Digital Halftoning

- Many image rendering technologies only have binary output. For example, printers can either "fire a dot" or not.
- Halftoning is a method for creating the illusion of continuous tone output with a binary device.
- Effective digital halftoning can substantially improve the quality of rendered images at minimal cost.

Thresholding

- Assume that the image falls in the range of 0 to 255.
- Apply a space varying threshold, $T(i, j)$.

$$b(i,j) = \begin{cases} 255 & \text{if } X(i,j) > T(i,j) \\ 0 & \text{otherwise} \end{cases} .$$

- What is $X(i, j)$?
- Lightness
  - Larger $\Rightarrow$ lighter
  - Used for display
- Absorbtance
  - Larger $\Rightarrow$ darker
  - Used for printing
- $X(i, j)$ will generally be in units of absorbtance.

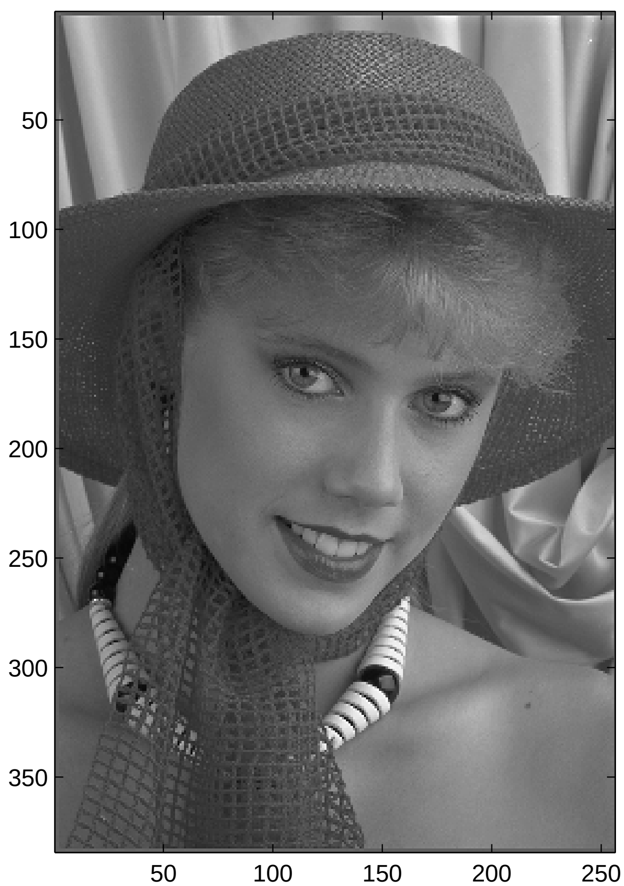
Constant Threshold

- Assume that the image falls in the range of 0 to 255.
- $255 \Rightarrow Black$ and $0 \Rightarrow White$
- The minimum squared error quantizer is a simple threshold
$$b(i,j) = \begin{cases} 255 & \text{if } X(i,j) > T \\ 0 & \text{otherwise} \end{cases} .$$
where $T = 127$.
- This produces a poor quality rendering of a continuous tone image.

