Image Processing

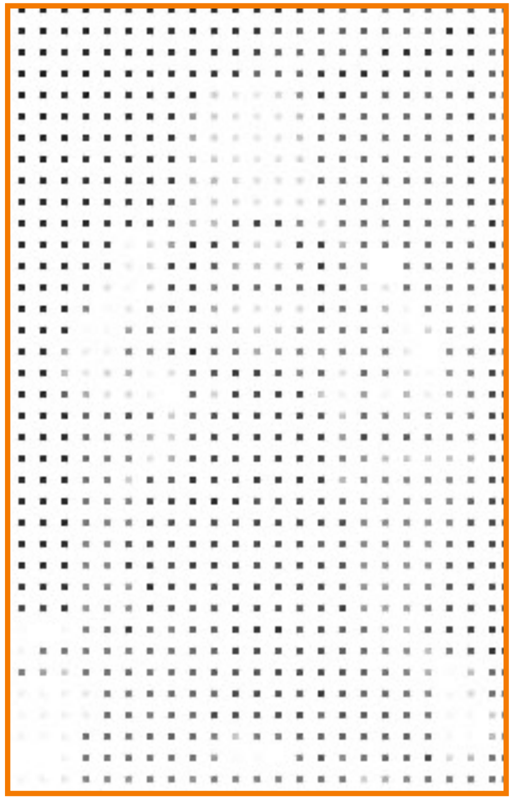
COS 426, Fall 2022

PRINCETON UNIVERSITY

What is a Digital Image?

- A digital image is a discrete array of samples representing a continuous 2D function

Continuous function



Discrete samples

Limitations on Digital Images

- Spatial discretization
- Quantized intensity
- Approximate color (RGB)
- (Temporally discretized frames for digital video)

Image Processing

- Changing pixel values
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Histogram equalization
- Filtering over neighborhoods
  - Blur & sharpen
  - Detect edges
  - Median
  - Bilateral filter
- Moving image locations
  - Scale
  - Rotate
  - Warp
- Combining images
  - Composite
  - Morph

Similar to Analog / Continuous

- Changing pixel values
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Histogram equalization
- Filtering over neighborhoods
  - Blur & sharpen
  - Detect edges
  - Median
  - Bilateral filter
- Moving image locations
  - Scale
  - Rotate
  - Warp
- Combining images
  - Composite
  - Morph

Account for Limitations of Digital

- Changing pixel values
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Histogram equalization
- Filtering over neighborhoods
  - Blur & sharpen
  - Detect edges
  - Median
  - Bilateral filter
- Moving image locations
  - Scale
  - Rotate
  - Warp
- Combining images
  - Composite
  - Morph

New Operations

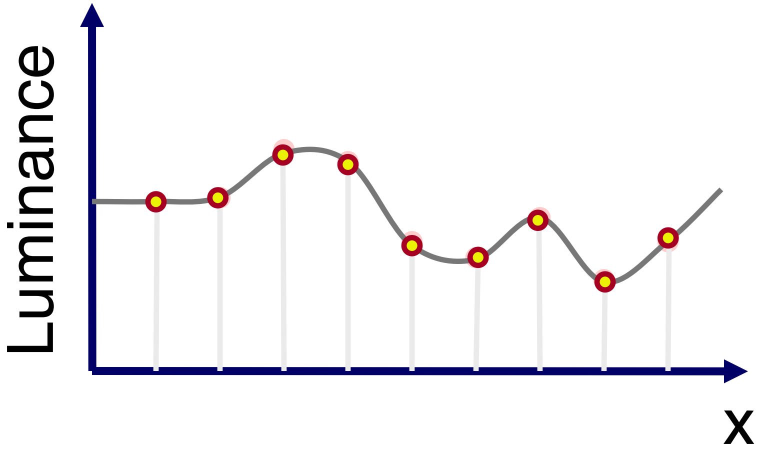
- Changing pixel values
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Histogram equalization
- Filtering over neighborhoods
  - Blur & sharpen
  - Detect edges
  - Median
  - Bilateral filter
- Moving image locations
  - Scale
  - Rotate
  - Warp
- Combining images
  - Composite
  - Morph
- Quantization
- Spatial / intensity tradeoff
  - Dithering

Digital Image Processing

- Changing pixel values
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Histogram equalization
- Filtering over neighborhoods
  - Blur & sharpen
  - Detect edges
  - Median
  - Bilateral filter
- Moving image locations
  - Scale
  - Rotate
  - Warp
- Combining images
  - Composite
  - Morph
- Quantization
- Spatial / intensity tradeoff
  - Dithering

Adjusting Brightness

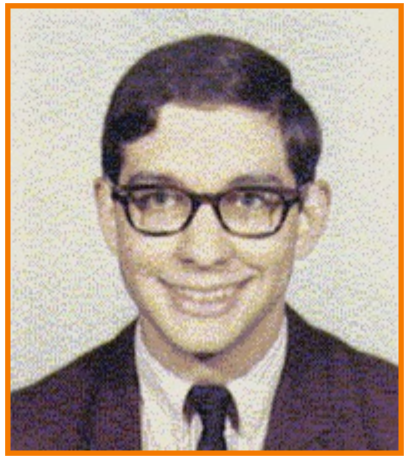
- What must be done to the RGB values to make this image brighter?



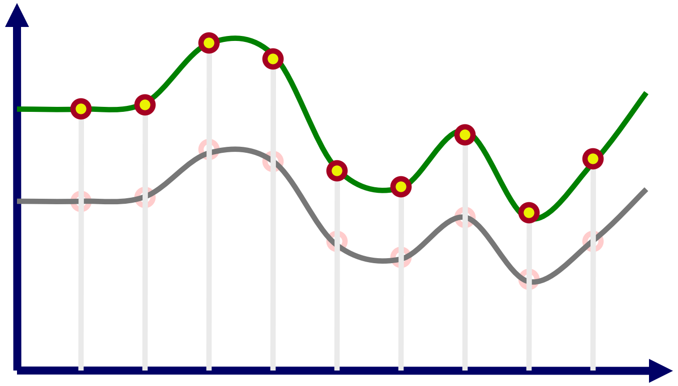
Adjusting Brightness

- Simply scale pixel components
  - Must clamp to range, e.g. [0..1] or [0..255]

Original



Brighter



Note: this is often "contrast" on your monitor!
"Brightness" adjusts black level (offset)

Adjusting Contrast

- Intuitively, “mid-tone” pixels should stay the same, dark ones get darker, light ones get lighter
- Preserve average *luminance*

Original

More Contrast

What is Luminance?

