Human Vision, Color and Basic Image Processing

Connelly Barnes

CS4810

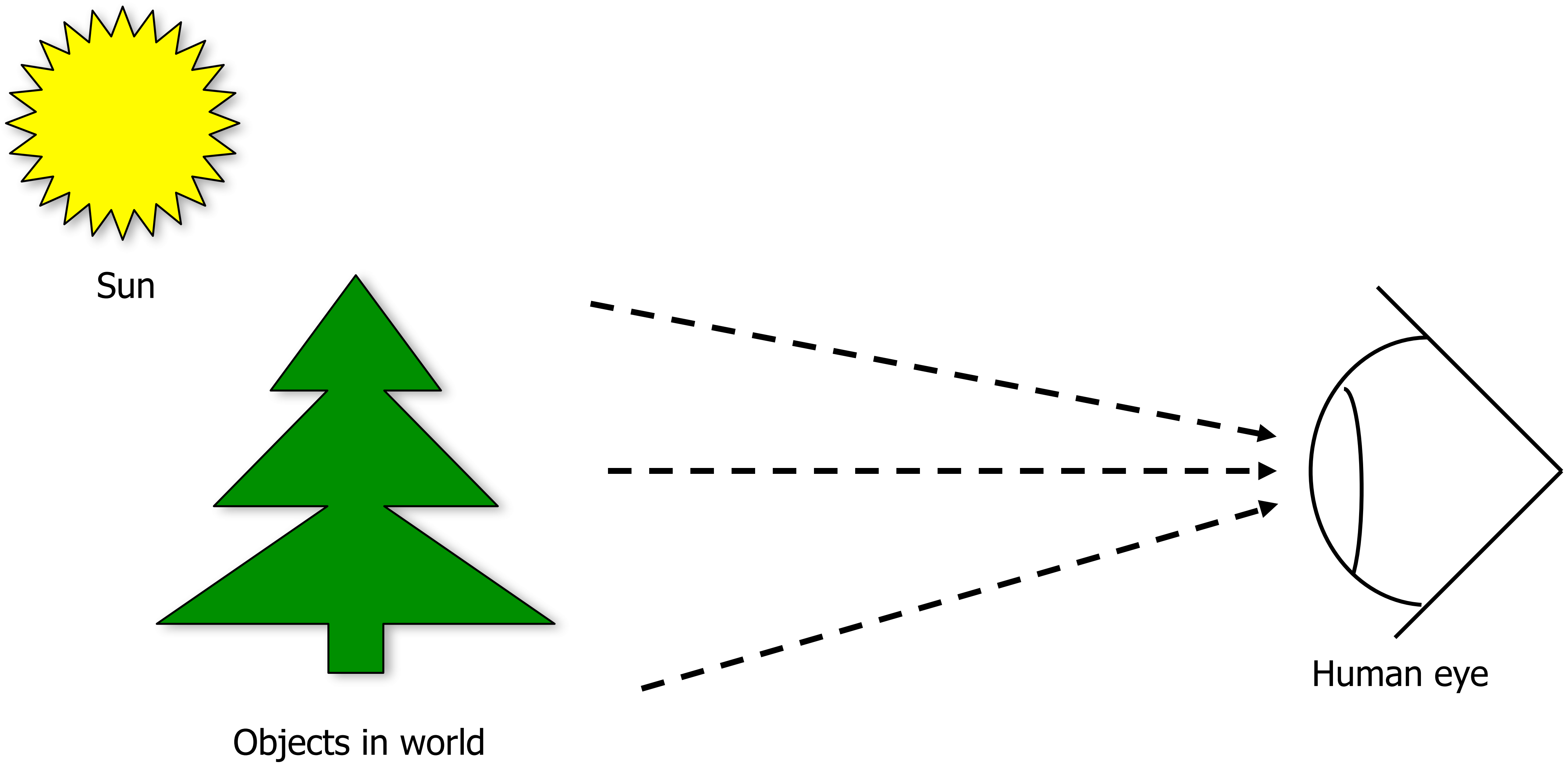
University of Virginia

Acknowledgement: slides by Jason Lawrence, Misha Kazhdan, Allison Klein, Tom Funkhouser, Adam Finkelstein and David Dobkin

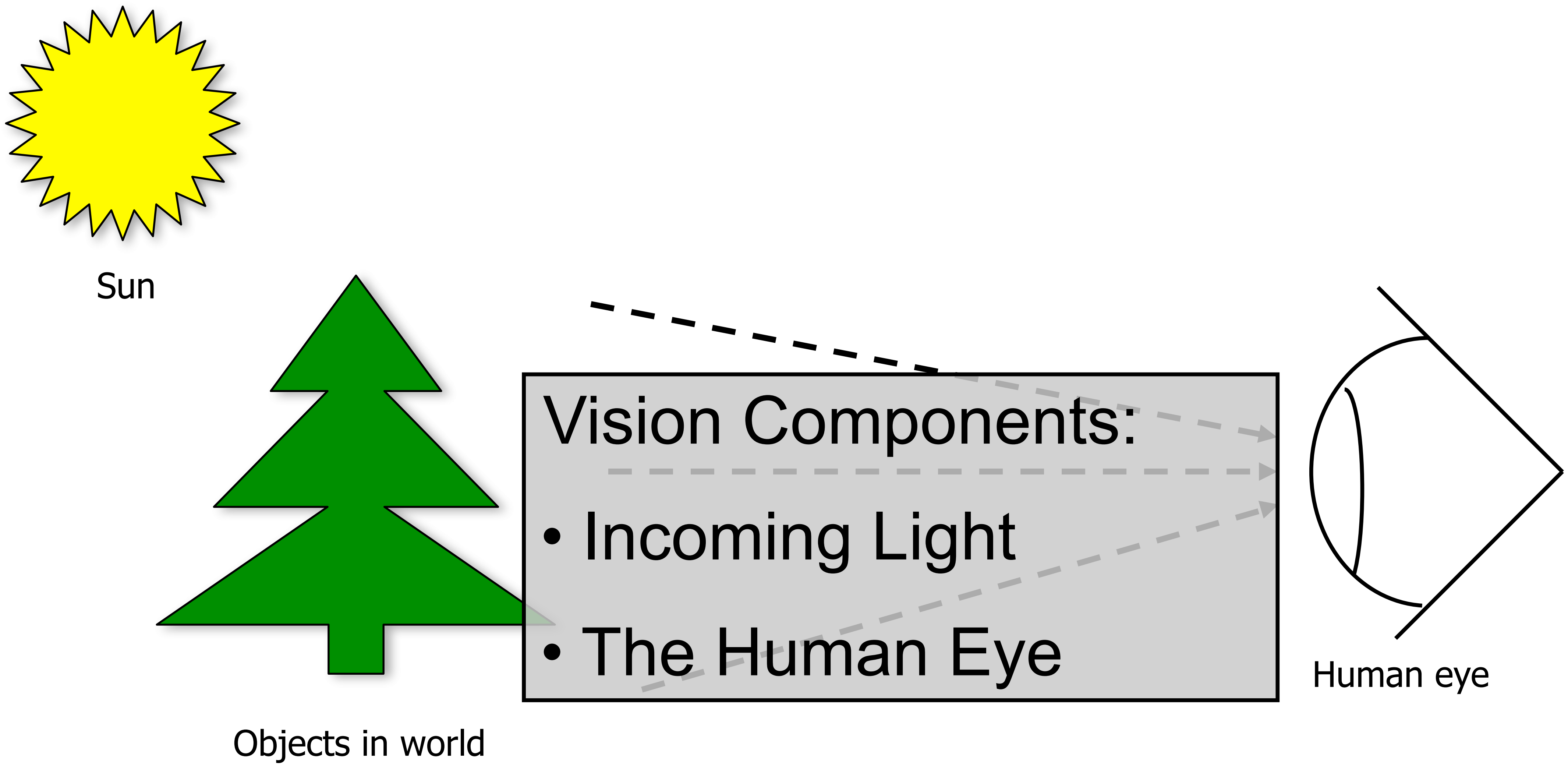
Outline

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

Human Vision



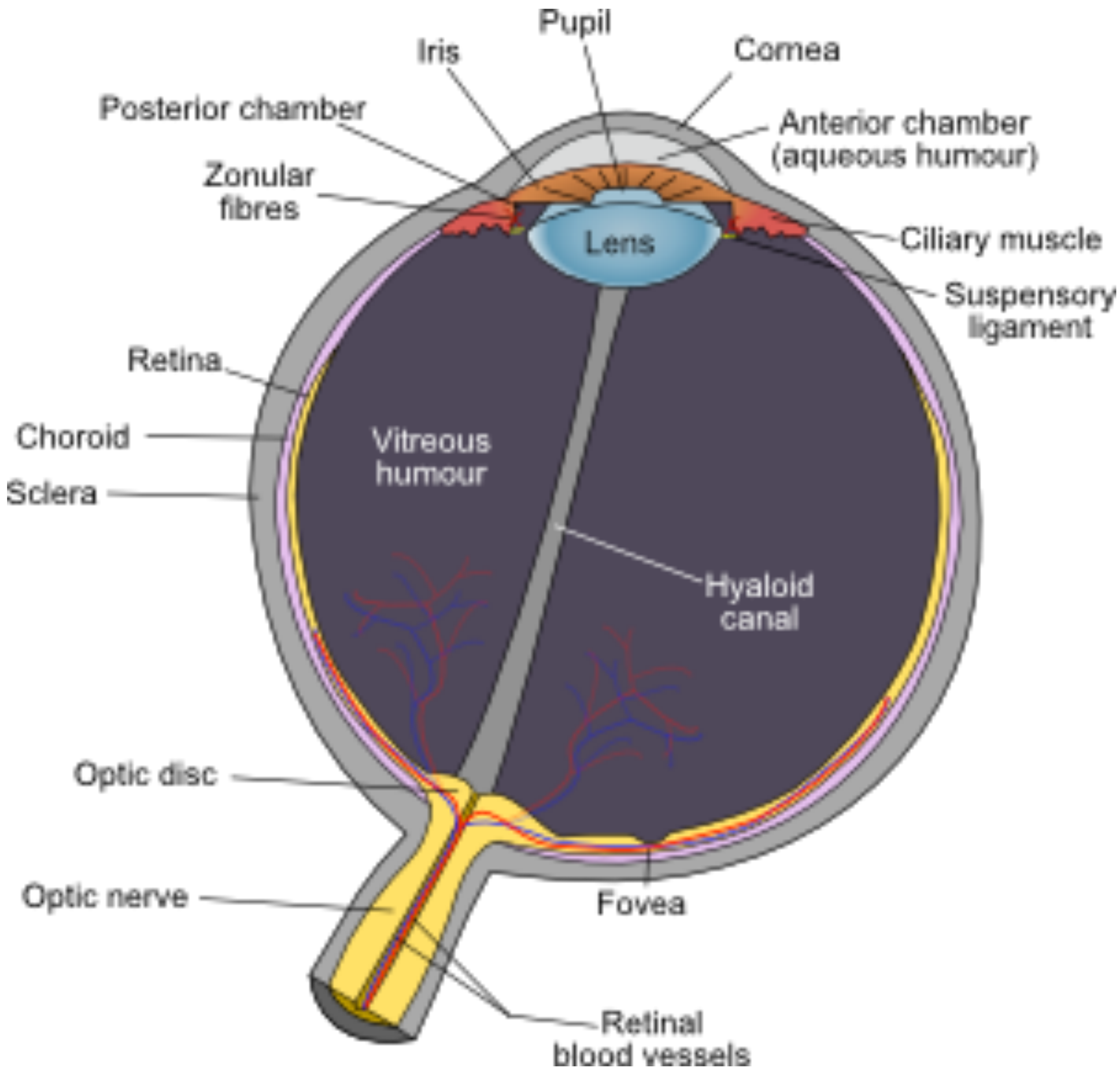
Human Vision



Vision Components:

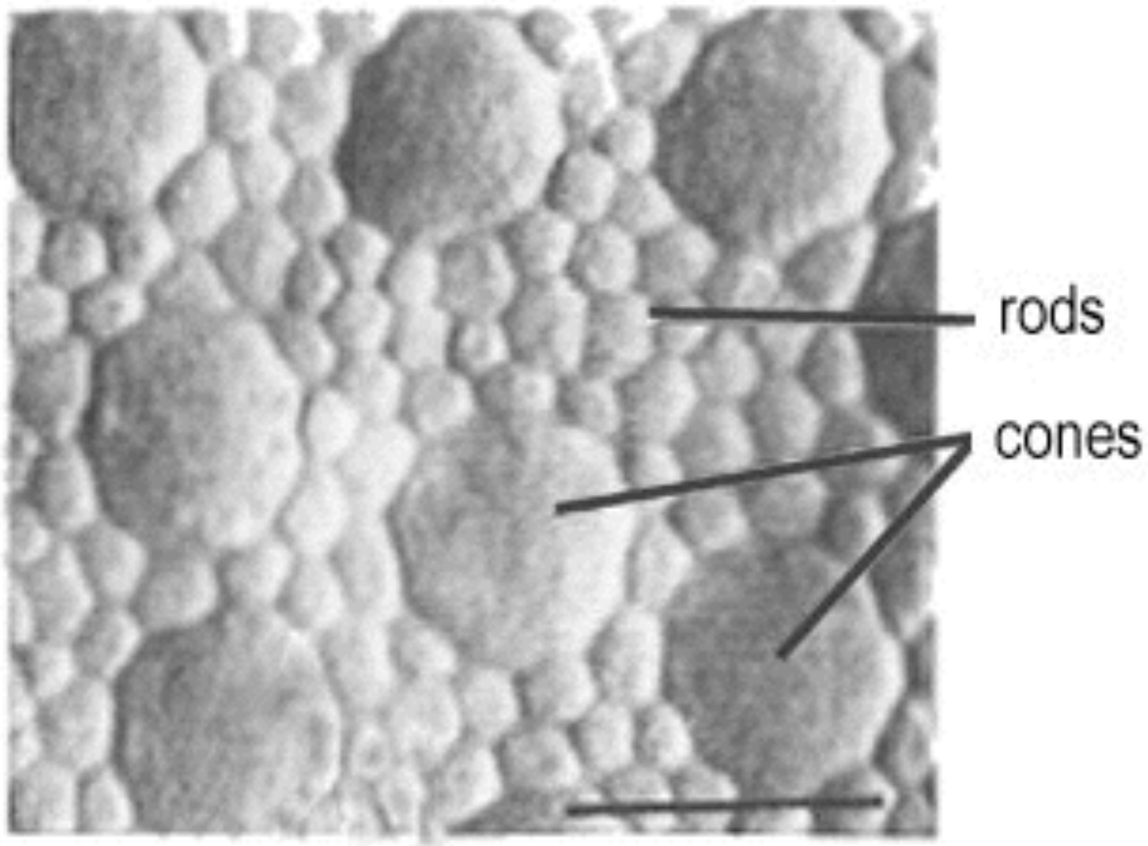
- Incoming Light
- The Human Eye

Typical Human Eye

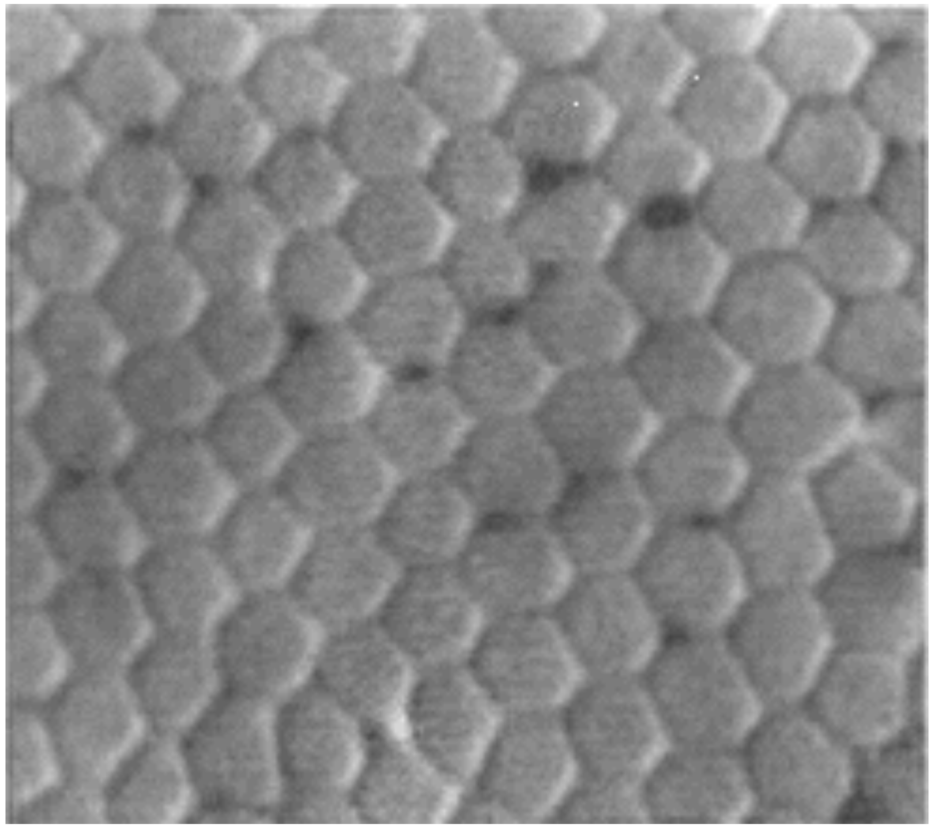


Color

- Two types of photo-sensitive cells ("photo receptors")



