Computer Graphics
-Quantization & Dithering

Junjie Cao @ DLUT

Spring 2017

http://jjcao.github.io/ComputerGraphics/

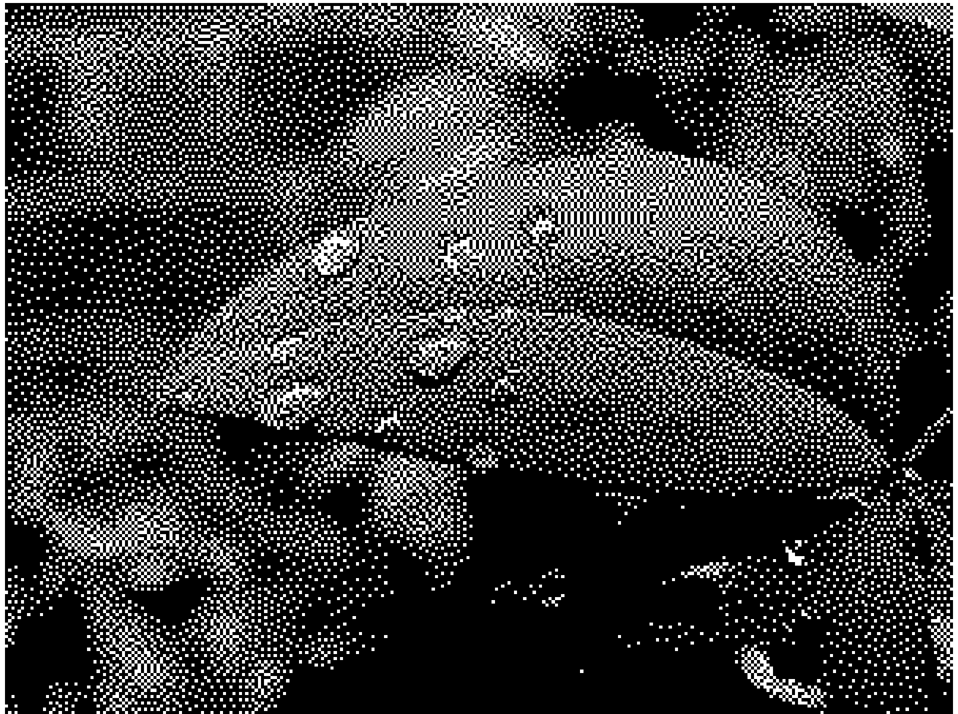


Image Resolution

- Intensity resolution
  - Each pixel has only “Depth” bits for colors/intensities
- Spatial resolution
  - Image has only “Width” x “Height” pixels
- Temporal resolution
  - Monitor refreshes images at only “Rate” Hz

Typical Resolutions

	Width x Height	Depth	Rate
NTSC	640 x 480	8	30
Workstation	1280 x 1024	24	75
Film	3000 x 2000	12	24
Laser Printer	6600 x 5100	1	-

Sources of Error

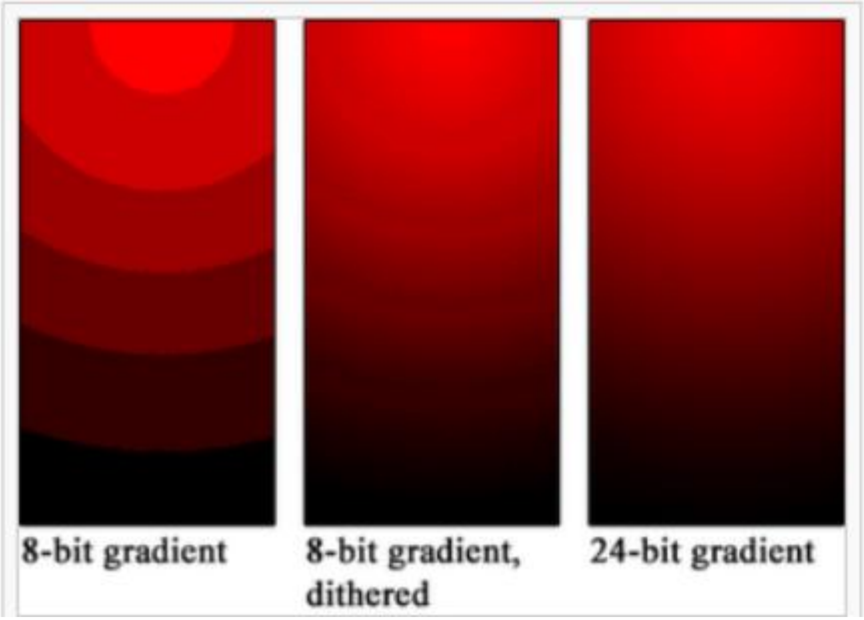
- Intensity quantization
  - Not enough intensity resolution
- Spatial aliasing
  - Not enough spatial resolution
- Temporal aliasing
  - Not enough temporal resolution

$$E^2 = \sum_{(x,y)} \left(I(x,y) - P(x,y) \right)^2$$

Quantization, Halftoning & Dithering

- How to print a color picture on a black-white printer?
- If you have just a 256 color screen, how can you display a true color picture with millions color?

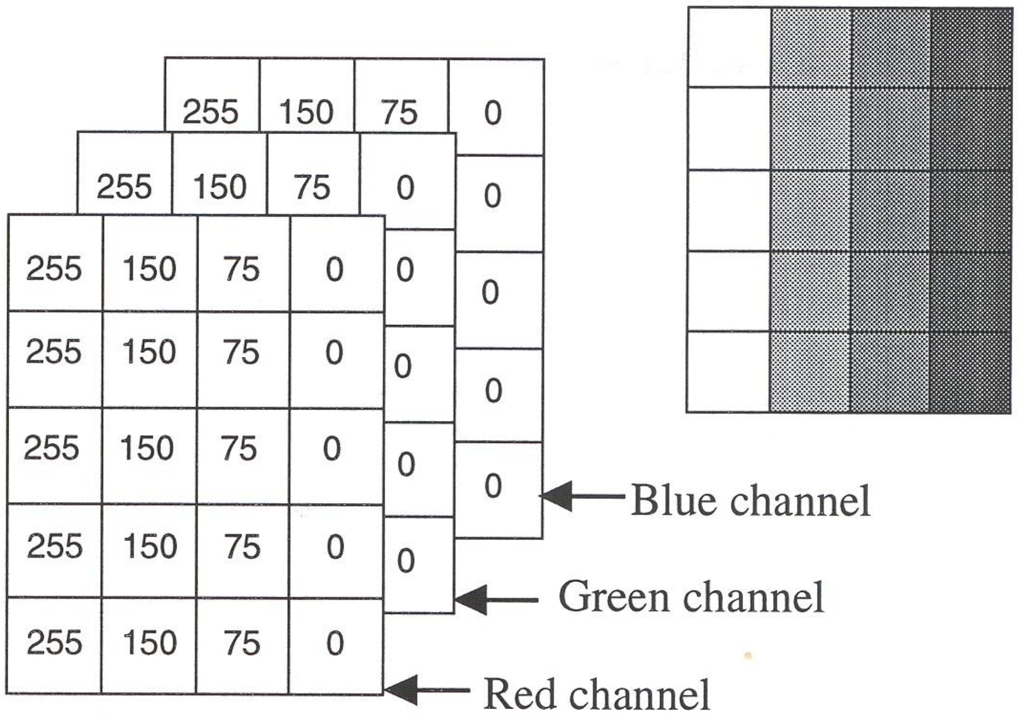
Simple threshold.



Colour banding (contouring (or false contouring))

Quantization - Why

- Signals in real world are continuous
- Physical devices have limited capability
  - Conserve memory & processing speed
  - Spatial resolution is limited: |pixels|
  - Intensity resolution: Frame buffers have limited number of bits per pixel
    - Binary output devices, printer



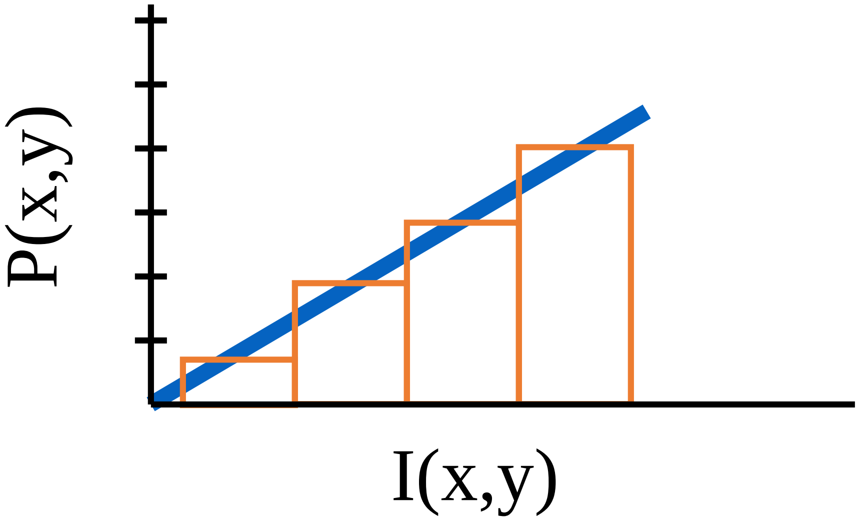
Quantization

- Definition: mapping a **large** set of input values to a (**countable**) **smaller** set.
  - Rounding & truncation

