0, 1, \ldots, \ldots, D-1\}$, we want to quantize each image pixel to K output levels $(K < D)$ so that a distortion criteria is minimized. As an example, we have $K=2$ if we want to halftone the image so that it can be printed using a binary laser or inkjet printer. In this case, the two output levels are 0 and D, respectively. Suppose we use the mean squared error criterion

$$D(x, y) = \sum_{n_1, n_2} (f(n_1, n_2) - b(n_1, n_2))^2$$

where $b(n_1, n_2)$ is the quantized output. Since the output levels are fixed, the mid-point condition in optimum scalar quantizers gives

$$b(n_1, n_2) = Q(f(n_1, n_2)) = \begin{cases} D & \text{if } f(n_1, n_2) \ge D/2 \\ 0 & \text{otherwise.} \end{cases}$$

This is constant thresholding of the image where the threshold is the mid-point of the dynamic range. Using the boats image in Fig. 2 as an example, we show in Fig. 3 the result of constant thresholding.

Although the quantizer is optimum with respect to the mean squared error distortion criterion, the resulting output quality is rather poor. This is due to the fact that mean squared error does not reflect the characteristics of human perception on images. A much better error criterion, often used in image processing applications, is the visual mean squared error criterion

$$\mathcal{E} = E\left[(v(n_1, n_2) * (b(n_1, n_2) - f(n_1, n_2)))^2\right] \tag{4}$$

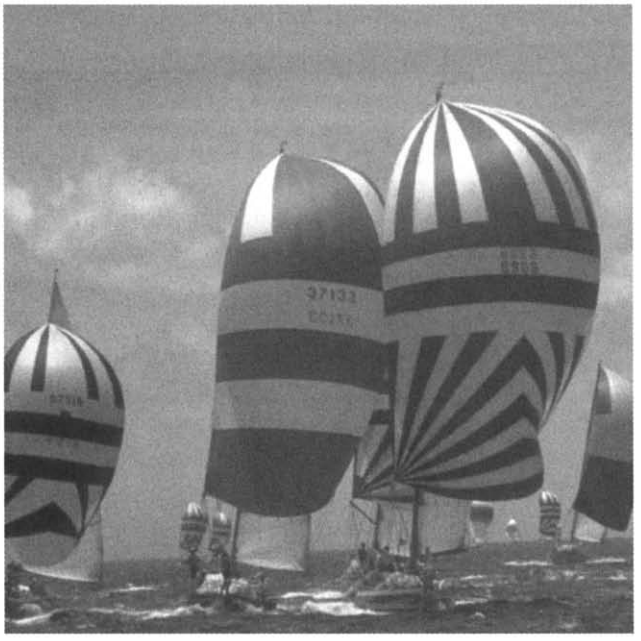


FIGURE 2 Original boats image.

FIGURE 3 Boats image thresholded to two levels using a constant threshold for each primary color plane.

where $*$ denotes convolution, $E(\cdot)$ is the expectation (averaging) operator, and $v(n_1, n_2)$ is the impulse response of a filter that approximates the spectral response of the human visual system. Notice that a neighborhood operation is involved in the visual mean squared error due to the convolution operation. This is consistent with the notion of halftoning that the human perception system filters (mixes) the dots in halftone images to give an impression of graylevel.

3 Halftoning

Halftoning is a process where a continuous tone image of, say 256 levels, is converted into an output image of 2 levels so that the original and the halftone images appear similar when observed from a distance. The corresponding problem of generating output images of multiple levels is called multitoning. The main application of halftoning and multitoning is printing and in displaying images with limited bit-depth displays such as liquid crystal displays. Since the techniques for multi-toning are straightforward extensions of halftoning (rendering at two levels), we will only focus on halftoning in the rest of this section.

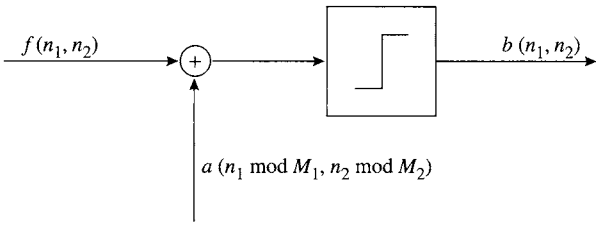
The three popular haltoning techniques are ordered dithering, error diffusion, and optimization techniques. Generally, ordered dithering requires the lowest computational complexity, while optimization techniques are the most computationally intensive. On the other hand, well-designed optimization techniques tend to generate halftones of the best quality. All these three halftoning techniques will be described in this chapter.

Before we go into the specifics of halftoning algorithms, we comment that the shape of a basic output unit from a real life printer is a very important factor to the ultimate quality of the output. Normally in image processing, we use rectangular grids and hence each pixel is considered to have a rectangular shape. In printing, the shape of each output pixel (printing dot) is *not* rectangular in shape. Furthermore, the dot shapes between laser and inkjet printers have very different characteristics. Inkjet dots can be well represented by a round shape that has an area coverage larger than the rectangular pixel size. Halftoning considerations using this oversized round dot model can be found in [5]. Output dot shapes of laser printers is round but the intensity transition from the edge to the center of the dot is more gradual. If we cut a cross section across a laser printer dot and draw the intensity profile, we have a curve that resembles a bell shape. More detailed discussions of laser dot shapes can be found in [6].

For simplicity, we describe in the following sections halftoning algorithms for monochrome images (black and white halftoning). Halftoning of color images can be done on a plane by plane basis. We give examples of color halftones using such an approach. Readers who are interested in plane-dependent halftoning, where the color planes are halftoned jointly, is referred to the literature [7, 8].

3.1 Ordered Dithering

Consider an image $f(n_1, n_2)$ of size N_1 by N_2 with pixel values in the range $\{0, 1, \ldots, D-1\}$. We want to generate a halftone $b(n_1, n_2)$ using ordered dithering. In this method, we use an



113	80	96	105	142	175	96	142
51	0	1	88	88	254	1	88
14	3	7	72	72	242	7	72
39	26	63	121	121	217	63	121
138	171	154	146	146	84	154	146
196	254	246	163	163	0	246	163
221	237	229	179	179	5	229	179
204	213	188	130	130	32	188	130

size M_1 by M_2

FIGURE 4 Pixel-by-pixel operation of ordered dithering. Here the values of the dither matrix elements correspond to images with 8-bit intensity representation.

array $a(n_1, n_2)$ of size M_1 by M_2, typically called a *dither matrix*, where the elements are threshold levels. Each element of the dither matrix is an integer that takes value in the range $\{0, 1, \ldots, D-1\}$. To generate the output, we perform an element-by-element thresholding of the image pixels using the elements of the dither matrix as shown in Fig. 4.