Rods and cones



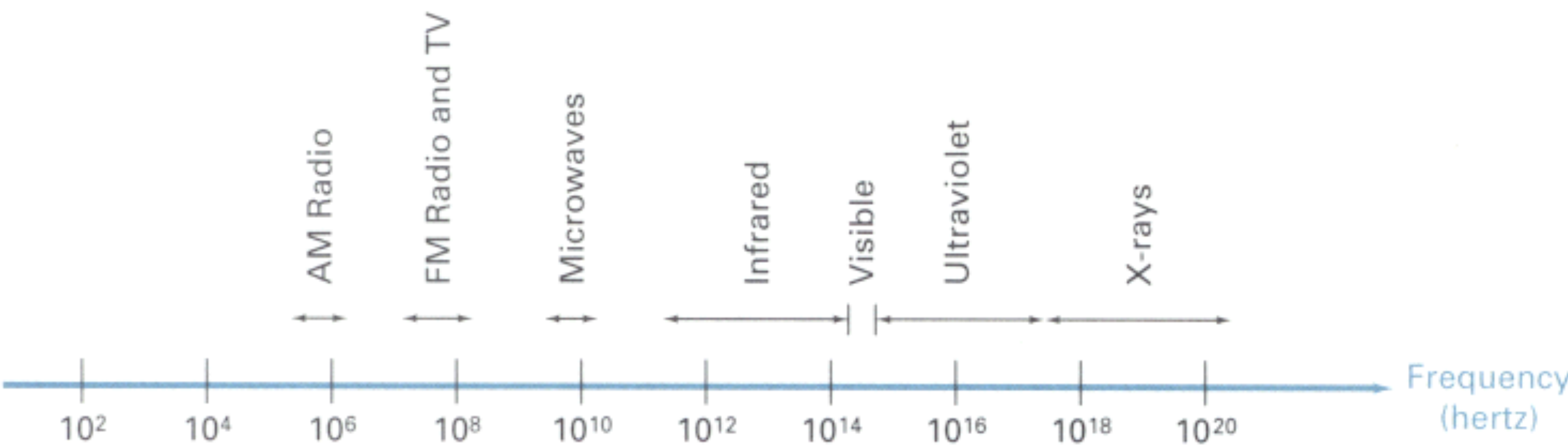
Cones in fovea

Rods and Cones

- Rods
  - More sensitive in low light: “scotopic” vision
  - More dense near periphery
- Cones
  - Only function with higher light levels: “photopic” vision
  - Densely packed at center of eye: fovea
  - Different types of cones → color vision

Electromagnetic Spectrum

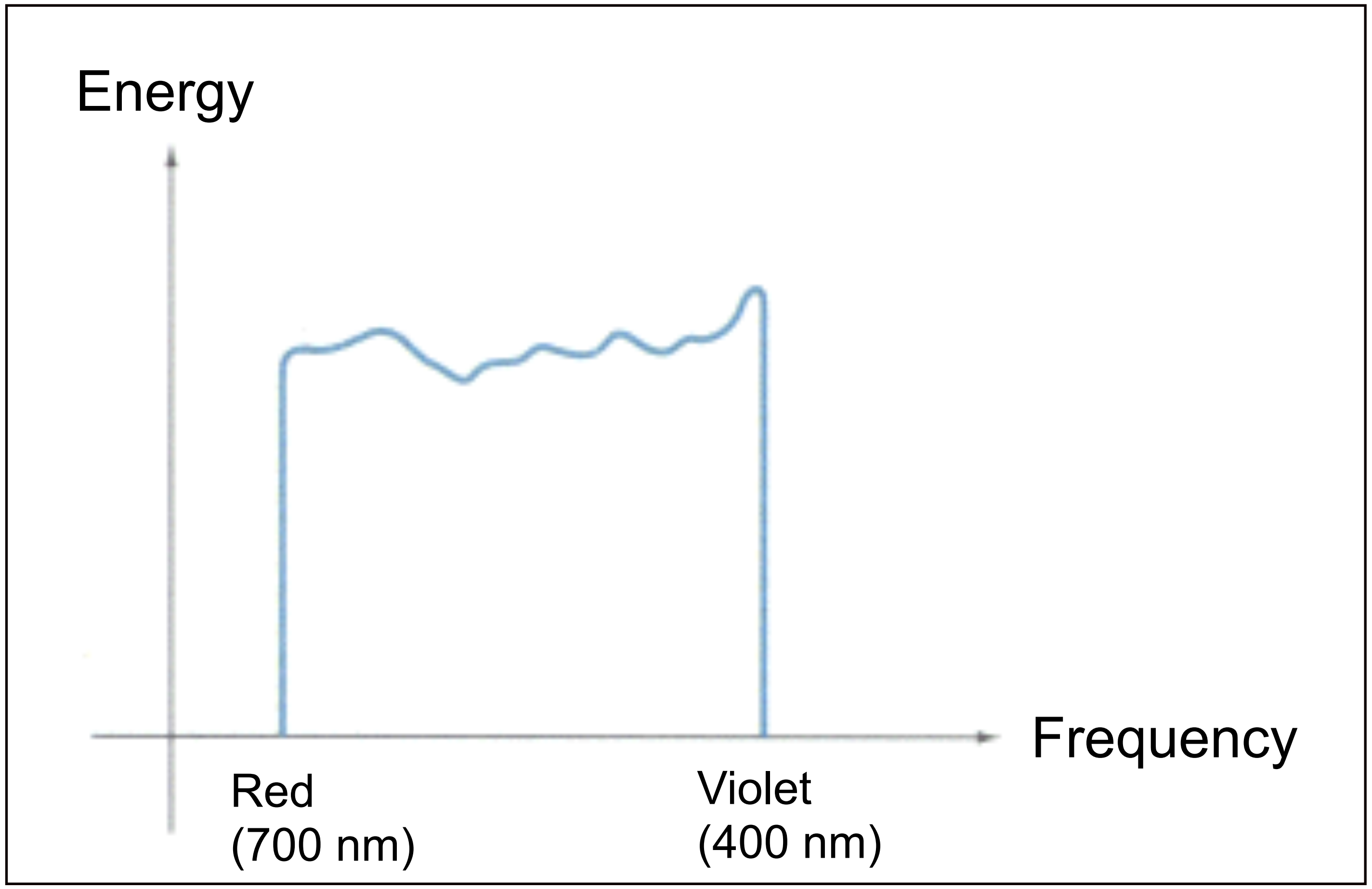
- Visible light frequencies range between ...
  - Red = 4.3 x 1014 hertz (700nm)
  - Violet = 7.5 x 1014 hertz (400nm)



Figures 15.1 from H&B

Visible Light

- The human eye can “see” light in the frequency range 400nm – 700nm



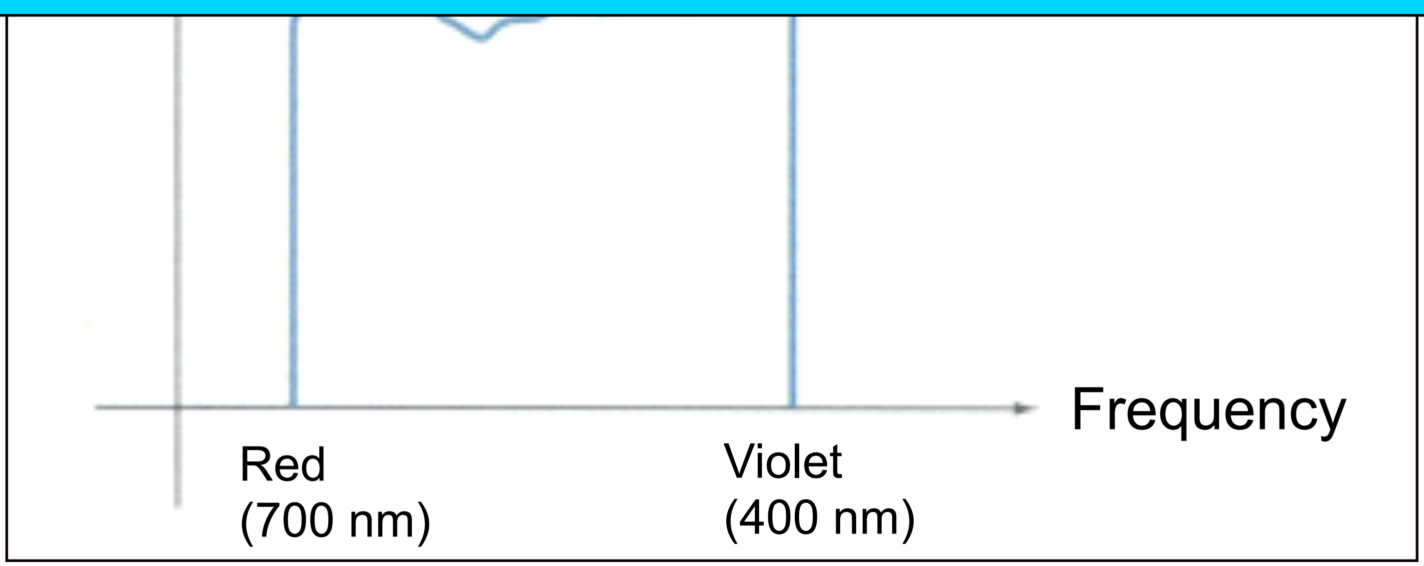
“White” Light

Figure 15.3 from H&B

Visible Light

- The human eye can “see” light in the frequency range 400nm – 700nm

This does not mean that we can see the difference between the different spectral distributions.

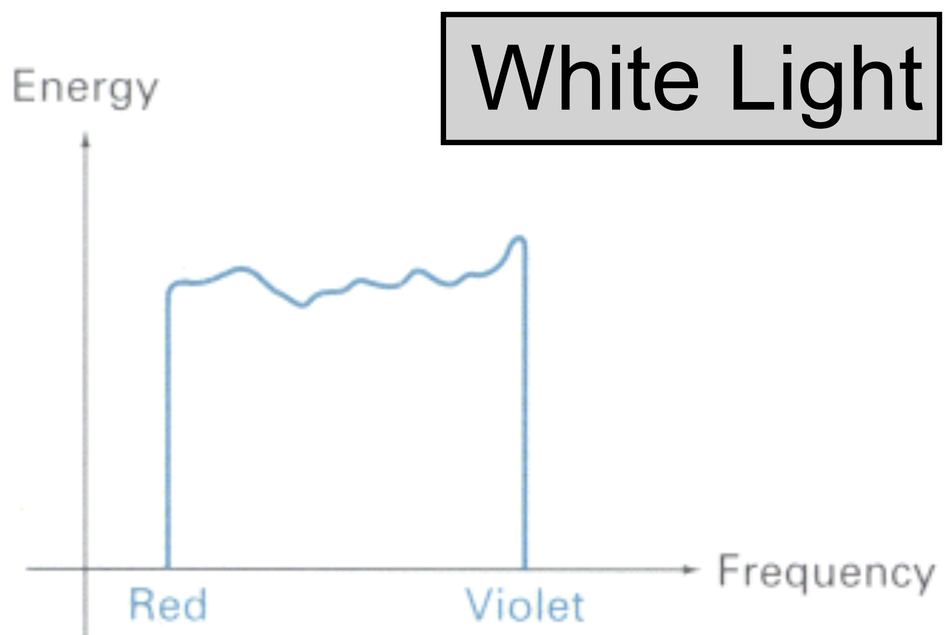


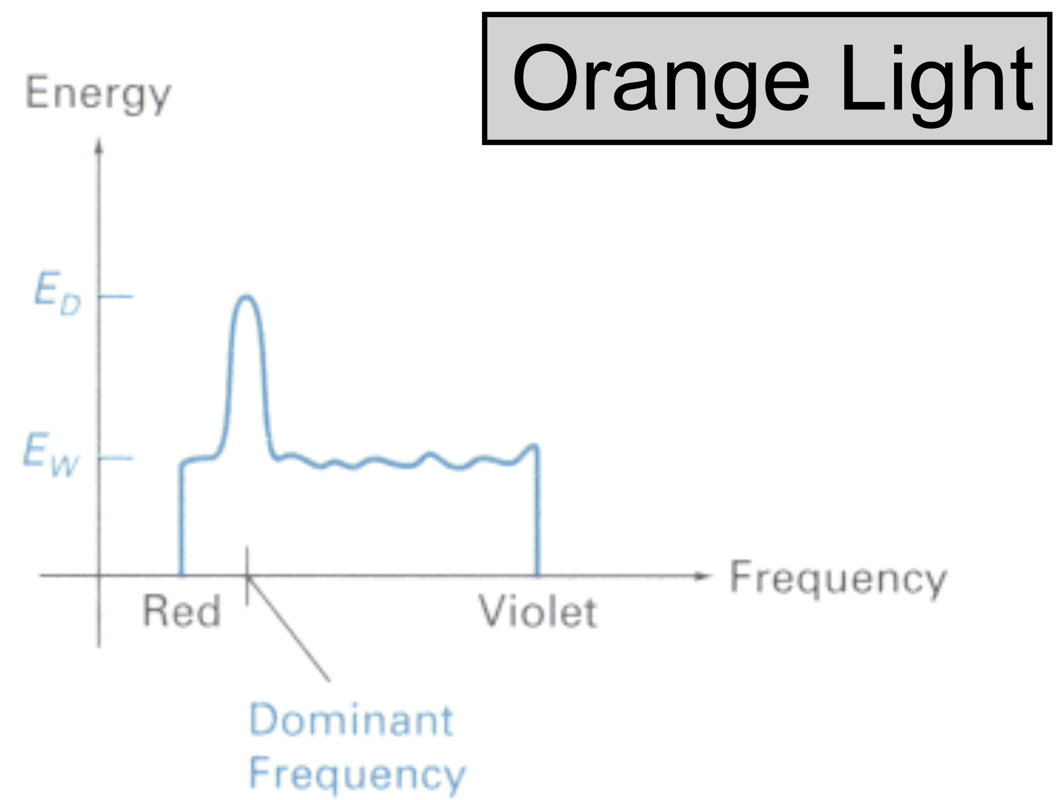
“White” Light

Figure 15.3 from H&B

Visible Light

- Color may be characterized by …
  - Hue = dominant frequency (highest peak)
  - Saturation = excitation purity (ratio of highest to rest)
  - Lightness = luminance (area under curve)