- Quantization v.s. sampling
  - **Intensity resolution** v.s. spatial resolution
  - **Loss of information**
  - **Irreversible** vs. reversible

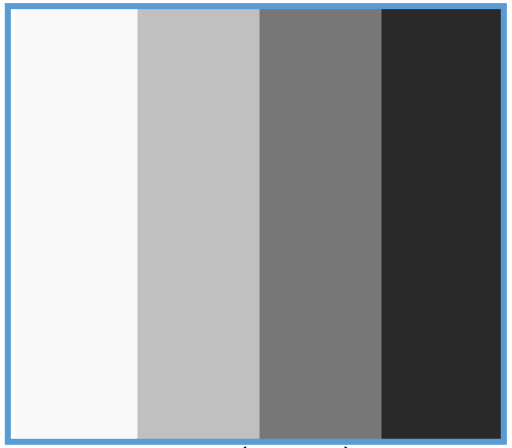
- How? Minimize error/distortion
  - Number of quantization levels
    - Tradeoff between resulting image quality versus amount of data needed
  - Value of each quantization level

Uniform Quantization



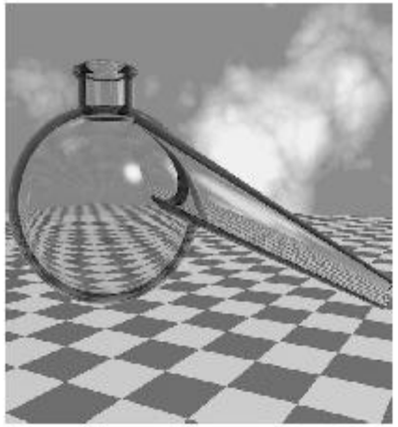
$$P(x, y) = \mathrm{trunc}(I(x, y) + 0.5)$$

I(x,y)

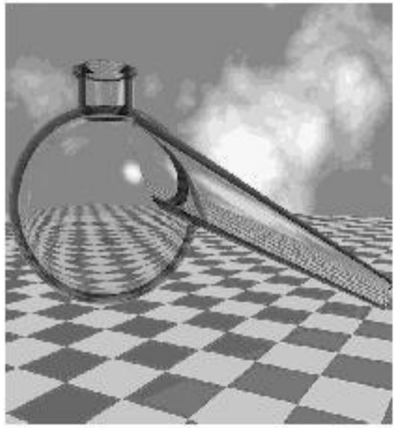


P(x,y)
(2 bits per pixel)

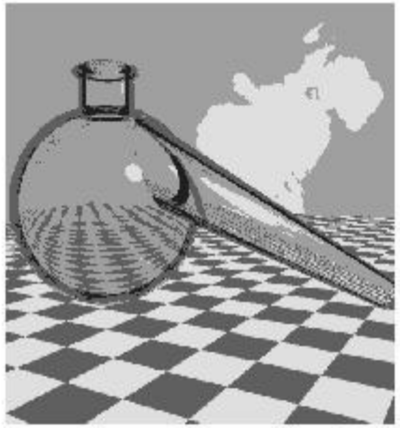
Uniform Quantization



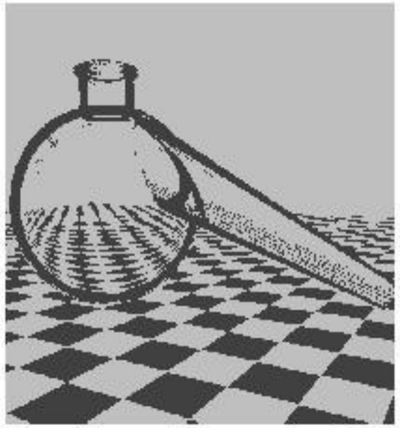
K=256



K=16



K=4



K=2

Quantization Error

Quantization introduces error

$$E^2 = \sum_{(x,y)} \left(\frac{\hat{v}(x,y)}{K-1} - v(x,y)\right)^2$$

$$0 \le v < 1$$

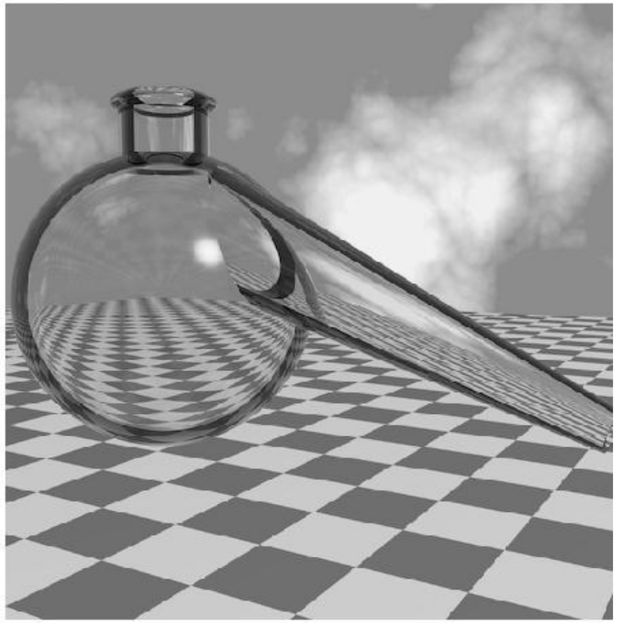
$$0 \le \hat{v} < K$$

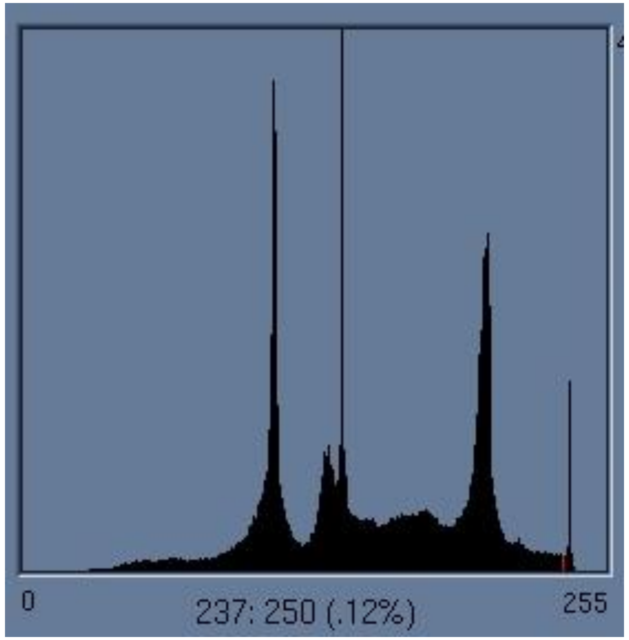
To reduce error:

1. Nonuniform quantization (minimize error)
2. Halftoning (trade-off intensity/space error)
3. Dithering

Image Histogram

- Uniformly quantize image to M levels.
- Plot number of pixels within each level.
- Divide by total number to get pixel probability distribution function *p(v)*

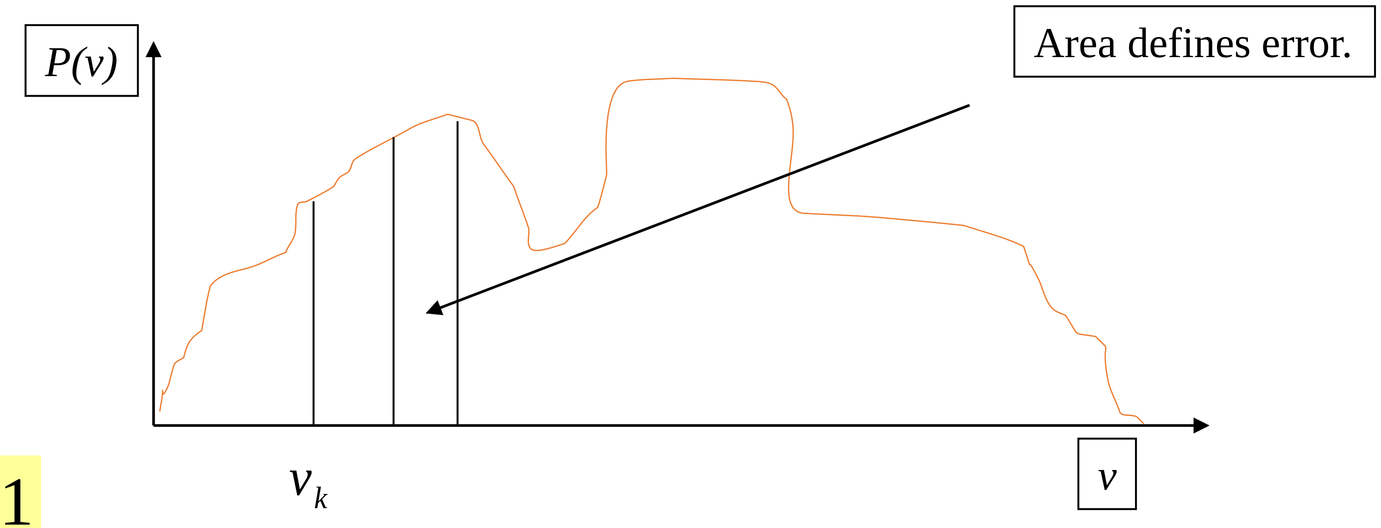




Non-uniform Quantization

Re-write error in terms of *p(v)*.

