Image Processing

Michael Kazhdan

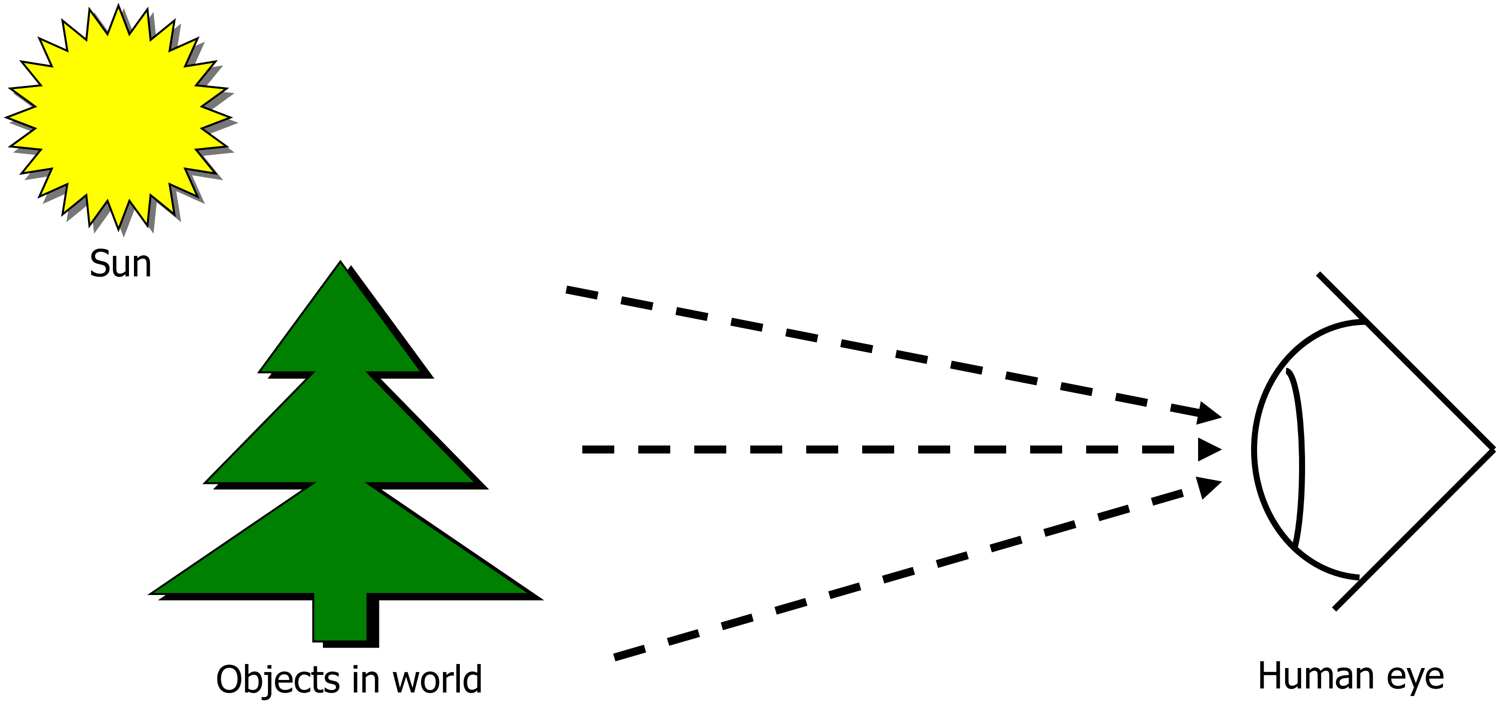
(601.457/657)

Outline

- Human Vision
- Image Representation
- Reducing Color Quantization Artifacts
- Basic Image Processing

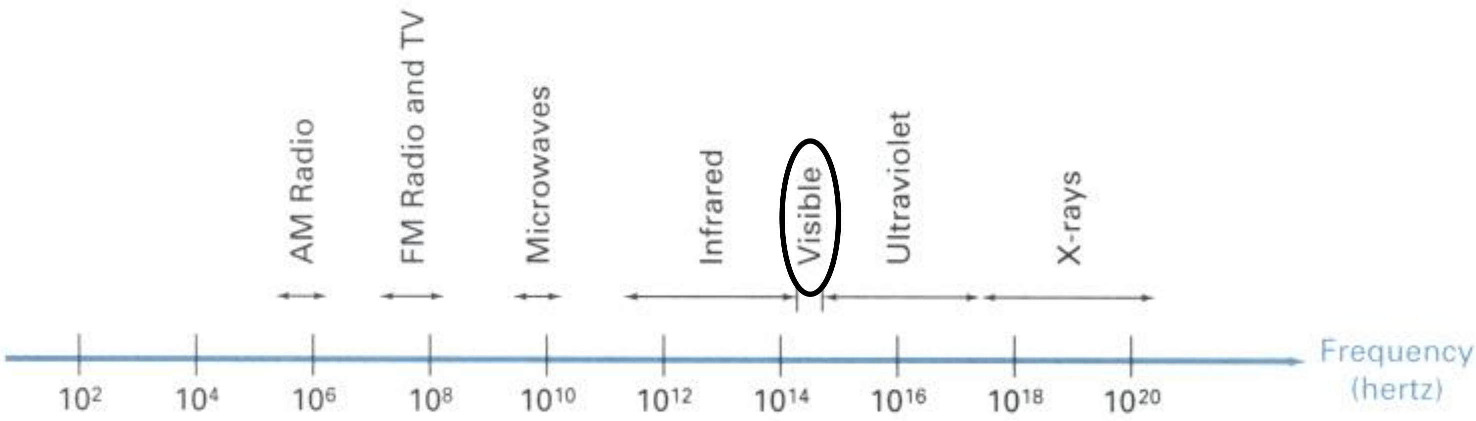
Human Vision

Model of Human Visual System



Electromagnetic Spectrum

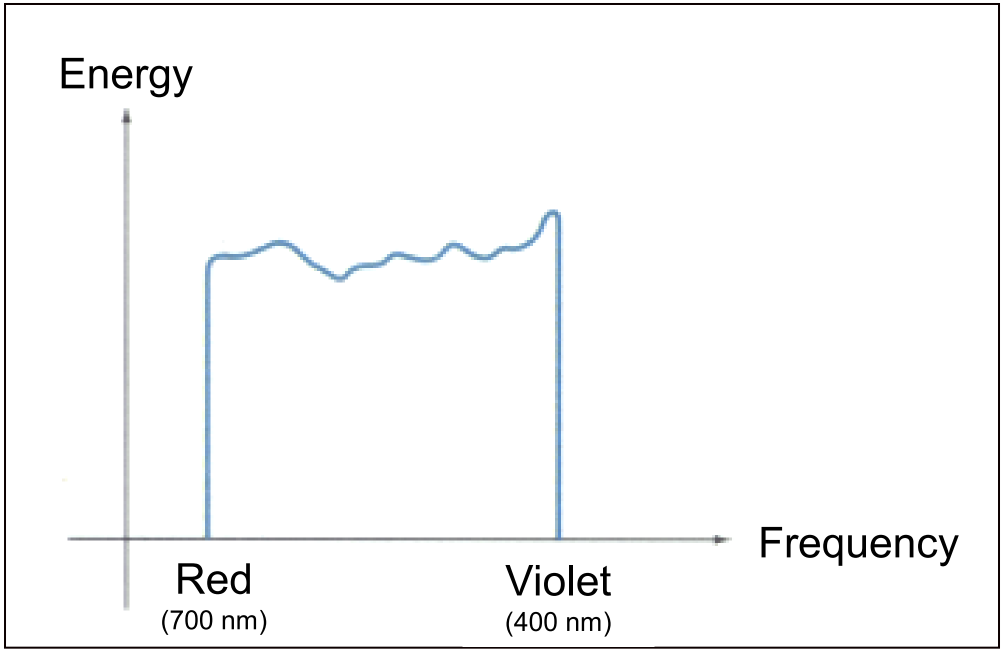
- Visible light frequencies range between ...
  - Red = 4.3×10^{14} hertz (700nm)
  - Violet = 7.5×10^{14} hertz (400nm)



Figures 15.1 from H&B

Visible Light

- What we see as “color” is described by the distribution/spectrum of light across the visible range.