Tristimulus Theory of Color

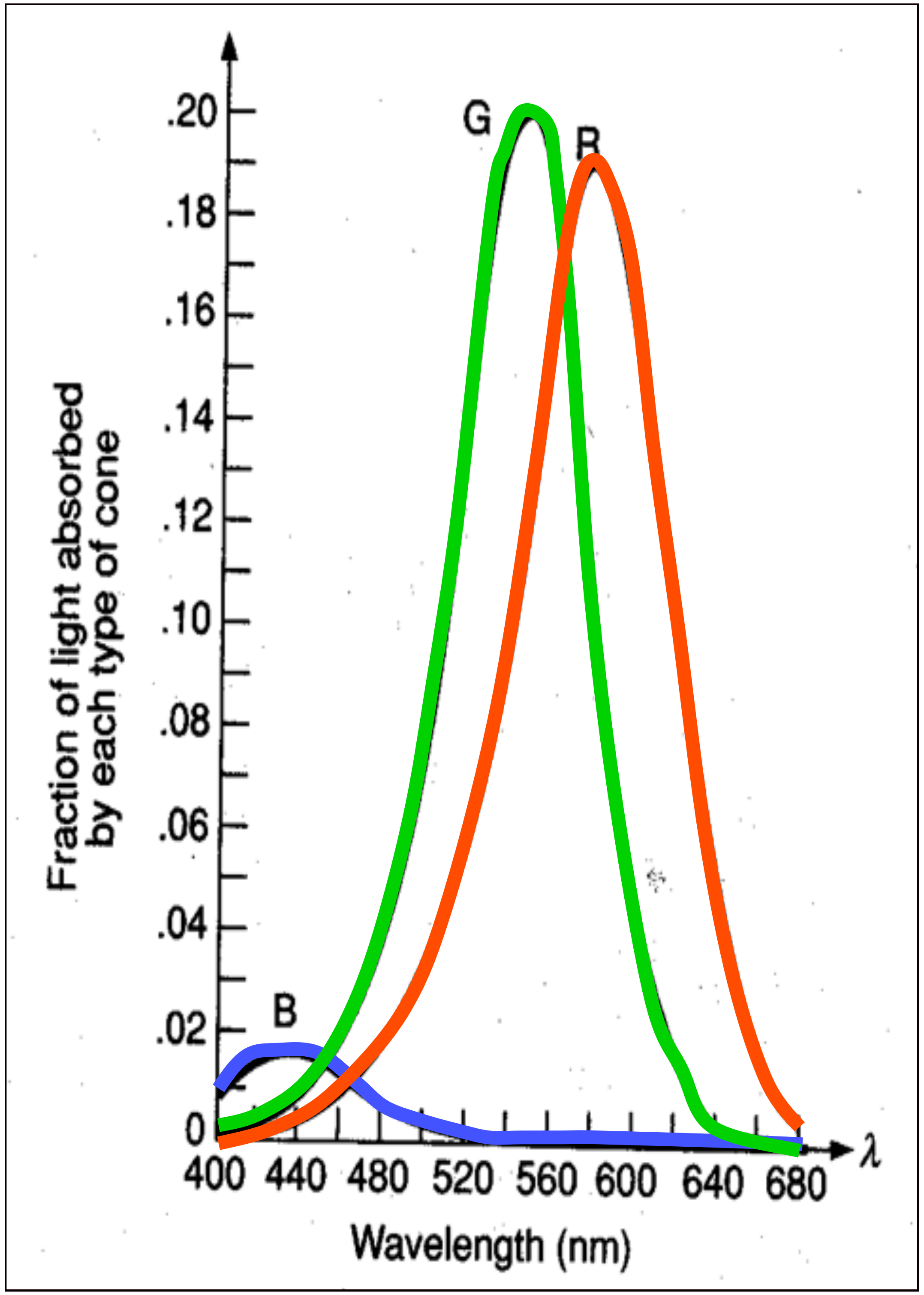


Figure 13.18 from FvDFH

Spectral-response functions of each of the three types of cones.

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

Tristimulus Color

- Any distribution of light can be summarized by its effect on 3 types of cones
- Therefore, human perception of color is a 3-dimensional space
- Metamerism: different spectra, same response
- Color blindness: fewer than 3 types of cones
  - Most commonly L cone = M cone

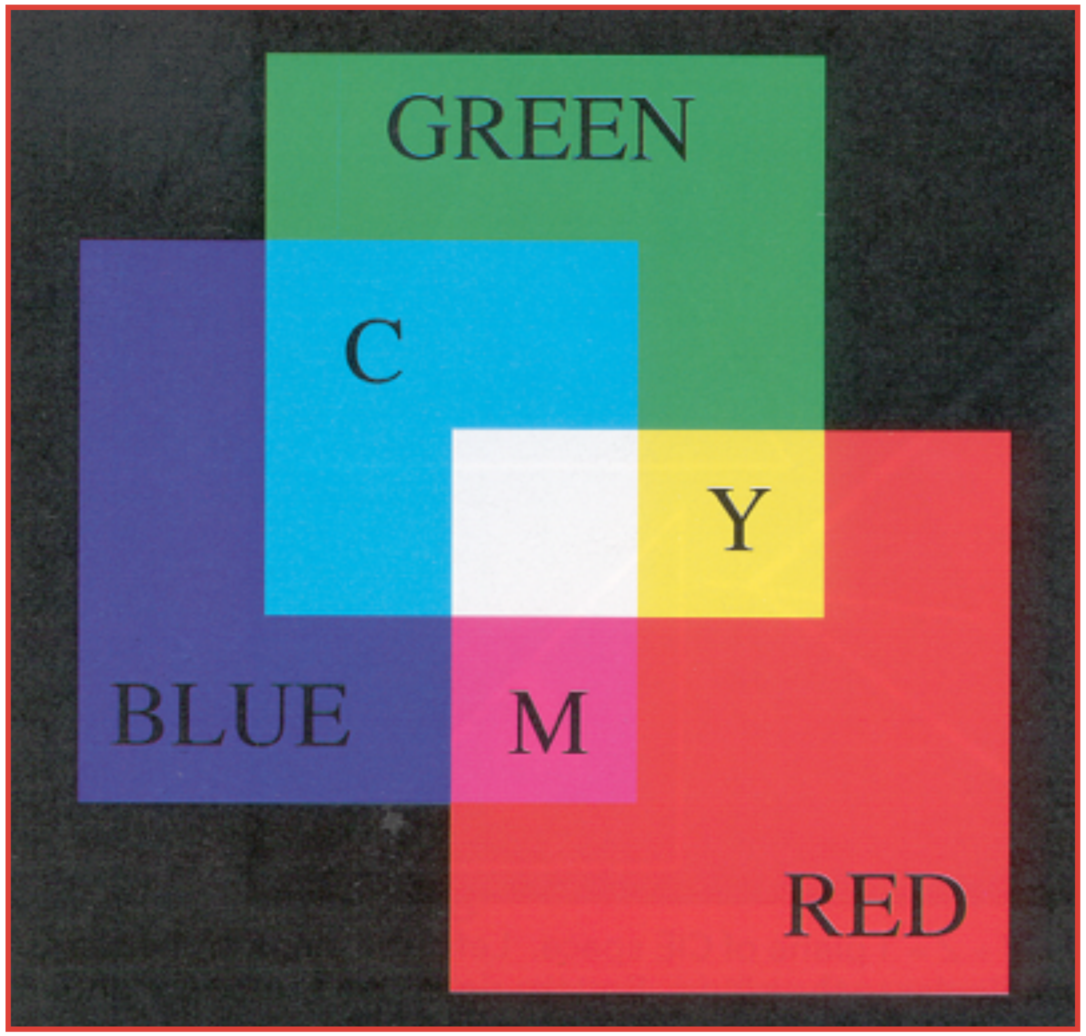
Color Models

- RGB
- XYZ
- CMYK
- HSV
- etc...

Different ways of parameterizing 3D space.

RGB most common and used in this class:
R=645.16nm, G=526.32nm, B=444.44nm

RGB Color Model

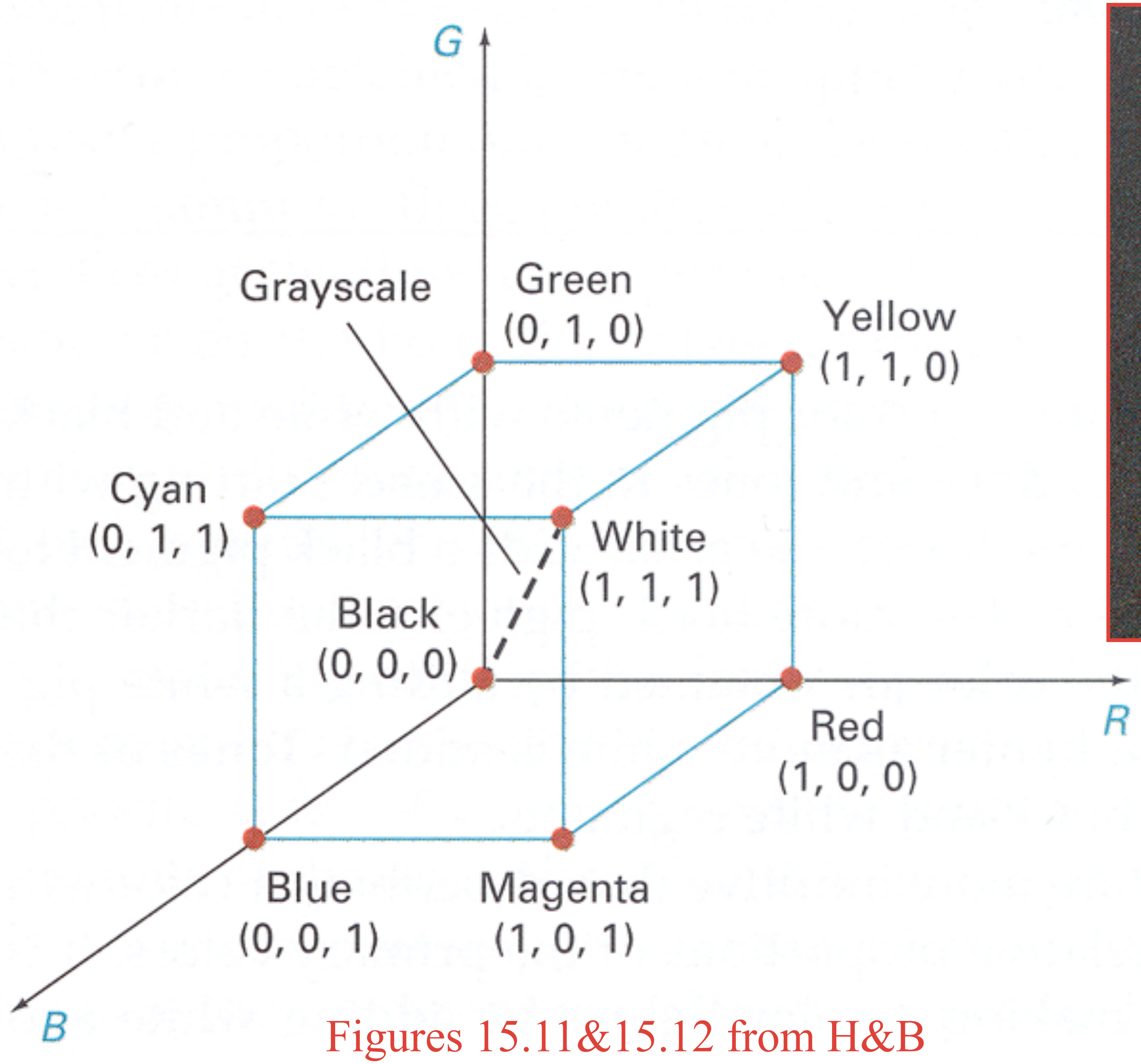


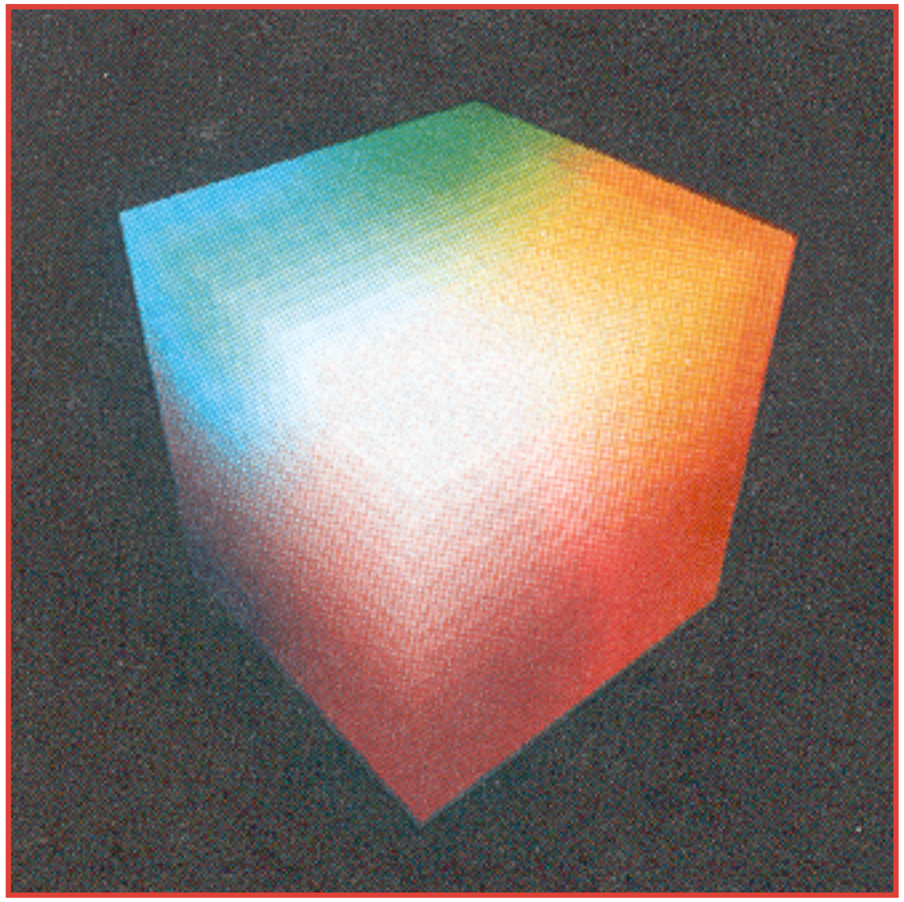
Colors are additive

Plate II.3 from FvDFH

R	G	B	Color
0.0	0.0	0.0	Black
1.0	0.0	0.0	Red
0.0	1.0	0.0	Green
0.0	0.0	1.0	Blue
1.0	1.0	0.0	Yellow
1.0	0.0	1.0	Magenta
0.0	1.0	1.0	Cyan
1.0	1.0	1.0	White
0.5	0.0	0.0	?
1.0	0.5	0.5	?
1.0	0.5	0.0	?
0.5	0.3	0.1	?