$$E^2 = \sum_{k=1}^{K} \int_{v_{k-1}}^{v_k} (v - \hat{v})^2 \, p(v) dv$$



$\hat{v}$ Quantised value of *v*, $0 \le \hat{v} < 1$

v_k Value of *v* at interval k, $0 \le v_k < 1$

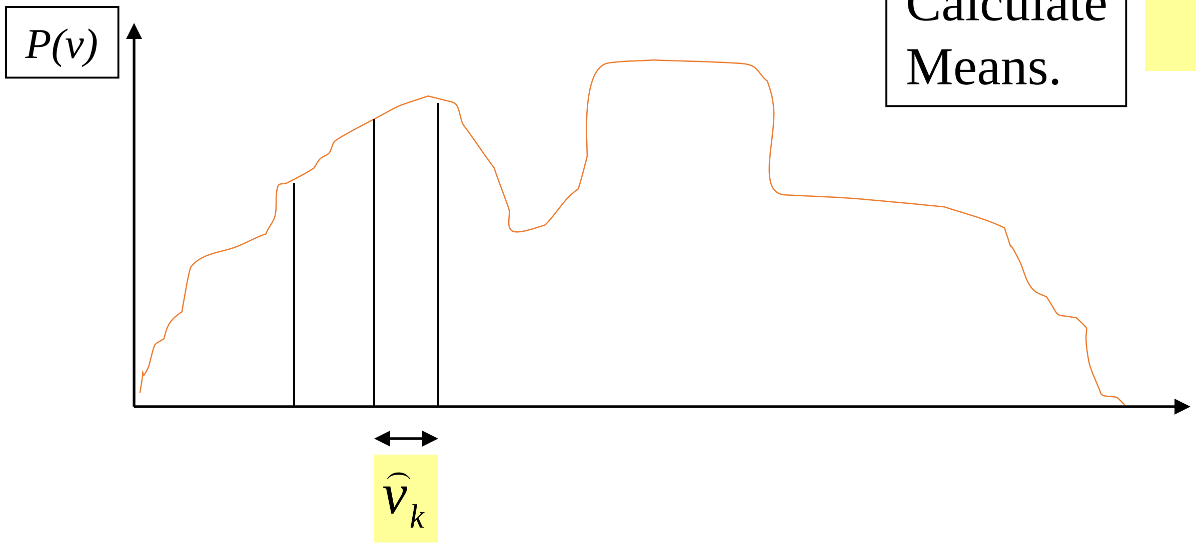
Note : v_k no longer integers

Least Squares Quantization

Known v_0, v_1, …, v_K, compute $\widehat{V}_k$
Differentiate, then …

$$E^2 = \sum_{k=1}^{K} \int_{v_{k-1}}^{v_k} (v - \hat{v})^2 \, p(v)\, dv \longrightarrow \widehat{V}_k = \frac{\int_{V_{k-1}}^{V_k} v p(v) dv}{\int_{V_{k-1}}^{V_k} p(v) dv}$$

Calculate Means.



Least Squares Quantization

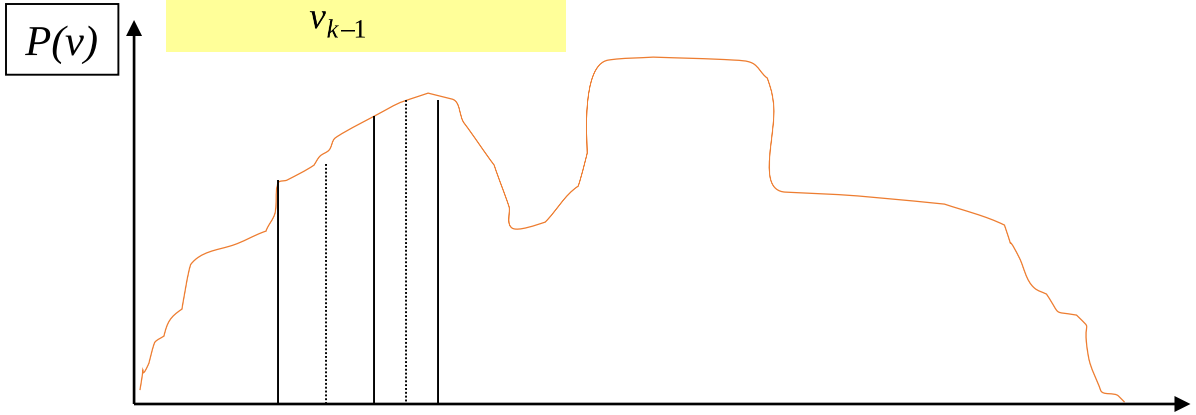
If v_0, v_1, …, v_K are not known …

$$\hat{v}_k = \frac{\int_{v_{k-1}}^{v_k} v p(v) dv}{\int_{v_{k-1}}^{v_k} p(v) dv}$$

interval

$$\bar{v}_k = \frac{1}{2}(\hat{v}_{k-1} + \hat{v}_k)$$

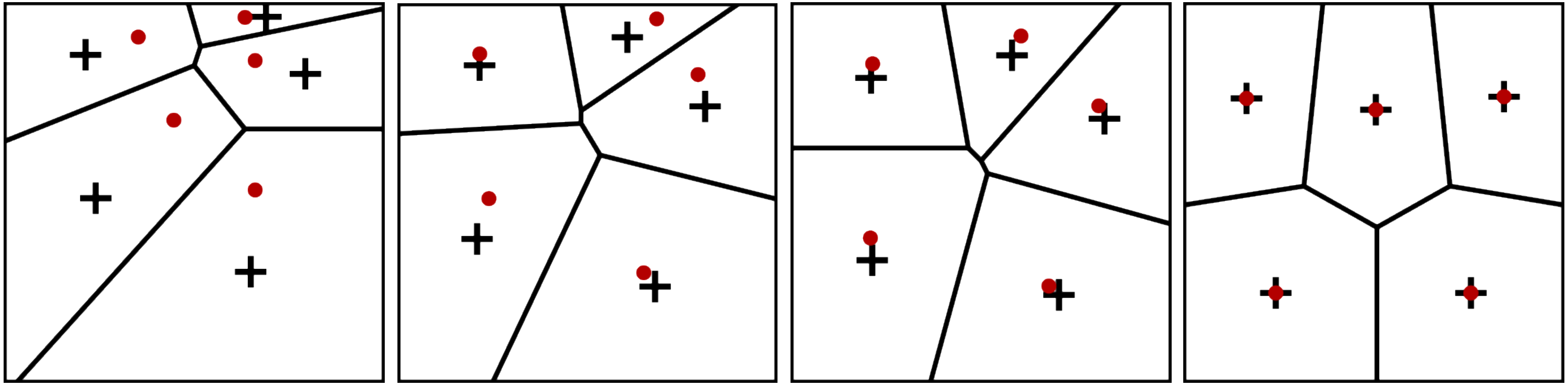
Solve by iteration.



$\hat{v}_{k-1}$ $\bar{v}_k$ $\hat{v}_k$

Move intervals $\bar{v}_k$ towards larger p(v)