In most applications, we have $M_1 \ll N_1$ and $M_2 \ll N_2$. That is, the size of the dither matrix is much smaller than the image size. Thus we periodically replicate $a(n_1, n_2)$ to give $\tilde{a}(n_1, n_2) = a(n_1 \bmod M_1, n_2 \bmod M_2)$ so that the entire image is covered. Then the halftoning operation can be represented mathematically as

$$b(n_1, n_2) = Q_{\tilde{a}(n_1,n_2)}(f(n_1, n_2)) = \begin{cases} 1 & \text{if } f(n_1, n_2) \geq a(n_1 \bmod M_1, n_2 \bmod M_2) \\ 0 & \text{otherwise.} \end{cases}$$

Here the output "1" represents a white pixel (do not print a dot) and "0" represents a black pixel (print a dot).

It is evident that the actual values of the dither matrix elements will have a great influence on the quality of the output halftone. It can be easily verified that the following 8×8 matrix

$$\begin{bmatrix} 113 & 80 & 96 & 105 & 142 & 175 & 159 & 150 \\ 51 & 0 & 1 & 88 & 200 & 254 & 250 & 167 \\ 14 & 3 & 7 & 72 & 225 & 242 & 233 & 183 \\ 39 & 26 & 63 & 121 & 208 & 217 & 192 & 134 \\ 138 & 171 & 154 & 146 & 117 & 84 & 101 & 109 \\ 196 & 254 & 246 & 163 & 57 & 0 & 2 & 92 \\ 221 & 237 & 229 & 179 & 20 & 5 & 10 & 76 \\ 204 & 213 & 188 & 130 & 45 & 32 & 67 & 125 \end{bmatrix} \quad (5)$$

will give halftones where the dots tend to be clustered within 4×4 cells. On the other hand, the matrix

$$\begin{bmatrix} 4 & 236 & 60 & 220 & 8 & 224 & 48 & 208 \\ 132 & 68 & 188 & 124 & 136 & 72 & 176 & 112 \\ 36 & 196 & 20 & 252 & 40 & 200 & 24 & 240 \\ 164 & 100 & 148 & 84 & 168 & 104 & 152 & 88 \\ 12 & 228 & 52 & 212 & 0 & 232 & 56 & 216 \\ 140 & 76 & 180 & 116 & 128 & 64 & 184 & 120 \\ 44 & 204 & 28 & 244 & 32 & 192 & 16 & 248 \\ 172 & 108 & 156 & 92 & 160 & 96 & 144 & 80 \end{bmatrix} \quad (6)$$

tends to generate halftones where the dots are dispersed. A very popular class of dispersed dither matrices is called Bayer dither [9], which has been shown to have certain optimality properties. Examples of clustered and dispersed halftones as well as their power spectra are shown in Figs. 5 and 6. It is evident from both the halftones and their spectra that these halftones exhibit strong periodicity induced by the size of the dither matrices.

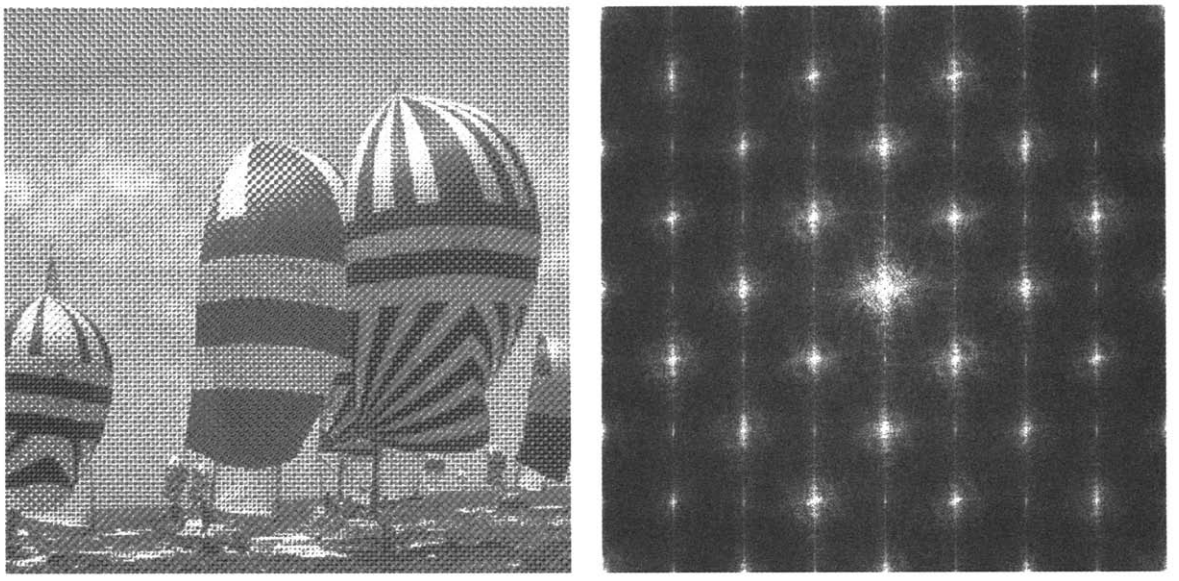
The major advantage of ordered dithering is its simplicity. To generate an output halftone pixel, only one thresholding or comparison operation is needed. As a result, ordered dithering is very popular in low cost printing applications. Experimental evidence has shown that it is generally preferred to use dispersed halftones in inkjet printers for improved smoothness, and clustered dot halftones in laser printers for better stability.

Traditionally, ordered dithering uses dither matrices of size between 8×8 to about 32×32. Recently, a class of ordered dither called *blue noise dithering* has been proposed that uses dither matrices of large sizes (e.g., 128×128 or larger) to generate halftones that are of higher quality than those of traditional ordered dithering with small dither matrices. These *blue noise dither matrices* are typically designed using iterative methods [10].

