– from "Fundamentals of Interactive Computer Graphics" by Foley and Van Dam, 1982.

This phenomenon is used in printing black and white photos in newspapers, magazines, and books, in a technique called *halftoning*. Each small resolution unit is imprinted with a circle of black ink whose area is proportional to the blackness (1 – intensity) of the area in the original photo. Figure 17.7 shows a greatly enlarged part of a halftone pattern. Newspaper halftones use a resolution of 60 to 80 dots per inch, while magazine and book halftones use up to 150 dots per inch.

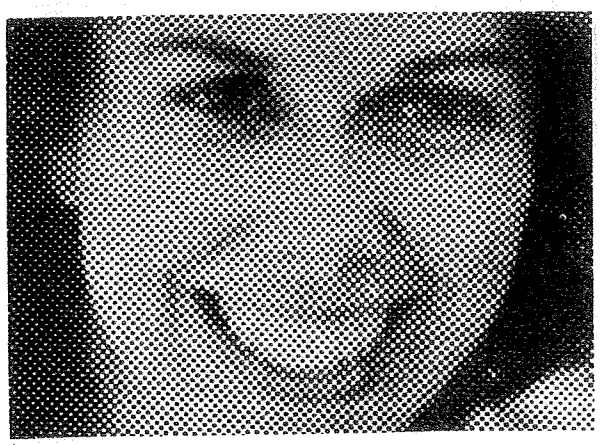


Fig. 17.7 Enlarged halftone pattern, Dot size varies inversely with intensity of original photo.

Graphics output devices can approximate the variable-area dots of halftone reproduction. For example, a 2 × 2 pixel area of a bi-level display can be used to produce five different intensity levels at the price of cutting the spatial resolution in half along each axis. The patterns shown in Fig. 17.8 can be used in the 2 × 2 areas. The idea is to fill the 2 × 2 area with a number of dots which is proportional to the desired intensity. Figure 17.9 shows a face defined on a 256 × 256 grid displayed on a 512 × 512 bi-level display by means of 2 × 2 patterns. In general, an $n \times n$ group of bi-level pixels can provide $n^2 + 1$ intensity levels. (We are trading spatial resolution, which is being decreased, for intensity resolution, which is being increased.) The use of a 3 × 3 pattern cuts spatial resolution by one-third on each axis, but provides 10 intensity levels. Of course, even larger patterns can be used, but the spatial versus intensity resolution trade-off is limited by our visual acuity (about one minute of arc in normal lighting).

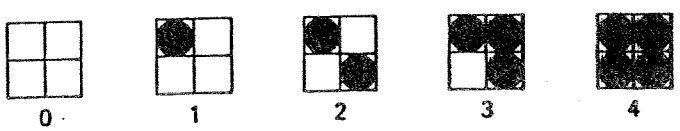


Fig. 17.8 Five intensity-level approximations by means of 2 × 2 patterns.

Fig. 17.9 Photo of 256×256 points displayed on 512×512 display by means of 2×2 patterns (courtesy J. Jarvis, Bell Laboratories).

One possible set of patterns for the 3×3 case is shown in Fig. 17.10. Note that these patterns can be represented by the matrix

$$\begin{bmatrix} 7 & 9 & 5 \\ 2 & 1 & 4 \\ 6 & 3 & 8 \end{bmatrix}.$$

To display a given intensity, all cells whose values are less than or equal to the intensity are set to on.

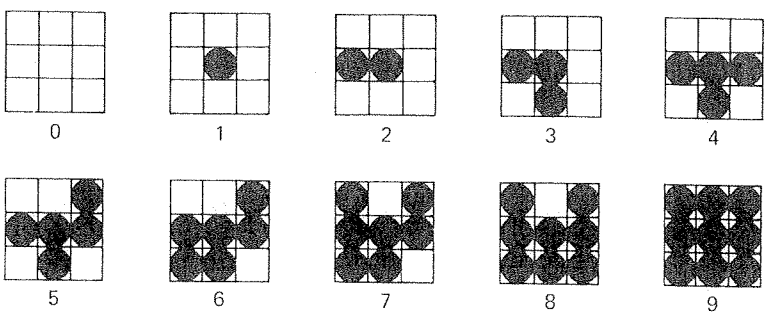


Fig. 17.10 Ten intensity-level approximations by means of 3×3 patterns.