Lloyd‘s algorithm & Least Squares Quantization

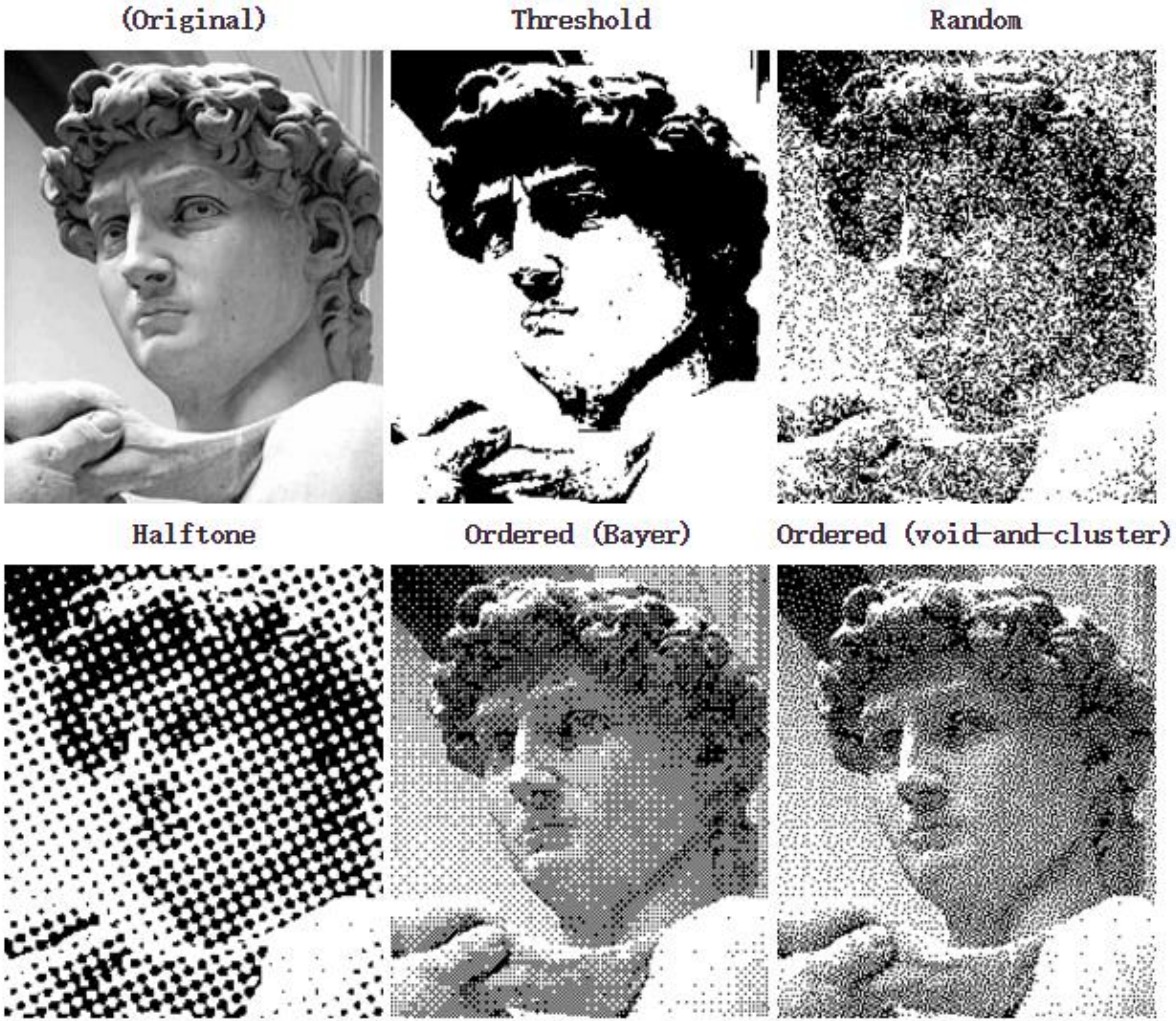


Lloyd, Stuart P. (1982), “Least squares quantization in PCM”, *IEEE Transactions on Information Theory*

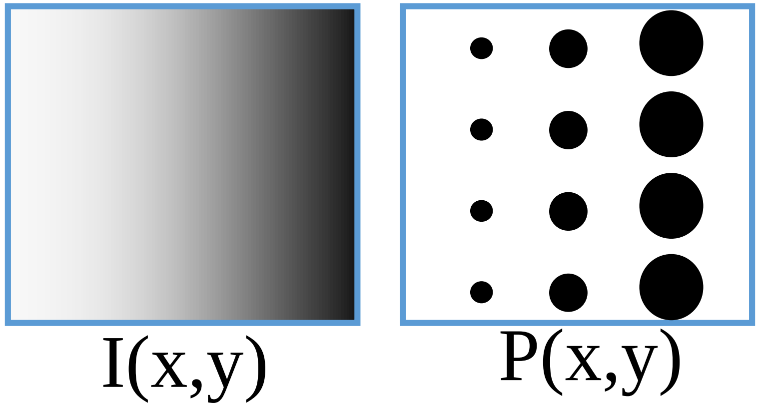
Quantization, Halftoning and dithering

- Quantization
  - Approximation intensity in intensity space, uniform or optimization

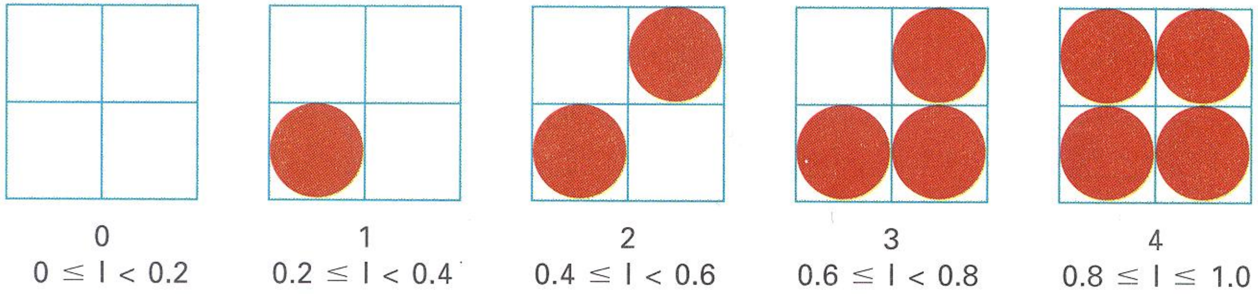
- Reduce visual artifacts due to quantization
  - Halftoning: Trade spatial resolution for intensity resolution (discussed later)
  - Dithering/pseudo random noise quantization


(Original)
Threshold
Random
Halftone
Ordered (Bayer)
Ordered (void-and-cluster)

Classical Halftoning



- Spatial averaging of the visual system.
  - Use dots of varying size to represent intensities
    - Area of dots proportional to intensity in image
  - **Use cluster of pixels to represent intensity**