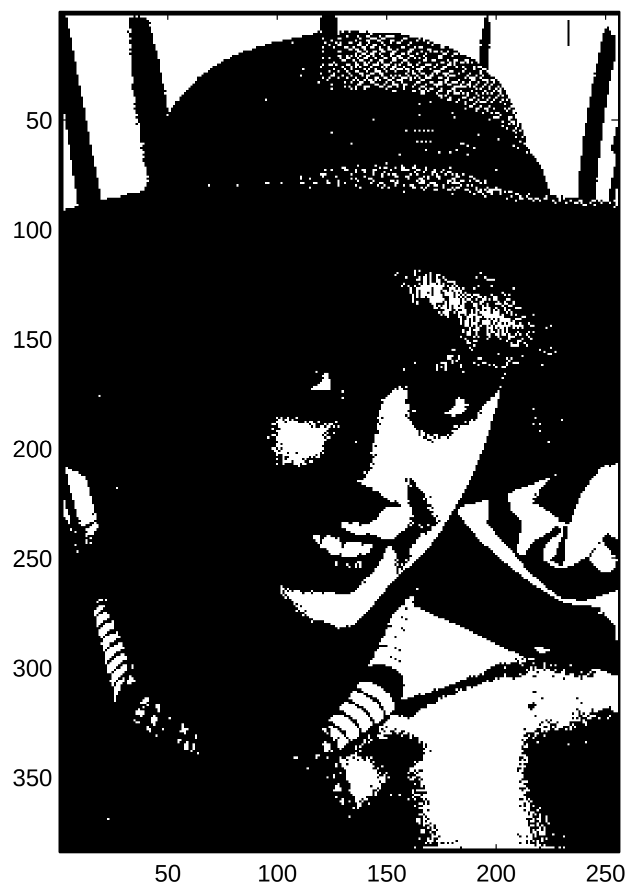
The Minimum Squared Error Solution

- Threshold each pixel
  - Pixel$>$ 127 Fire ink
  - Pixel$\leq$ 127 do nothing

Original Image



Thresholded Image

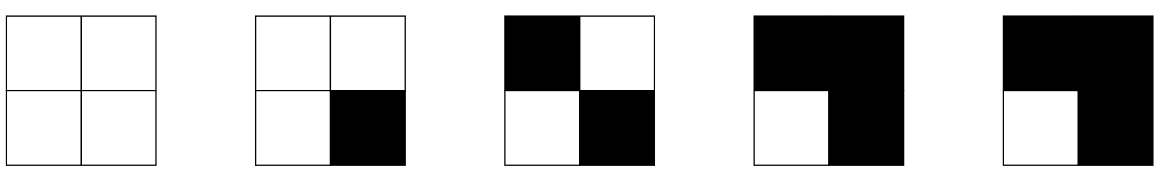


Ordered Dither

- For a constant gray level patch, turn the pixel “on” in a specified order.
- This creates the perception of continuous variations of gray.
- An $N \times N$ index matrix specifies what order to use.

$$I_2(i,j) = \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix}$$

- Pixels are turned on in the following order.



0 1 2 3 4

Implementation of Ordered Dither via Thresholding

- The index matrix can be converted to a "threshold matrix" or "screen" using the following operation.

$$T(i,j) = 255\frac{I(i,j)+0.5}{N^2}$$

- The $N \times N$ matrix can then be "tiled" over the image using periodic replication.

$$T(i \bmod N, j \bmod N)$$

- The ordered dither algorithm is then applied via thresholding.

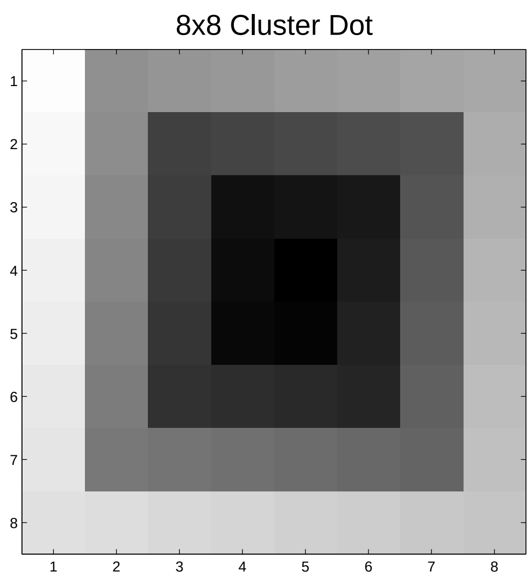
$$b(i,j) = \begin{cases} 255 & \text{if } X(i,j) > T(i \bmod N, j \bmod N) \\ 0 & \text{otherwise} \end{cases} .$$