- Measures perceived “gray-level” of pixel
  - L = 0.30*red + 0.59*green + 0.11*blue

0.5

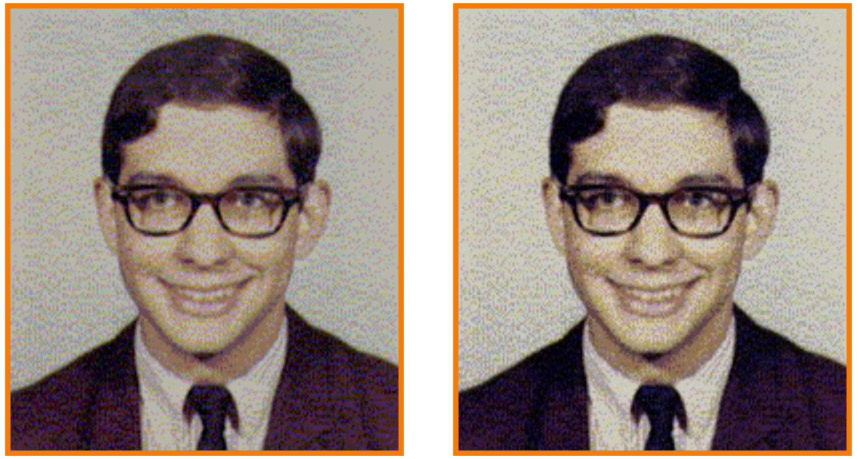
0.0

0.7

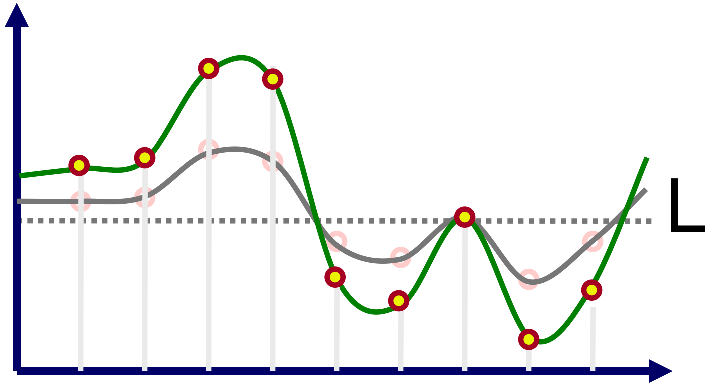
0.9

Adjusting Contrast

- Compute mean luminance L for all pixels
  - luminance = 0.30*r + 0.59*g + 0.11*b
- Scale deviation from L for each pixel component
  - Must clamp to range (e.g., 0 to 1)