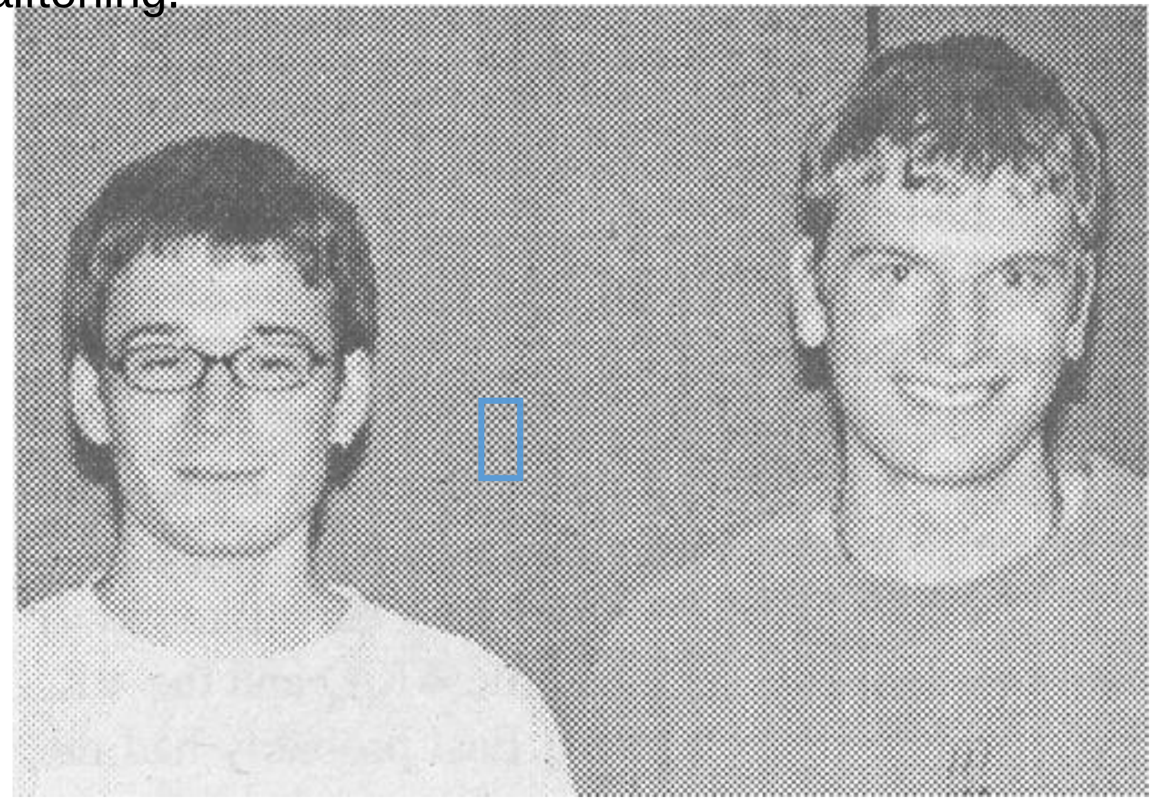


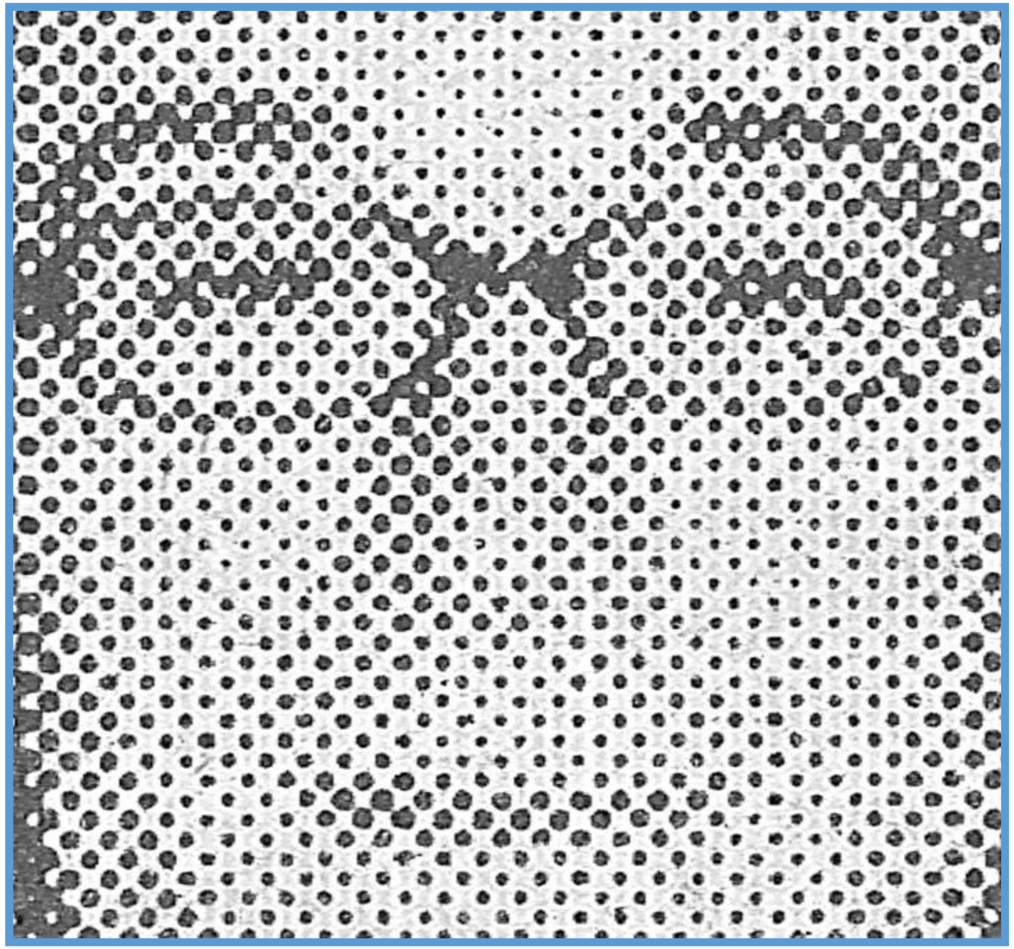
- When image is to be displayed/printed on a device with a much larger resolution than original image, **resolution may be traded of for dynamic range**.

- In printing (6600 x 5100), this is usually possible.

Classical Halftoning

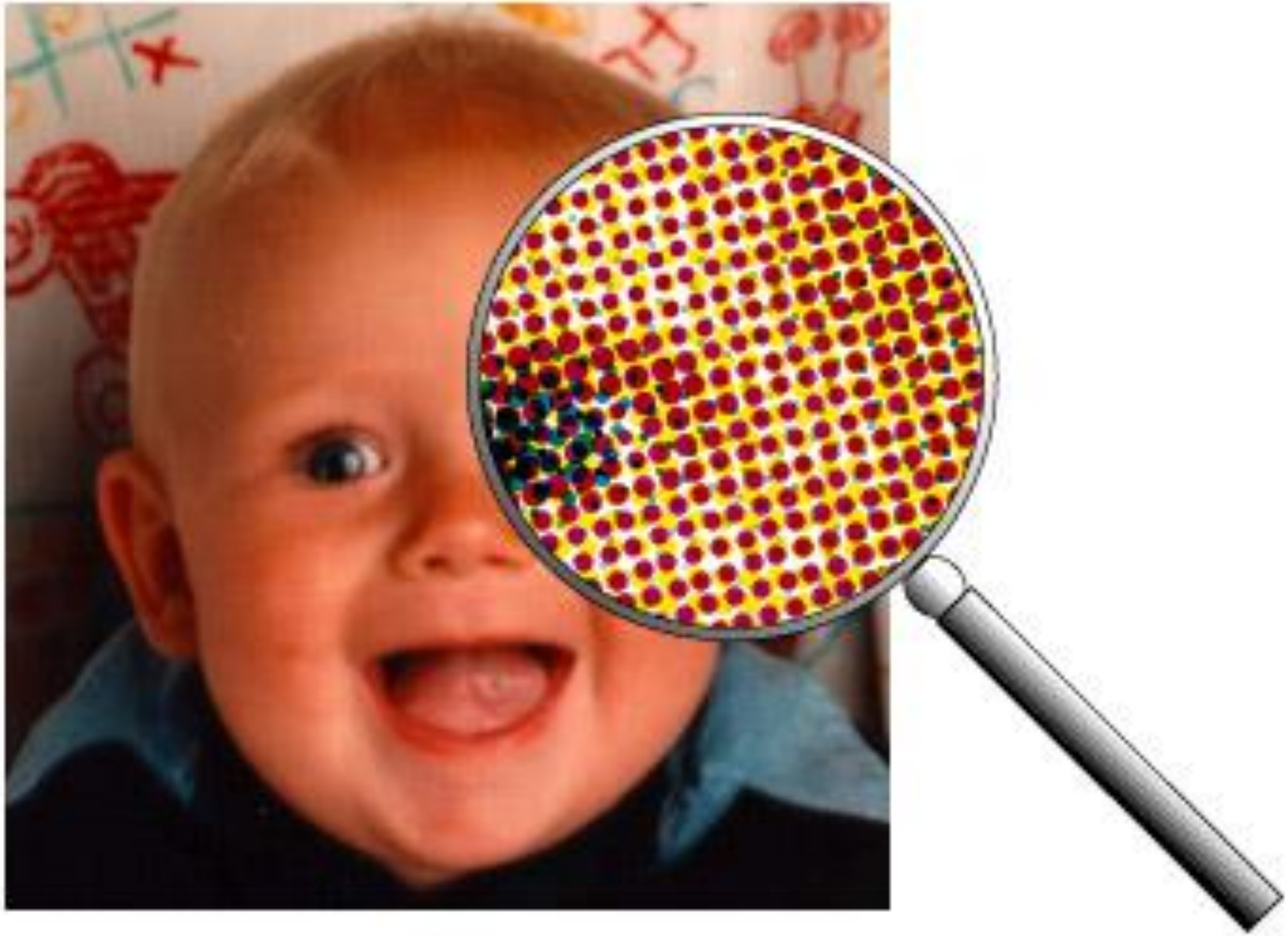
Almost all printed images in books, newspapers, and from laser printers are done using some form of halftoning.





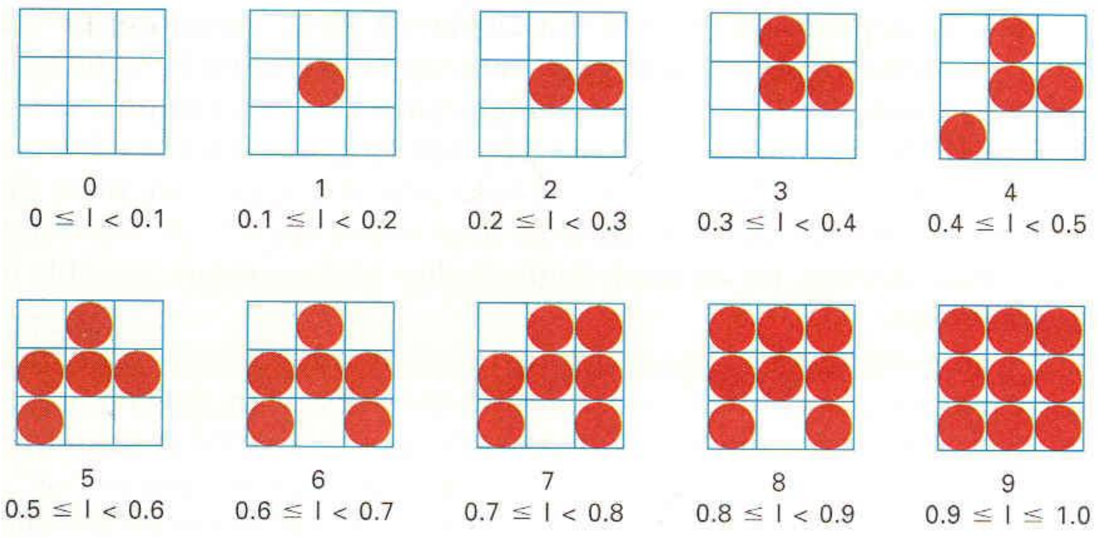
Newspaper image from *North American Bridge Championships Bulletin*, Summer 2003