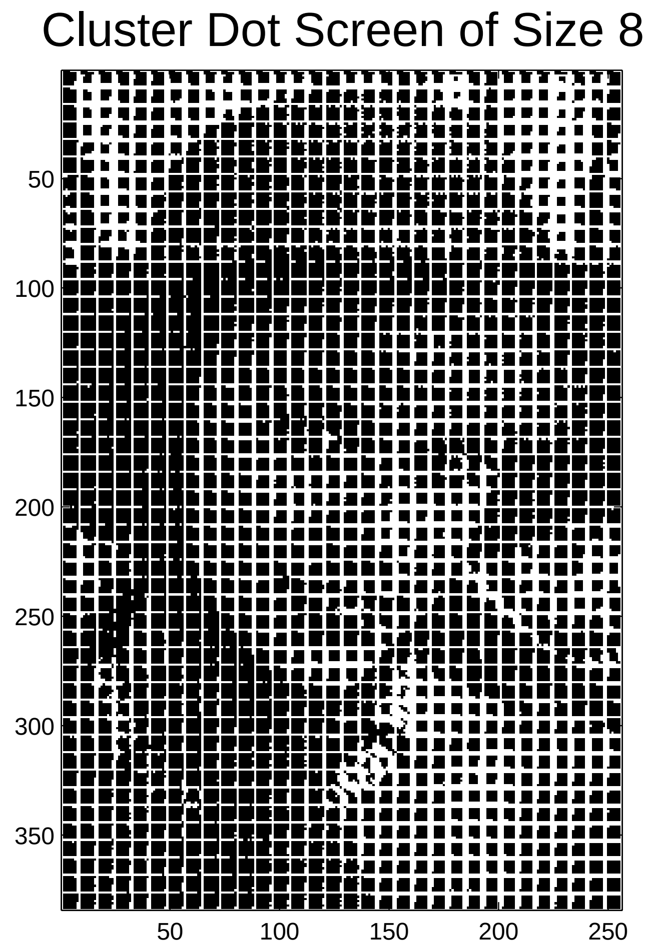
Clustered Dot Screens

- Definition: If the consective thresholds are located in spatial proximity, then this is called a "clustered dot screen.
- Example for 8×8 matrix:

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

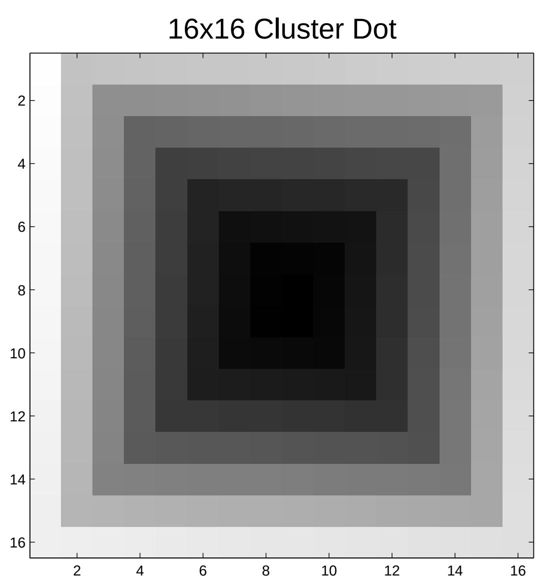
Example: 8×8 Clustered Dot Screening

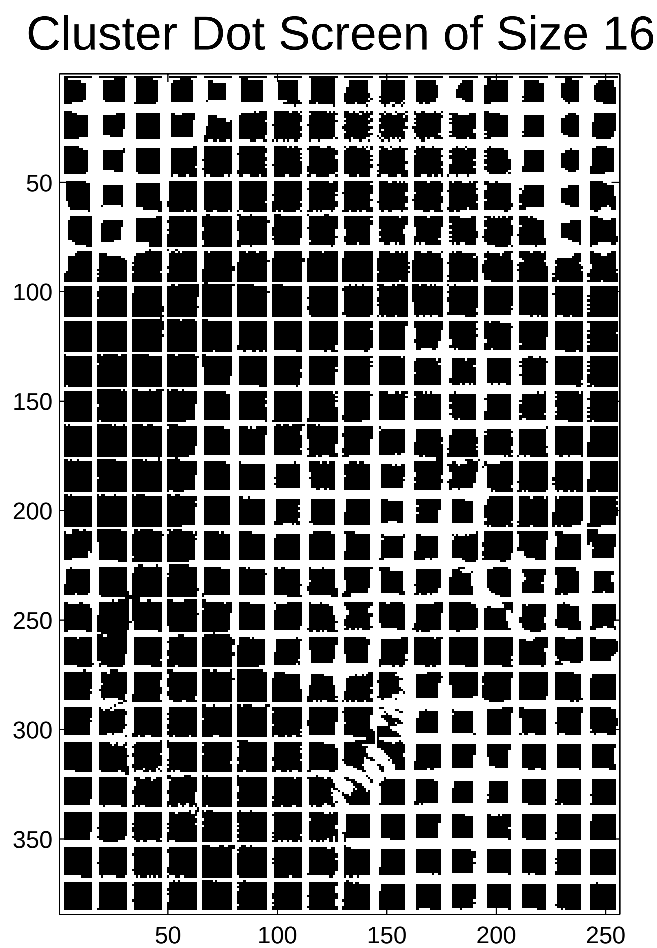




- Only supports 65 gray levels.

Example: 16×16 Clustered Dot Screening





- Support a full 257 gray levels, but has half the resolution.

Properties of Clustered Dot Screens

- Requires a trade-off between number of gray levels and resolution.
- Relatively visible texture
- Relatively poor detail rendition
- Uniform texture across entire gray scale.
- Robust performance with non-ideal output devices
  - Non-additive spot overlap
  - Spot-to-spot variability
  - Noise

Dispersed Dot Screens

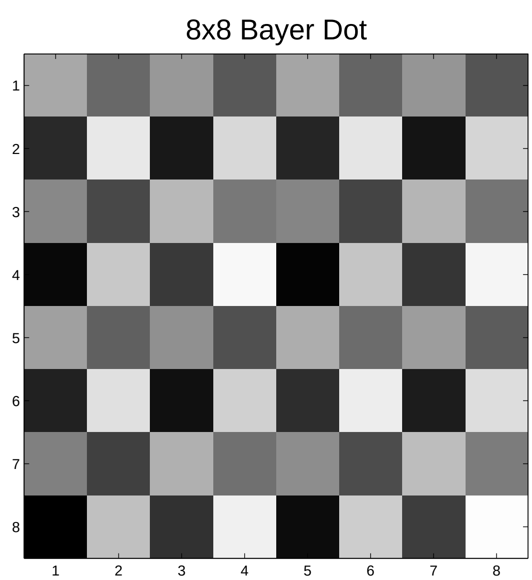
- Bayer's optimum index Matrix (1973) can be defined recursively.