White Light

Figure 15.3 from H&B

Human Vision

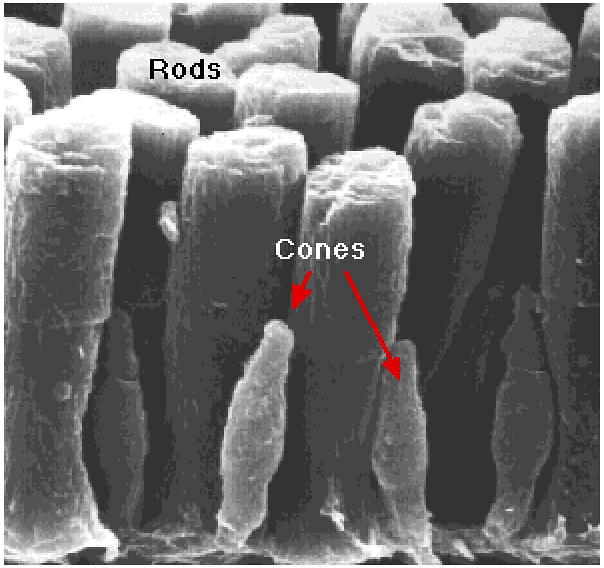
The human retina contains two types of photoreceptors, cones and rods.

Cones:

- 6-7 million cones in the retina
- Responsible for **photopic** vision
- Color sensitive:
  - 64% red, 32% green, 2% blue
- Distributed in the fovea centralis

Rods:

- 120 million rods in the retina
- 1000x more light sensitive than cones
- Responsible for **scotopic** vision
- Short-wavelength sensitive
- Responsible for peripheral vision



Tristimulus Theory of Color

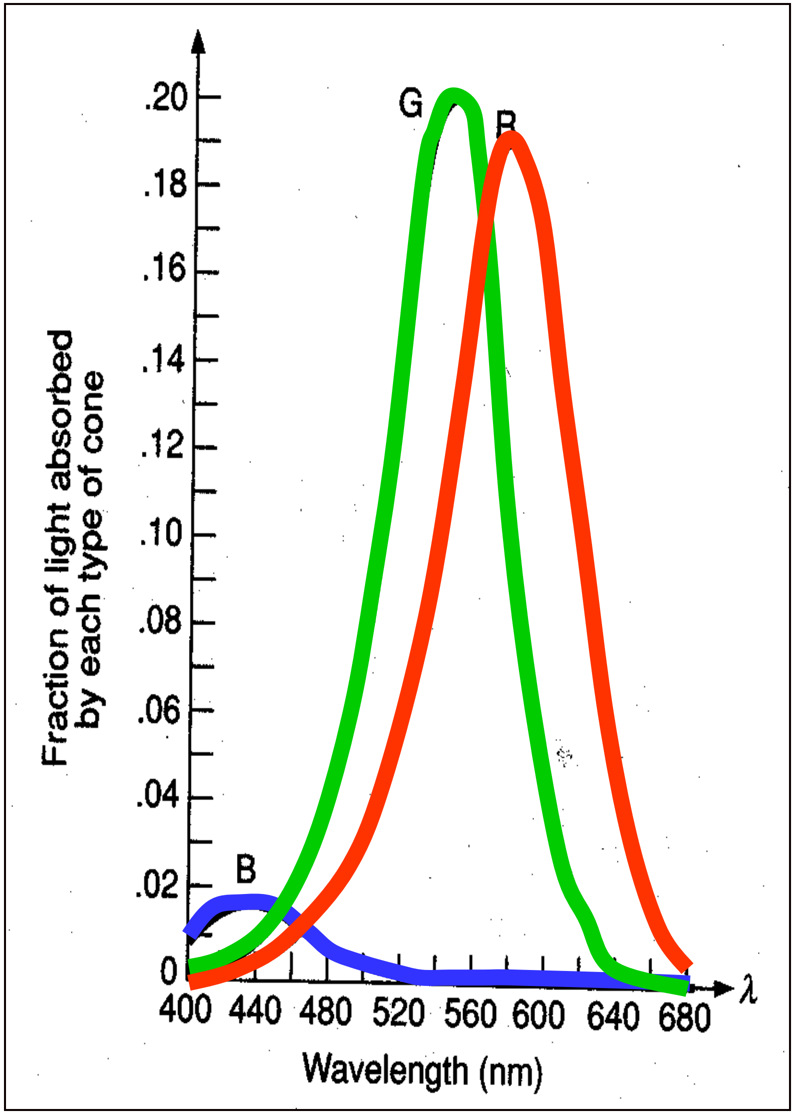


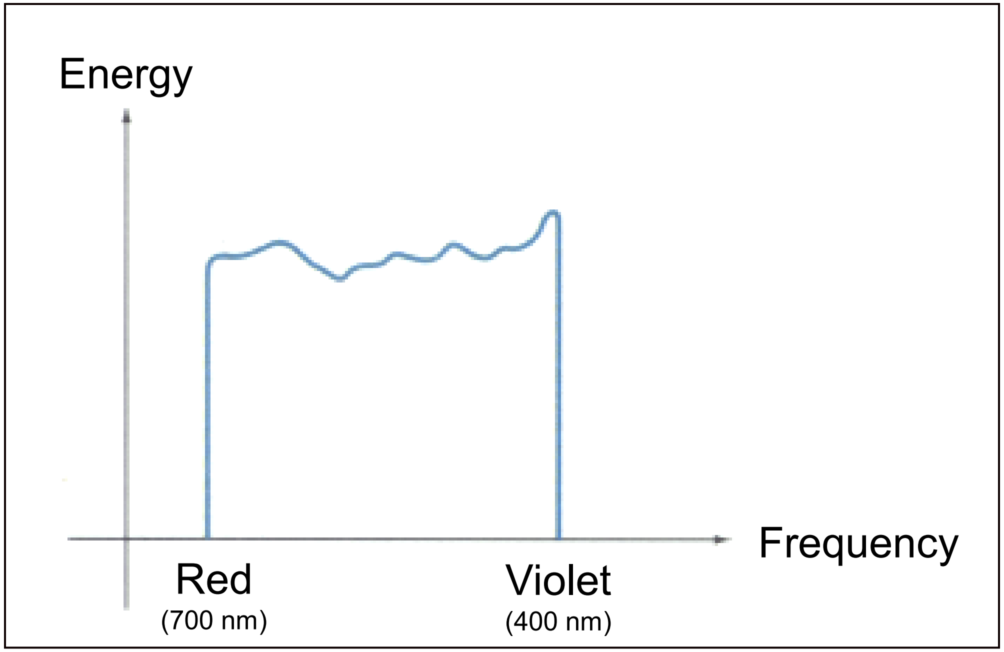
Figure 13.18 from FvDFH

Spectral-response functions of each of the three types of cones on the human retina.

This motivates encoding color as a combination of red, green, and blue (RGB).

Visible Light

- What we see as "color" is described by the distribution/spectrum of light across the visible range.



White Light

Figure 15.3 from H&B

Visible Light

- What we see as “color” is described by the distribution/spectrum of light across the visible range.

This does not mean that we can see the difference between all spectral distributions in the visible range.

Metamers = Two spectral distributions that look the same

Red
(700 nm)

Violet
(400 nm)

White Light

Figure 15.3 from H&B

Outline

- Human Vision
- **Image Representation**
- Reducing Color Quantization Artifacts
- Basic Image Processing

Image Representation

What is an image?

Image Representation

An image is a 2D rectilinear array of pixels:

A *width* x *height* array where each entry of the array stores a single pixel.

Continuous image

Digital image

Image Representation

What is a pixel?

Continuous image

Digital image

Image Representation

A pixel is something that represents color

- Luminance pixels
  - » Grey-scale images (aka “intensity images”)
- Red, Green, Blue pixels (RGB)
  - » Color images

Pixel channel value:

» Conceptually, in the continuous range [0,1)*

» In practice, in a discrete range (e.g. {0,1,…,254,255})

*[0,1) is the continuous range of numbers including 0 but not including 1.

Resolutions

- Space: *width* x *height* pixels
- Intensity/Color: n bits per pixel

	width x height	bit depth
Handheld	2220 x 1080	24
Monitor (4K)	3840 x 2160	24
CCDs	6000 x 4000	36
Laser Printer	6600 x 5100	3

Resolutions

- Space: *width* x *height* pixels
- Intensity/Color: n bits per pixel
- Time: n Hz (fps)

	width x height	bit depth	Hz
Handheld	2220 x 1080	24	60
Monitor (4K)	3840 x 2160	24	144
CCDs	6000 x 4000	36	50
Laser Printer	6600 x 5100	3	-

Image Quantization Artifacts

- With only a small number of bits associated to each color channel of a pixel there is a limit to intensity resolutions of an image
  - A black and white image allocates a single bit to the luminance channel of a pixel.