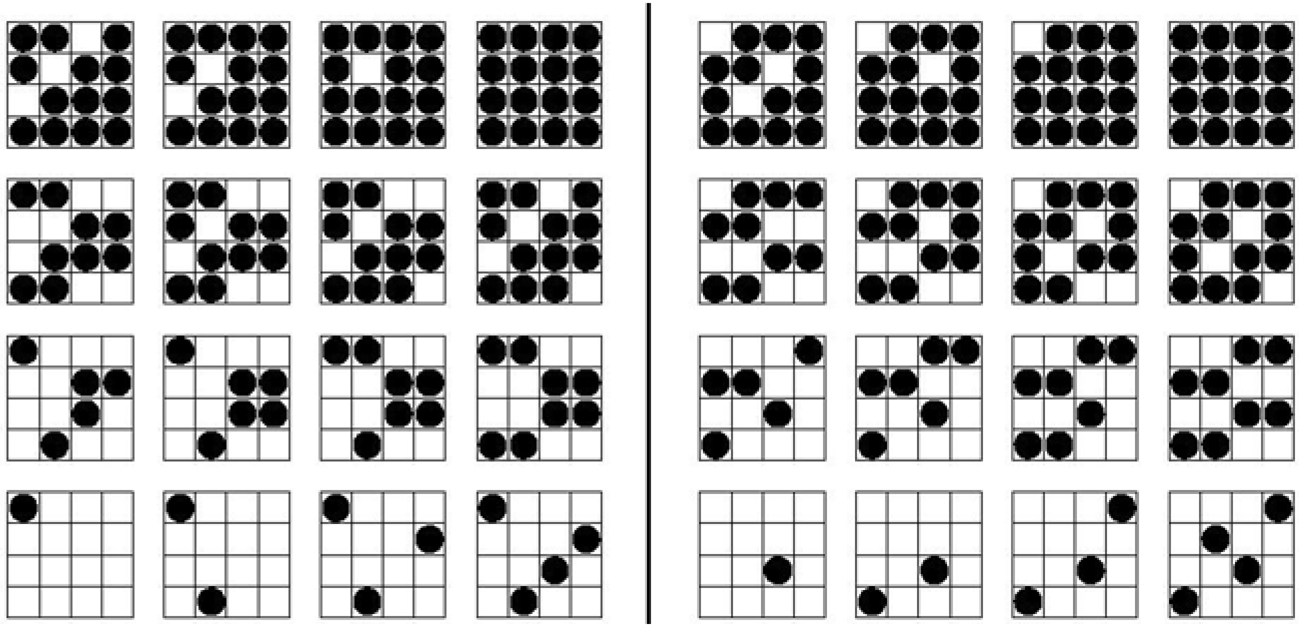
Halftoning and Colors



Halftone pattern design

- Use cluster of pixels to represent intensity
  - Intensities/gray scale vs number of black dots
  - Pattern for determined |dots|:
    - avoid using a pattern where all 8 white pixels are grouped together and all 8 black pixels are grouped together.
    - Avoid using a fixed pattern for a given intensity throughout the image
      - Unnatural: noticeable appearance of patterns
      - Add some random: diff pattern for a given intensity

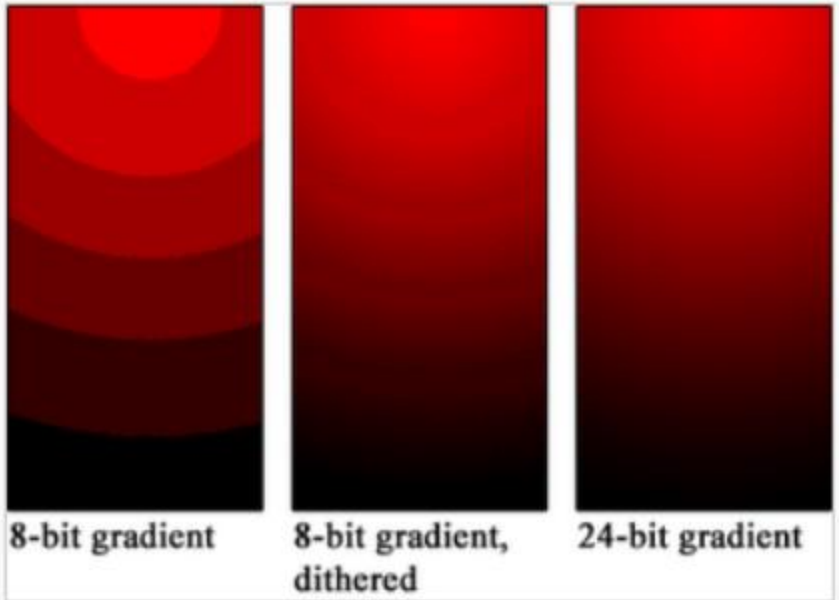




Dithering/pseudo random noise quantization

- Idea: add a small amount of random noise to the signal before quantizing
- This works because our eyes have limited spatial resolution.

- By having some pixels in a small neighborhood take on one quantized level and some other pixels take on a different quantized level, the transition between the two levels happens more gradually, as a result of averaging due to limited spatial resolution.



An illustration of colour banding

Dithering

- Uniform quantization discards all errors
  - i.e. all “rounding” errors
- Distribute errors among pixels
  - Exploit spatial integration in our eye
  - Display greater range of perceptible intensities

- Dithering is also used in audio, by the way

- Classification
  - **Error diffusion**
  - **Random dither/Robert’s algorithm**
  - **Ordered dither**

Error diffusion: Floyd-Steinberg dithering

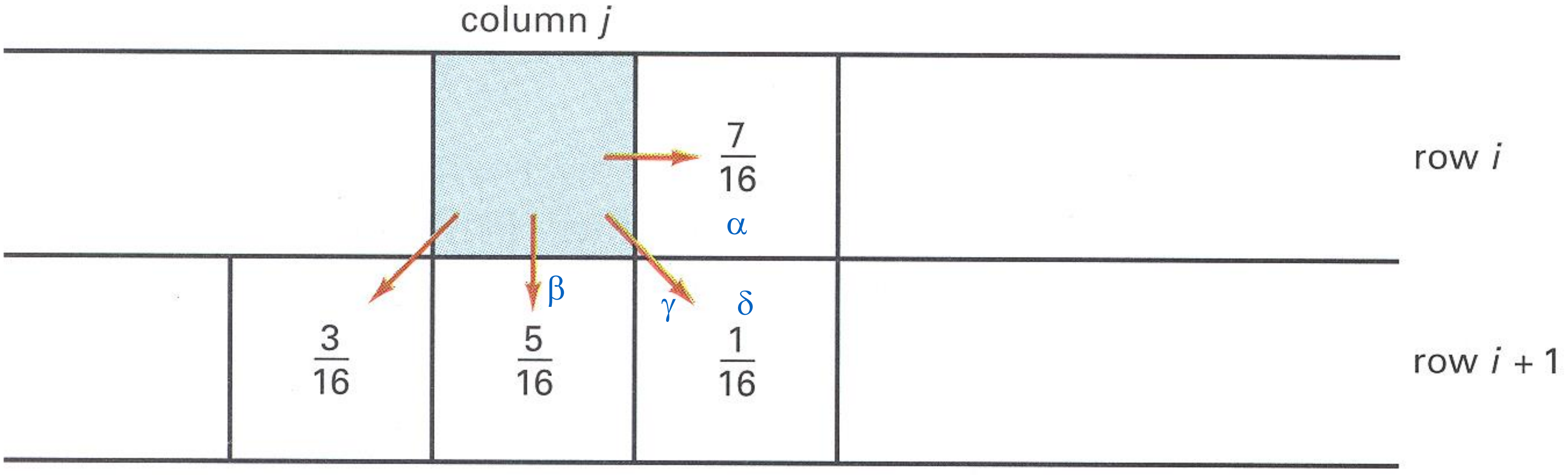
- Any “rounding” errors are distributed to other pixels
  - Specifically to the pixels below and to the right
    - 7/16 of the error to the pixel to the right
    - 3/16 of the error to the pixel to the lower left
    - 5/16 of the error to the pixel below
    - 1/16 of the error to the pixel to the lower right

- Assume the 1 in the middle gets “rounded” to 0

$$\begin{bmatrix} 0.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \end{bmatrix} \longrightarrow \begin{bmatrix} 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.44 \\ 0.19 & 0.31 & 0.06 \end{bmatrix}$$

Error Diffusion Dither

- Spread quantization error over neighbor pixels
  - Error dispersed to pixels right and below

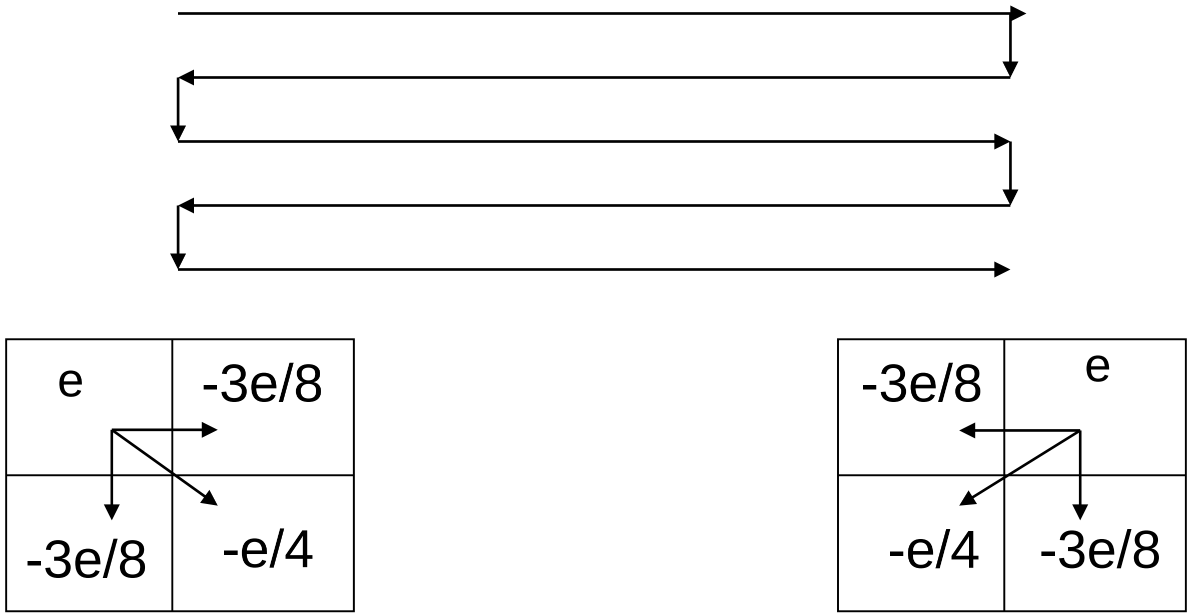


$$\alpha + \beta + \gamma + \delta = 1.0$$

Figure 14.42 from H&B

Floyd-Steinberg Error Diffusion

With this method, the average quantization error is reduced by propagating the error from each pixel to some of its neighbors in the scan order.