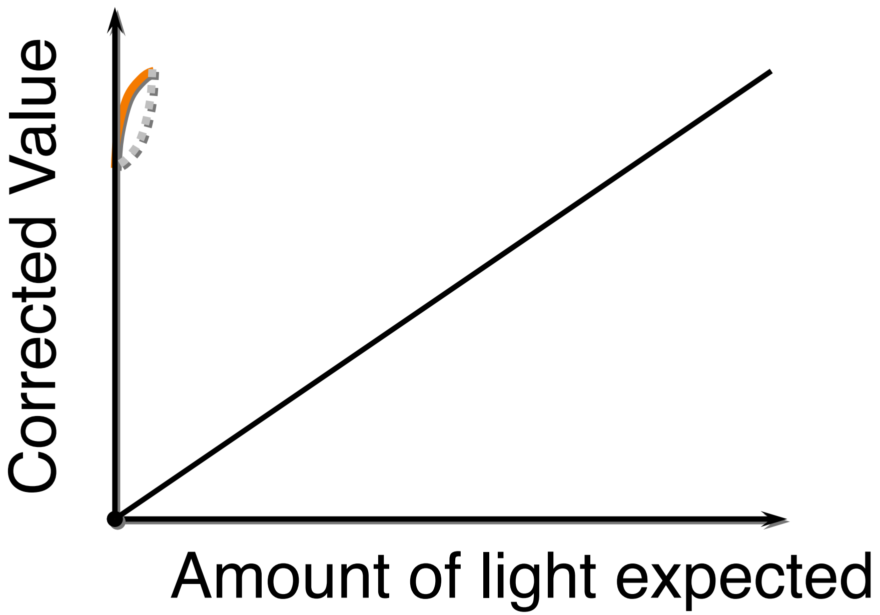
Original More Contrast



Adjusting Gamma

- Function originally accounting for nonlinearity in cameras and displays

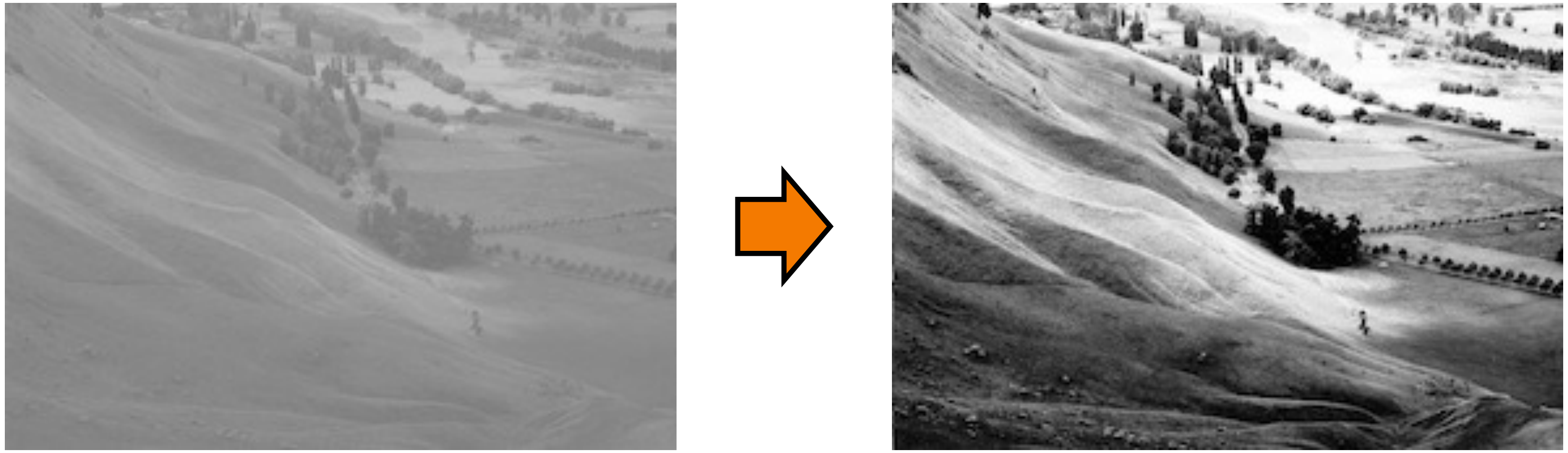
$$I_{out} = I_{in}^{\gamma}$$



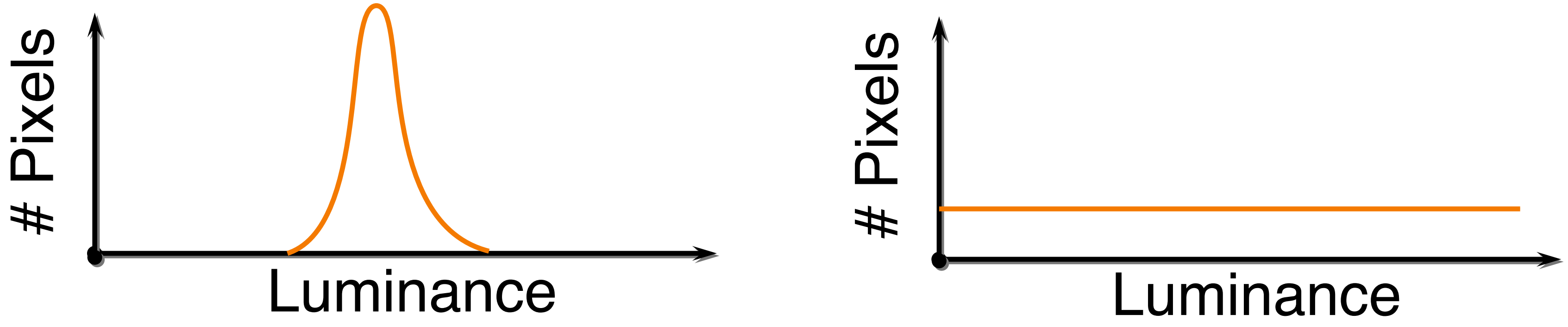
- γ depends on camera and monitor

Histogram Equalization

- Change distribution of luminance values to cover full range [0-1]

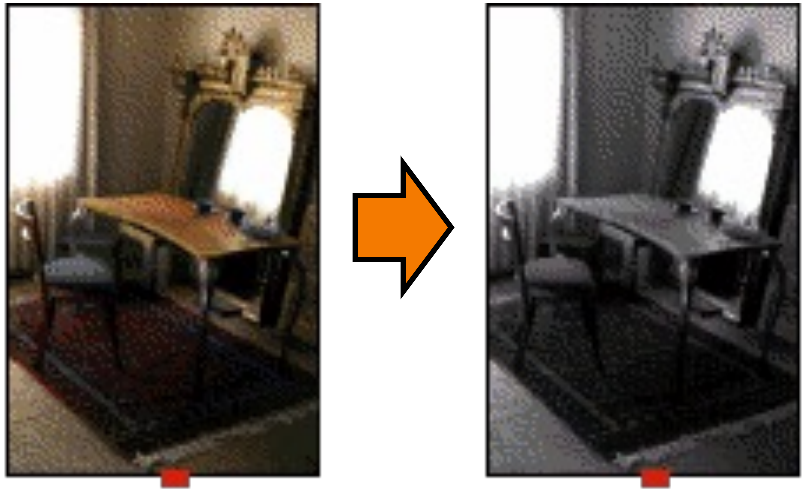


http://en.wikipedia.org/wiki/Histogram_equalization



Grayscale

- Convert from color to gray-levels



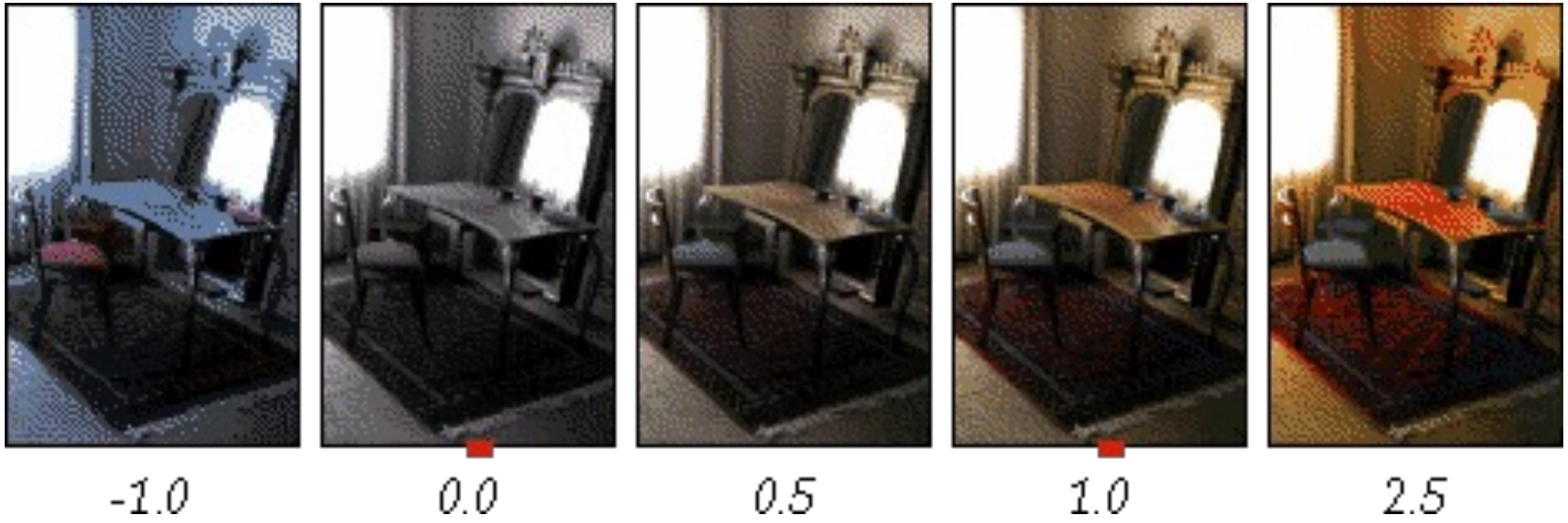
Original

Grayscale
(“black&white” photo)