The $n \times n$ pixel patterns used to approximate the halftones must be designed so that they are not conspicuous in an area of identical intensity values. For instance, if we used the pattern in Fig. 17.11 rather than the one in Fig. 17.10, then horizontal lines would be visible in any large area of the image that has intensity 3. Another consideration in choosing patterns is that they form a *growth sequence,* in which a pixel that is intensified for intensity level j is intensified for all levels $k > j$. This minimizes the differences in the patterns for successive intensity levels, thereby minimizing the contouring effects.

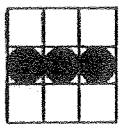


Fig. 17.11 A pattern that should not be used to approximate halftones.

Halftone approximation is not limited to bi-level displays. Suppose we have a display with two bits per pixel and hence four intensity levels. The halftone technique can be used to increase further the number of intensity levels. If we use a 2×2 pattern, we have a total of four pixels at our disposal, each of which can take on three values besides black, allowing us to display $4 \times 3 + 1 = 13$ intensities. One possible set of growth sequence patterns for this case is shown in Fig. 17.12. The intensities of the individual pixels sum to the intensity level represented by each pattern.

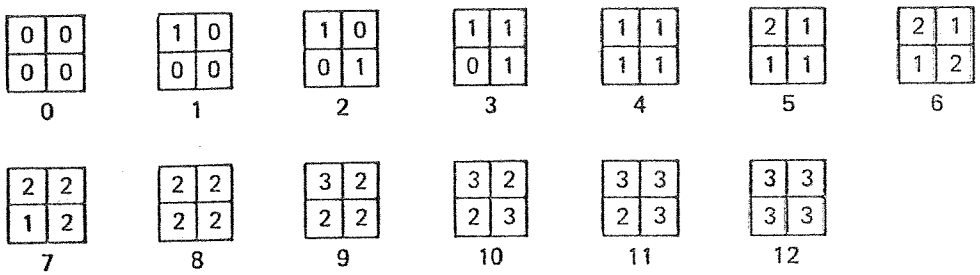


Fig. 17.12 Halftone patterns for intensity levels 0 to 12 based on 2×2 patterns of 4-level pixels.

These preceding techniques are appropriate if the resolution of the image to be displayed is lower than the resolution of the display device, allowing the use of multiple display pixels for one image pixel. What if the image and device resolutions are the same? The *ordered dither* technique can be used to display an $m \times m$ image with multiple levels of intensity on an $m \times m$ bi-level display.

In ordered dither, the decision to intensify or not to intensify the pixel at point (x, y) depends on the desired intensity $I(x, y)$ at that point and on an $n \times n$ dither matrix $D^{(n)}$. The dither matrix is indexed from 0 to $n - 1$ along its rows and col-

umns. Each of the integers 0 to $n^2 - 1$ appears once in the matrix. For instance, when $n = 2$, we have

$$D^{(2)} = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}. \tag{17.8}$$

This is the same representation as used for growth sequences. To process the point at (x, y), we first compute

$$i = x \text{ modulo } n, \qquad j = y \text{ modulo } n.$$

Then if

$$I(x, y) > D_{ij}^{(n)}, \tag{17.9}$$

the point at (x, y) is intensified; otherwise it is not. Notice that large areas of fixed intensity are displayed exactly as by the previous methods, so the effect of ordered dither is apparent only in areas of changing intensity. Matrices for various values of n have been developed by Bayer [BAYE73] to minimize the amount of texture they introduce into displayed images.

The dither matrix used to display an image must be able to specify the number of intensity levels in the image. Recurrence relations have thus been developed [JUDI74] to compute $D^{(2n)}$ from $D^{(n)}$. Their application to $D^{(2)}$ produces:

$$D^{(4)} = \begin{bmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{bmatrix} \tag{17.10}$$

and

$$D^{(8)} = \begin{bmatrix} 0 & 32 & 8 & 40 & 2 & 34 & 10 & 42 \\ 48 & 16 & 56 & 24 & 50 & 18 & 58 & 26 \\ 12 & 44 & 4 & 36 & 14 & 46 & 6 & 38 \\ 60 & 28 & 52 & 20 & 62 & 30 & 54 & 22 \\ 3 & 35 & 11 & 43 & 1 & 33 & 9 & 41 \\ 51 & 19 & 59 & 27 & 49 & 17 & 57 & 25 \\ 15 & 47 & 7 & 39 & 13 & 45 & 5 & 37 \\ 63 & 31 & 55 & 23 & 61 & 29 & 53 & 21 \end{bmatrix}. \tag{17.11}$$

Figure 17.13 shows a face drawn on a 512 × 512 bi-level display by using $D^{(8)}$. Compare this bi-level result to the multilevel pictures shown earlier in this section. Further examples of pictures displayed by means of ordered dither are in [JARV76a, JARV76b], as are descriptions of still other ways to display continuous-tone images on bi-level displays.