  ⇒ The number of different colors that can be represented by a single pixel is 2.
  - A 24-bit image allocates 8 bits each to the red, green, and blue channels of a pixel.

  ⇒ The number of different colors that can be represented by a single pixel is 2^{24}=~16,000,000.

Outline

- Human Vision
- Image Representation
- **Reducing Color Quantization Artifacts**
  - **Halftoning and Dithering**
- Basic Image Processing

Pixel Representation

<u>Disclaimer</u>:

In the next slides, we will assume that images are gray-scale (single-channel).

We assume that the <u>original</u> image has continuous (italics) pixel values :

$$I(x, y) \in [0,1)$$

We will represent them using a finite number of bits per pixel so that the values are discrete (bold):

$$\mathbf{I}(x, y) \in \{0, \dots, n-1\}$$

Discretization

In particular, using b bits per pixel, we can represent $n = 2^b$ different colors.

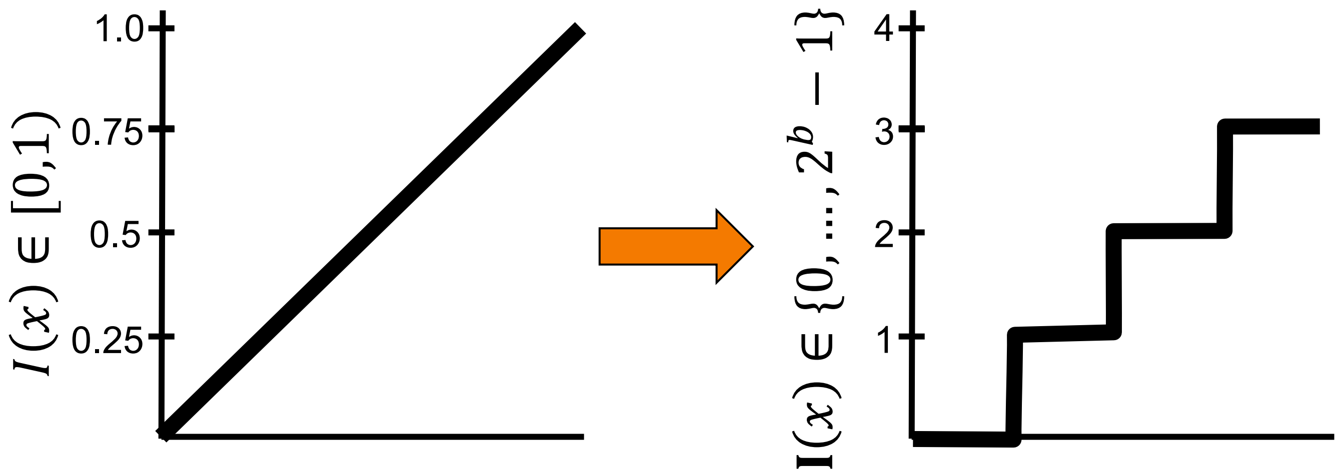
$$\Downarrow$$

$$\mathbf{I}(x, y) \in \{0, \dots, 2^b - 1\}$$

Quantization

With b bits per pixel, you can coarsely represent an image by quantizing the color values:

$$\mathbf{I}(x,y) = Q_b\big(I(x,y)\big) = \mathrm{floor}\big(I(x,y)\cdot 2^b\big)$$



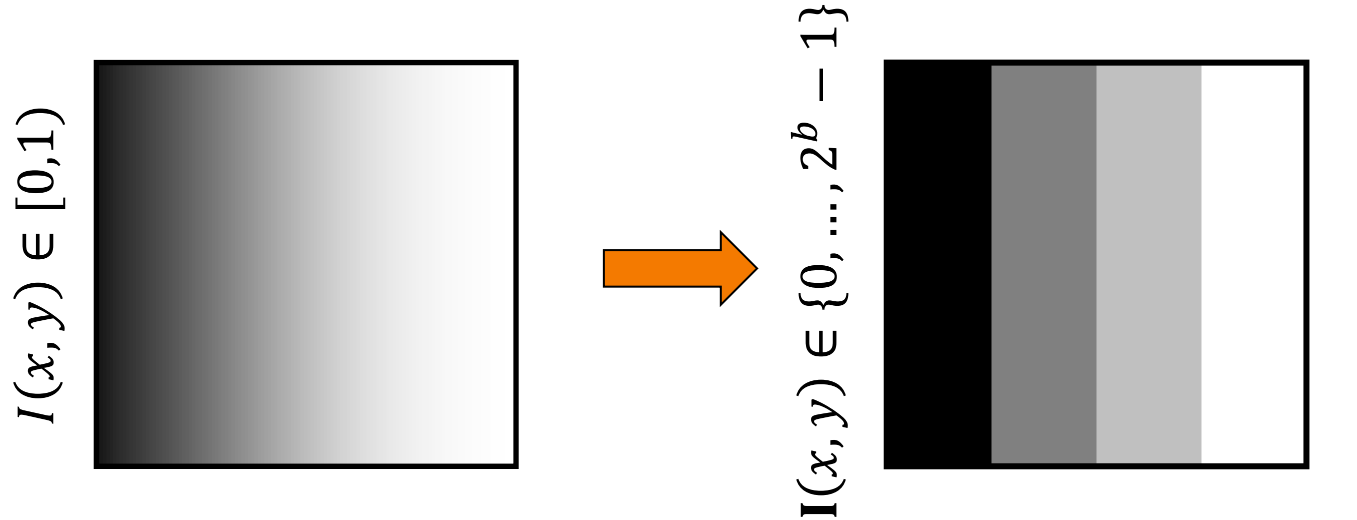
$b = 2$ bits per pixel

Note: We quantize so that each “bucket” gets the same measure of values.

Quantization

With b bits per pixel, you can coarsely represent an image by quantizing the color values:

$$\mathbf{I}(x,y) = Q_b\big(I(x,y)\big) = \text{floor}\big(I(x,y) \cdot 2^b\big)$$



$b = 2$ bits per pixel

Quantization

Image with decreasing bits per pixel

- Quantization can cause contours <u>away</u> from image edges.

$b = 8$ bits

$b = 4$ bits

$b = 2$ bits

$b = 1$ bits

Reducing Color Quantization Artifacts

For (still) images, the combination of image resolution and intensity/color resolution define the total informational content.

Key Idea:

We can trade off between these.

Reducing Effects of Quantization

Trade spatial resolution for intensity resolution:

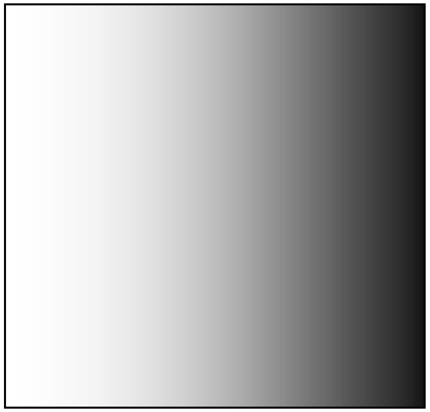
- Half-toning
- Dithering

Both exploit spatial integration in our eye to display a greater range of *perceptible* intensities.