Compute luminance L, set every pixel to (L,L,L)

Adjusting Saturation

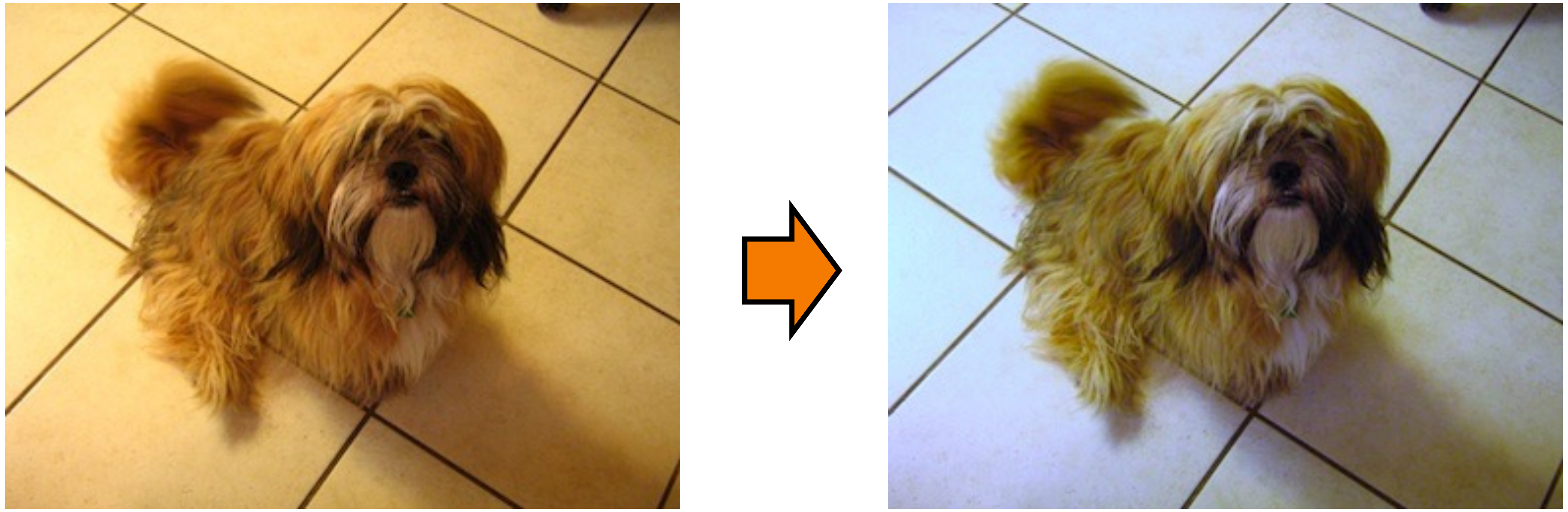
- Increase/decrease color saturation of every pixel



Interpolate / extrapolate between image and grayscale version

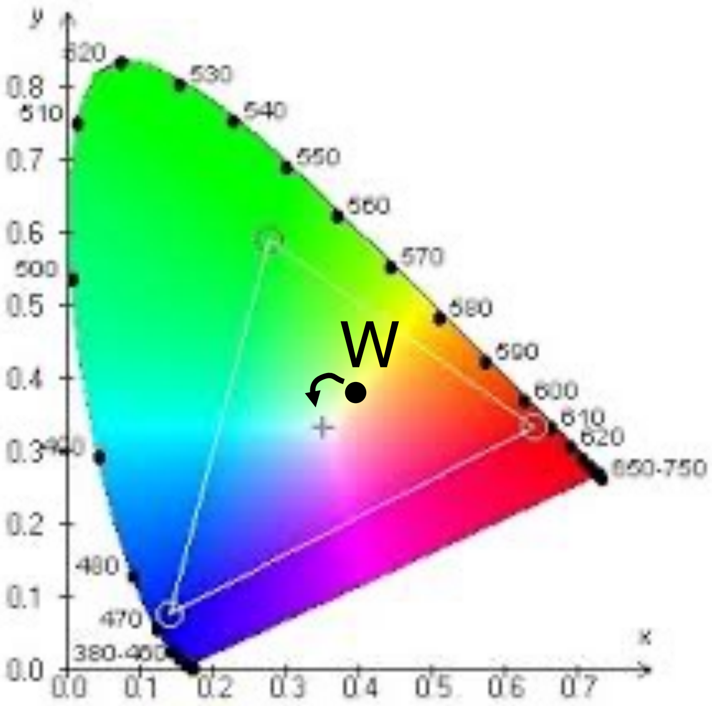
White Balance

- Adjust colors so that a given RGB value is mapped to white



White Balance

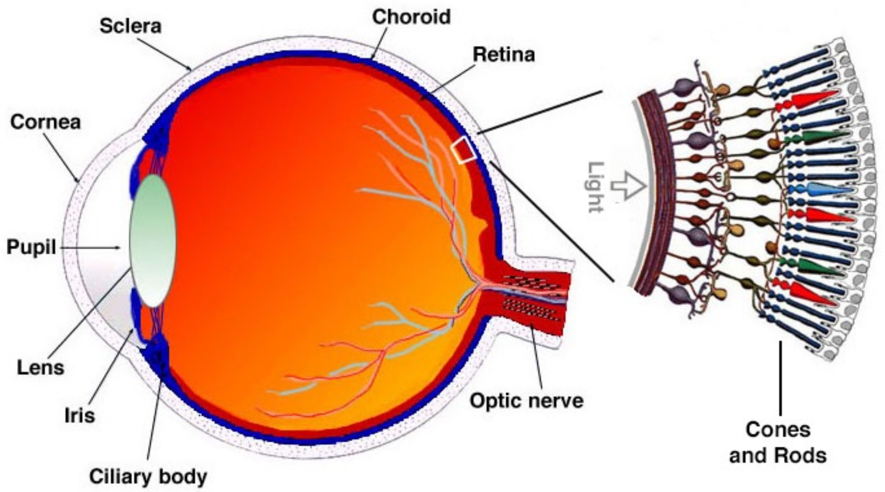
- Conceptually:
  - Provide an RGB value W that should be mapped to white
  - Perform transformation of color space