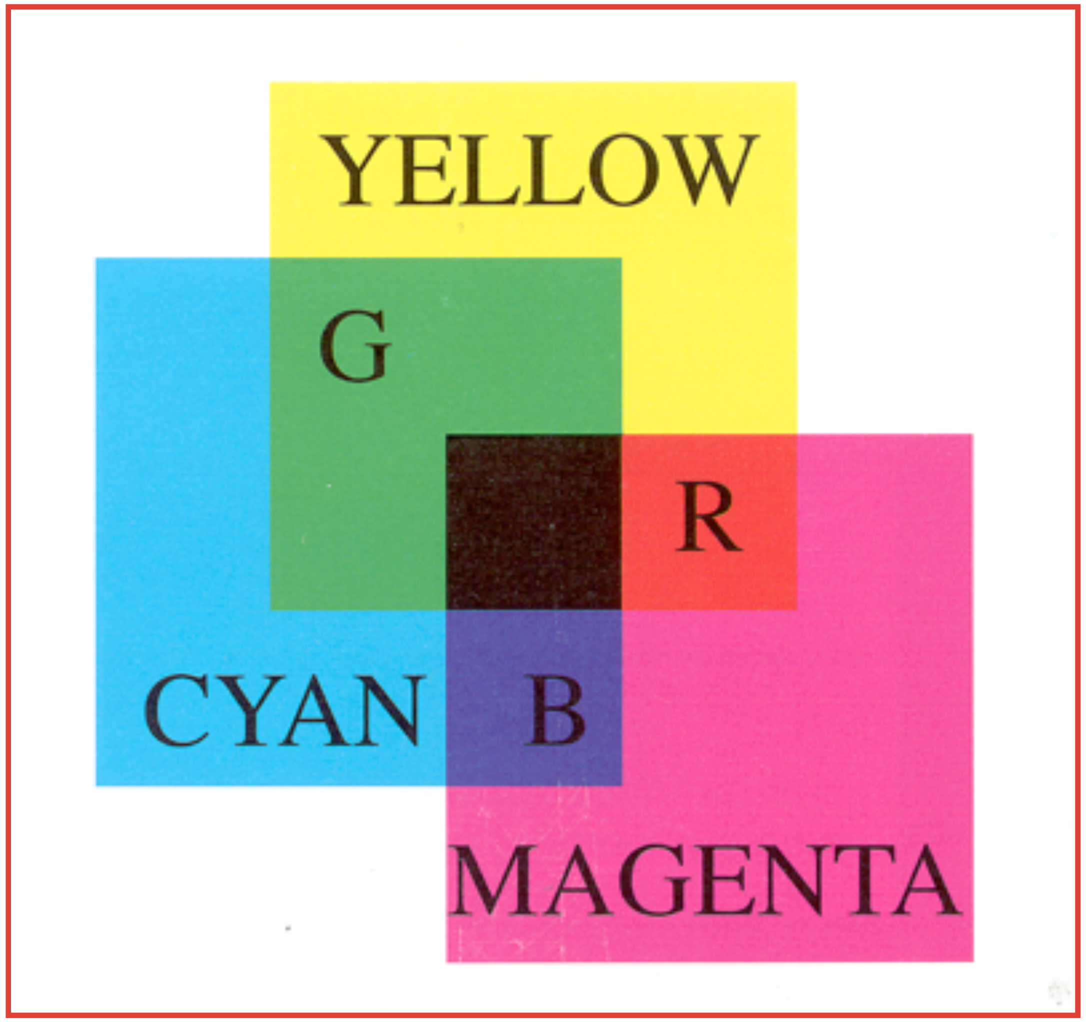
RGB Color Cube





Figures 15.11&15.12 from H&B

CMY(K) Color Model



Colors are subtractive

Plate II.7 from FvDFH

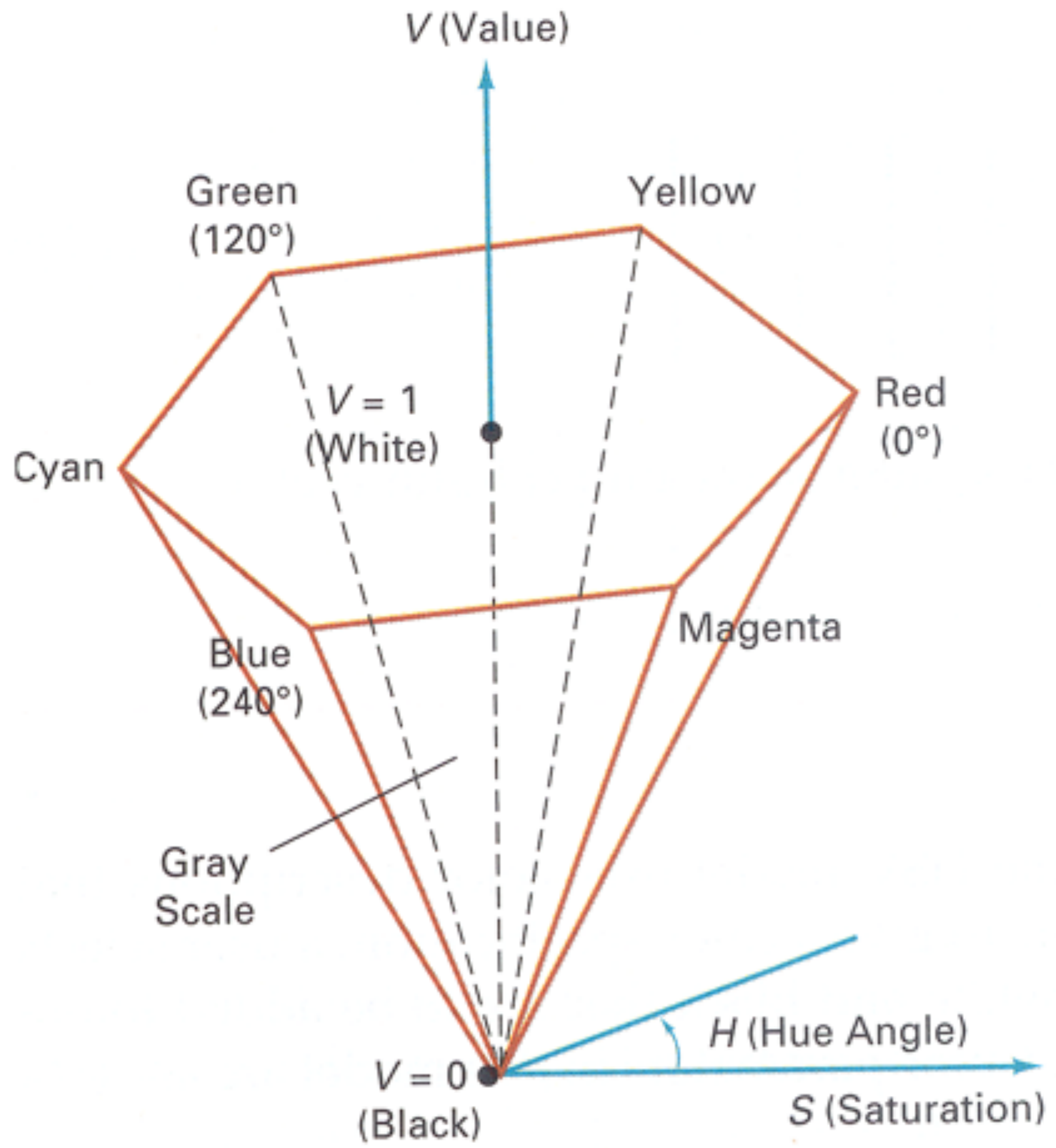
C	M	Y	Color
0.0	0.0	0.0	White
1.0	0.0	0.0	Cyan
0.0	1.0	0.0	Magenta
0.0	0.0	1.0	Yellow
1.0	1.0	0.0	Blue
1.0	0.0	1.0	Green
0.0	1.0	1.0	Red
1.0	1.0	1.0	Black
0.5	0.0	0.0	?
1.0	0.5	0.5	?
1.0	0.5	0.0	?

Discussion question

- CMY(K) is frequently used for printer ink cartridge colors
- RGB is frequently used for displays

- Why would CMY(K) be used for printers and RGB for screens?

HSV Color Model



H	S	V	Color
0	1.0	1.0	Red
120	1.0	1.0	Green
240	1.0	1.0	Blue
*	0.0	1.0	White
*	0.0	0.5	Gray
*	*	0.0	Black
60	1.0	1.0	?
270	0.5	1.0	?
270	0.0	0.7	?

Figure 15.16&15.17 from H&B

Outline

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

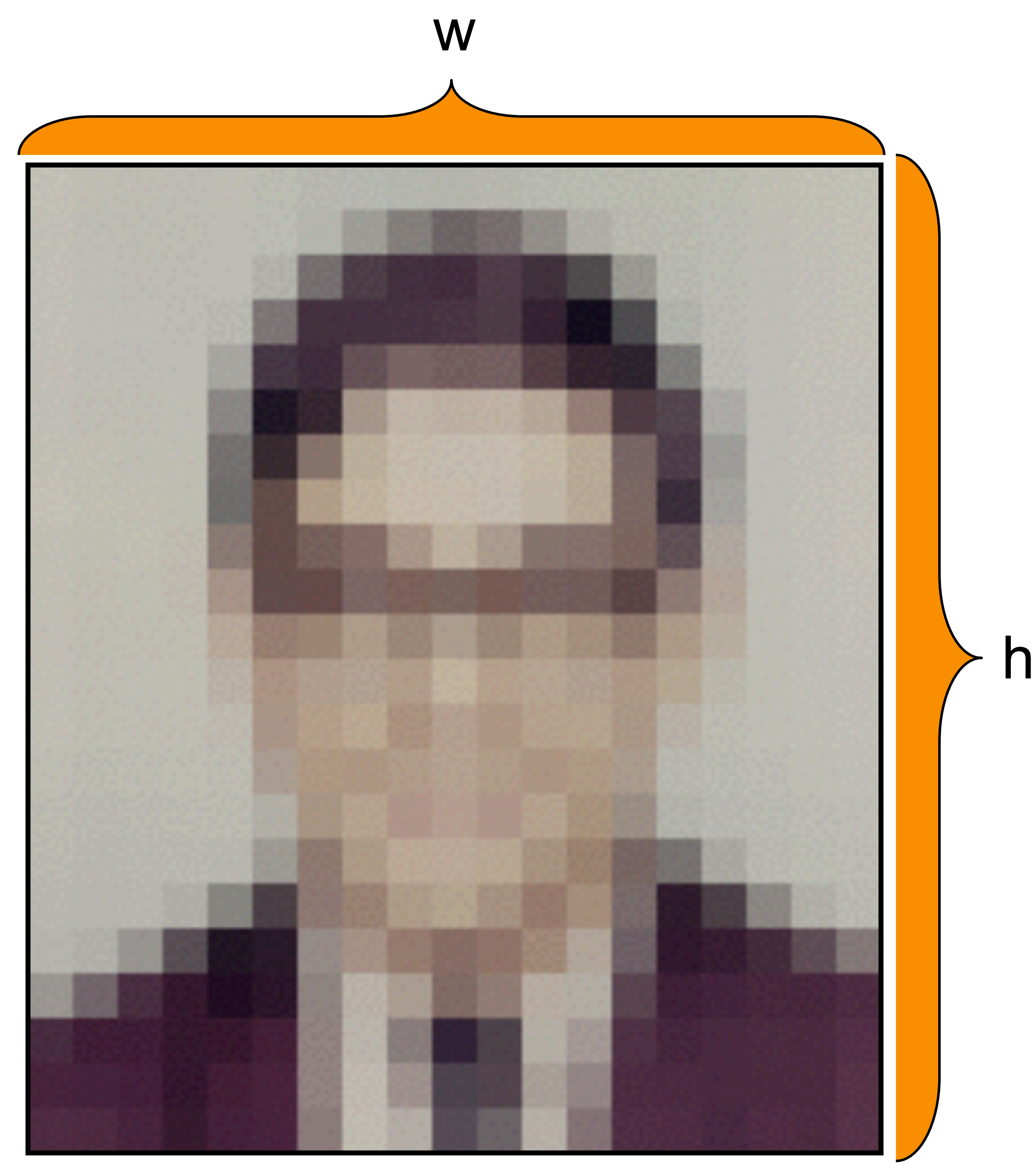
Image Representation

- 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 captures the notion of "intensity" and possibly "color"
- Luminance pixels
  - Grey-scale images (aka "Intensity images")
  - 0 – 1.0 or 0 – 255
- Red, Green, Blue pixels (RGB)
  - Color images
  - 0 – 1.0 or 0 – 255

Image Resolution

- Spatial resolution: width x height pixels
- Intensity/Color resolution: n bits per pixel
- Temporal resolution: n Hz (fps)

	Width x Height	Bit Depth	Hz
NTSC	640 x 480	8	30
iPhone5	640 x 1136	24	60
Monitor	1920 x 1200	24	75
CCDs	3000 x 2000	36	-
Laser Printer	6600 x 5100	1	-

Image Quantization Artifacts

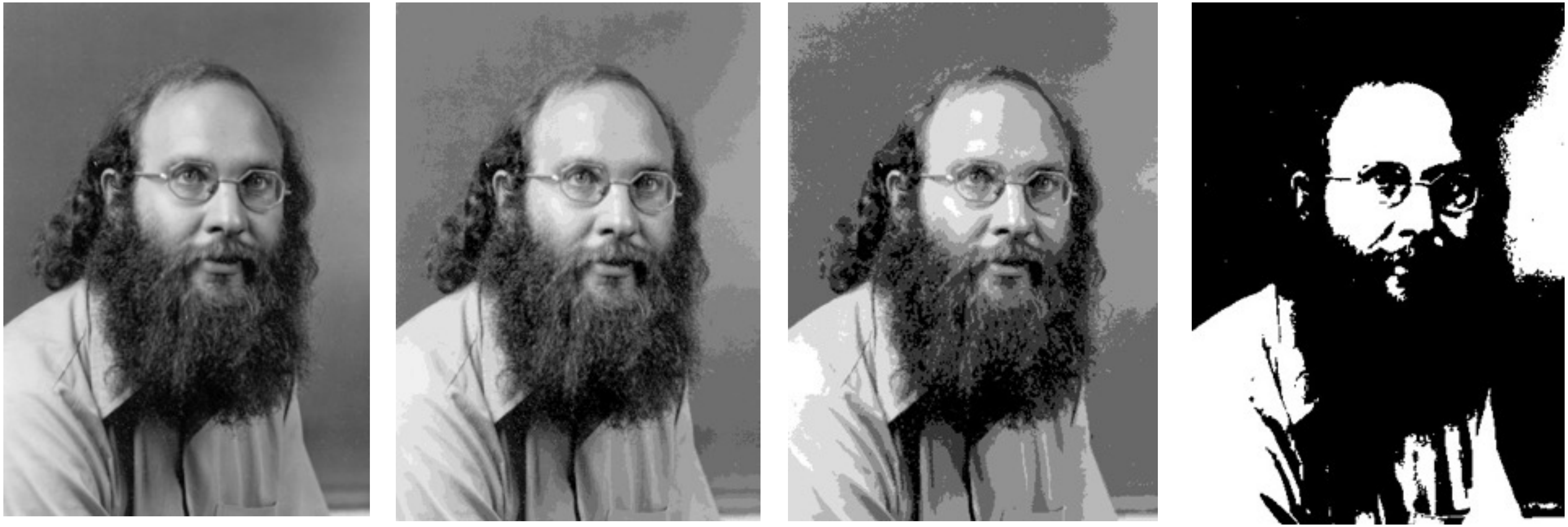
- With only a small number of bits associated to each color channel of a pixel there is a limit to intensity resolutions of an image
  - A black and white image allocates a single bit to the luminance channel of a pixel.
    - The number of different colors that can be represented by a pixel is 2.
  - A 24 bit bitmap image allocates 8 bits to the red, green, and blue channels of a pixel.
    - The number of different colors that can be represented by a pixel is 2^{24} = 16.8 million.