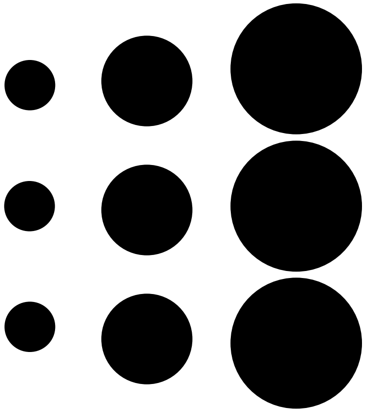
Half-Toning: Historically

Half-toning:

- Consider the average intensity in a **region**
- Draw varying-size dots representing the average
  - » Area of dots determined by the average intensity in the covered area

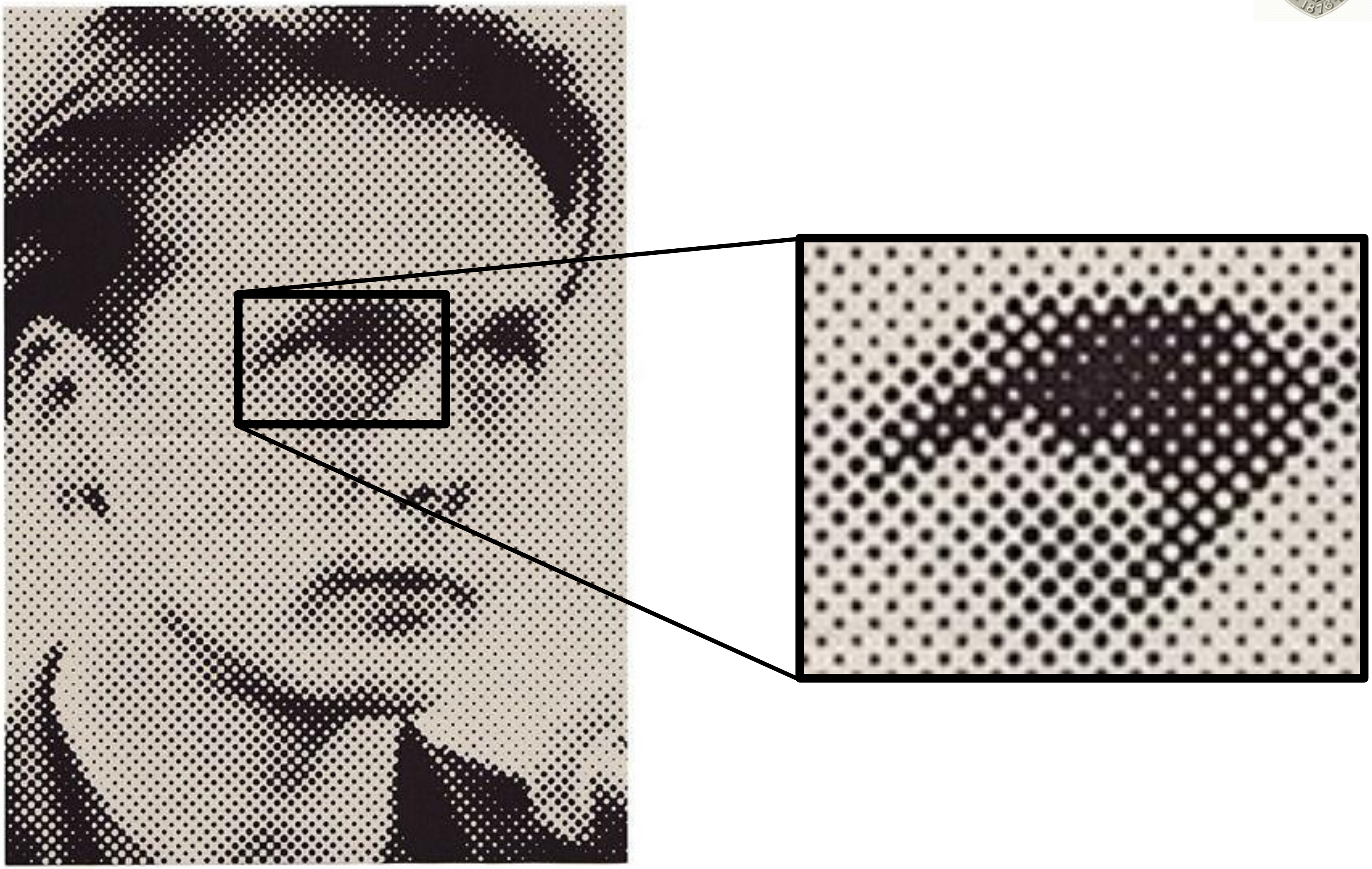


$I(x,y)$



$\mathbf{I}(x,y)$

Half-Toning: Historically



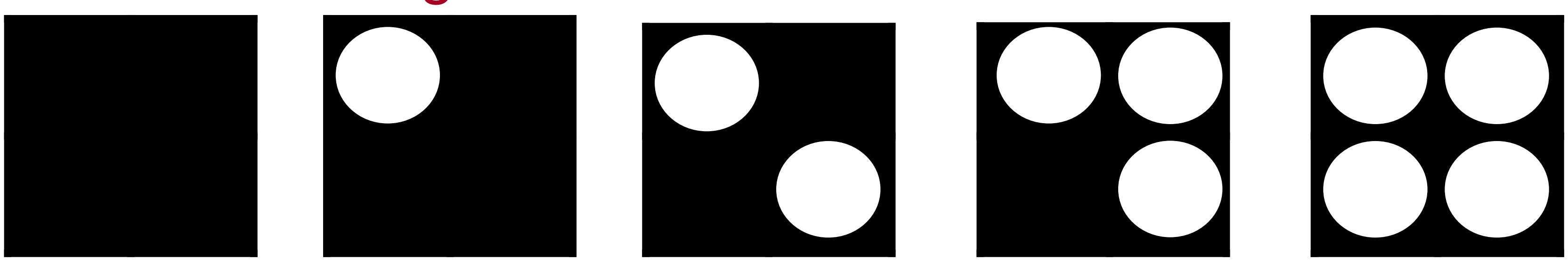
https://i.pinimg.com/736x/e4/18/83/e41883601f5e242a10e71270443be07b.jpg

Half-Toning: Digital

- Consider the **average** intensity in a $k \times k$ block
- Turn on a variable number of the pixels in the block
  - » Number of pixels determined by the average intensity in the covered area

<u>Note</u>:

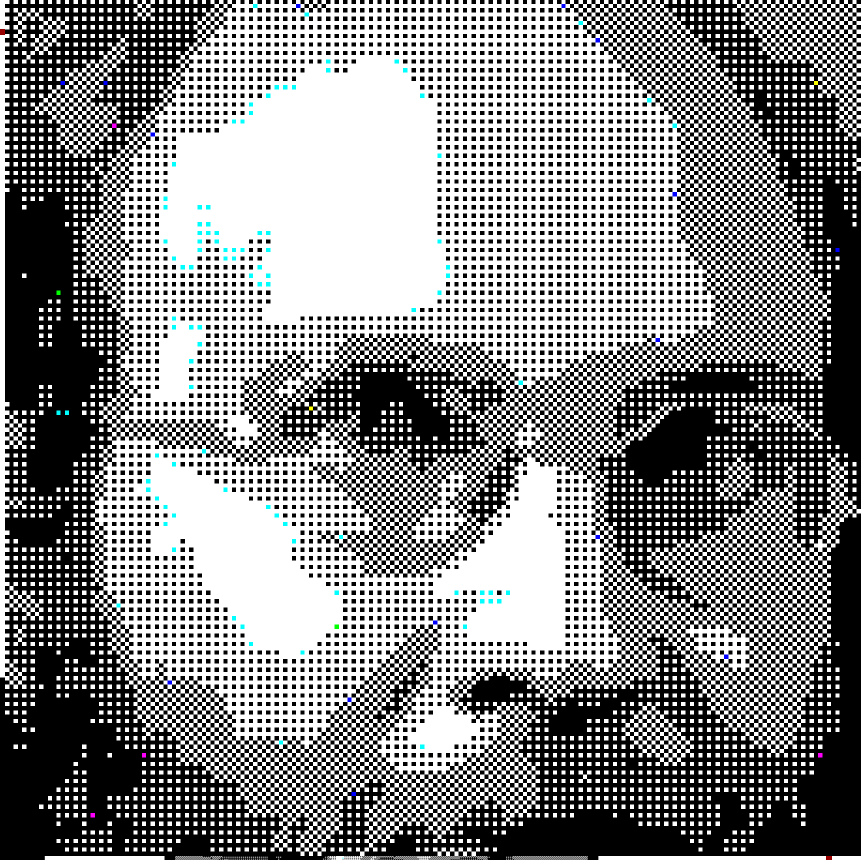
- Half-toning pattern matters
  - » Want to avoid vertical, horizontal lines
- Loss of information
  - » 16 configurations → 5 intensities



$0 \le I < 0.2$ $\quad$ $0.2 \le I < 0.4$ $\quad$ $0.4 \le I < 0.6$ $\quad$ $0.6 \le I < 0.8$ $\quad$ $0.8 \le I < 1$

Half-Toning: Digital

- The use of a regular grid still causes contouring.