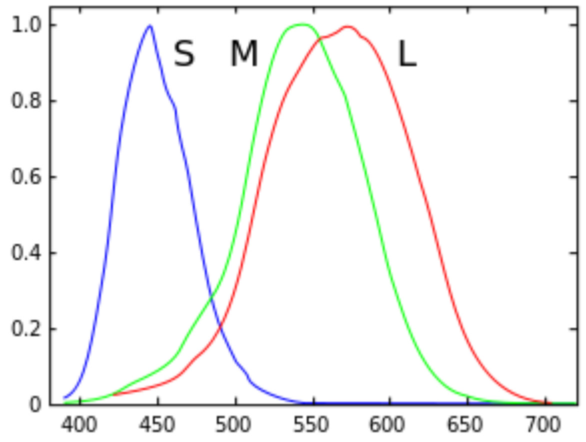


White Balance

Von Kries method: adjust colors in LMS color space

- LMS primaries represent the responses of the three different types of cones in our eyes





White Balance

For each pixel RGB:

1) Convert to XYZ color space

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9502 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

2) Convert to LMS color space

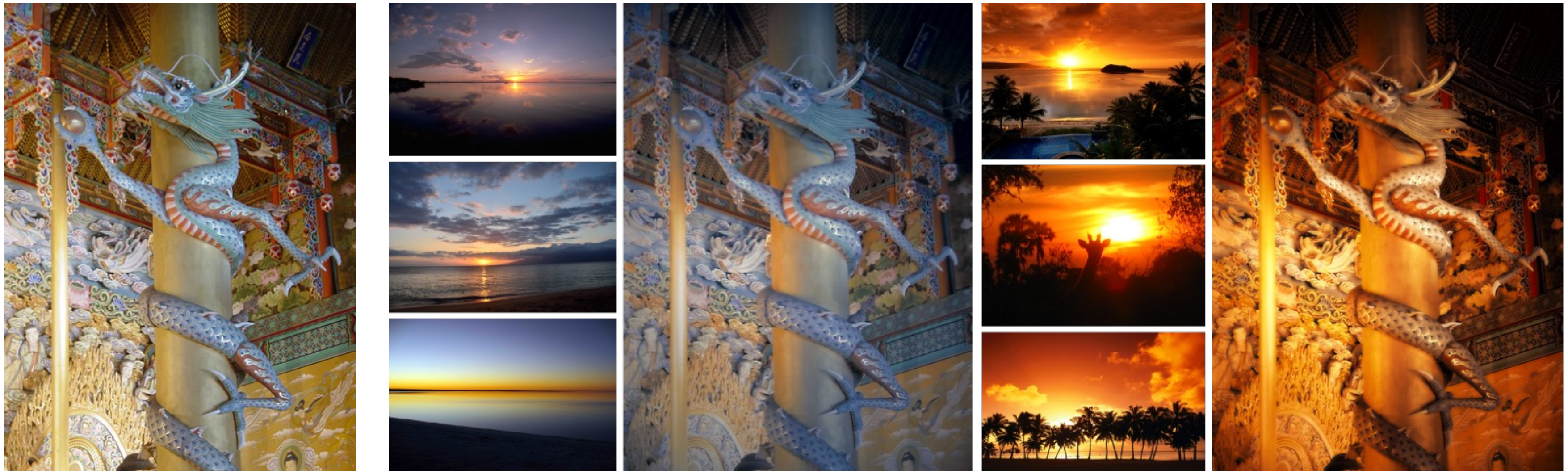
$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.40024 & 0.7076 & -0.08081 \\ -0.2263 & 1.16532 & 0.0457 \\ 0 & 0 & 0.91822 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

3) Divide by $L_W M_W S_W$

4) Convert back to RGB

Color Histogram Transfer

- Adjust colors so that their distribution (histogram) matches a target distribution



Source image | Target colors | Result | Target colors | Result

Fancier version of this idea from “AutoStyle: Automatic Style Transfer from Image Collections to Users' Images” by Princeton student Yiming Liu et al.

Digital Image Processing

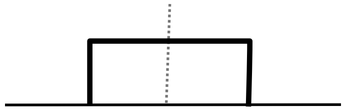
- Changing pixel values
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Histogram equalization
- Filtering over neighborhoods
  - Blur & sharpen
  - Detect edges
  - Median
  - Bilateral filter
- Moving image locations
  - Scale
  - Rotate
  - Warp
- Combining images
  - Composite
  - Morph
- Quantization
- Spatial / intensity tradeoff
  - Dithering

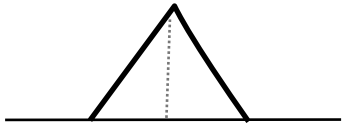
Blur

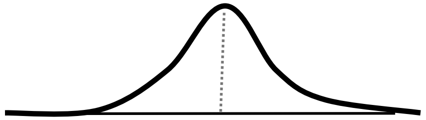
- What is the basic operation for each pixel when blurring an image?