Dithering:

— display gray-level (> 1 bit) image on a monochrome device, e.g. printer.

Example:

Ordered Dither — display $m \times m$ image on an $m \times m$ display.

Algorithm: $i = x \bmod n$, $j = y \bmod n$

if $I(x, y) > D_{ij}^{(n)}$

then the point at (x, y) is intensified.

Choose $n = 2$, $D^{(2)} = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}$

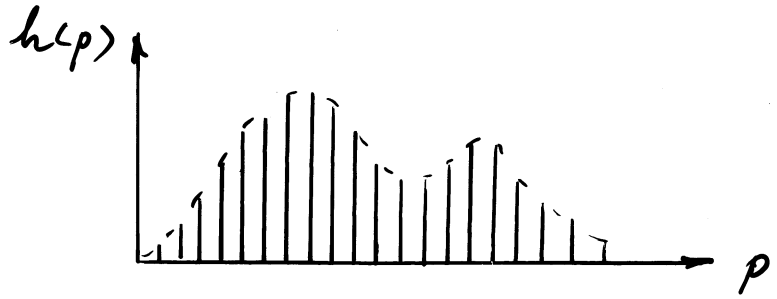
$I(x, y)$

1	1	1	1	1	1	1	1		·		·		·	·
1	1	1	1	1	1	1	1							
1	1	3	3	3	3	1	1		·		·	·	·	·
1	1	3	3	3	3	1	1					·		·
1	1	3	3	3	3	1	1		·		·	·	·	·
1	1	3	3	3	3	1	1					·		·
1	1	1	1	1	1	1	1		·		·		·	·
1	1	1	1	1	1	1	1							

I. Low-level Image Processing Techniques.

1. Histogram Transformation

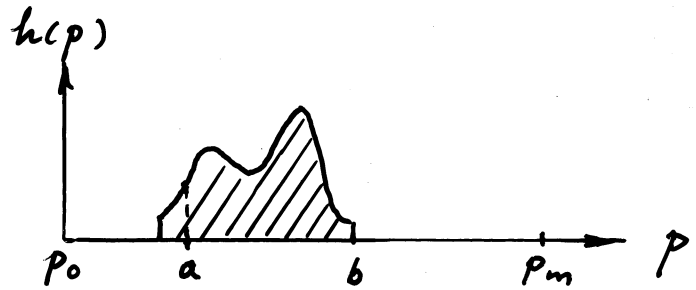
Gray-level histogram — the frequency of occurrence of each gray level in the image.



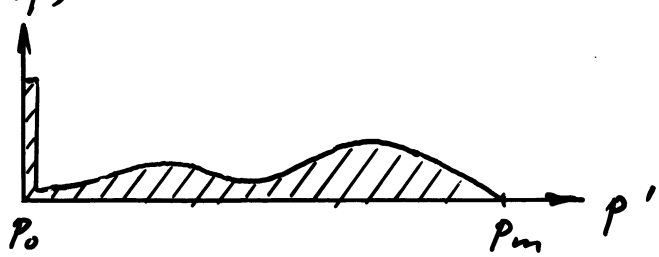
Histogram transformation — change the histogram in order to enhance the image.

(1) Linear Gray-level Transformation

underexposed
(dark, poor contrast)



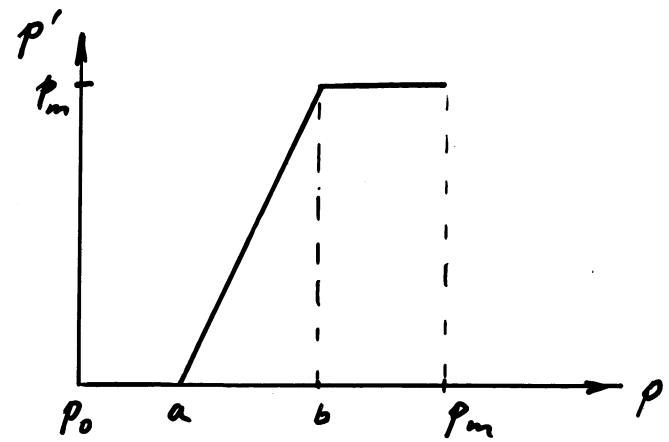
stretched to → h'(p')



Stretching:

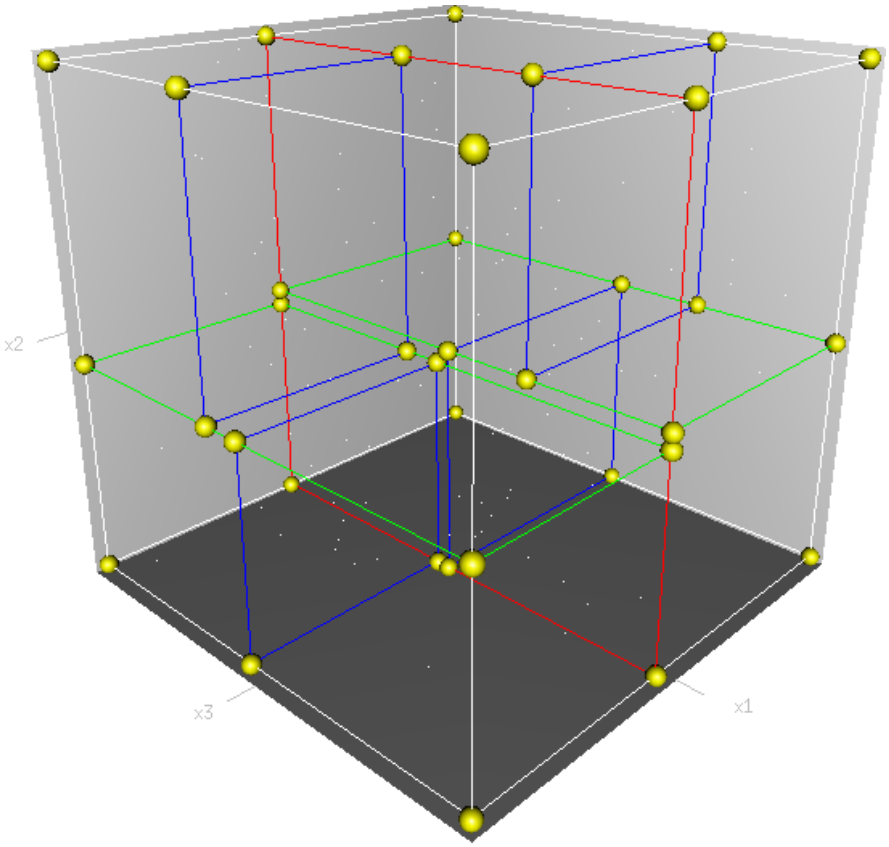
$$p' = T(p)$$

$$p' = \begin{cases} \dfrac{p_m - p_0}{b-a}(p-a) + p_0 & \text{if } a \le p \le b \\ p_0 & \text{if } p < a \\ p_m & \text{if } p > b \end{cases}$$



A special case: compressed to binary images.
(thresholding)

Example: Color Quantization by *Median Cut* in the color histogram space

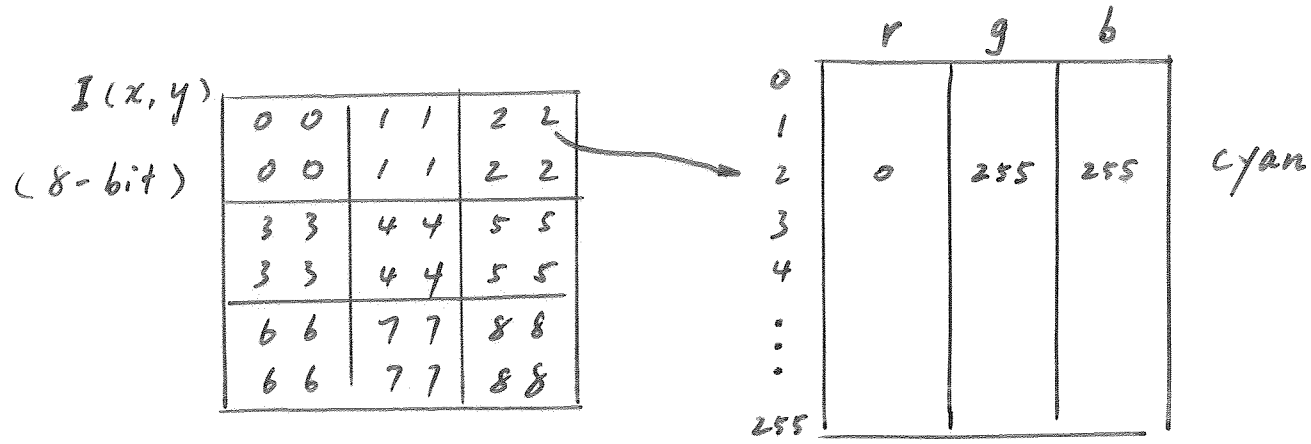


Red: first cut Green: second cut Blue: third cut

- Only three cuts/bits are shown. In general, more cuts/bits can be used when needed.
- The resulting cells will have approximately the same pixel count.
- This data structure is also known as *k-d tree*.