Original
(8 bits)

Quantized
(1 bit)

Half-toned
(1 bit)

Dithering

Per pixel, round to the floor/ceiling value.

- Similar to quantization:
  - » Rounding based on pixel value
- Unlike quantization:
  - » Rounding threshold is not fixed

Ordered Dither (Binary Displays)

Pseudo-random quantization thresholds described by a $k \times k$ matrix D_k with entries in the range $\{1, \ldots, k^2\}$

// For a pixel at position (x,y):

// Locate the index in the block:

$$i = x \bmod k$$

$$j = y \bmod k$$

// Get fractional component

$$e = I(x, y)$$

// Round up/down

$$\text{if}\left(e > \frac{D_k(i,j)}{k^2+1}\right) \mathbf{I}(x, y) = 1$$

$$\text{else} \quad \mathbf{I}(x, y) = 0$$

$$D_2 = \begin{bmatrix} 1 & 3 \\ 4 & 2 \end{bmatrix}$$

Ordered Dither (b-Bit Displays)

Pseudo-random quantization thresholds described by a $k \times k$ matrix D_k with entries in the range $\{1, \ldots, k^2\}$

// For a pixel at position (x,y):

// Locate the index in the block:

$$i = x \bmod k$$

$$j = y \bmod k$$

// Get fractional component

$$c = I(x, y) \cdot (2^b - 1)$$

$$e = c - \text{floor}(c)$$

// Round up/down

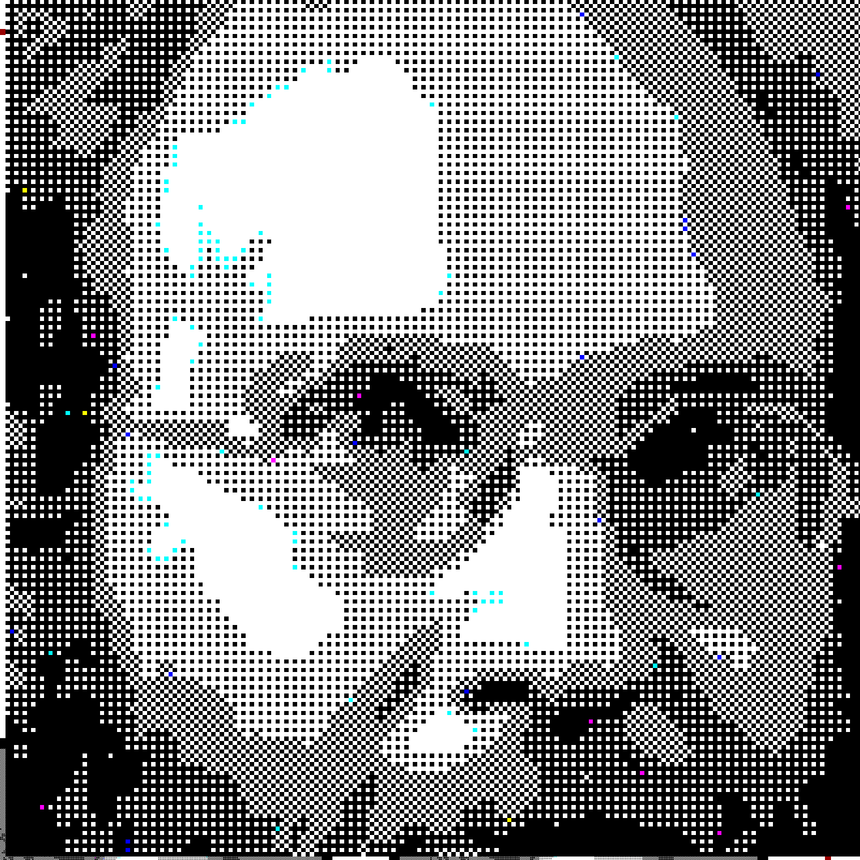
if$\left(e > \frac{D_k(i,j)}{k^2+1}\right)$ $\mathbf{I}(x, y) = \text{ceil}(c)$

else $\mathbf{I}(x, y) = \text{floor}(c)$

$$D_2 = \begin{bmatrix} 1 & 3 \\ 4 & 2 \end{bmatrix}$$

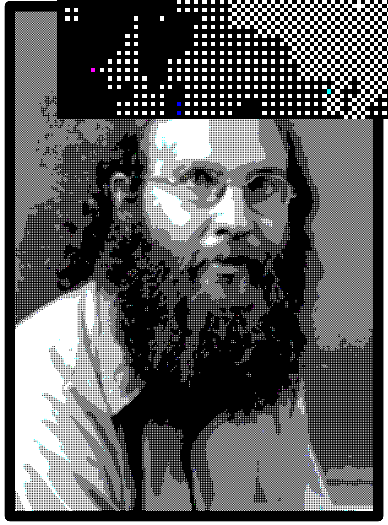
Ordered Dither

- Similar to half-toning. (And similar issue with contouring due to regularity.)

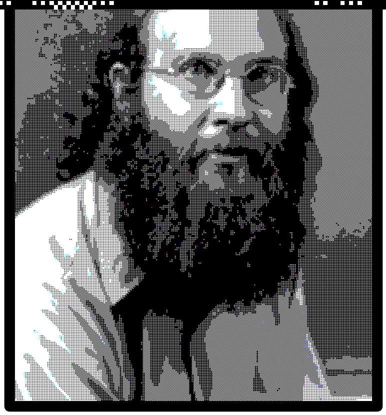


Original
(8 bits)

Uniform
(1 bit)



Half-toned
(1 bit)

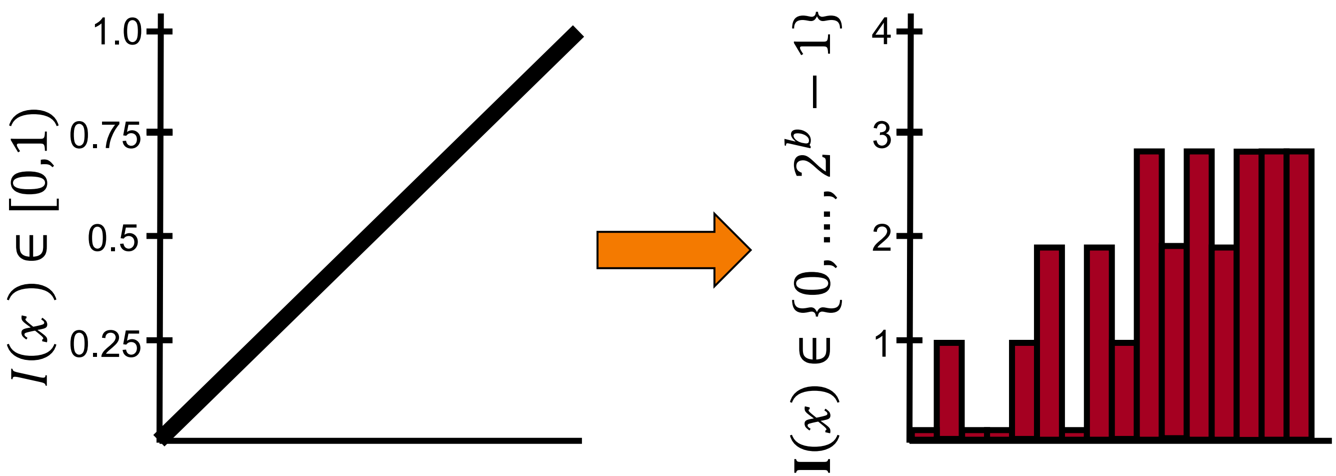


Ordered
(1 bit)

Random Dither

Rounding threshold defined randomly, rather than by a regular grid:

- The local average color is unchanged
- Because noise is *zero-mean*, our eye averages it out



If a pixel is black, adding random noise, you are less likely to turn it into a white pixel then if the pixel were gray.

$$\mathbf{I}(x,y) = Q_b\left(I(x,y) + \frac{\text{noise}(x,y)}{2^b}\right)$$

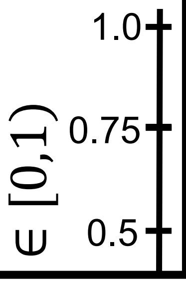
Random Dither

Rounding threshold defined randomly, rather than by a regular grid:

- The loc

Q: How much noise?

- out



A: Just enough to make it to the previous/next intensity value:

$$\text{noise}(x,y) \in (-1.0, 1.0)$$

s black,
ndom
u are

Warning!