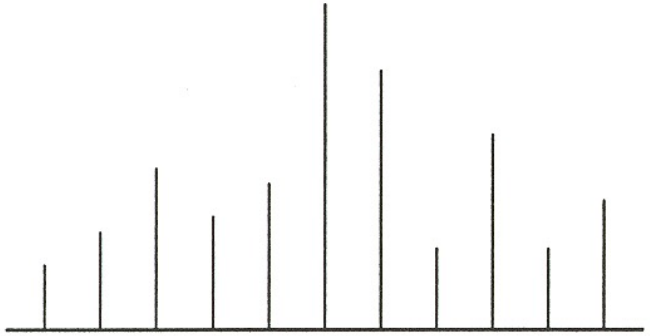
Basic Operation: Convolution

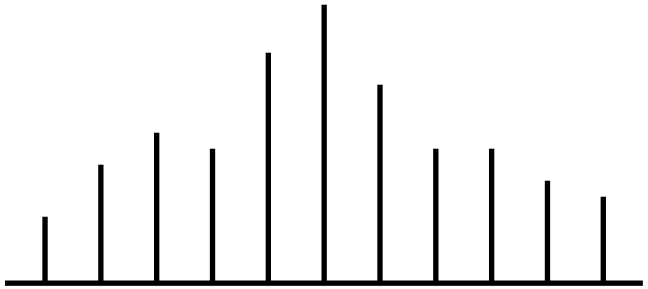
- Output is weighted sum of values in neighborhood of input image
  - Pattern of weights is the “filter” or “kernel”