Outline

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

Quantization

- Image with decreasing bits per pixel
  - Note contouring!



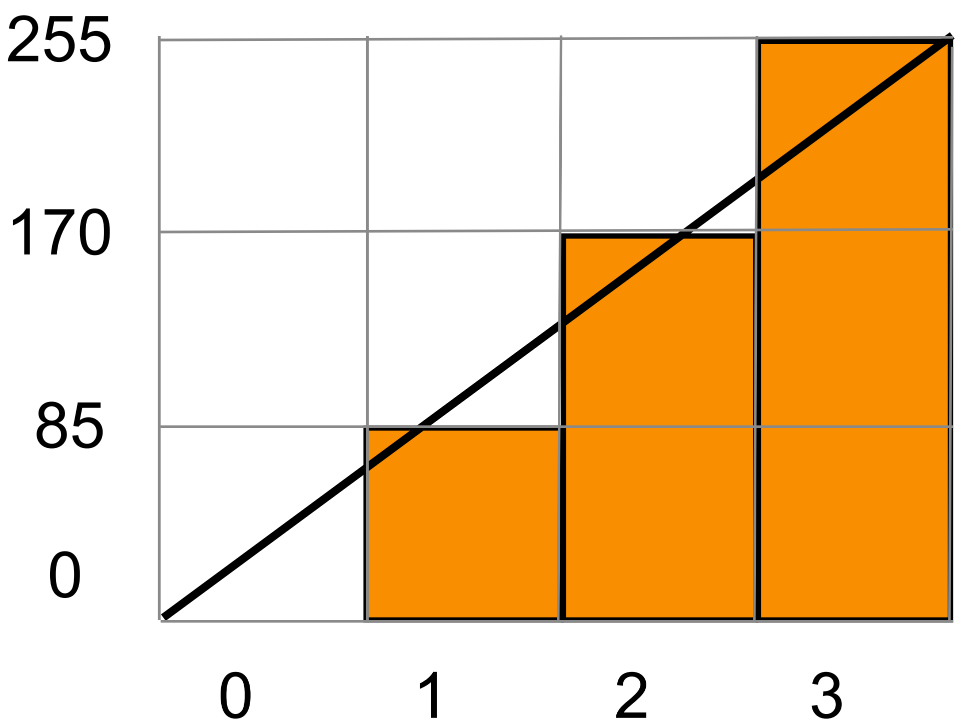
8 bits 4 bits 2 bits 1 bit

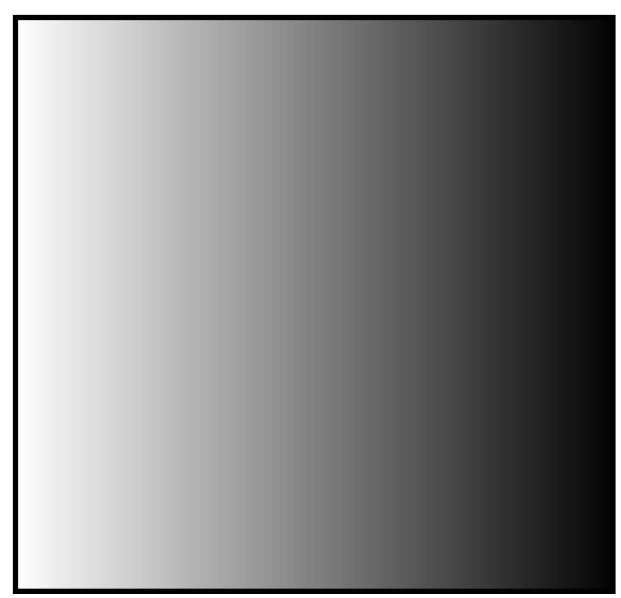
Quantization

- When you have a small number of bits per pixel, you can coarsely represent an image by quantizing the color values:

$$P(x, y) = Q(I(x, y)) = \text{floor}\left(\frac{I(x, y)}{256} 2^b\right)$$

b is the number of bits per pixel





I(x, y)

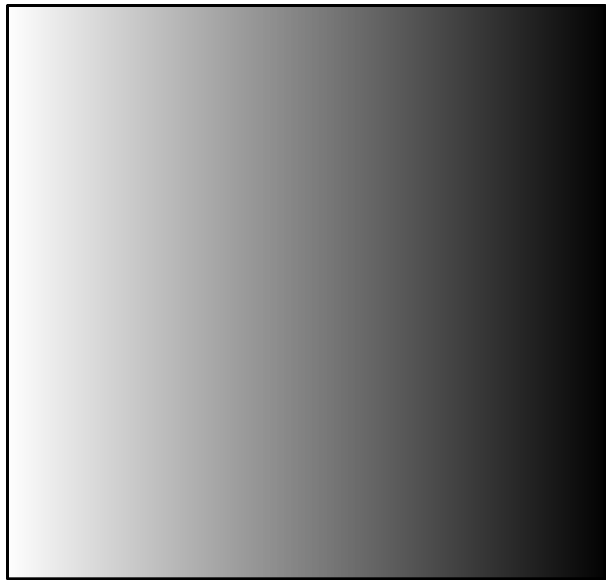
Q(x, y)
2 bits per pixel

Reducing Effects of Quantization

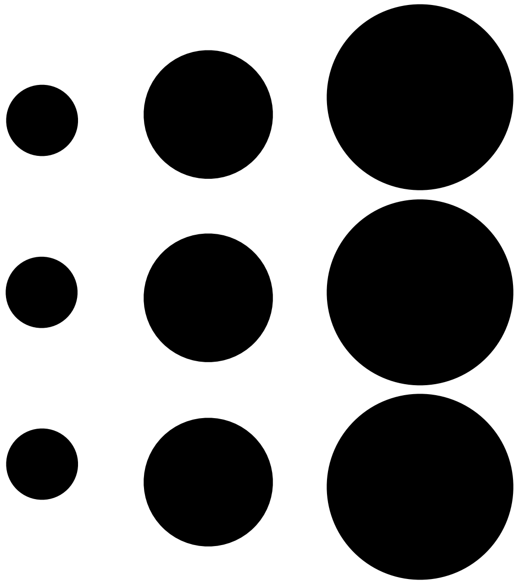
- Trade spatial resolution for intensity resolution
- Halftoning
- Dithering
  - Random dither
  - Ordered dither
  - Error diffusion dither

Classical Halftoning

- Varying-size dots represent intensities
- Area of dots inversely proportional to intensity

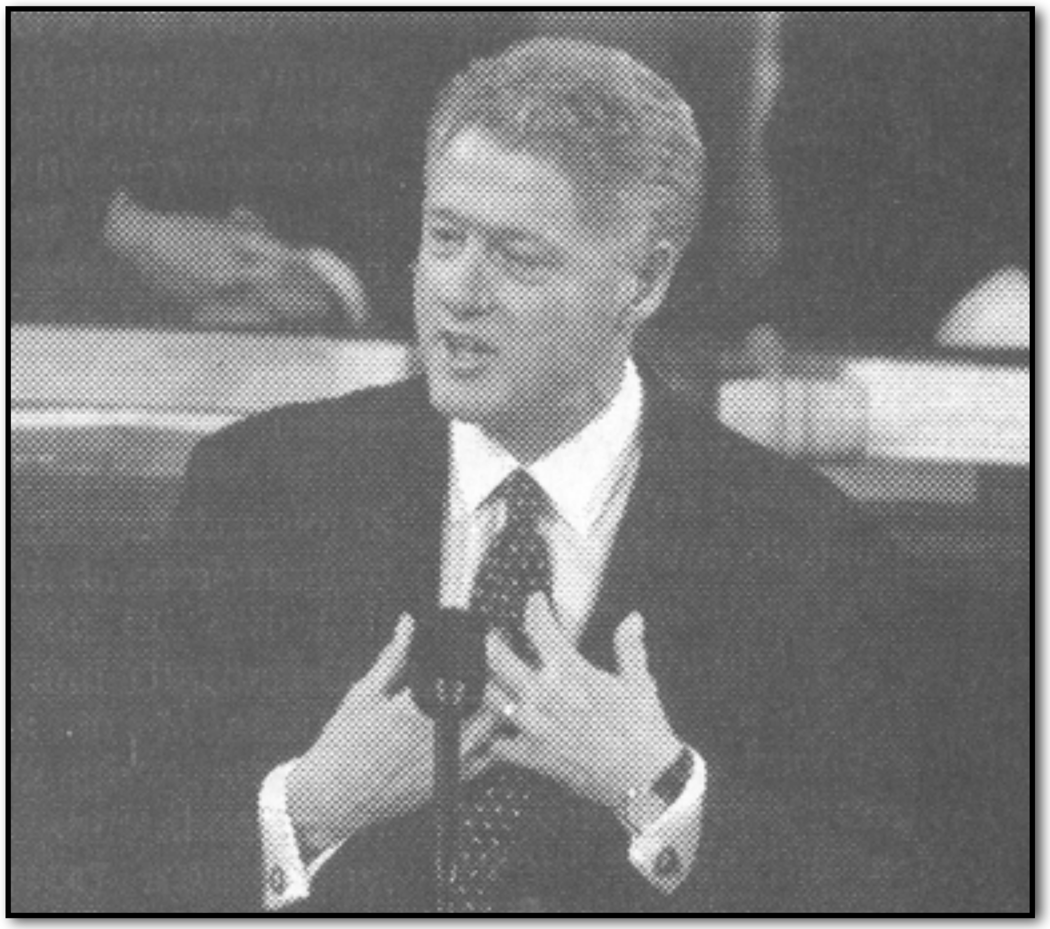


I(x, y)

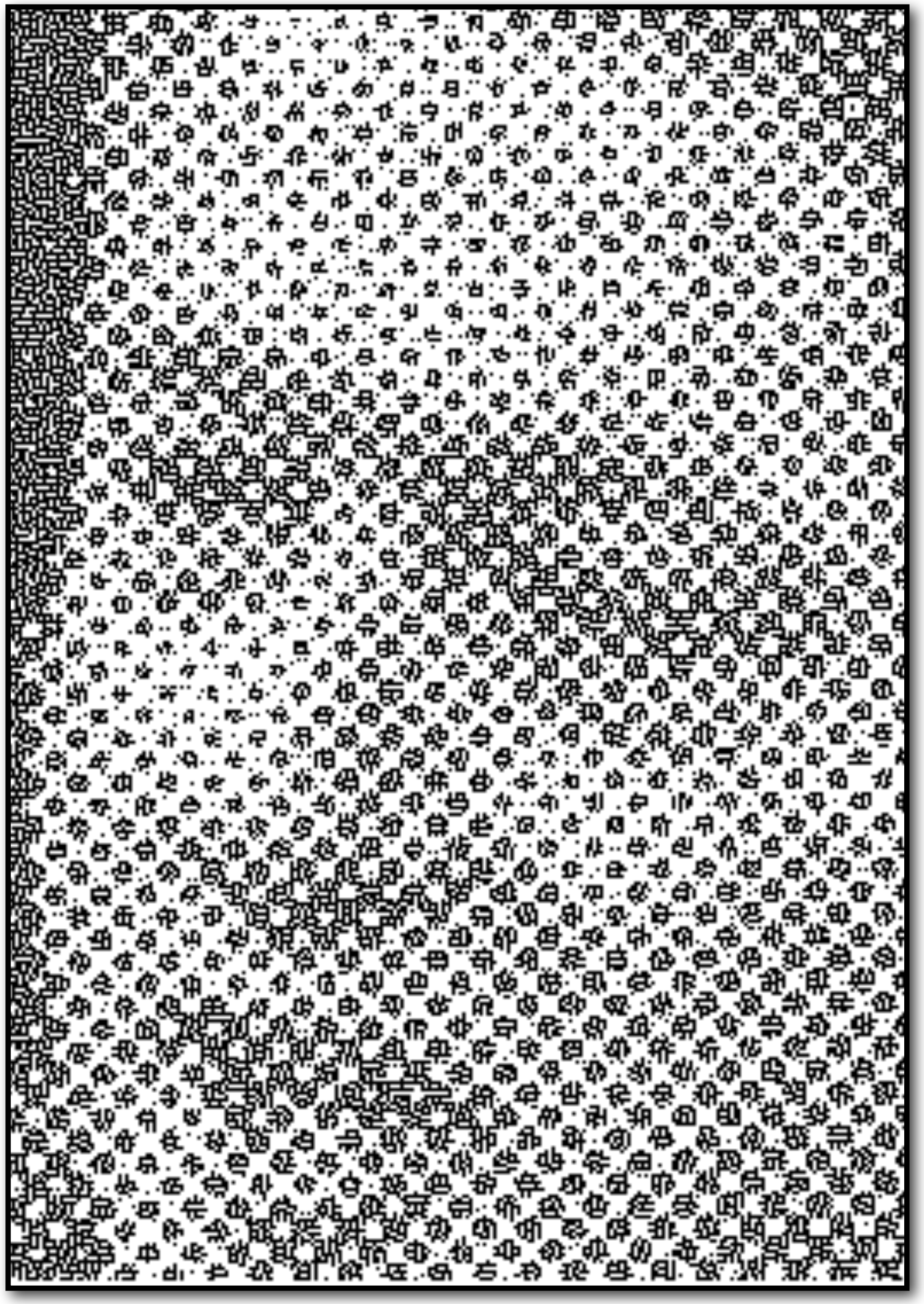


P(x, y)