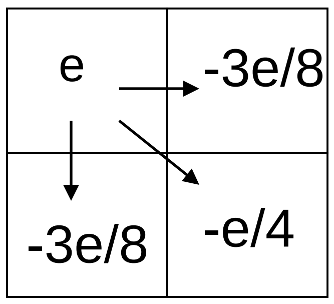


Note that the error propagation weights must sum to one

Error Diffusion

Idea: Quantize, then distribute error to neighbours

```
for(y=0; y<ny; y++)
 for(x=0; x<nx; x++){
  vq[x][y] = quantize(v[x][y]);
  e = v[x][y] - vq[x][y];
  v[x+1][y] += 3/8*e;
  v[x][y+1] += 3/8*e;
  v[x+1][y+1] += 1/4*e;
}
```



Floyd-Steinberg dithering

- Floyd-Steinberg dithering is a specific error dithering algorithm

Original
(8 bits)

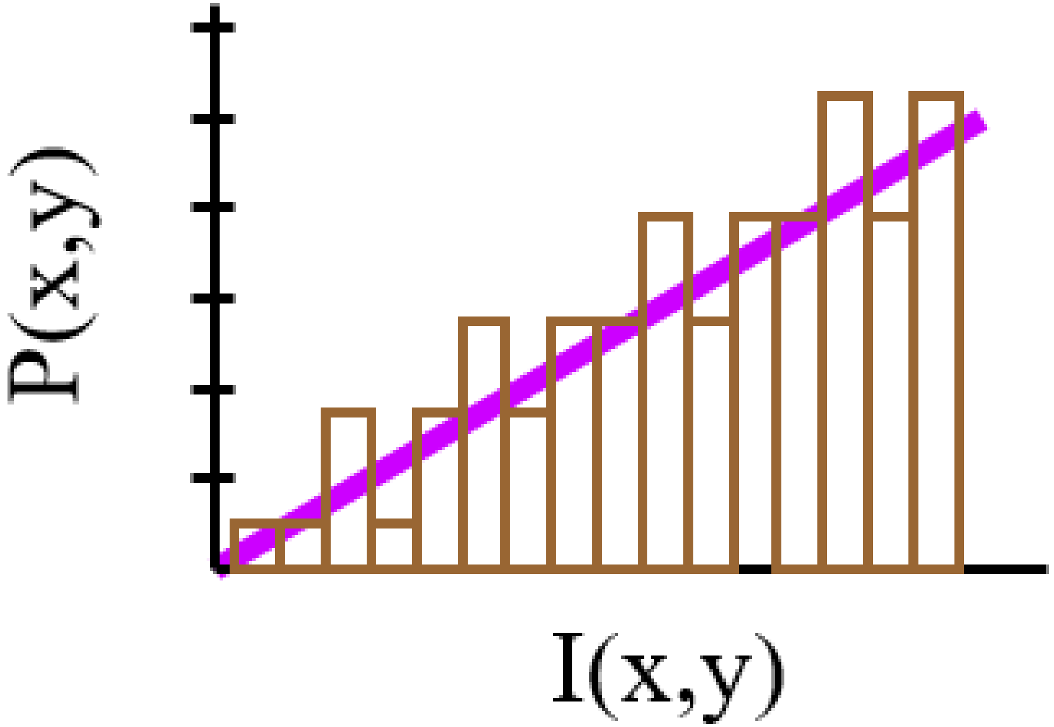
Uniform
Quantization
(1 bit)

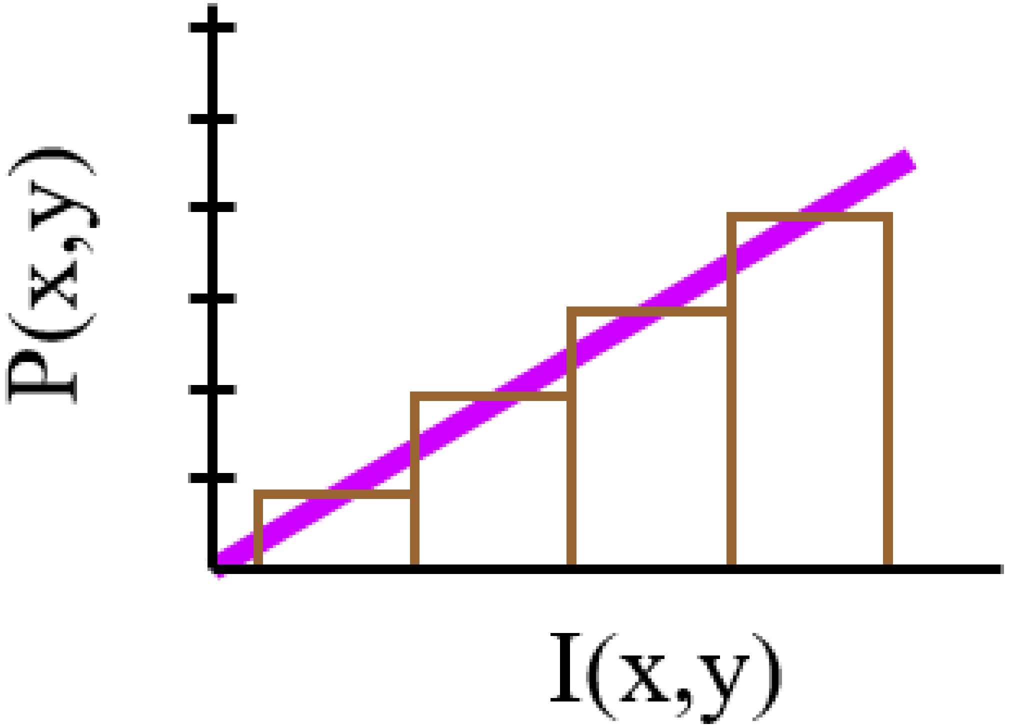
Floyd-Steinberg
Dither (1 bit)

Random Dithering - **Robert's Algorithm**

- First add noise
- Then quantize

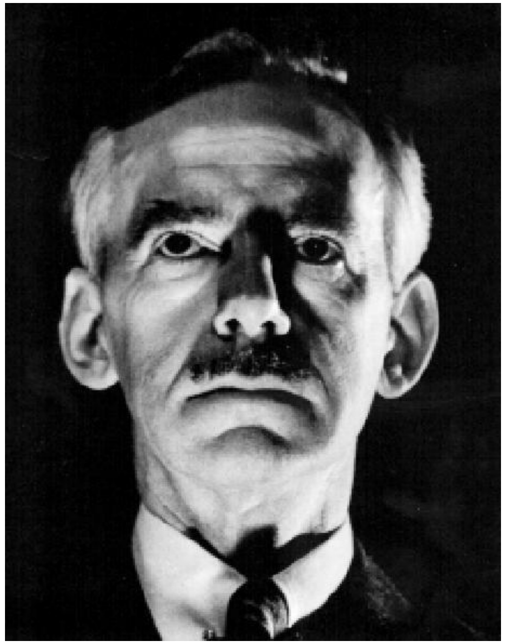
$$P(x, y) = \mathrm{trunc}(I(x, y) + \mathrm{noise}(x,y) + 0.5)$$





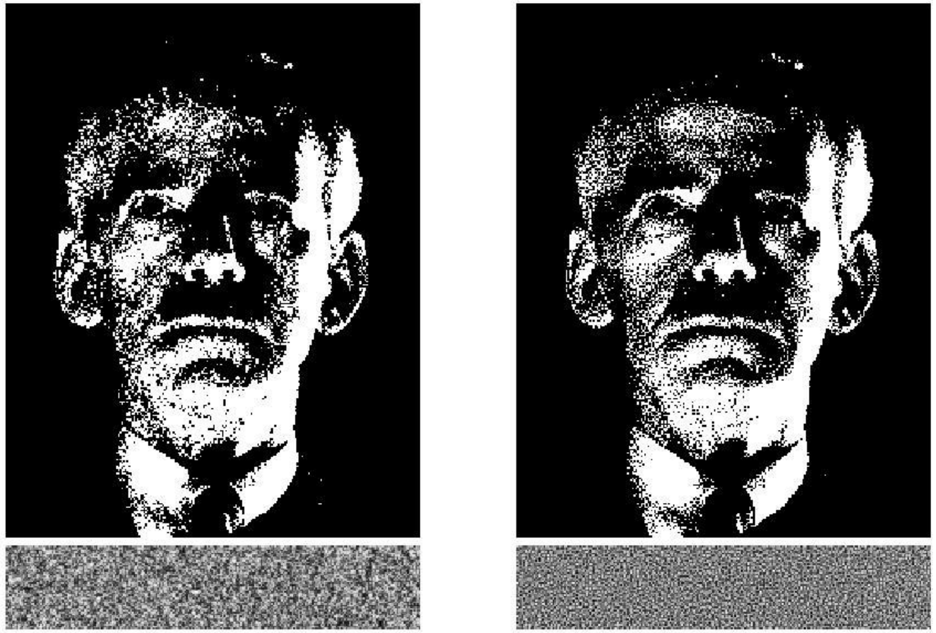
- Of course, the noise should not be too large since other features of the signal can also get disrupted by the noise.
- Usually the noise is small enough to produce changes to only adjacent quantization levels.