Adding noise may take you out of the $[0,1)$ range.

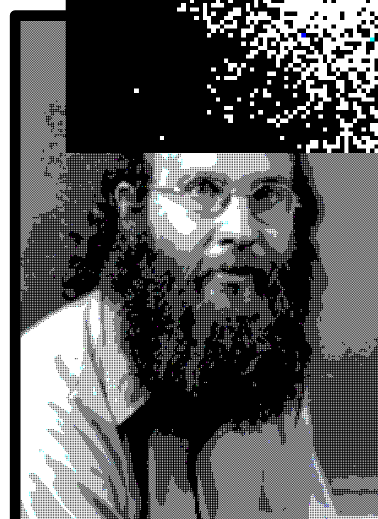
$$\mathbf{I}(x,y) = Q_b\left(I(x,y) + \boxed{\frac{\text{noise}(x,y)}{2^b}}\right)$$

Ordered Dither

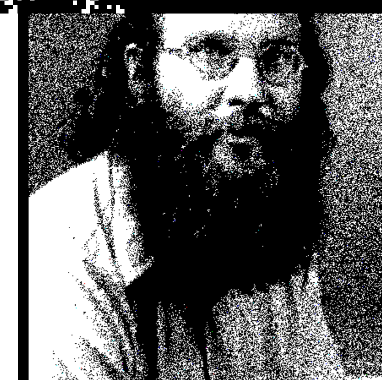
- No structure resulting from regular thresholding

Original
(8 bits)

Uniform
(1 bit)



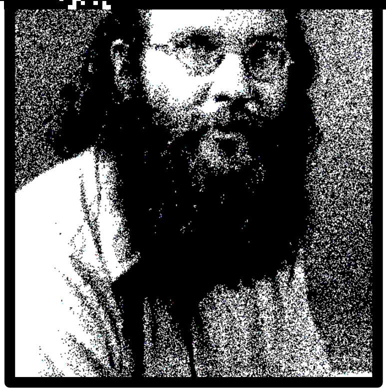
Ordered
(1 bit)



Random
(1 bit)

Ordered Dither

- No structure resulting from regular thresholding



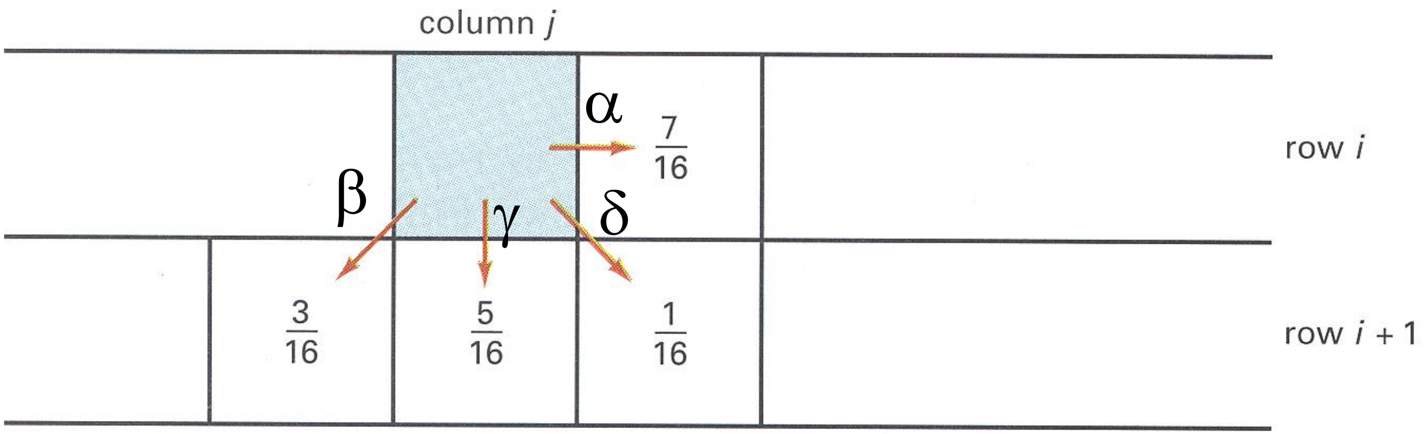
Original
(8 bits)

Limitation:
Do not correct when quantization error is introduced.

Error <u>Diffusion</u> Dither

Floyd Steinberg Dither:

- Process pixels left-to-right and top-to-bottom
- Quantize at current pixel
- **Distribute quantization residual over unvisited (right and below) neighboring pixels**



$$\alpha + \beta + \gamma + \delta = 1$$

Figure 14.42 from H&B

Error Diffusion Dither

```
for( int j=0 ; j<height ; j++ ) for ( int i=0 ; i<width ; i++ )
{
```

$Dest_{i,j}$ = quantize($Source_{i,j}$)

error = $Source_{i,j} - Dest_{i,j}$

$Source_{i+1,j}$ += α * error

$Source_{i-1,j+1}$ += β * error

$Source_{i,j+1}$ += γ * error

$Source_{i+1,j+1}$ += δ * error

```
}
```

$$\alpha = \frac{7}{16}$$

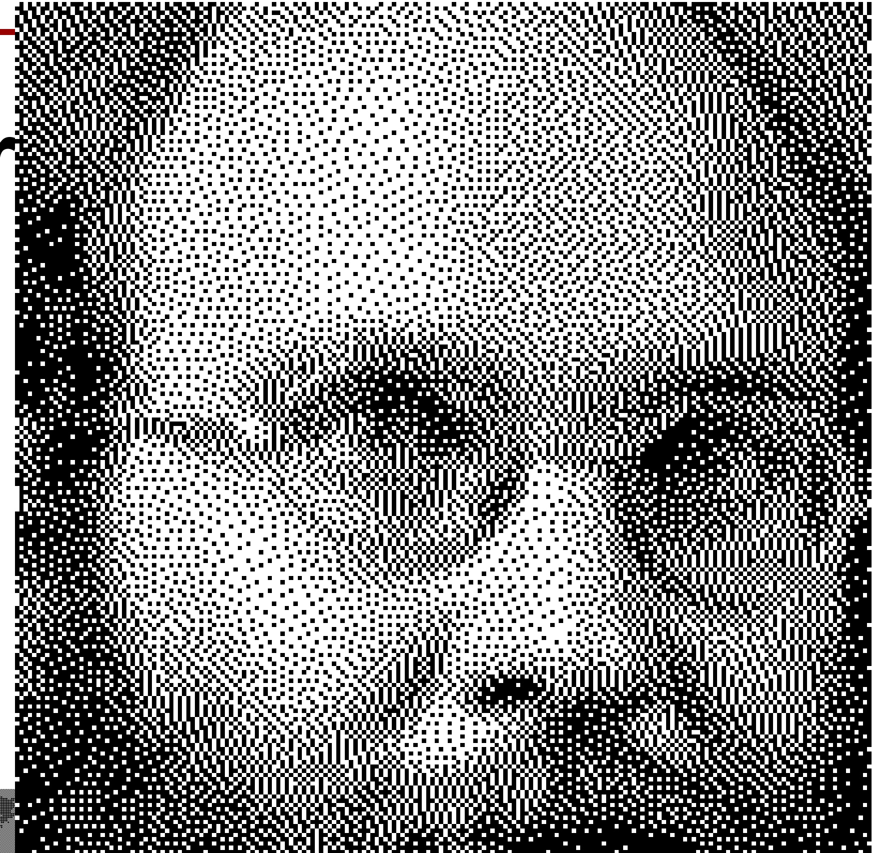
$$\beta = \frac{3}{16}$$

$$\gamma = \frac{5}{16}$$

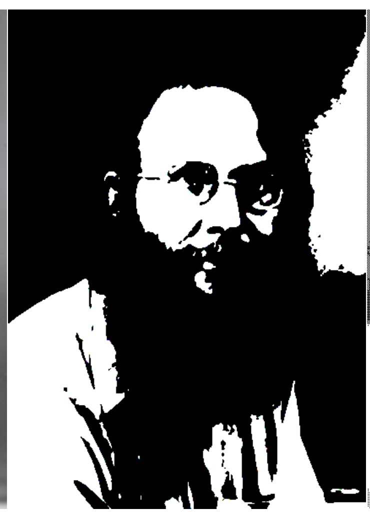
$$\delta = \frac{1}{16}$$

Floyd-Steinberg Dither

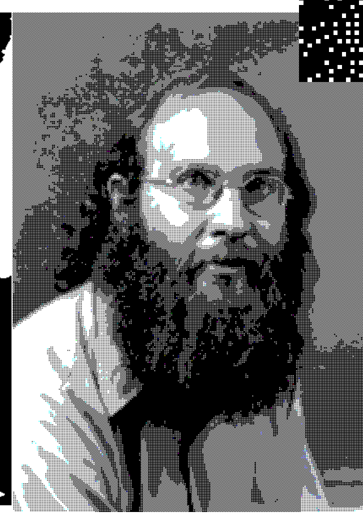
Error Diffusion Dither



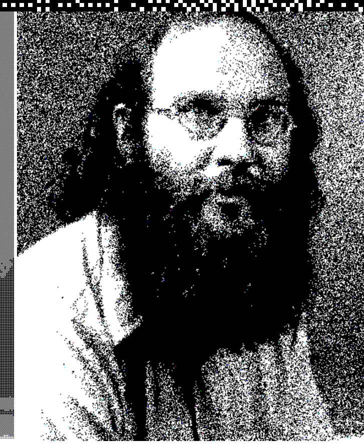
Original
(8 bits)