Box Filter


Triangle Filter

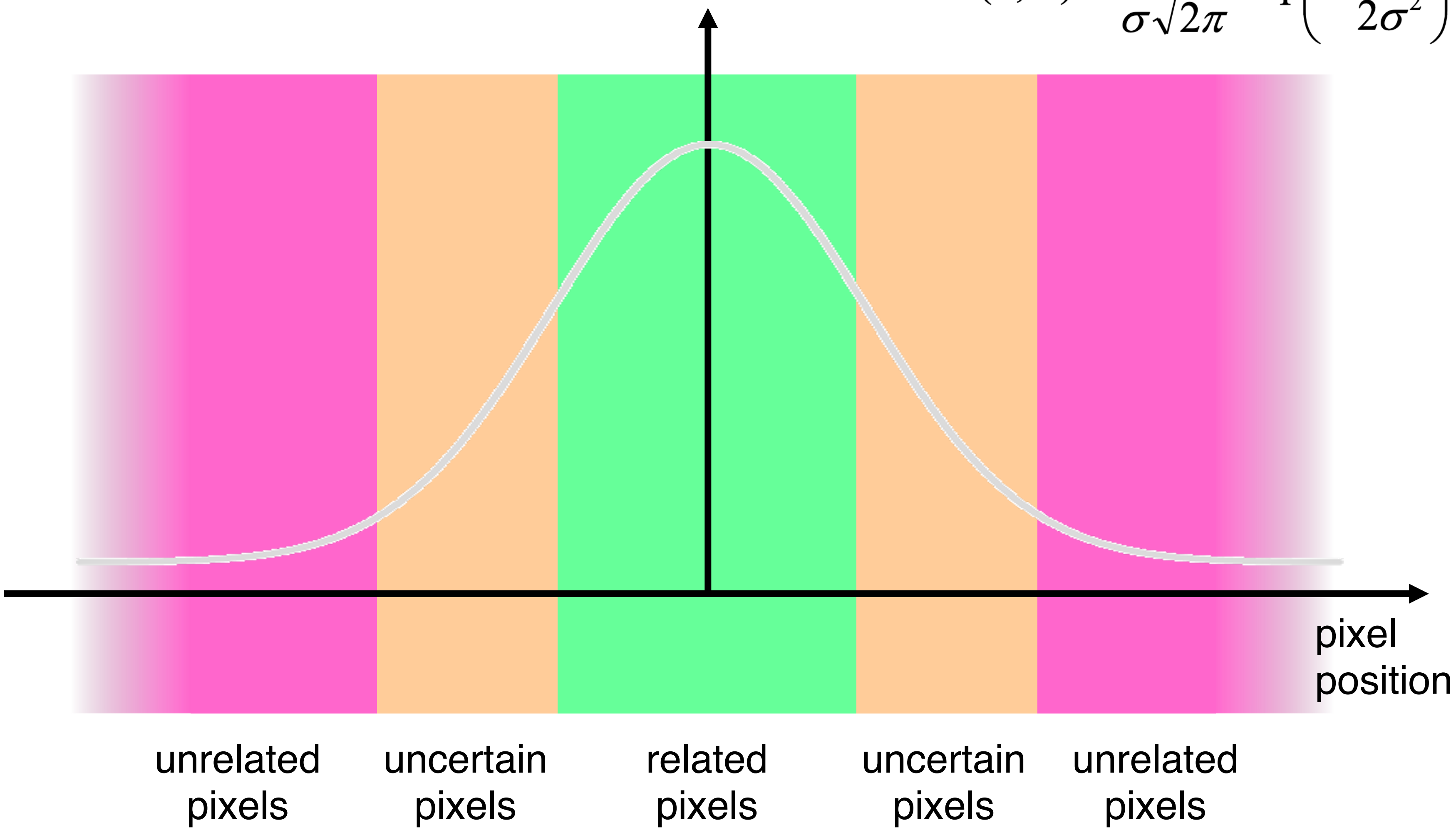

Gaussian Filter


Input


Output

Convolution with a Gaussian Filter

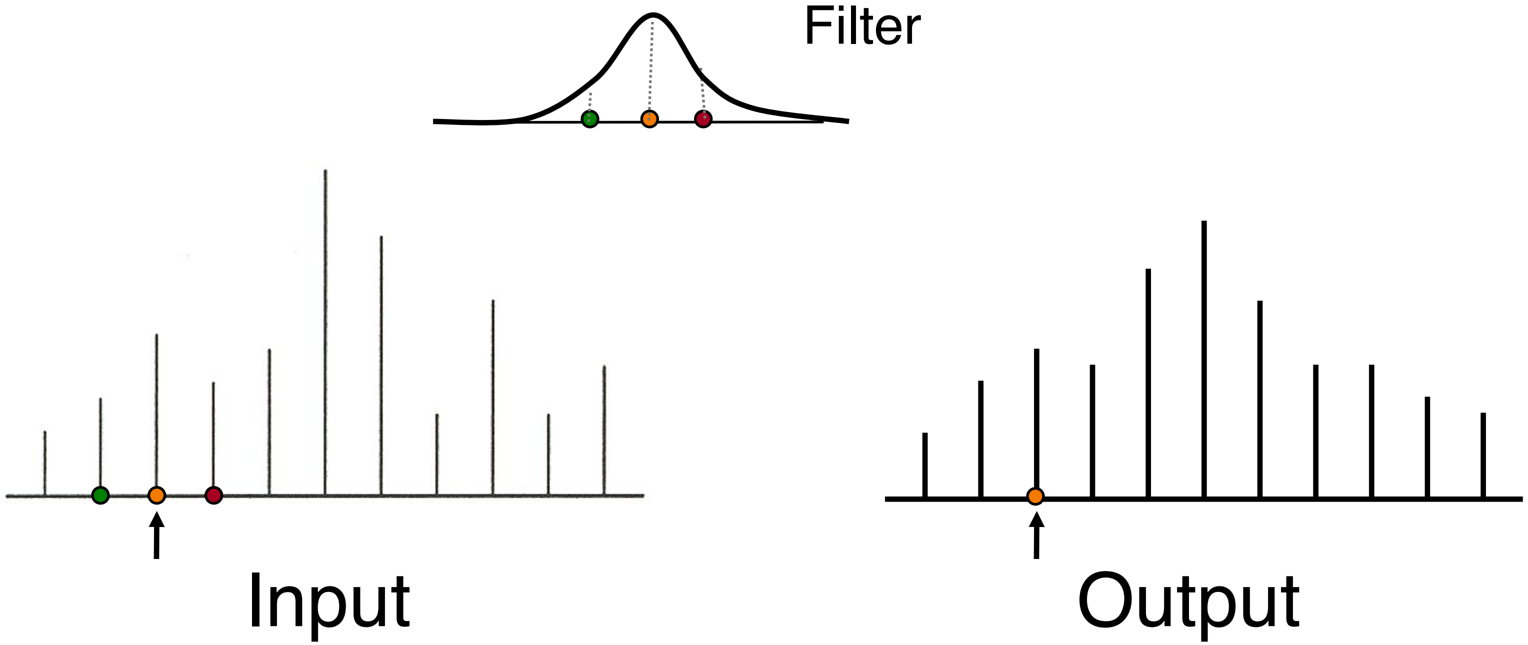
$$G(x,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$



Convolution with a Gaussian Filter

- Output is weighted sum of values in neighborhood of input image

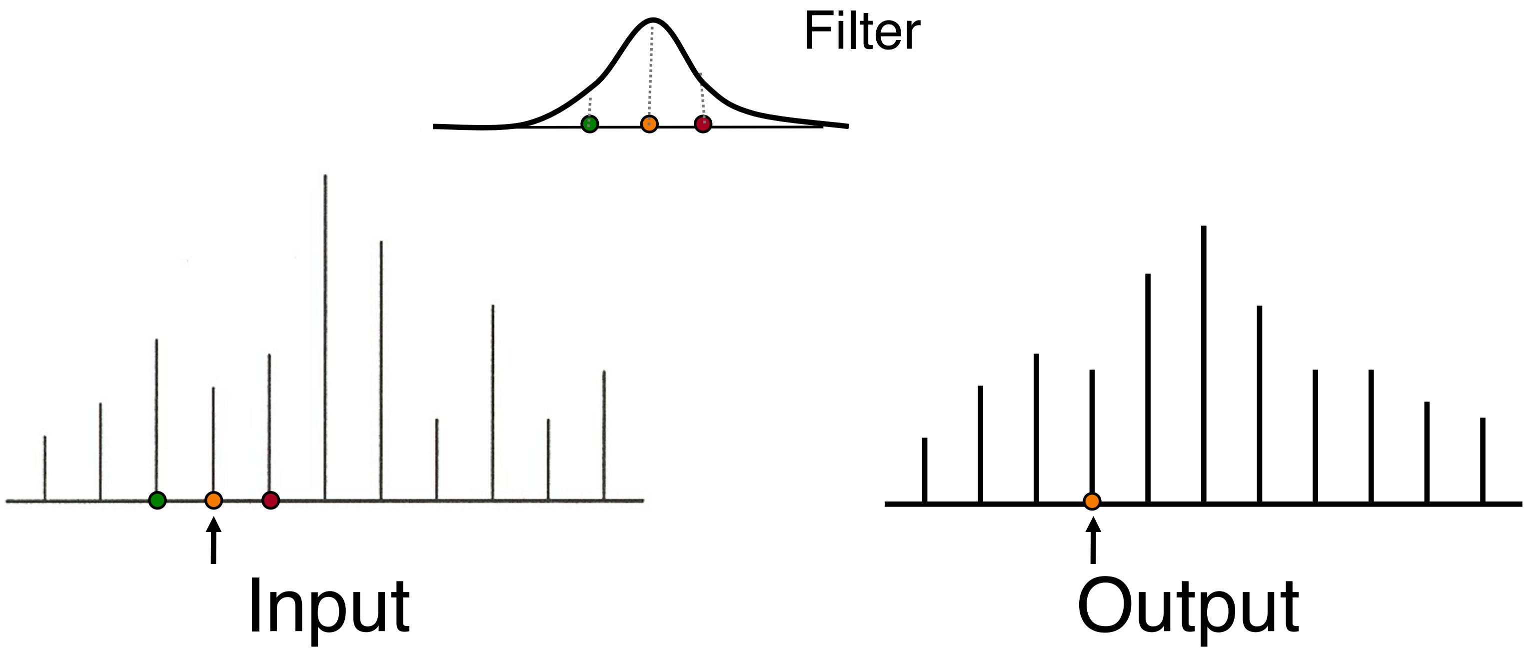
$$G(x,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$