Classical Halftoning



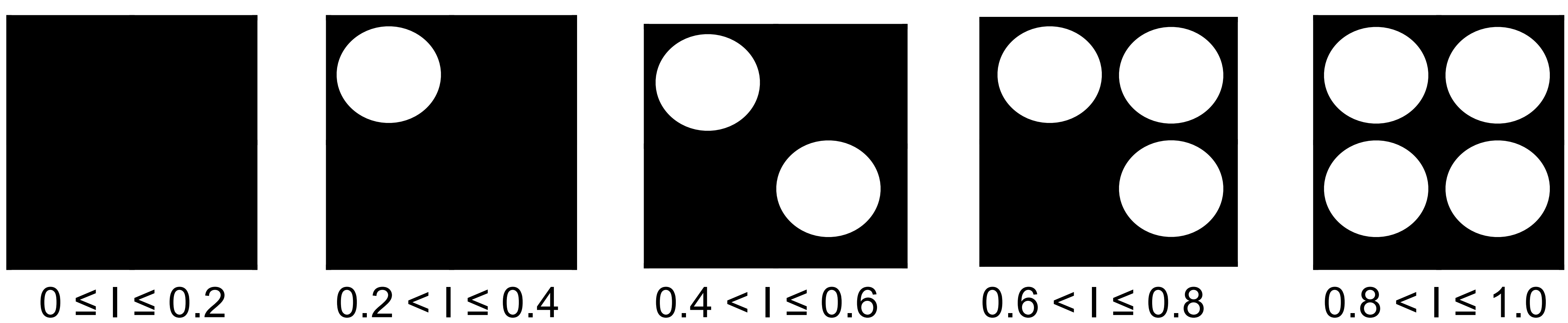
Newspaper Image



From New York Times, 9/21/99

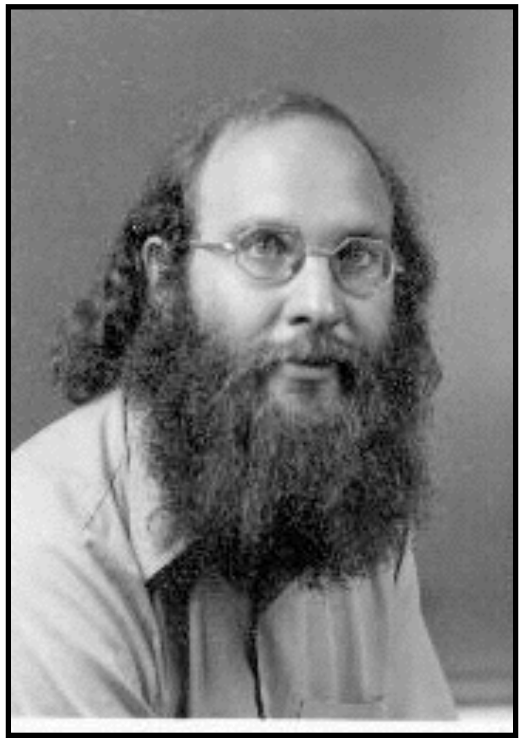
Digital Halftoning

- Use cluster of pixels to represent intensity
- Trades spatial resolution for intensity resolution
- Note that halftoning pattern matters
  - Want to avoid vertical, horizontal lines



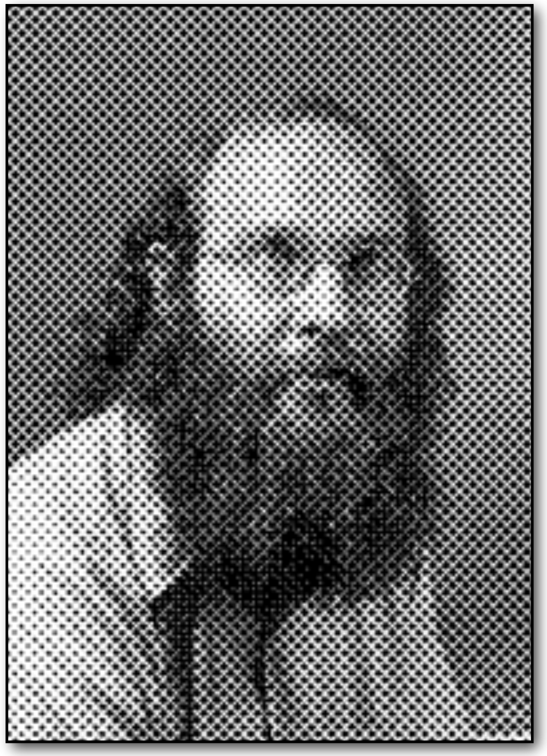
Digital Halftoning

- Use cluster of pixels to represent intensity
- Trades spatial resolution for intensity resolution
- Note that halftoning pattern matters



Original
(8 bits)

Quantized
(1 bit)



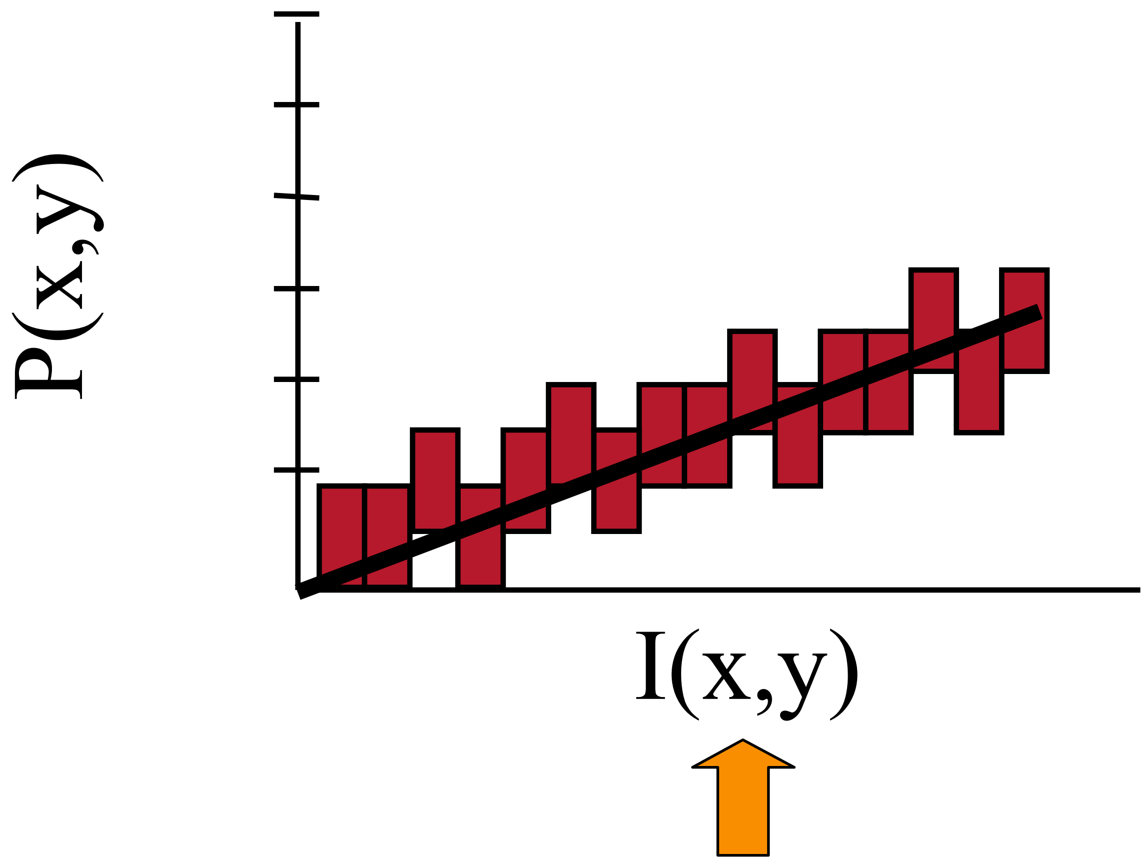
Halftoned
(1 bit)

Dithering

- Distribute errors among pixels
  - Exploit spatial integration in our eye
  - Display greater range of perceptible intensities

Random Dither

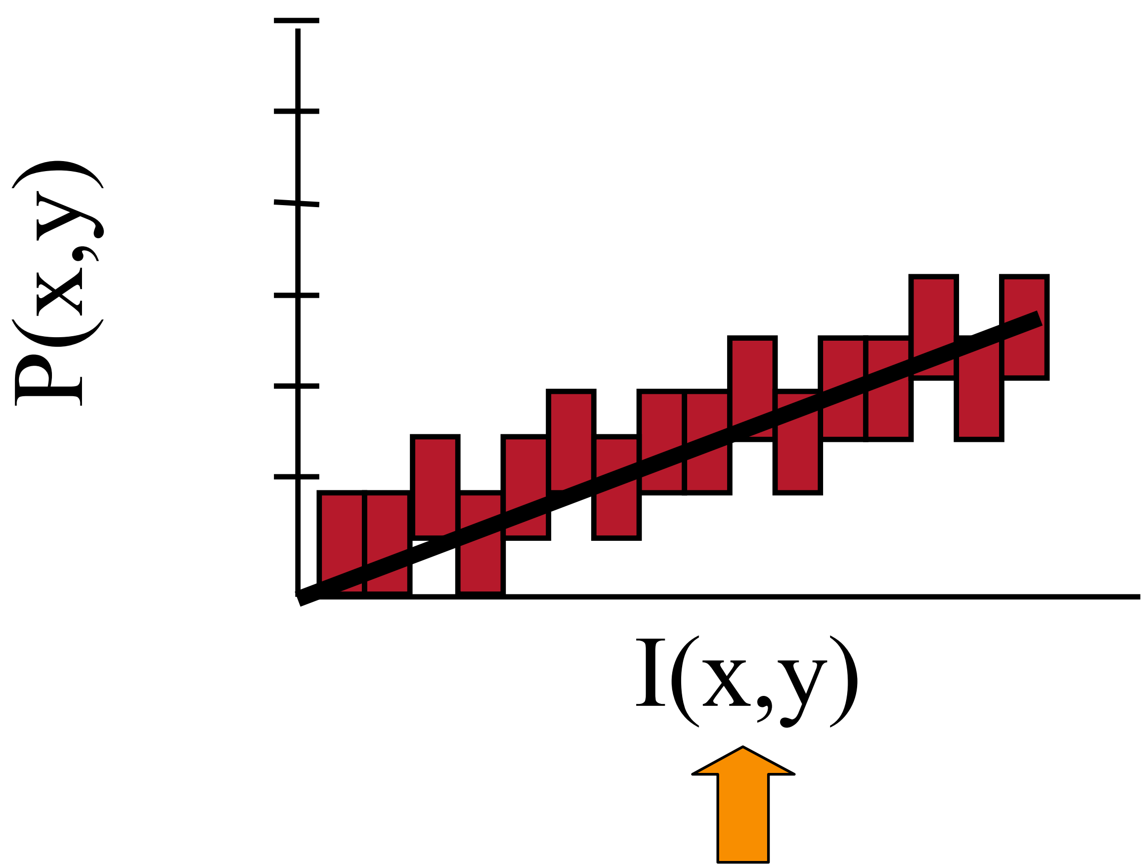
- Randomize quantization errors
- Errors appear as noise



$$P(x, y) = Q(I(x, y) + \text{noise}(x, y))$$

Random Dither

- Randomize quantization errors
- Errors appear as noise



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

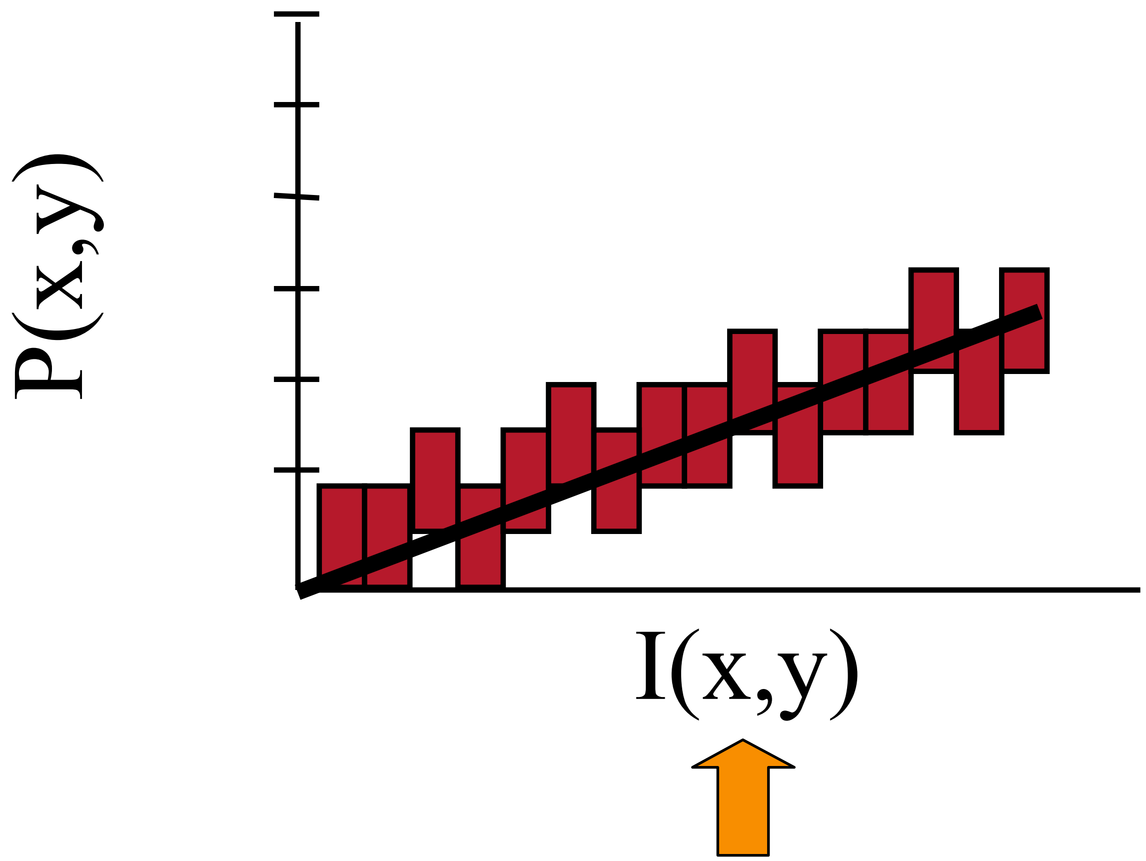
$$P(x,y) = Q(I(x,y) + \text{noise}(x,y))$$

Random Dither

- Randomize quantization errors
- Errors appear as noise

How much noise should we add?

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



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

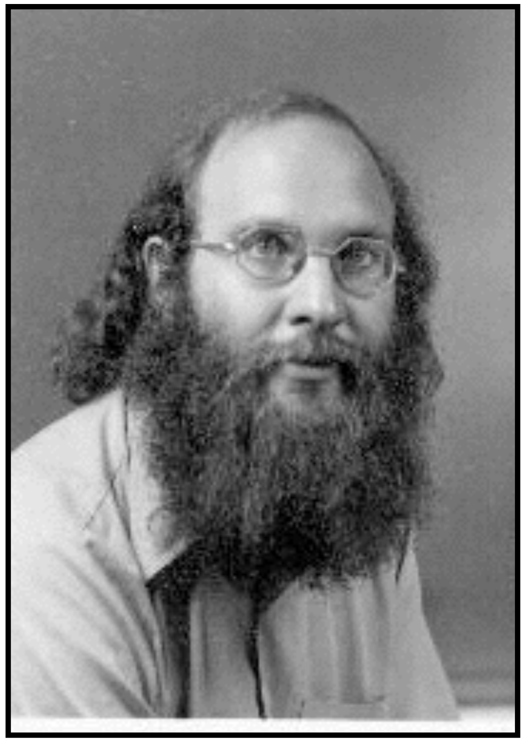
Random Dither

- Randomize quantization errors
- Errors appear as noise

How much noise should we add?

Enough so that we can effect rounding but not so much that we overshoot:
[-0.5/s,0.5/s],
where s is the intensity of a single dither level in the output: s = 2^(b-1).
(Note also the assignment uses a color range from 0 to 255).