3.2 Error Diffusion

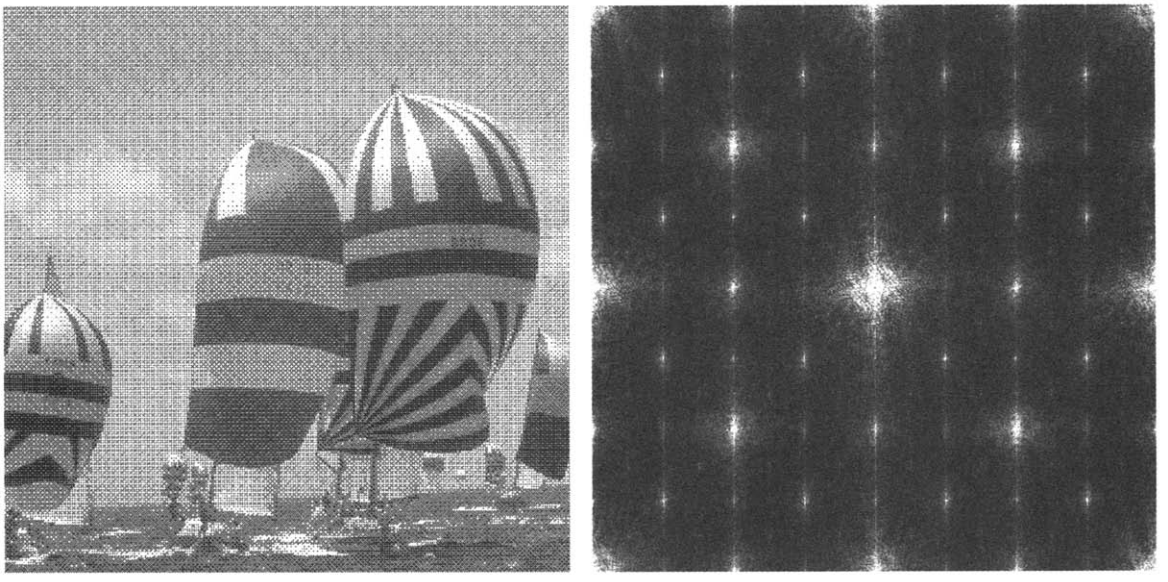
Error diffusion [11] is an excellent method for generating high quality halftones that are particularly suitable for low to medium resolution devices. A block diagram showing an error diffusion system is given in Fig. 7. It turns out [12] that error diffusion is the two-dimension equivalent of sigma–delta modulation (also called delta–sigma modulation) [13], which was developed for performing high resolution analog-to-digital conversion using a one-bit quantizer embedded in a feedback loop.

To apply the error diffusion algorithm in producing a halftone $b(n_1, n_2)$, we need to scan the input image $f(n_1, n_2)$ in some fashion. One of the most popular strategies is raster scanning, where we scan the image row by row from the top to the bottom, and within each row from the left to the right.



(a) (b)

FIGURE 5 Boats image (a) halftoned using clustered dot dither at 150 dots per inch, and (b) the halftone power spectrum.



(a) (b)

FIGURE 6 Boats image (a) halftoned using Bayer dither [9] at 150 dots per inch, and (b) the halftone power spectrum.

At each pixel location, we perform the operations

$$u(n_1, n_2) = f(n_1, n_2) - \sum_{m_1, m_2} h(m_1, m_2) e(n_1 - m_1, n_2 - m_2) \quad (7)$$

$$b(n_1, n_2) = Q(u(n_1, n_2)) = \begin{cases} 1 & \text{if } u(n_1, n_2) \geq 0.5 \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

$$e(n_1, n_2) = b(n_1, n_2) - u(n_1, n_2) = Q(u(n_1, n_2)) - u(n_1, n_2) \quad (9)$$

In the signal processing literature, $u(n_1, n_2)$ is called either the *state variable* of the system or the *modified input.*

Notice that we need to buffer the binary quantizer error values $e(n_1, n_2)$ because of the convolution operation in (7). The extent of the buffer required is dependent on the support of the *error diffusion kernel* $h(m_1, m_2)$. The filter $h(m_1, m_2)$ generally has a low pass characteristic, and the coefficients often satisfy

$$\sum_{(m_1, m_2)} h(m_1, m_2) = 1.$$

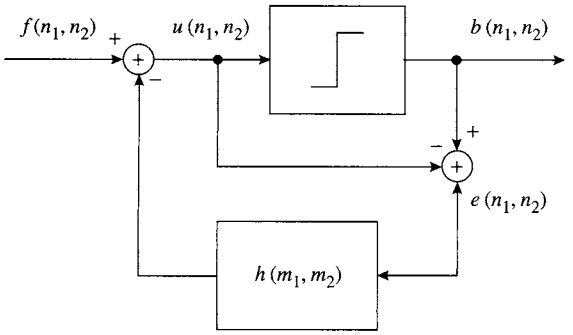


FIGURE 7 A block diagram of the error diffusion system.

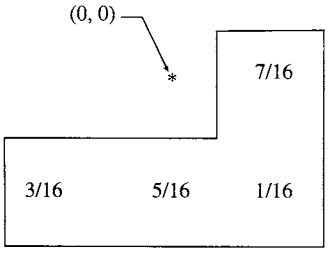


FIGURE 8 The Floyd–Steinberg error diffusion filter coefficients.

A very popular kernel suggested by Floyd and Steinberg [11] is shown in Fig. 8, with

$$h(0, 1) = 7/16, \quad h(1, -1) = 3/16,$$
$$h(1, 0) = 5/16, \quad h(1, 1) = 1/16.$$

This kernel consists of four coefficients, and hence the complexity of the error diffusion algorithm is quite mild. Two other popular error diffusion kernels are proposed by Jarvis, Judice and Ninke [14] and Stucki [15]. A Floyd–Steinberg error diffused halftone and its power spectrum is shown in Fig. 9.