Random Dither



Original

simple threshold



Robert’s results with pink and blue noise

- Even for small amounts of noise, however, there will be some loss in spatial resolution. Nevertheless, the resulting signal is often more desirable.
- Adding noise and then quantizing will only **increase the mean squared error**, yet we’ve seen that this can result in a perceptually more pleasing signal.
- This shows that standard quantitative criteria such as mean square error **do not necessarily reflect subjective (perceived) quality**.
- Although dithering in images will introduce some dot-like artifacts, the image is often visually more pleasing since the false contours due to the abrupt transitions are less noticeable

The trouble with noise

- Difficult to compute quickly.
- Not reproducible.
- Pre-compute pseudo-random function and store in table.
- Small tiled patterns sufficient

Ordered Dithering

- Break the image into small blocks
- Define a *threshold matrix*
  - Use a different threshold for each pixel of the block
  - Compare each pixel to its own threshold
- The thresholds can be clustered, which looks like newsprint
- The thresholds can be “random” which looks better

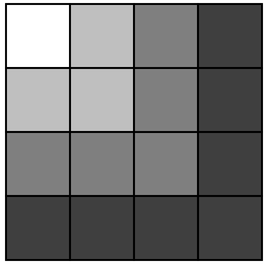
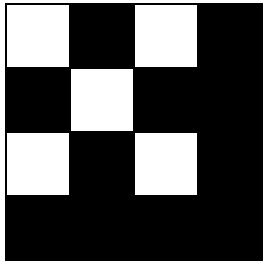


Image block

$$\begin{bmatrix} 1 & 0.75 & 0.5 & 0.25 \\ 0.75 & 0.75 & 0.5 & 0.25 \\ 0.5 & 0.5 & 0.5 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$$

Threshold matrix $\frac{1}{16}\begin{bmatrix} 2 & 16 & 3 & 13 \\ 10 & 6 & 11 & 7 \\ 4 & 14 & 1 & 15 \\ 12 & 8 & 9 & 5 \end{bmatrix}$



Result

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Ordered Dither

- Pseudo-random quantization errors
  - Matrix stores pattern of thresholds

$$D_2 = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}$$

```
i = x mod n
j = y mod n
d= [0,2;3,1]/4;
if (I(x,y) > D(i,j))
        P(x,y) = white
else
        P(x,y) = black
```

Ordered Dither

- Bayer's ordered dither matrices
  - Reflections and rotations of these are used as well

$$D_2 = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}$$

$$D_n = \begin{bmatrix} 4D_{n/2}+0 & 4D_{n/2}+2 \\ 4D_{n/2}+3 & 4D_{n/2}+1 \end{bmatrix}$$

$$D_4 = \begin{bmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{bmatrix}$$

$$\frac{1}{5}\begin{bmatrix} 1 & 3 \\ 4 & 2 \end{bmatrix}$$

$$\frac{1}{10}\begin{bmatrix} 1 & 8 & 4 \\ 7 & 6 & 3 \\ 5 & 2 & 9 \end{bmatrix}$$

$$\frac{1}{17}\begin{bmatrix} 1 & 9 & 3 & 11 \\ 13 & 5 & 15 & 7 \\ 4 & 12 & 2 & 10 \\ 16 & 8 & 14 & 6 \end{bmatrix}$$

$$\frac{1}{65}\begin{bmatrix} 1 & 49 & 13 & 61 & 4 & 52 & 16 & 64 \\ 33 & 17 & 45 & 29 & 36 & 20 & 48 & 32 \\ 9 & 57 & 5 & 53 & 12 & 60 & 8 & 56 \\ 41 & 25 & 37 & 21 & 44 & 28 & 40 & 24 \\ 3 & 51 & 15 & 63 & 2 & 50 & 14 & 62 \\ 35 & 19 & 47 & 31 & 34 & 18 & 46 & 30 \\ 11 & 59 & 7 & 55 & 10 & 58 & 6 & 54 \\ 43 & 27 & 39 & 23 & 42 & 26 & 38 & 22 \end{bmatrix}$$

Ordered Dither

Original
(8 bits)

Uniform
Quantization
(1 bit)

4x4 Ordered
Dither
(1 bit)

Dither Comparison

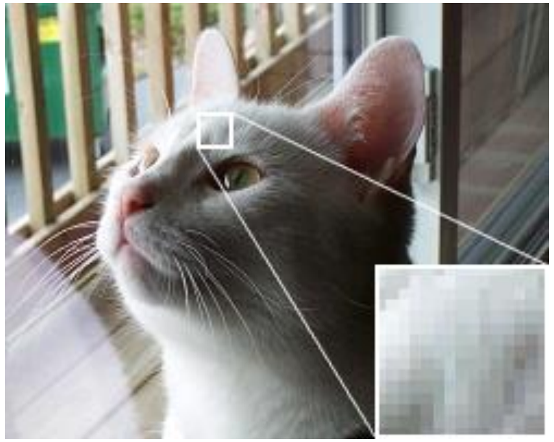
Original
(8 bits)

Random
Dither
(1 bit)

Ordered
Dither
(1 bit)

Floyd-Steinberg
Dither
(1 bit)

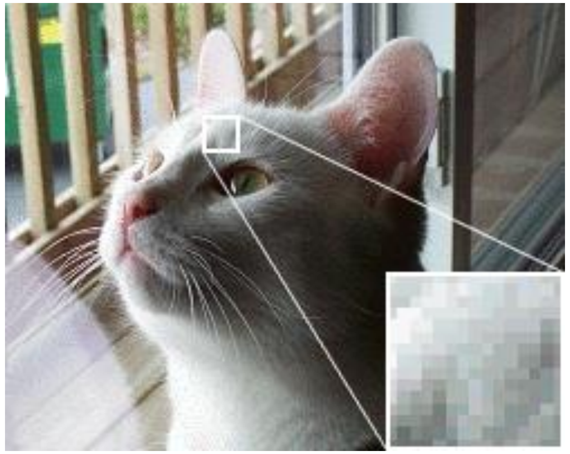
Color dithering comparison



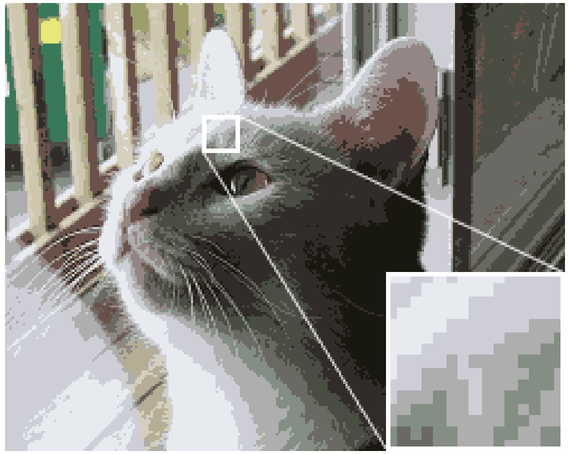
Original image

Web-safe palette, no dithering

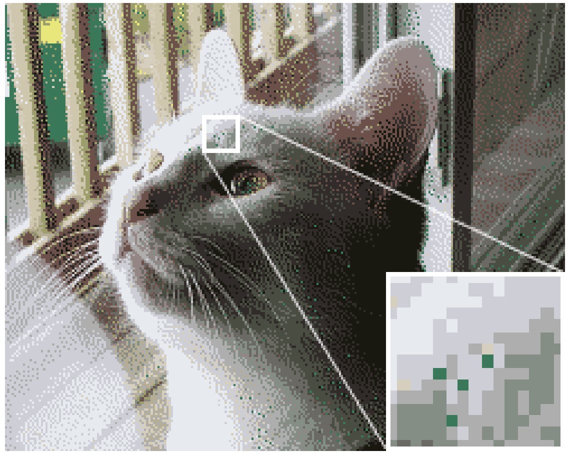
Web-safe palette, FS dithering



Optimized 256 color palette
FS dithering



Optimized 16 color palette
No dithering



Optimized 16 color palette
FS dithering

Summary

- Intensity resolution
  - Each pixel has only “Depth” bits for colors/intensities
- Spatial resolution
  - Image has only “Width” x “Height” pixels
- Temporal resolution
  - Monitor refreshes images at only “Rate” Hz

- Halftoning and dithering
  - Reduce visual artifacts due to quantization
  - Distribute errors among pixels
    - Exploit spatial integration in our eye