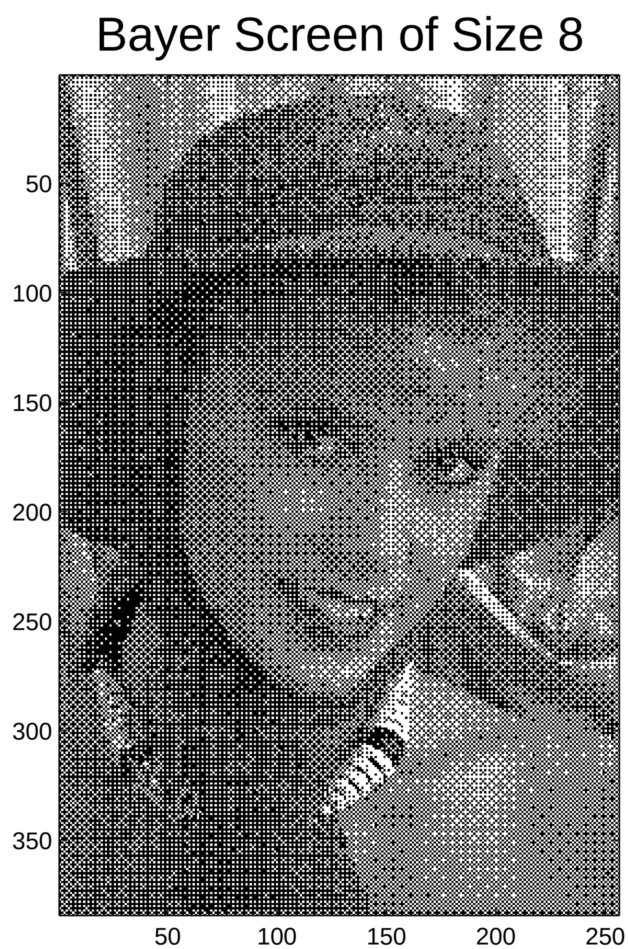
$$I_2(i,j) = \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix}$$

$$I_{2n}(i,j) = \begin{bmatrix} 4 * I_n(i,j) + 1 & 4 * I_n(i,j) + 2 \\ 4 * I_n(i,j) + 3 & 4 * I_n(i,j) \end{bmatrix}$$

- Yields finer amplitude quantization over larger area.
- Retains good detail rendition within smaller area.

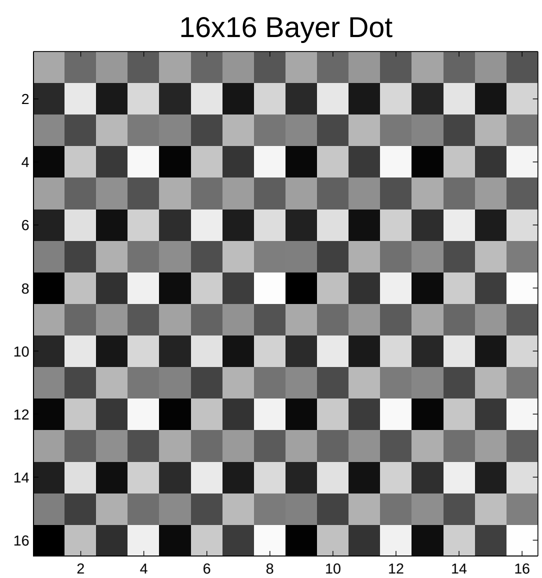
Example: 8×8 Bayer Dot Screening

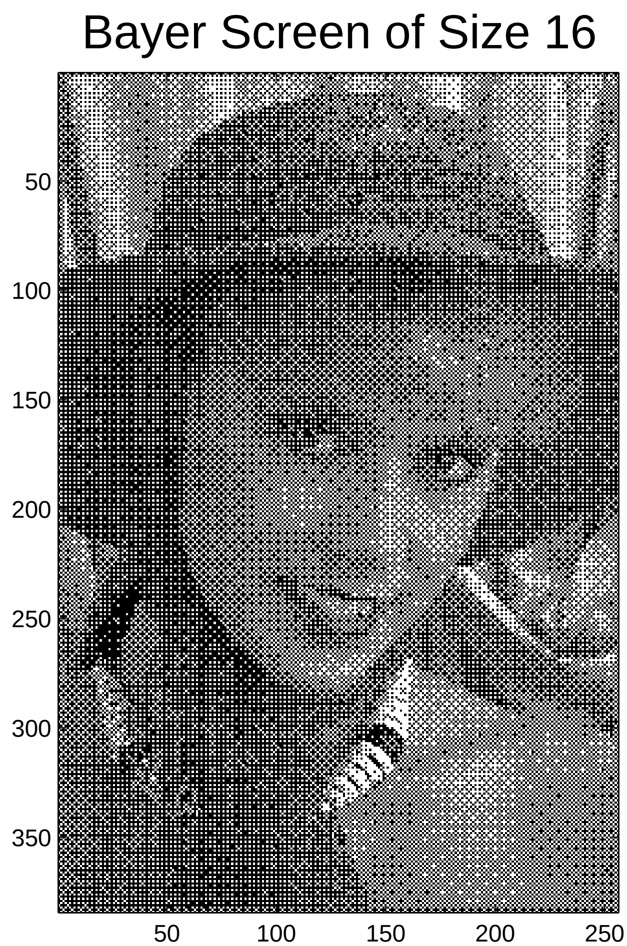




- Again, only 65 gray levels.