Using (7) and (9), we can write

$$b(n_1, n_2) = f(n_1, n_2) + \sum_{m_1, m_2} g(m_1, m_2)\, e(n_1 - m_1, n_2 - m_2) \tag{10}$$

where

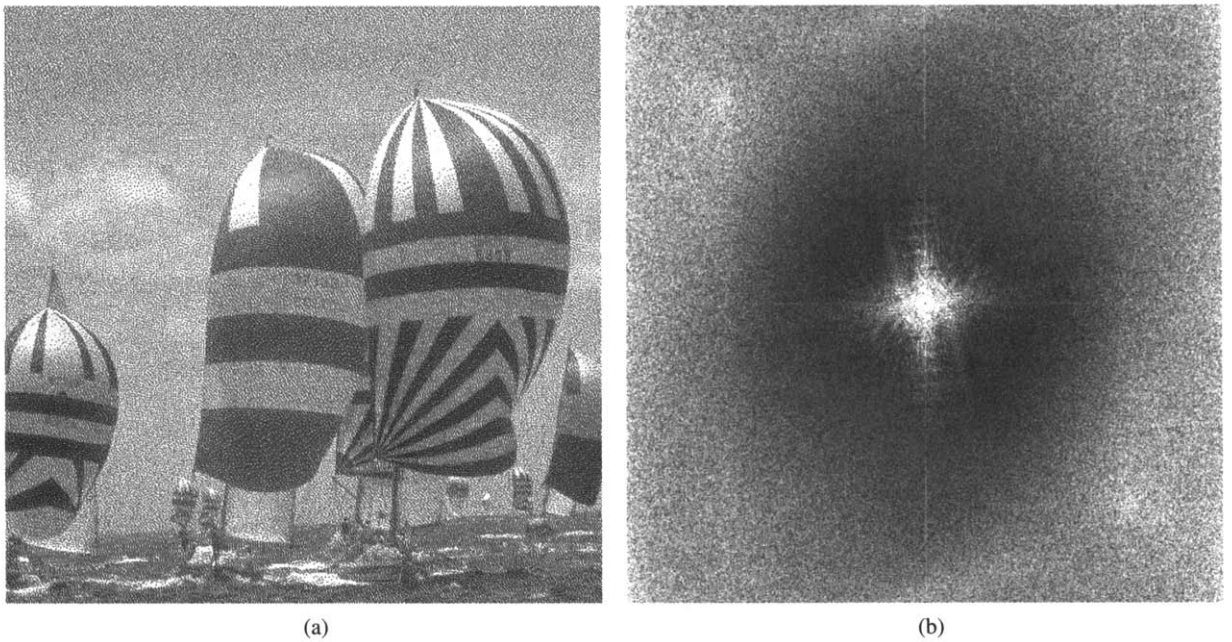
$$g(m_1, m_2) = \delta(m_1, m_2) - h(m_1, m_2),$$

and $\delta(m_1, m_2)$ is the Kronecker delta. Using the Floyd Steinberg filter kernel as an example, we have

$$g(m_1, m_2) = \begin{cases} 1 & (m_1, m_2) = (0, 0) \\ -7/16 & (m_1, m_2) = (0, 1) \\ -3/16 & (m_1, m_2) = (1, -1) \\ -5/16 & (m_1, m_2) = (1, 0) \\ -1/16 & (m_1, m_2) = (1, 1) \\ 0 & \text{otherwise.} \end{cases}$$

A nice interpretation of (10) is that the output is the sum of the input and a filtered version of the quantizer noise.

To calculate the power spectral density of $b(n_1, n_2)$, we need to calculate its autocorrelation function. This calculation



(a) (b)

FIGURE 9 Boats image (a) halftoned using Floyd-Steinberg error diffusion [11] at 150 dots per inch, and (b) the halftone power spectrum.

requires the autocorrelation function of $e(n_1, n_2)$ and the cross correlation function $E[f(n_1, n_2)e(n_1 + m_1, n_2 + m_2)]$, both as function of the statistical properties of the input image $f(n_1, n_2)$. This turns out to be a very difficult task because of the nonlinear nature of the system induced by the binary quantizer. A general solution for the two-dimensional case is still an open problem, although solutions to the one-dimensional case (sigma–delta modulation) have already been found [16, 17].

Despite the difficulty in an exact mathematic analysis of error diffusion, we can compute the spectrum of error diffused halftones empirically. An example of such a power spectrum is shown in Fig. 9. It is of interest to note that the noise of error diffusion is primarily in the high frequency region, which agrees with the intuition that the quantization noise in a high quality halftone should be mostly located in the high frequency region.

Although Floyd–Steinberg error diffusion generally produces high quality output, it also generates undesirable artifacts such as "worms" and undesirable patterns. Consequently there has been a large number of reports in the literature focusing on the improvement of error diffusion. These techniques can generally be categorized into two classes. The first class of techniques focuses on "breaking up" the undesirable output patterns' of error diffusion. This includes using alternative scanning strategies [18, 19] and injecting random noises into the error diffusion system [18, 20].

Another class of techniques attempts to optimize the error diffusion filter with respect to a distortion criterion. To this end, one can use the frequency weighted mean squared error in (4). Substituting (10) into (4), we have

$$\mathcal{E} = E\big[(v(n_1, n_2) * g(n_1, n_2) * e(n_1, n_2))^2\big].$$

It is evident that $\mathcal{E}$ is a function of $h(n_1, n_2)$. Hence at least in principle, we can find an optimum error diffusion kernel $h(n_1, n_2)$ to minimize $\mathcal{E}$.

As discussed earlier, error diffusion is a highly non-linear system where $e(n_1, n_2)$ depends on $f(n_1, n_2)$ and the structure of the feedback loop in a very complicated way. This makes the optimization problem very difficult. In the literature, there are reports in solving this optimization problem either by making certain assumptions on the quantizer noise [21], using the LMS algorithm in adaptive signal processing to solve the optimization problem [22], or using an iterative optimization approach [23].

3.3 Optimization Halftoning Techniques

Consider a distortion criterion $d(f(n_1, n_2), b(n_1, n_2))$ between a continuous tone image $f(n_1, n_2)$ and a halftone $b(n_1, n_2)$. Given a specific $f(n_1, n_2)$, we can find $b(n_1, n_2)$ that minimizes the average distortion $E[d(f(n_1, n_2), b(n_1, n_2))]$. There are many methods for finding a signal to minimize a certain cost function. Here, any specific method for minimizing $E[d(f(n_1, n_2), b(n_1, n_2))]$ will give rise to a halftoning algorithm.