More on Color LUT

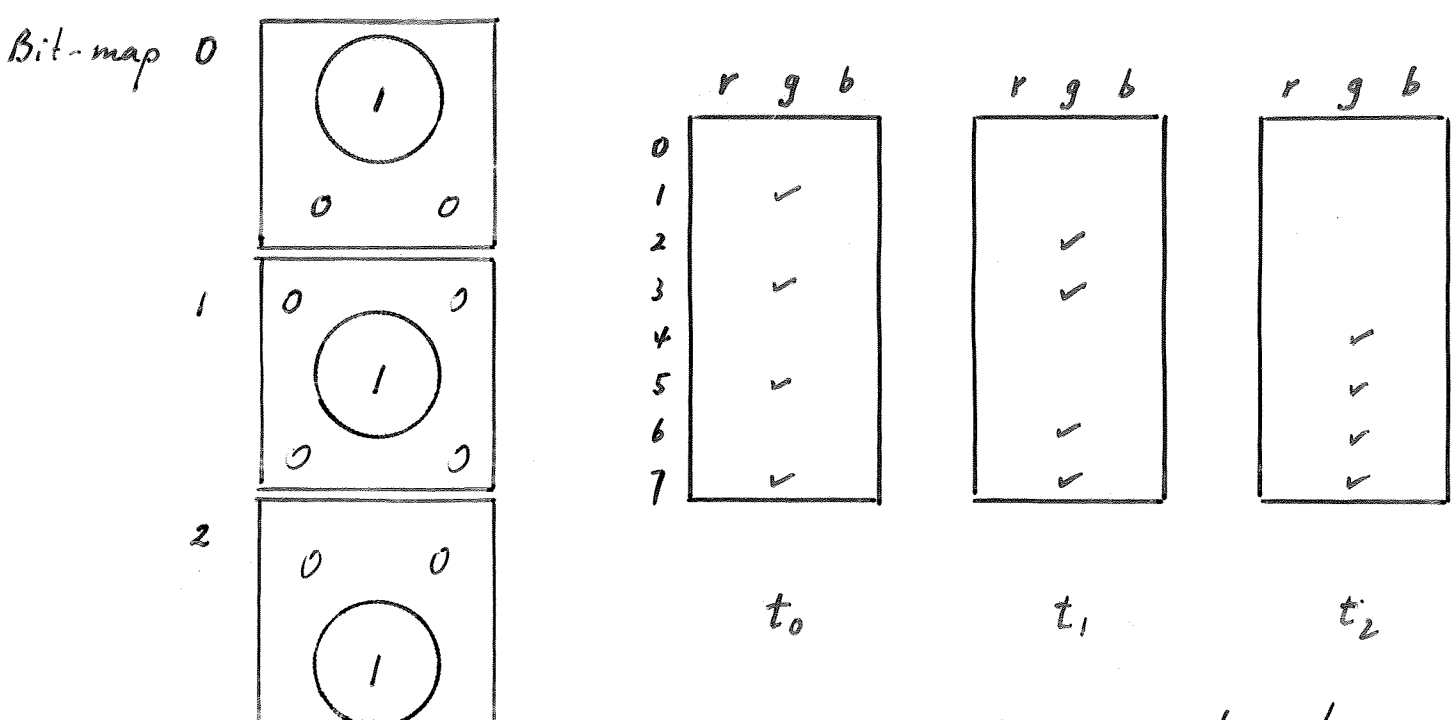
1. Color Palette Image:



2. Simple Animation Using Bit-map Images:

Example: Bouncing Ball

(if balls do not overlap, then easy. (how?))



✓ — any foreground color for the ball / disk.

PNG (Portable Network Graphics)

- 1-, 2-, 4- and 8-bit palette support (like GIF)
- 1-, 2-, 4-, 8- and 16-bit grayscale support
- 8- and 16-bit-per-channel (that is, 24- and 48-bit) truecolor support
- full alpha blending in 8- and 16-bit modes, not just simple on-off transparency like GIF
- two-dimensional interlacing scheme
- gamma correction for cross-platform "brightness" control
- lossless compression, typically 10% to 30% better than GIF
- full Year 2000 (Y2K) support (good for *at least* 63 millennia!)
- free and complete reference implementation with full source code