Random Dither



Original
(8 bits)

Uniform
Quantization
(1 bit)

Random
Dither
(1 bit)

Ordered Dither

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

For Binary Displays

```
i = x mod n
j = y mod n
if (I(x,y)/255 > D(i,j) / (n^2+1))
        P(x,y) = 1
else
        P(x,y) = 0
```

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

Ordered Dither

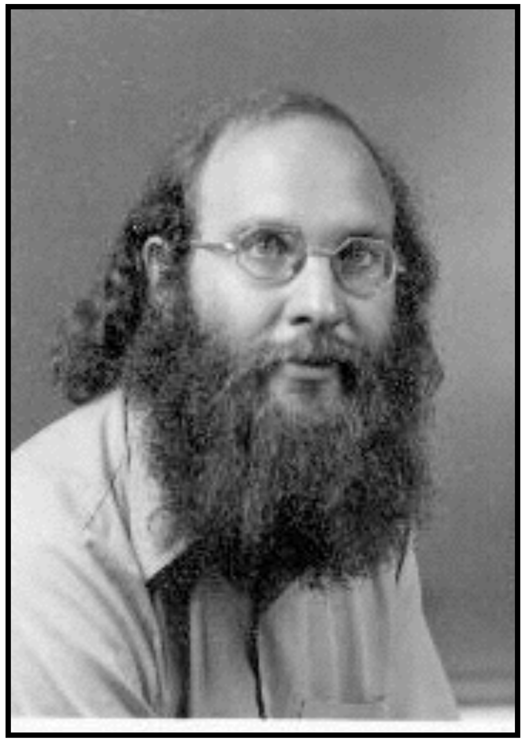
- Pseudo-random quantization errors
- Matrix stores pattern of thresholds

For b-Bit Displays

```
i = x mod n
j = y mod n
c = (I(x,y)/255)*(2^b-1)
e = c - floor(c)
if (e > D(i,j) / (n^2+1) )
        P(x,y) = ceil(c)
else
        P(x,y) = floor(c)
```

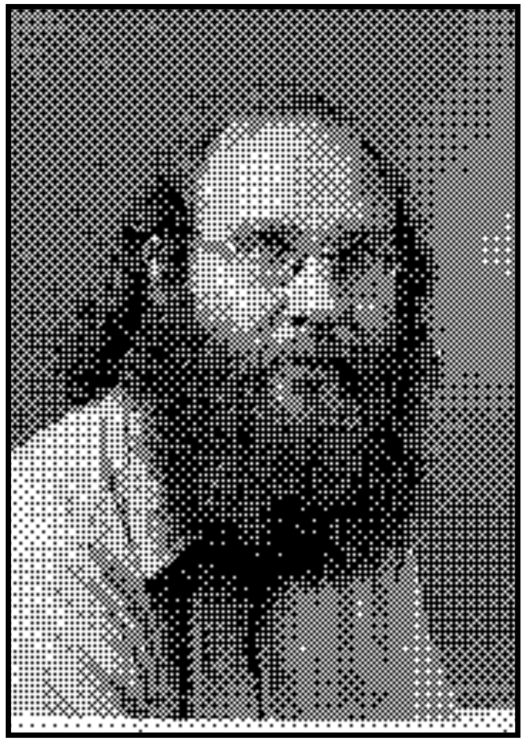
$$D_2 = \begin{bmatrix} 1 & 3 \\ 4 & 2 \end{bmatrix}$$

Ordered Dither



Original
(8 bits)

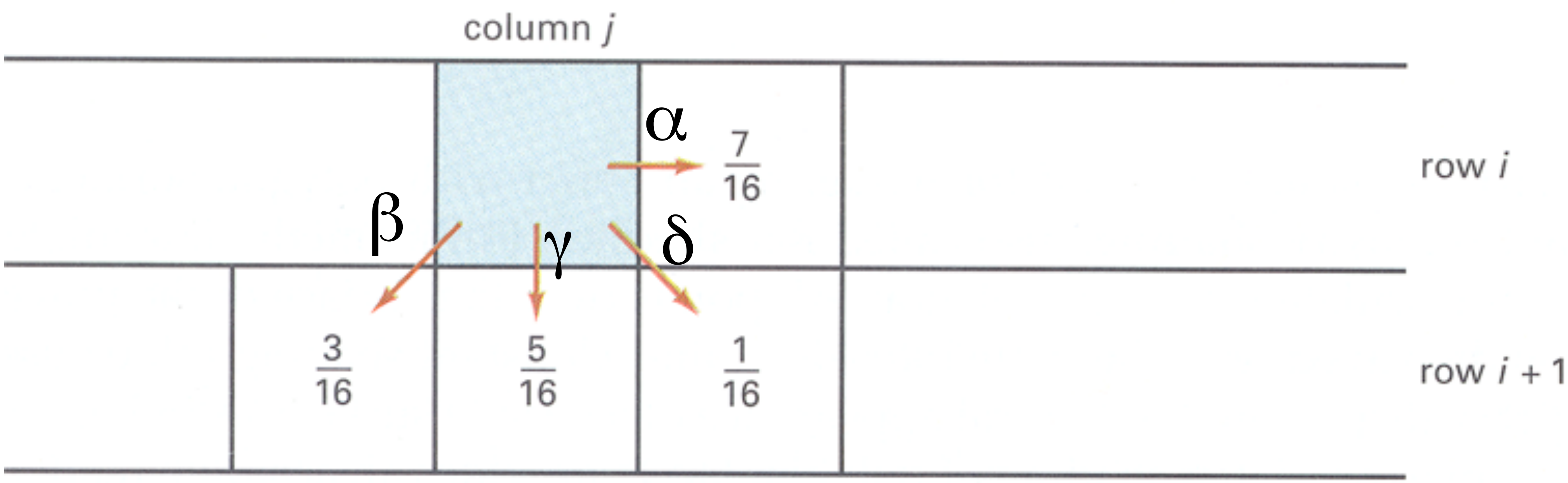
Random
Dither
(1 bit)



Ordered
Dither
(1 bit)

Error Diffusion Dither

- Spread quantization error over neighbor pixels
  - Error dispersed to pixels right and below
- Floyd-Steinberg Dither Method:



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

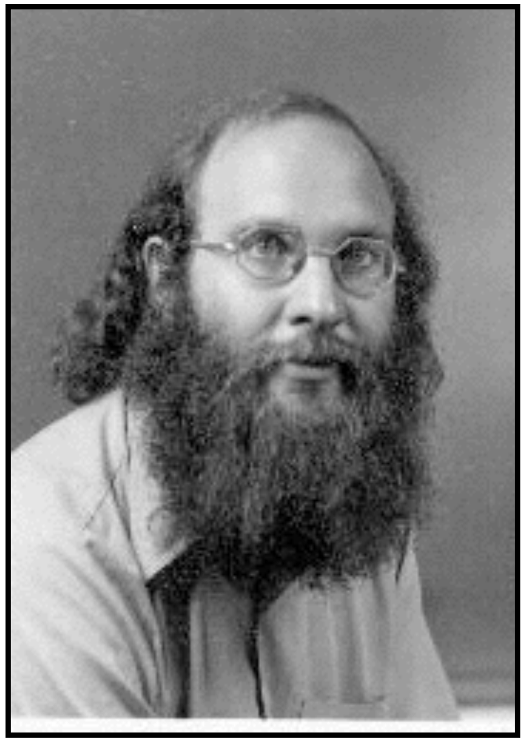
Figure 14.42 from H&B

Floyd-Steinberg Dither

$\alpha = 7/16$
$\beta = 3/16$
$\gamma = 5/16$
$\delta = 1/16$

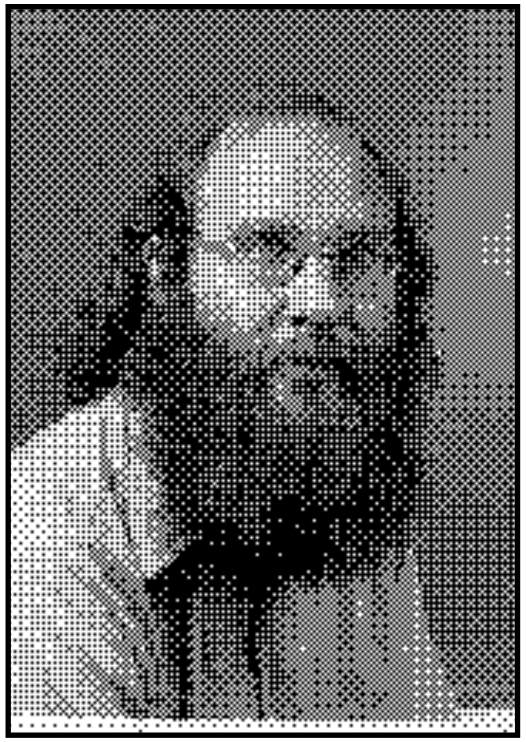
```
for (i = 0; i < width; i++)
    for (j = 0; j < height; j++)
        Dest[i,j] = quantize(Source[i,j])
        error = Source[i,j] - Dest[i,j]
        Source[i,j+1] = Source[i,j+1] + α * error
        Source[i+1,j-1] = Source[i+1,j-1] + β * error
        Source[i+1,j] = Source[i+1,j] + γ * error
        Source[i+1,j+1] = Source[i+1,j+1] + δ * error
```

Floyd-Steinberg Dither

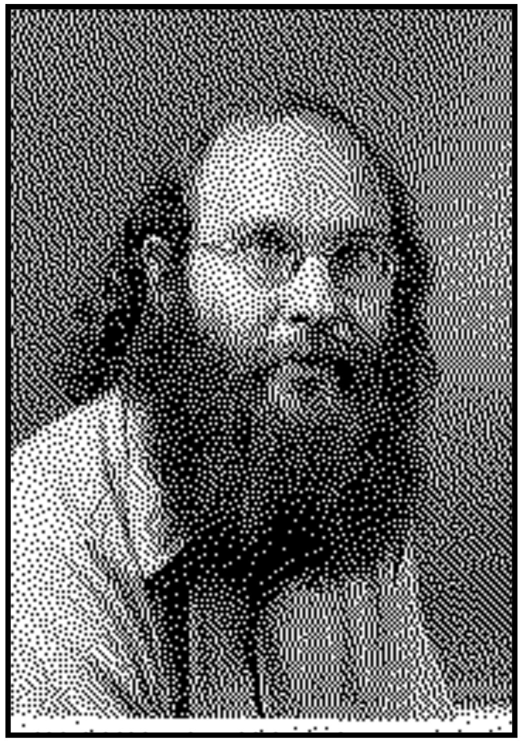


Original
(8 bits)

Random
Dither
(1 bit)



Ordered
Dither
(1 bit)