Suppose we consider the frequency weighted squared error

$$d(f(n_1, n_2), b(n_1, n_2)) = (v(n_1, n_2) * (b(n_1, n_2) - x(n_1, n_2)))^2.$$

One way to find a halftone is to use a *greedy minimization* approach as follows: We scan the image $f(n_1, n_2)$ in raster scan fashion. At each pixel location (n_1, n_2), we determine the binary output pixel as

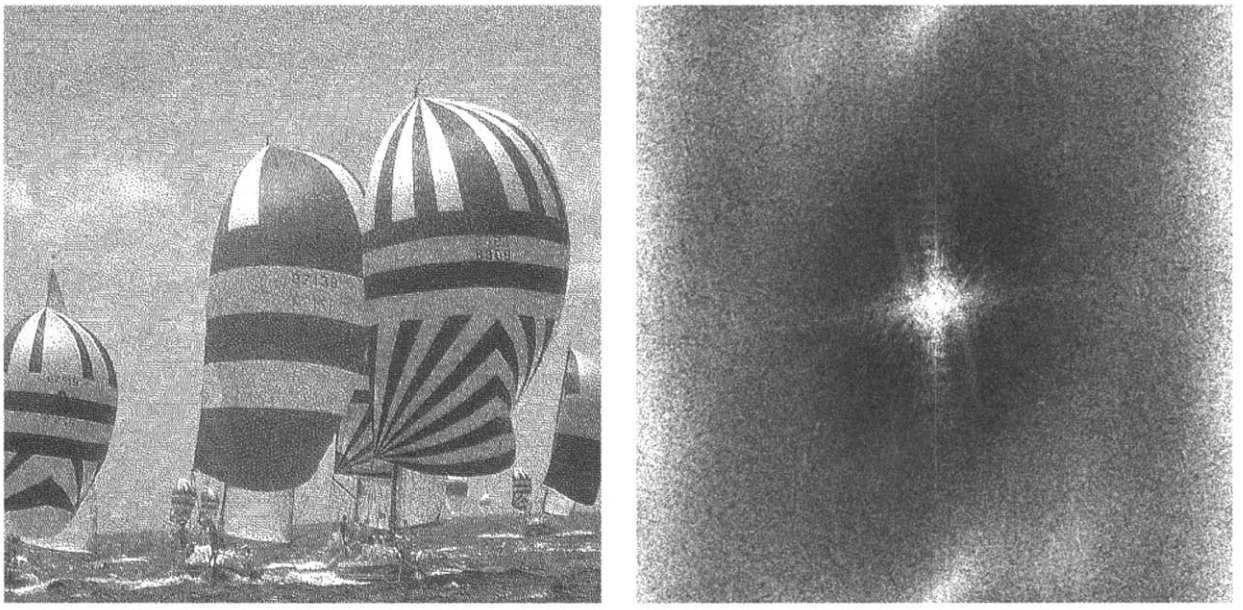
$$b(n_1, n_2) = \underset{b(n_1,n_2)\in\{0,1\}}{\arg\min}\ (v(n_1, n_2) * (b(n_1, n_2) - x(n_1, n_2)))^2.$$

Since there are only two possible values of $b(n_1, n_2)$ for each (n_1, n_2), we can compute $d(f(n_1, n_2), b(n_1, n_2))$ for both values, and then pick the one that results in a smaller distortion. As long as $v(n_1, n_2)$ has a causal support with respect to the scanning strategy, this minimization procedure can be performed very easily.

In general, greedy minimization does not generate output halftones of satisfactory quality. Since the greedy approach performs the minimization on a pixel by pixel basis with respect to a scanning strategy, there is no guarantee that it actually minimizes the overall average distortion. That is, any decision made by the greedy approach at a local pixel location will affect the local distortion of "future" pixels. Experimental results comparing greedy minimization and more sophisticated techniques indeed show that greedy minimization is far from optimal [24].

One method for generating halftones of excellent quality is direct binary search (DBS) [25]. The DBS algorithm goes through the image in multiple passes. In the kth pass, one uses the output halftone of the previous pass $b_{k-1}(n_1, n_2)$ and the original continuous tone image $f(n_1, n_2)$ to obtain an improved output halftone $b_k(n_1, n_2)$. In the first pass, one needs an initial halftone $b_0(n_1, n_2)$ that can be generated, for example, using greedy optimization. In each pass over the image, the DBS algorithm attempts to improve the quality of the output using the following strategy: At each pixel location (n_1, n_2), one considers possible improvement to the output halftone by flipping (inverting) the current halftone pixel or by swapping the current halftone pixel with one of its eight neighbors. The halftone pixel $b_k(n_1, n_2)$ at location (n_1, n_2) is chosen to be the one that results in the smallest distortion among the 10 possibilities (center pixel unchanged, center pixel inverted, center pixel swapped with one of its neighbors). Efficient methods for implementing the DBS algorithm have been suggested [25].

Other excellent methods for optimization based halftoning include tree coding [24], least square solution [26], combined halftoning-compression [27] and many others. Figure 10



(a) (b)

FIGURE 10 Boats image (a) halftoned using a tree coding algorithm [24] at 150 dots per inch, and (b) the halftone power spectrum.

shows a halftone generated using the tree coding algorithm [24] and the corresponding halftone power spectrum.

3.4 Halftoning and Compression

The interplay between halftoning and compression is an interesting and important topic [28]. This is important for both storage purposes (e.g., memory insider a printer) and for communications reasons (e.g., remote printing, fax). Consider a color page of US letter size (8.5 inches by 11 inches). Assuming a print area of 8 by 10 inches, a printer of 600 dots per inch resolution, and binary printing at 3 bits per pixel (i.e., 2 levels per color plane), we need 10.8 Mbytes to store the halftone of just one page. It is obvious why compression is an important issue. There are two general categories of solutions related to compression and halftoning.

The first category of techniques attempt to directly compress the halftones. Using the remote printing example, the continuous tone image is halftoned and then compressed at the source. The bit stream is transmitted. At the receiver, the bit stream is decoded and the resulting halftone is finally printed. Examples of work toward this area include lossless compression of halftones [29], lossy compression of halftones [30–32], joint halftoner-compression design [27, 33], and entropy constrained halftoning [24, 34].