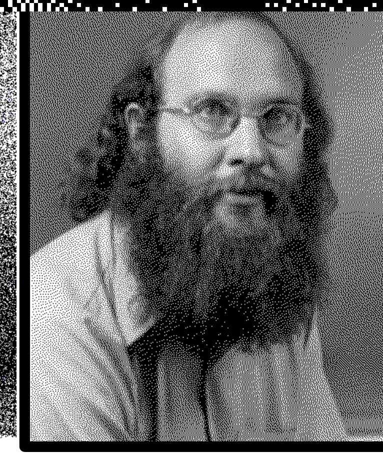
Uniform
(1 bit)



Ordered
(1 bit)



Random
(1 bit)



Floyd-Steinberg
(1 bit)

Outline

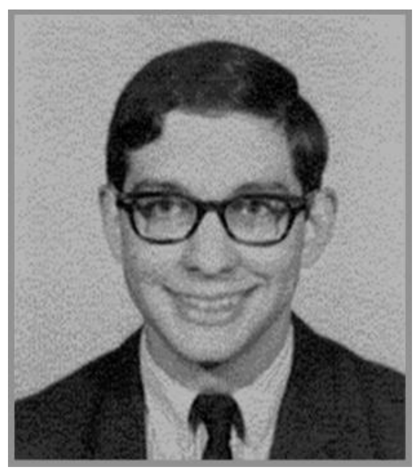
- Human Vision
- Image Representation
- Reducing Color Quantization Artifacts
- **Basic Image Processing**
    - **Single Pixel Operations**

Computing Grayscale

- The human retina perceives red, green, and blue as having different levels of brightness.

- To compute the *luminance* (perceived brightness) of a pixel, we take the <u>weighted</u> average of the RGB values:
  - $\mathbf{L}_p = 0.30 \cdot \mathbf{r}_p + 0.59 \cdot \mathbf{g}_p + 0.11 \cdot \mathbf{b}_p$

Original



Grayscale

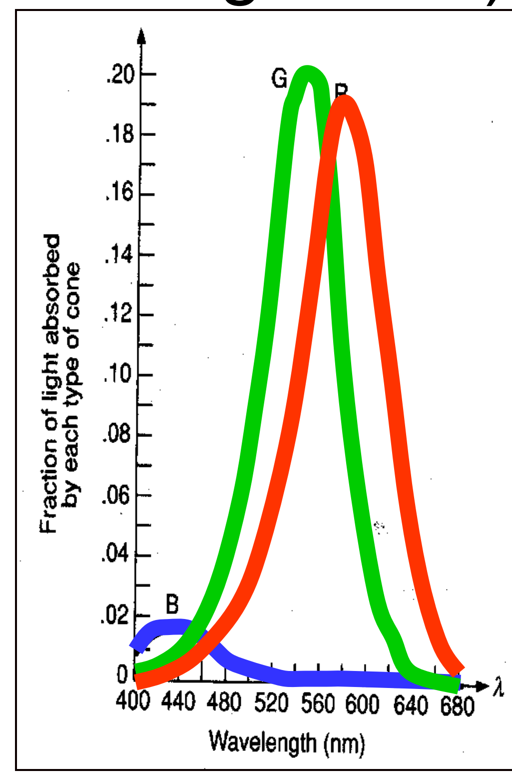


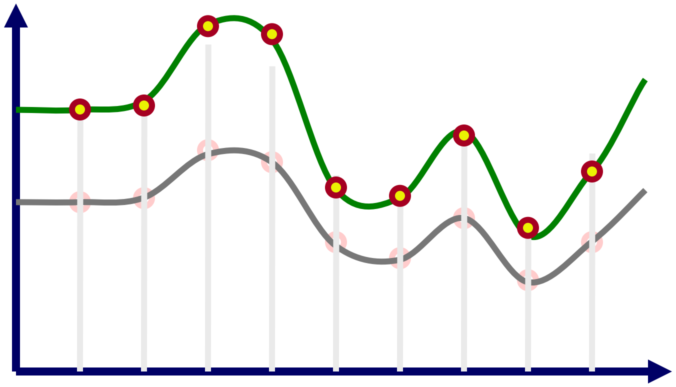
Figure 13.18 from FvDFH

Adjusting Brightness

- Scale pixel components
  - [WARNING] Clamp to range -- e.g. to [0,255)

Original

Brighter

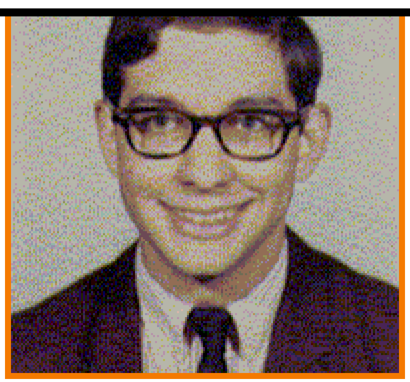


$$\mathbf{I}_p \leftarrow \mathbf{I}_p \cdot \alpha$$

Adjusting Brightness

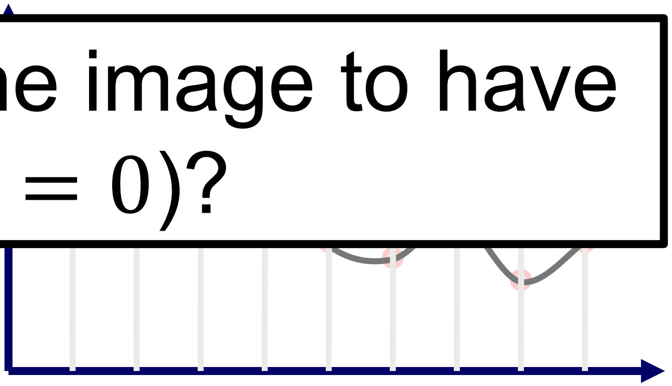
- Scale pixel components
  - [WARNING] Clamp to range -- e.g. to [0,255)

What do we get if we set the image to have no brightness ($\alpha = 0$)?



Original

Brighter

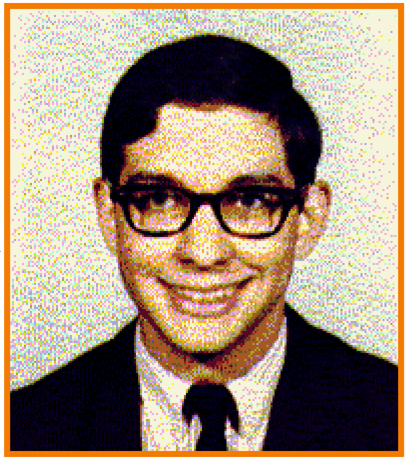


$$\mathbf{I}_p \leftarrow \mathbf{I}_p \cdot \alpha$$

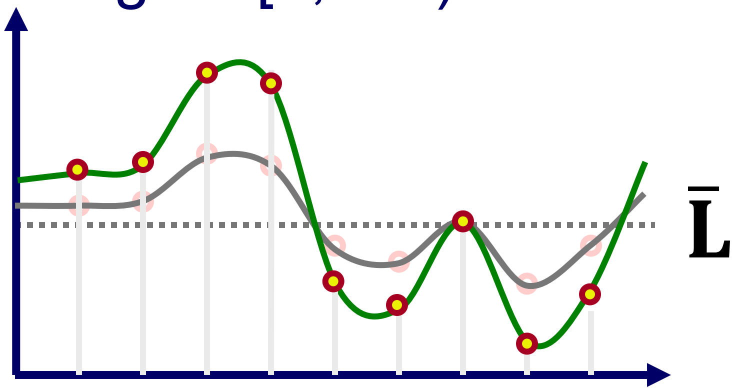
Adjusting Contrast

- Compute <u>mean image</u> luminance $\bar{\mathbf{L}}$, averaged over all pixels
  - $\bar{\mathbf{L}} = \text{Average}(0.30 \cdot \mathbf{r}_p + 0.59 \cdot \mathbf{g}_p + 0.11 \cdot \mathbf{b}_p)$
- Scale <u>difference</u> from $\bar{\mathbf{L}}$ for each pixel component
  - [WARNING] Clamp to range -- e.g. to [0,255)