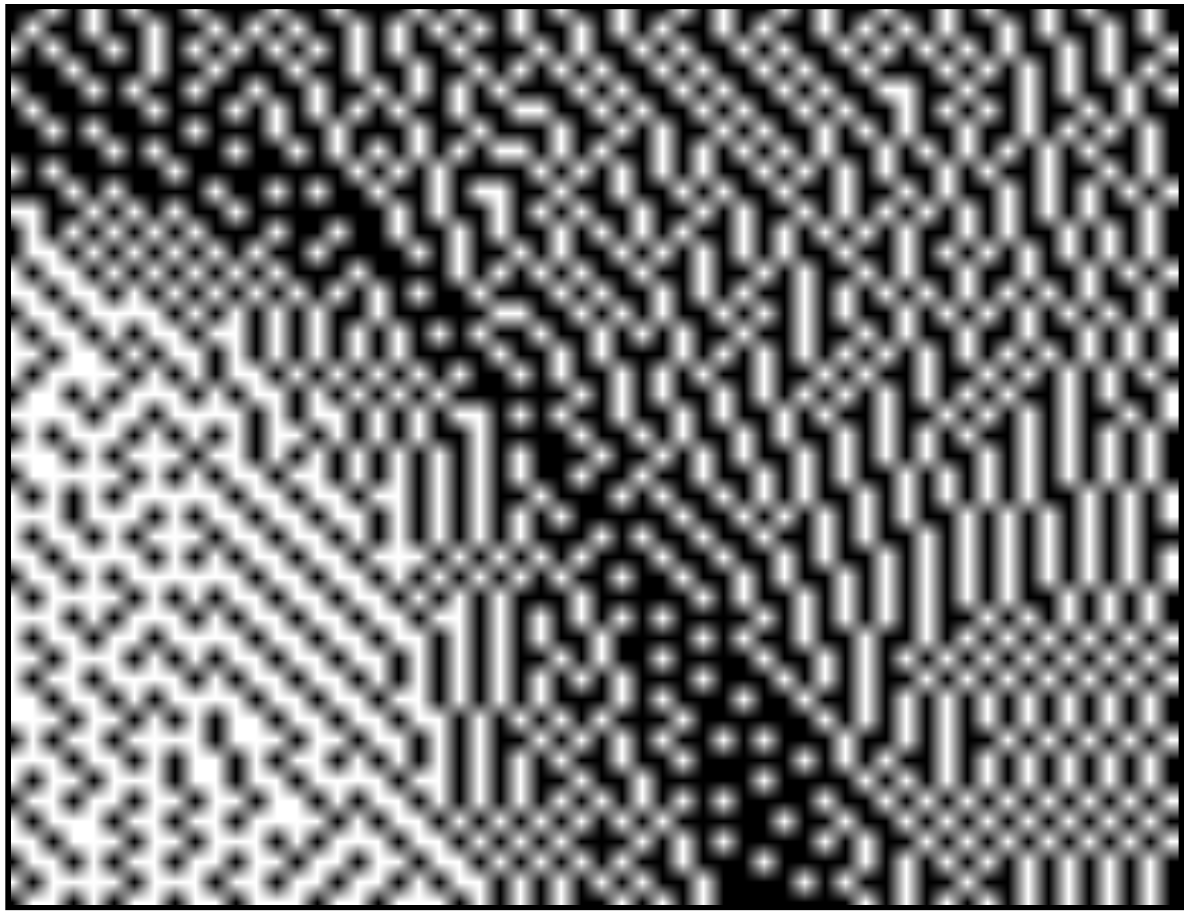


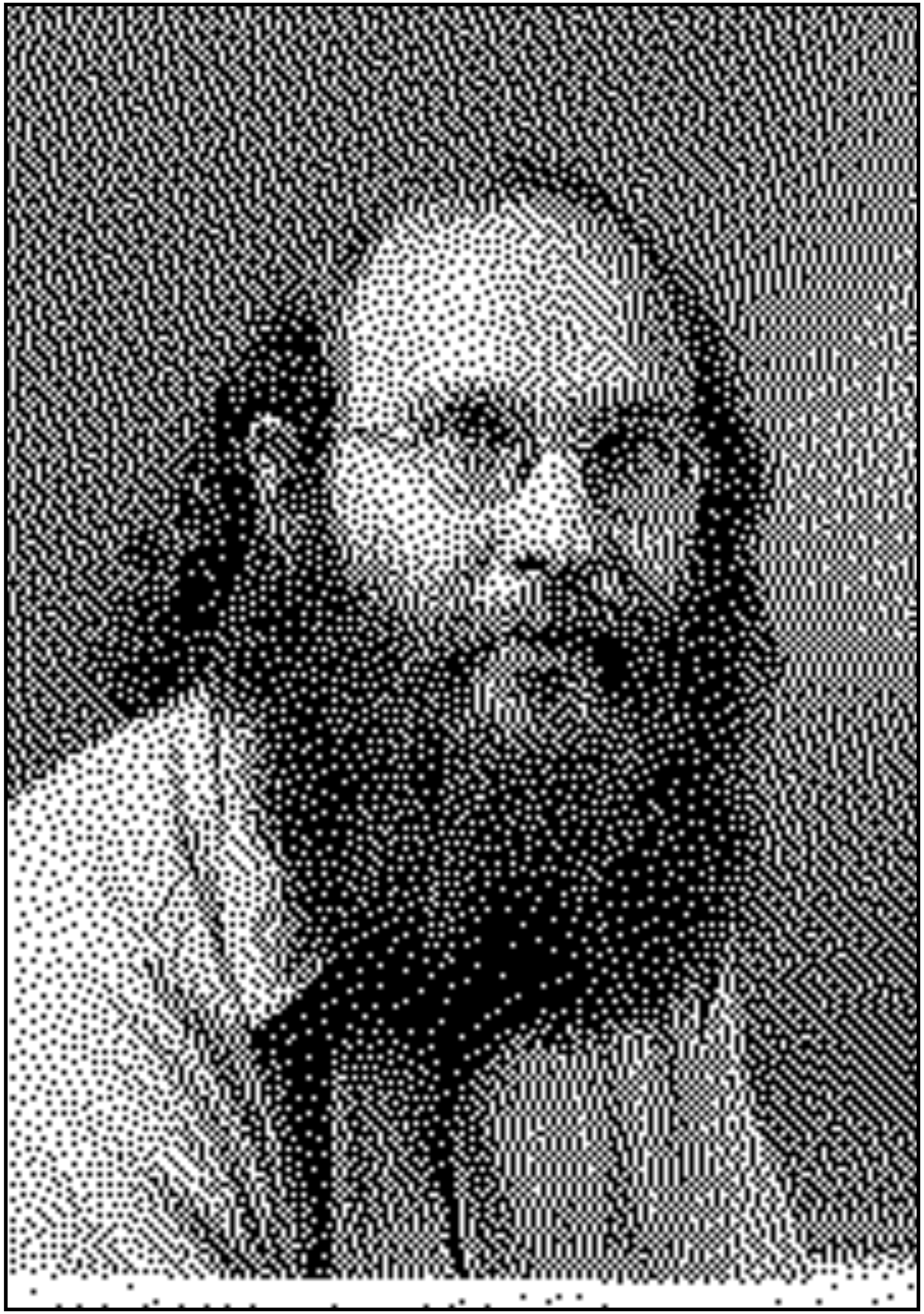
Floyd-Steinberg
Dither
(1 bit)

Discussion Q.

- How might one get a result even better than Floyd-Steinberg?

Close up view:





Outline

- Human Vision
- Image Representation
- Reducing Color Quantization Artifacts
- **Basic Image Processing**
    - Single Pixel Operations
    - Multi-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 need to take the weighted average of the RGBs: $L = 0.30*r + 0.59*g + 0.11*b$

Original

Grayscale

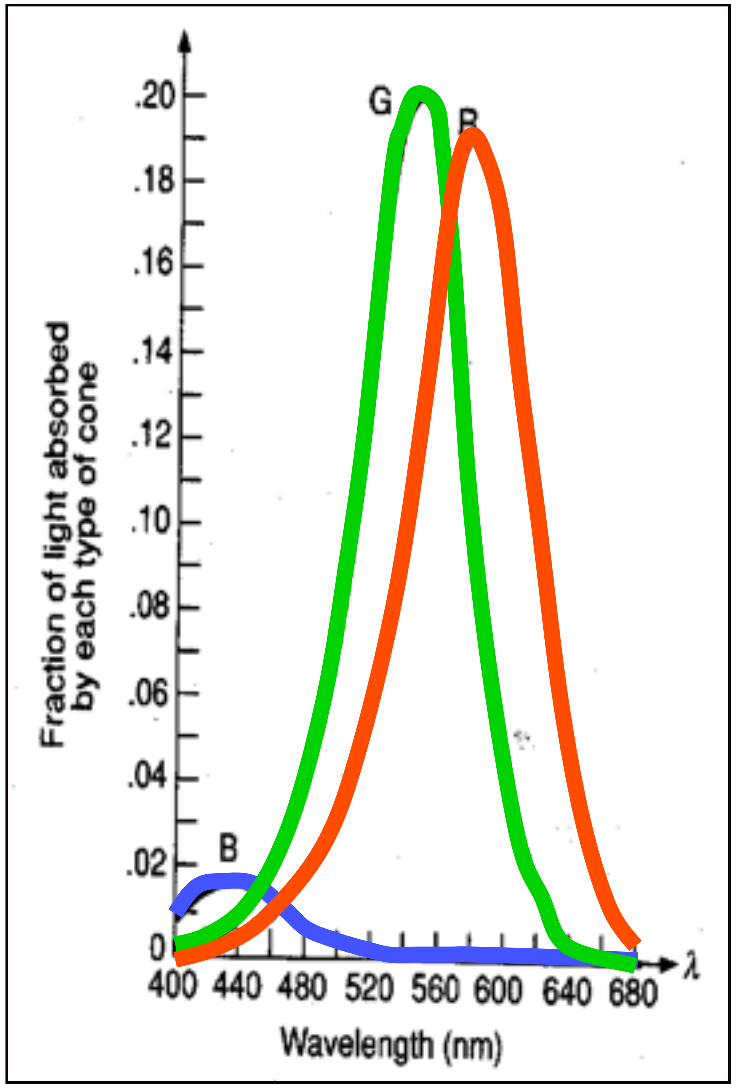


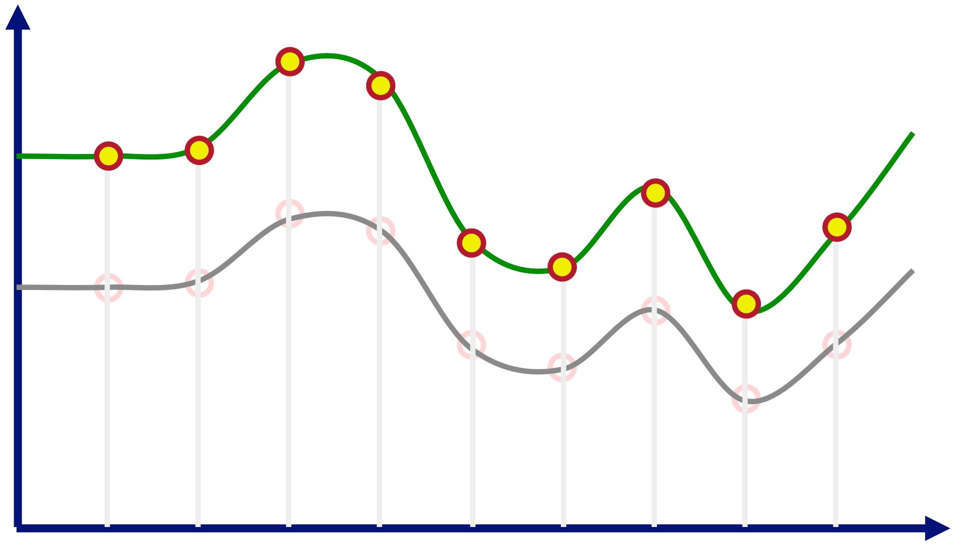
Figure 13.18 from FvDFH

Adjusting Brightness

- Simply scale pixel components
  - Must clamp to range (e.g., 0 to 255)

Original

Brighter

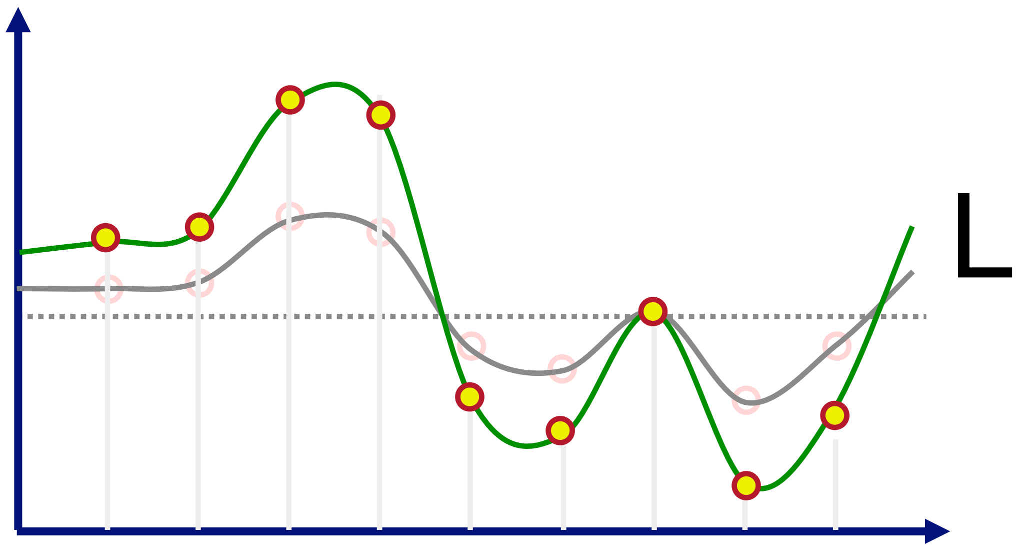


Adjusting Contrast

- Compute mean luminance L for all pixels
  - L = 0.30*r + 0.59*g + 0.11*b
- Scale deviation from L for each pixel color component (RGB)
  - Must clamp to range (e.g., 0 to 255)

Original

More Contrast



Adjusting Saturation

- Compute luminance L(p) for each pixel p
  - L(p) = 0.30*r(p) + 0.59*g(p) + 0.11*b(p)
- Scale deviation from L(p) for each pixel component (RGB)
  - Must clamp to range (e.g., 0 to 255)

Original

More Saturation

Image Processing by Interpolation

- Nice discussion of these operations: http://www.graficaobscura.com/interp/index.html

Image Processing by Interpolation

- Nice discussion of these operations: http://www.graficaobscura.com/interp/index.html

$$out = (1-alpha)*in0 + alpha*in1$$

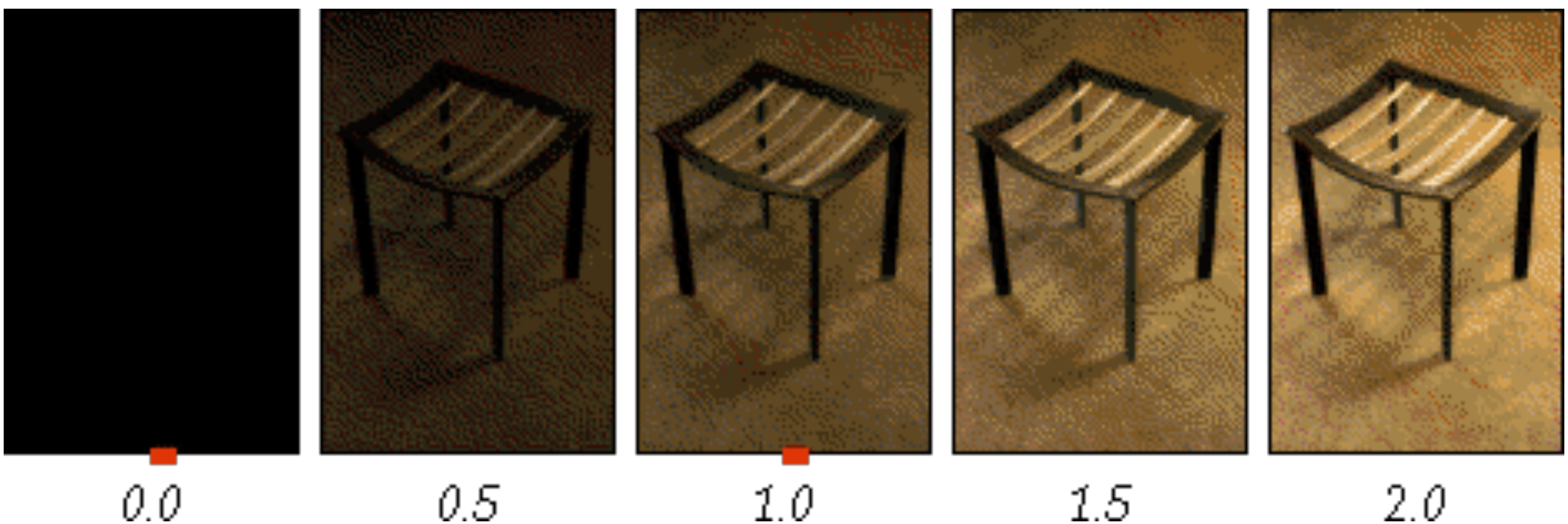


Image Processing by Interpolation

- Nice discussion of these operations:
  http://www.graficaobscura.com/interp/index.html

$$out = (1-alpha)*in0 + alpha*in1$$

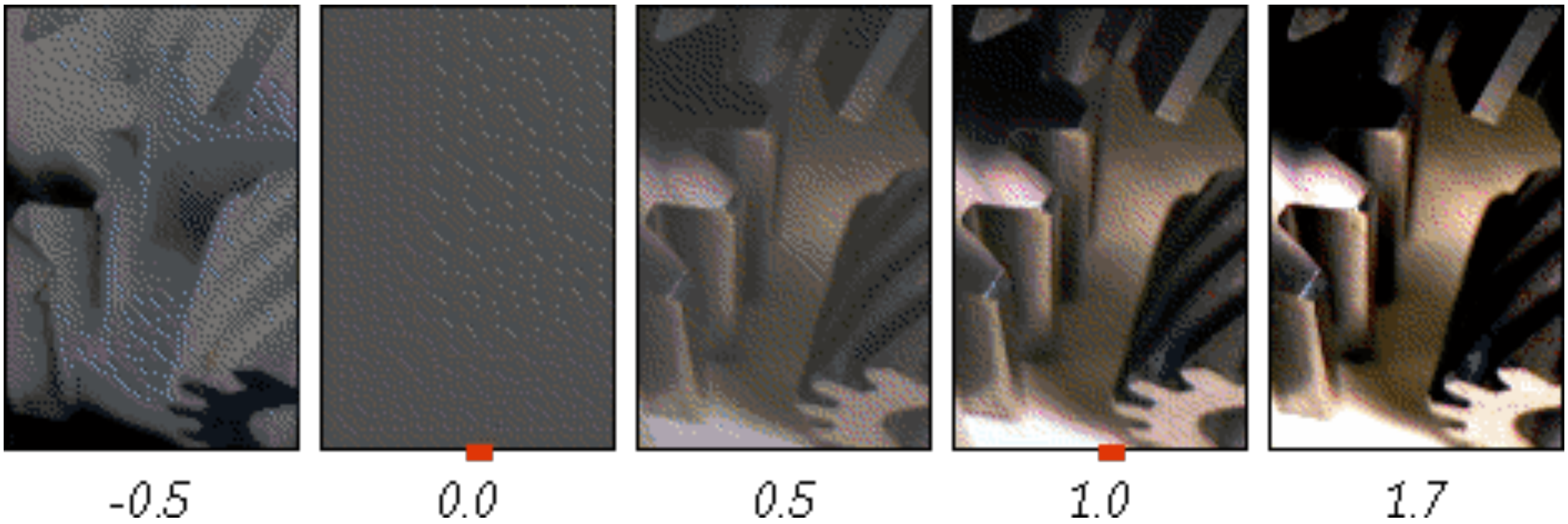


Image Processing by Interpolation

- Nice discussion of these operations: http://www.graficaobscura.com/interp/index.html

$$out = (1-alpha)*in0 + alpha*in1$$