Another category of solutions focuses on the optimization of continuous tone image compression taking into account the characteristics of halftoning. In the remote printing example, a continuous image is compressed at the source, and the compressed bit stream is transmitted. At the receiver, the bit stream is decompressed. In this case, the receiver needs to halftone the decoded image before sending it for printing. Examples of work in this area include optimized subband coders [35] and optimized JPEG coder [36], both designed with respect to the spectral characteristics of halftoning.

4 Color Quantization

Consider the problem of displaying a color image $f(n_1, n_2)$ where each pixel is represented by K bits (e.g., $K=24$), i.e., assuming 3 primary color planes with $K/3$ bits precision for each primary color. Suppose the display is only capable of handling L bits ($L < K$) or 2^L different colors. This restriction is usually due to memory size constraint in the video display buffer. To display an image using such a device, we convert $f(n_1, n_2)$ to an image $c(n_1, n_2)$ where each pixel $c(n_1, n_2)$ is described by an L-bit index. In practice, we construct a look-up-table that maps each index to an RGB color pixel value. The problem of color quantization is to design the color palette (hence the look-up table), and to determine how each pixel is to be mapped to the colors in the palette.

Although this problem may appear to be similar to halftoning, there are significant differences. First, the number of levels used in printing is small (usually between 2 and 16), whereas in color displays, we can usually have 256 different colors or more. Second, the colors (typically CMY or CMYK) of the inks and toners for color printing is fixed once a printer design is completed. In display, we can choose the optimum colors for displaying an image. This collection of colors is called a *color palette*. Third, the primary colors in

printing are image independent. On the other hand, the color palette in color quantization is often image dependent.

A very simple color quantization method is uniform quantization. For example, we can assign 3 bits each to the red and green channels, and 2 bits to the blue channel. We then use a uniform quantizer for each color channel. It is perhaps obvious that this method will generally give poor output results. In the rest of this section, we consider color quantization methods that can produce far better results.

4.1 Color Palette Design

There are many ways to design a color palette. The problem is similar to the optimum quantizer design problem. There are several quantization techniques, including sequential scalar quantization and vector quantization, that can be applied for designing color palettes [37–39]. A very popular method for designing image dependent palettes is the median cut algorithm. It is a relatively simple algorithm both conceptually and computationally. At the same time, it produces very good results.

The median cut algorithm was originally reported in a thesis in 1980. It was subsequently published in a paper in 1982 [40]. The philosophy behind the algorithm is that each color within the color palette is used for representing approximately the same number of pixels in the original image $f(n_1, n_2)$.

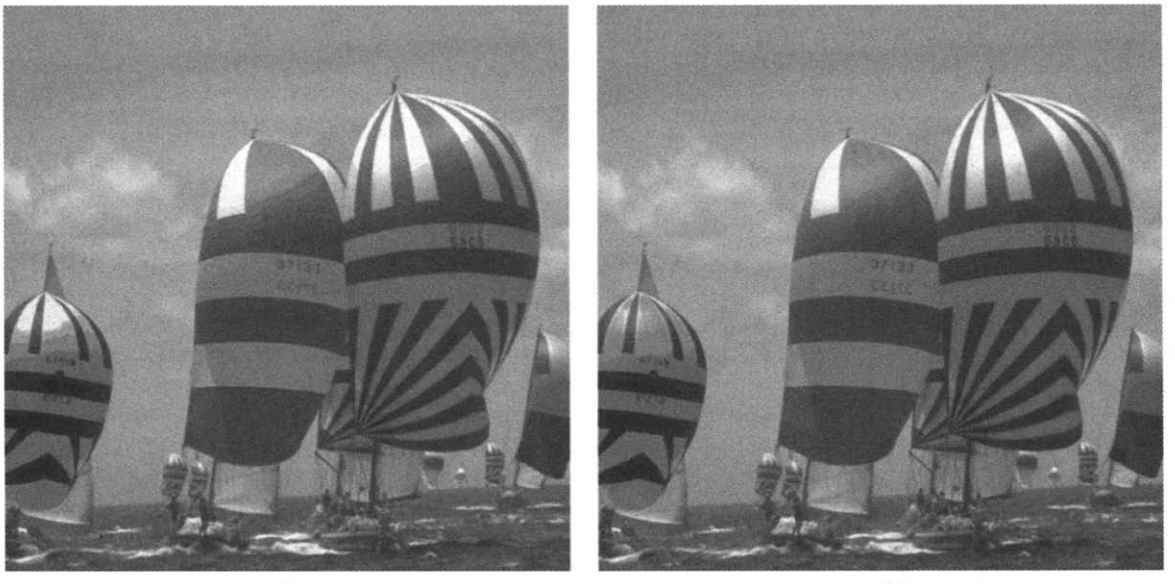
Consider a 24-bit image $f(n_1, n_2)$. We first compute the color histogram of the image. For a 24-bit image, there can be a maximum of 16 million different colors. A histogram with that many entries could be difficult to handle. Hence Heckbert proposed to "pre-quantize" the original image into a 15-bit representation using a 5-bit uniform quantizer for each color channel. Hence there are 32,768 colors after this step.

From the color histogram, one finds a rectangular box that tightly encloses the available colors within the image. The idea of the median cut algorithm is to partition the box recursively until we have 256 rectangular boxes. At each step of the algorithm, we choose the box that encloses the largest number of pixels to be partitioned. To perform the partition of a box, Heckbert proposed to make a cut along the coordinate that spans the largest range. The specific cut is made at the median point so that approximately equal number of pixels fall into each half of the sub-divided boxes. This procedure is recursively applied until we have 256 boxes.

4.2 Error Diffusion Approach

Once a color palette is designed, the next step is to map each color pixel to one of the colors in the color palette. Nearest neighbor mapping is a straightforward approach, where we map each color pixel to the nearest color in the color palette. This is equivalent to fixed vector quantization using the color palette as the code book. It turns out that this approach generally produces output images where false contouring is quite visible, particularly when the size of the color palette is small (e.g., 256).

A better approach is to apply error diffusion for generating the output image. Here we use a system as shown in Fig. 7, except that we replace the one bit quantizer by a vector quantizer with the color palette as the code book. Experimental results have shown that this approach generates output images that are significantly better than nearest neighbor



(a) (b)

FIGURE 11 Boats image color quantized to 256 colors using the median cut algorithm [40] with (a) nearest neighbor mapping, and (b) error diffusion.