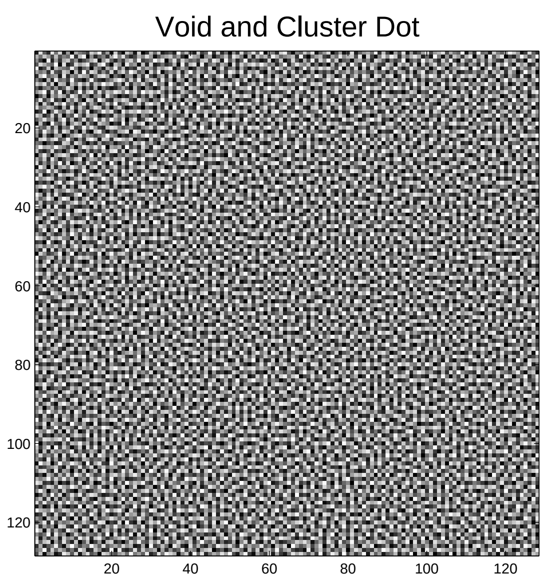
Example: 16×16 Bayer Dot Screening

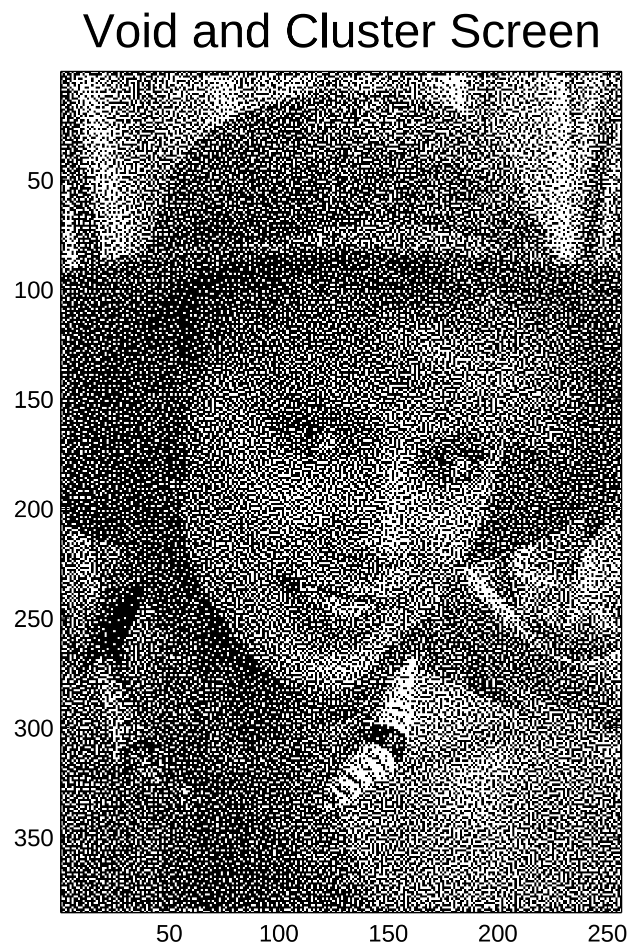




- Doesn't look much different than the 8×8 case.
- No trade-off between resolution and number of gray levels.

Example: 128×128 Void and Cluster Screen (1989)





- Substantially improved quality over Bayer screen.