Original



More Contrast

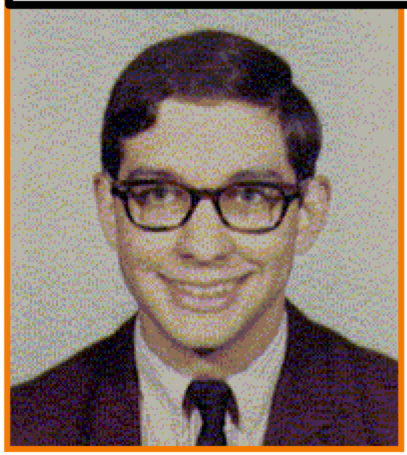


$$\mathbf{I}_p \leftarrow \left(\mathbf{I}_p - \bar{\mathbf{L}}\right)\alpha + \bar{\mathbf{L}}$$

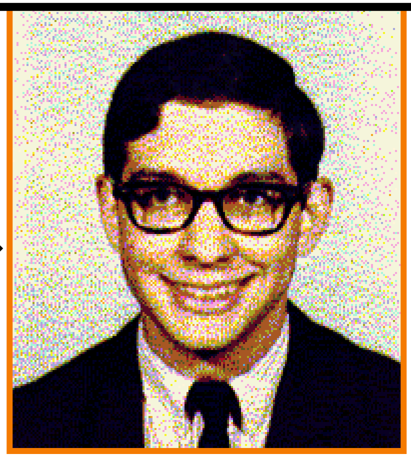
Adjusting Contrast

- Compute <u>mean image</u> luminance $\bar{\mathbf{L}}$, averaged over all pixels
  - $\bar{\mathbf{L}} = \text{Average}(0.30 \cdot \mathbf{r}_p + 0.59 \cdot \mathbf{g}_p + 0.11 \cdot \mathbf{b}_p)$

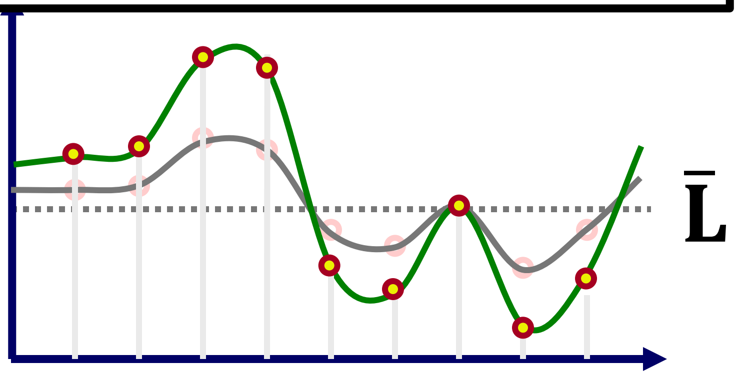
What do we get if we set the image to have no contrast ($\alpha = 0$)?



Original



More Contrast

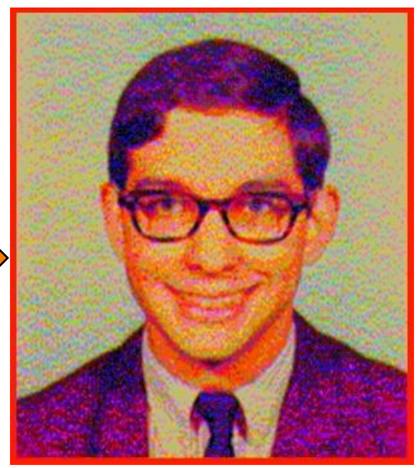


$$\mathbf{I}_p \leftarrow (\mathbf{I}_p - \bar{\mathbf{L}})\alpha + \bar{\mathbf{L}}$$

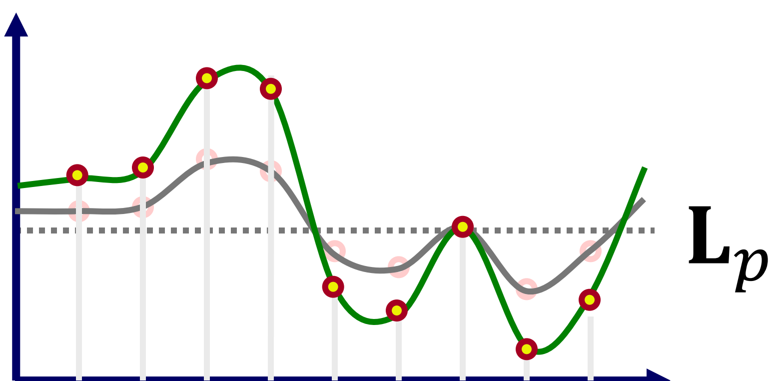
Adjusting Saturation

- Compute <u>per-pixel</u> luminance $\mathbf{L}_p$
  - $\mathbf{L}_p = 0.30 \cdot \mathbf{r}_p + 0.59 \cdot \mathbf{g}_p + 0.11 \cdot \mathbf{b}_p$
- Scale <u>deviation</u> from $\mathbf{L}_p$ for each pixel component
  - [WARNING] Clamp to range -- e.g. to [0,255)

Original



More Saturation



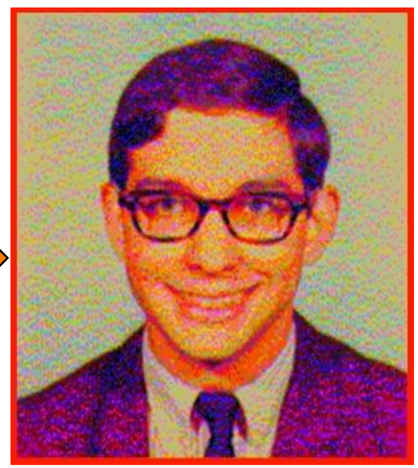
$$\mathbf{I}_p \leftarrow (\mathbf{I}_p - \mathbf{L}_p)\alpha + \mathbf{L}_p$$

Adjusting Saturation

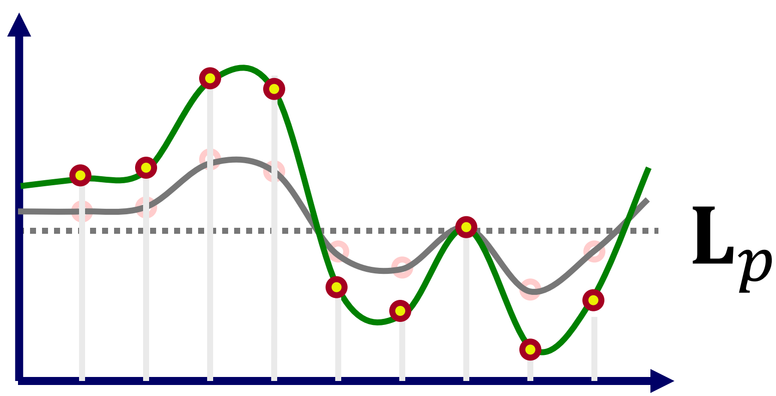
- Compute <u>per-pixel</u> luminance $\mathbf{L}_p$
  - $\mathbf{L}_p = 0.30 \cdot \mathbf{r}_p + 0.59 \cdot \mathbf{g}_p + 0.11 \cdot \mathbf{b}_p$

What do we get if we set the image to have no saturation ($\alpha = 0$)?

Original



More Saturation



$\mathbf{I}_p \leftarrow (\mathbf{I}_p - \mathbf{L}_p)\alpha + \mathbf{L}_p$