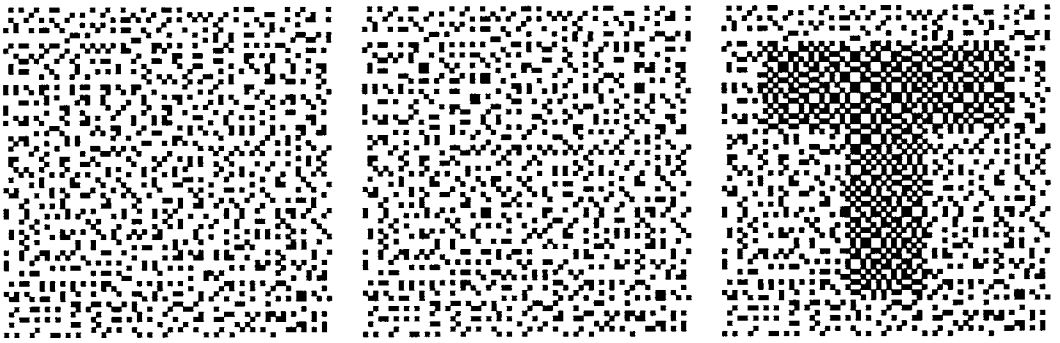


FIGURE 12 An illustration of the idea of binary watermarking.

mapping. Figure 11(a) and (b) show color quantized images using the median cut algorithm with nearest neighbor mapping and error diffusion, respectively. False contouring is quite obvious using the nearest neighbor mapping approach.

5 Halftone Watermarking and Embedding

Early work on the embedding of information into binary images can be traced to visual cryptography. Shamir [41] suggests a method that embeds a watermark into two binary images (the carriers). The requirement is that the resolution of the watermark should be no more than one half the resolution of the carriers. Each pixel of the binary secret image (the watermark) is broken up into two collection of subpixels, and each collection is printed as shown in the left and the center images, respectively, in Fig. 12. When each of these prints are viewed independently, the black and white pixels appear to be randomly positioned. When the two prints are superimposed on each other, the human observer will see the union of the two sets of pixels as shown in the right image of Fig. 12, and the secret image appears.

There has been consideration of embedding one halftone into another halftone. The embedding of lower a bit-depth halftone into a higher bit-depth halftone of the same image, as well as embedding a monochrome halftone into a color halftone of the same image, have been reported [42]. More recently, an algorithm has been proposed to embed a halftone $a(n_1, n_2)$ of a continuous tone image $f(n_1, n_2)$ into a halftone of a second continuous tone image $g(n_1, n_2)$ [43], resulting in a composite halftone $b(n_1, n_2)$. The method uses a multiresolution error diffusion algorithm [22] with a suitably generated random key to embed $a(n_1, n_2)$ into $b(n_1, n_2)$ while $b(n_1, n_2)$ is being generated from $g(n_1, n_2)$. Figure 13 is an example showing the composite halftone $b(n_1, n_2)$ that appears to be the halftone of only the boat image $g(n_1, n_2)$. However, when a user extracts a subset of pixels from $b(n_1, n_2)$ using an appropriate key, the halftone $a(n_1, n_2)$ of the mandrill image appears. It can be shown that the method is secure in the sense that if user does not have the correct key, the probability of extracting a recognizable portion of the image is on the order of 2^{-117}.

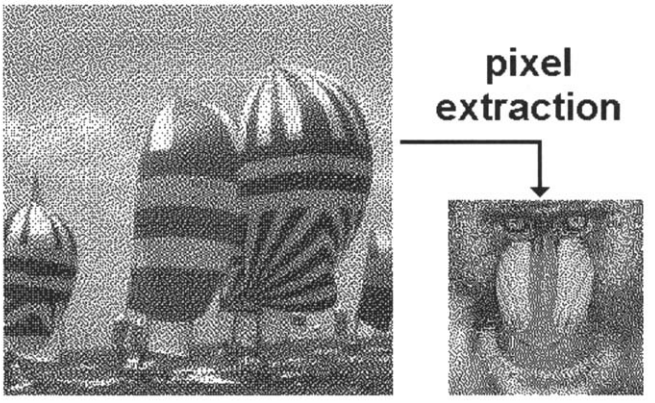


FIGURE 13 An example showing a halftone of mandrill embedded into a halftone of the boat image [43].

Information embedding and watermarking halftones is a very interesting problem. The interested readers are refereed to [28] for further information.

6 Conclusion

We have discussed in this chapter the basics of scalar quantization, halftoning, color quantization, halftone watermarking and embedding. We reviewed some constraints imposed by printers and displays, which motivate the problems of halftoning and color quantization. The basics of optimum scalar quantization and its application to image quantization are presented. In halftoning we described and compared the common approaches of halftoning. Examples are given for several representative halftoning algorithms. We described and showed examples of the median cut algorithm for color quantization. We also described and showed examples of halftone watermarking and embedding. The readers who are interested in further probing into these areas are referred to the bibliography as well as the references therein.

References

[1] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, 28, 129–137, March 1982.

[2] J. Max, "Quantizing for minimum distortion," *IRE Transactions on Information Theory*, 6, 7–12, March 1960.

[3] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, COM-28, 84–95, (1980).

[4] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Boston: Kluwer Academic Publishers, 1992.

[5] T. N. Pappas, C. K. Dong, and D. L. Neuhoff, "Measurement of printer parameters for model-based halftoning," *Journal of Electronic Imaging*, 2, 193–204, July 1993.

[6] Q. Lin and J. Wiseman, "Impact of electrophotographic printer dot modeling on halftone image quality," in *SID Digest of Technical Papers*, Seattle WA, May 1993, 147–150.

[7] J. R. Sullivan, R. L. Miller, and T. J. Wetzel, "Color digital halftoning with vector error diffusion," United States Patent 5070413, December 1991.

[8] Q. Lin and J. Allebach, "Color FM screen design using DBS algorithm," in *Proceedings of IS&T/SPIE Symposium on Electronic Imaging*, (San Jose, CA), January 1998.