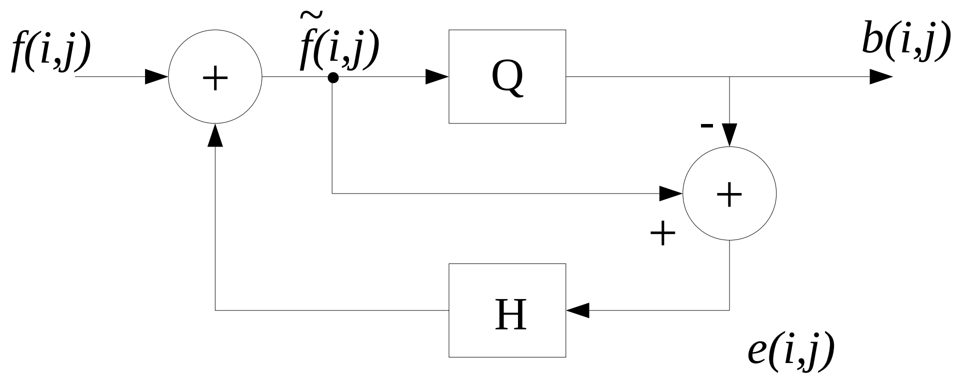
Properties of Dispersed Dot Screens

- Elliminate the trade-off between number of gray levels and resolution.
- Within any region containing K dots, the K thresholds should be distributed as uniformly as possible.
- Textures used to represent individual gray levels have low visibility.
- Improved detail rendition.
- Transitions between textures corresponding to different gray levels may be more visible.
- Not robust to non-ideal output devices
  - Requires stable formation of isolated single dots.

Error Diffusion

- Error Diffusion
  - Quantizes each pixel using a neighborhood operation, rather than a simple pointwise operation.
  - Moves through image in raster order, quantizing the result, and "pushing" the error forward.
  - Can produce better quality images than is possible with screens.

Filter View of Error Diffusion



- Equations are

$$b(i,j) = \begin{cases} 255 & \text{if } \tilde{f}(i,j) > T \\ 0 & \text{otherwise} \end{cases}$$

$$e(i,j) = \tilde{f}(i,j) - b(i,j)$$

$$\tilde{f}(i,j) = f(i,j) + \sum_{k,l \in S} h(k,l) e(i-k, j-l)$$

- Parameters
  - Threshold is typically $T = 127$.
  - $h(k.l)$ are typically choosen to be positive and sum to 1

Pixel View of Error Diffusion

1. Initialize $\tilde{f}(i,j) \leftarrow f(i,j)$
2. For each pixel in the image (in raster order)
   (a) Compute

$$b(i,j) = \begin{cases} 255 & \text{if } \tilde{f}(i,j) > T \\ 0 & \text{otherwise} \end{cases}$$

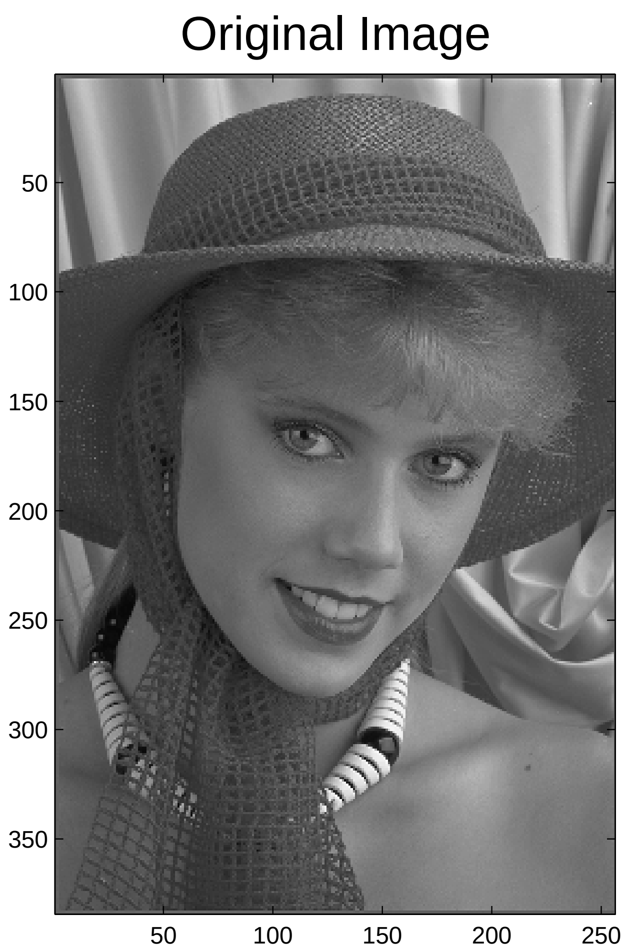
   (b) Diffuse error forward using the following scheme

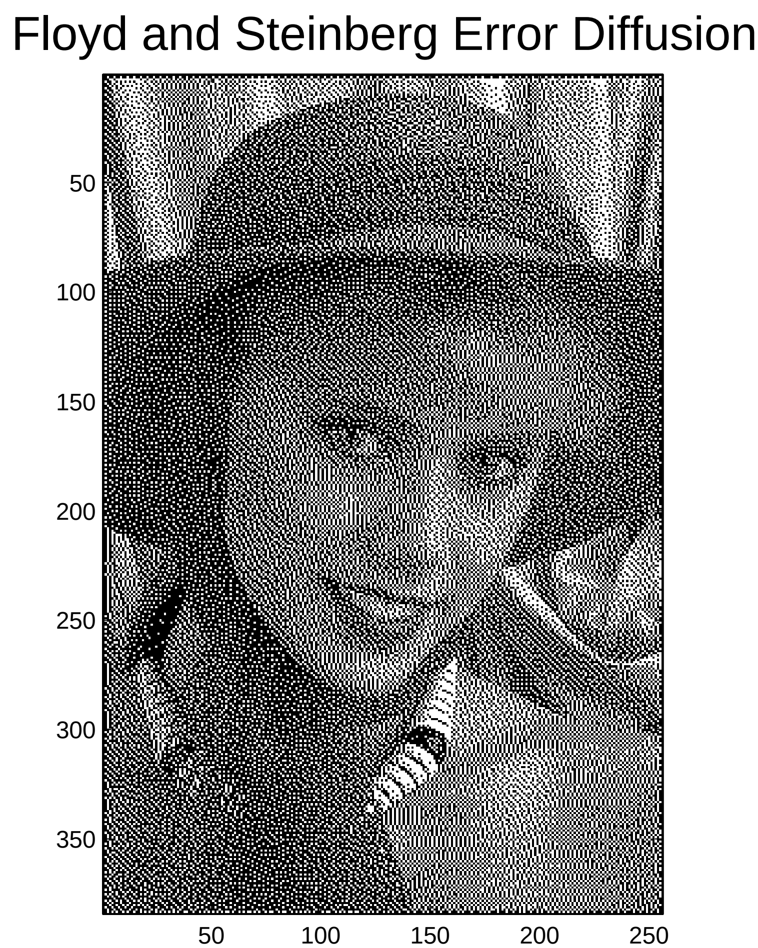
	$e(i,j)=\tilde{f}(i,j)-b(i,j)$	$\tilde{f}(i,j+1)$ += $h(0,1)*e(i,j)$
$\tilde{f}(i+1,j-1)$ += $h(1,-1)*e(i,j)$	$\tilde{f}(i+1,j)$ += $h(1,0)*e(i,j)$	$\tilde{f}(i+1,j+1)$ += $h(1,1)*e(i,j)$

3. Display binary image $b(i,j)$

Floyd Steinberg Error Diffusion (1976)

- Process pixels in neighborhoods by "diffusing error" and quantizing.





Error Image in Floyd Steinberg Error Diffusion

- Process pixels in neighborhoods by "diffusing error" and quantizing.