[9] B. E. Bayer, "An optimum method for two-level rendition of continuous-tone pictures," in *Proceedings of IEEE International Conference in Communications*, 1973, 26.11–26.15.

[10] R. Ulichney, "The void-and-cluster method for dither array generation," in *Proceedings of SPIE*, 1913, February 1993, 332–343.

[11] R. Floyd and L. Steinberg, "An adaptive algorithm for spatial grey scale," in *SID International Symposium, Digest of Technical Papers*, 1975, 36–37.

[12] D. Anastassiou, "Error diffusion coding for A/D conversion," *IEEE Transactions on Circuits and Systems*, CAS-36, 1175–1186, September 1989.

[13] H. Inose, Y. Yasuda, and J. Murakami, "A telemetering system by code modulation — Δ–Σ modulation," *IRE Transactions on Space Electronics and Telemetry*, SET-8, 204–209, September 1962.

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

[15] P. Stucki, "MECCA—a multiple-error correcting computation algorithm for bilevel image hardcopy reproduction," IBM Research Laboratory, Zurich, Switzerland, Tech. Rep. RZ1060, 1981.

[16] P. W. Wong and R. M. Gray, "Sigma-delta modulation with i.i.d. Gaussian inputs," *IEEE Transactions on Information Theory*, IT-36, 784–798, July 1990.

[17] W. Chou, P. W. Wong, and R. M. Gray, "Multi-stage sigma-delta modulation," *IEEE Transactions on Information Theory*, IT-35, 784–796, July 1989.

[18] R. A. Ulichney, "Dithering with blue noise," *Proc. IEEE*, 76, 56–79, Jan 1988.

[19] I. H. Witten and M. Neal, "Using Peano curves for bilevel display of continuous-tone images," in *Proceedings of IEEE CG&A*, May 1982, 47–52.

[20] K. T. Knox and R. Eschbach, "Threshold modulation in error diffusion," *Journal of Electronic Imaging*, 2, 185–192, July 1993.

[21] B. W. Kolpatzik and C. A. Bouman, "Optimized error diffusion for image display," *Journal of Electronic Imaging*, 1, 277–292, July 1992.

[22] P. W. Wong, "Adaptive error diffusion and its application in multiresolution rendering," *IEEE Trans. Image Processing*, 5, 1184–1196, (1996).

[23] P. W. Wong and J. P. Allebach, "Optimum error diffusion kernel design," in *Proceedings of SPIE/IS&T Symposium on Electronic Imaging*, invited paper, San Jose, CA, January 1997.

[24] P. W. Wong, "Entropy constrained halftoning using multipath tree coding," *IEEE Trans. Image Processing*, 6, 1567–1579, Nov. 1997.

[25] T. J. Flohr, C. B. Atkins, and J. P. Allebach, "Can DBS ever be a practical halftoning technique?" in *SID Digest of Technical Papers*, Seattle WA, May 1993, 143–146.

[26] T. N. Pappas and D. L. Neuhoff, "Least-square model-based halftoning," in *Proceedings of SPIE*, 1666, San Jose, CA, 1992, 165–176.

[27] R. A. Vander Kam, P. A. Chou, and R. M. Gray, "Combined halftoning and entropy-constrained vector quantization," in *SID Digest of Technical Papers*, Seattle, WA, May 1993, 223–226.

[28] P. W. Wong and N. Memon, "Image processing for halftones: focusing on inverse halftoning, compression, and watermarking," *IEEE Signal Processing Magazine*, 59–70, July 2003.

[29] ISO/IEC International Standard 11544, "Coded representation of bi-level and limited-bits-per-pixel grayscale and color images," February 1993.

[30] S. Forchhammer and K. S. Jensen, "Data compression of scanned halftone images," *IEEE Trans. Commun.*, 42, 1881–1893, Feb. 1994.

[31] R. A. Vander Kam and R. M. Gray, "Lossy compression of clustered-dot halftones using subcell prediction," in *Proceedings of Data Compression Conference*, Snowbird, UT, 1995, 112–121.

[32] M. Y. Ting and E. A. Riskin, "Error diffused image compression using a halftone-to-grayscale decoder and predictive pruned tree-structured vector quantization," *IEEE Trans. Image Processing*, 3, 854–858, November 1994.

[33] R. A. Vander Kam, P. A. Chou, E. A. Riskin, and R. M. Gray, "An algorithm for joint vector quantizer and halftoner design," in *Proceedings of ICASSP*, San Francisco, CA, March 1992.

[34] P. W. Wong and H. Nguyen, "Entropy constrained error diffusion," in *SID 95 Digest of Technical Papers*, Orlando, FL, May 1995, 905–908.

[35] D. L. Neuhoff and T. N. Pappas, "Perceptual coding of images for halftone display," *IEEE Trans. Image Processing*, 3, 341–354, July 1994.

[36] R. A. Vander Kam, P. W. Wong, and R. M. Gray, "JPEG-compliant perceptual coding for a grayscale image printing pipeline," *IEEE Trans. Image Processing*, 8, 1–14, Jan1999.

[37] M. T. Orchard and C. A. Bouman, "Color quantization of images," *IEEE Transactions on Signal Processing*, 39, 2677–2690, December 1991.

[38] R. Balasubramanian, C. A. Bouman, and J. P. Allebach, "Sequential scalar quantization of color images," *Journal of Electronic Imaging*, 3, 45–59, January 1994.

[39] B. W. Kolpatzik and C. A. Bouman, "Optimized universal color palette design for error diffusion," *Journal of Electronic Imaging*, 4, 131–143, January 1995.

[40] P. Heckbert, "Color image quantization for frame buffer display," *Computer Graphics*, 16, 3, 297–307, April 1982.

[41] A. Shamir, "Method and apparatus for protecting visual information with printed cryptographic watermarks," United States Patent 5488664, January 1996.

[42] J. R. Goldschneider, E. A. Riskin, and P. W. Wong, "Embedded multilevel error diffusion," *IEEE Trans. Image Processing*, 6, 956–964, July 1997.

[43] P. W. Wong, "Multi-resolution binary image embedding," in *Proceedings of the Conference on Security and Watermarking of Multimedia Contents in the SPIE/IS&T Symposium on Electronic Imaging*, San Jose, CA, January 2003, 423–429.