Convolution with a Gaussian Filter

- Output is weighted sum of values in neighborhood of input image

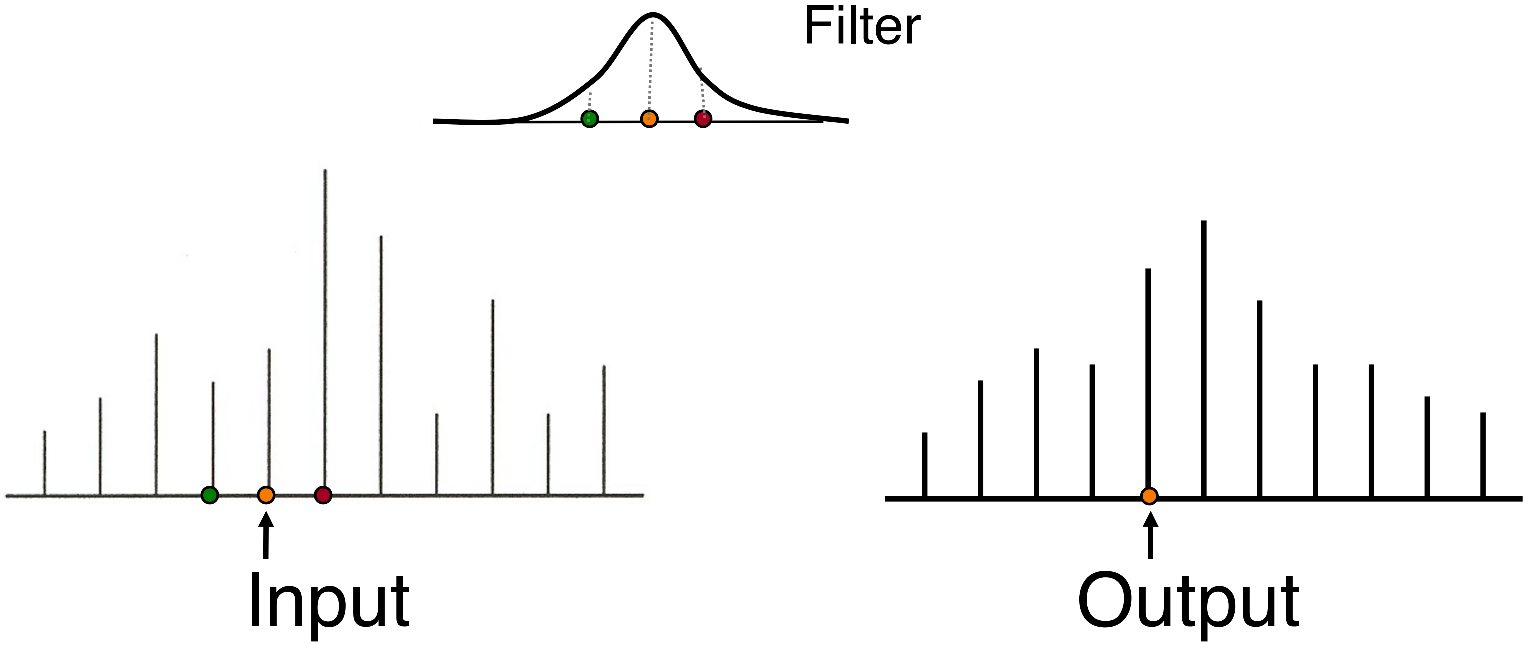
$$G(x,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$



Convolution with a Gaussian Filter

- Output is weighted sum of values in neighborhood of input image

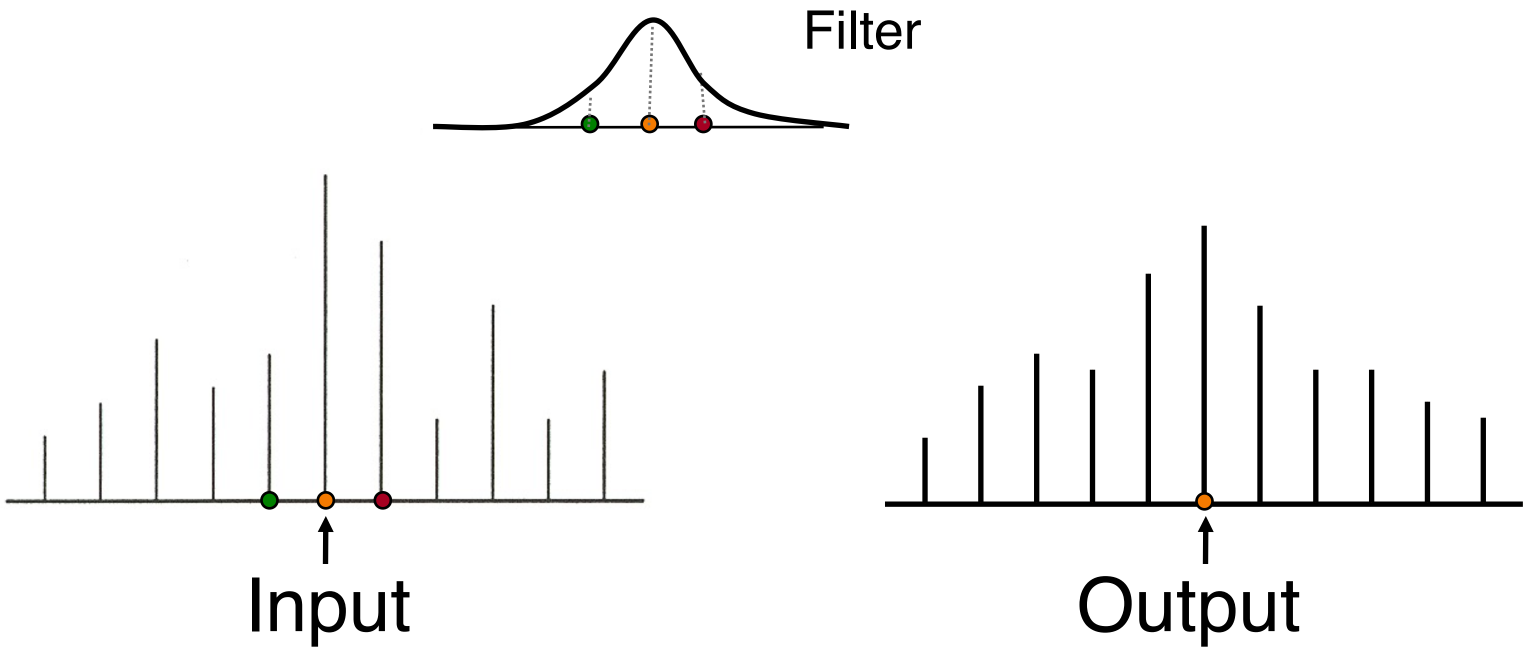
$$G(x,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$



Convolution with a Gaussian Filter

- Output is weighted sum of values in neighborhood of input image

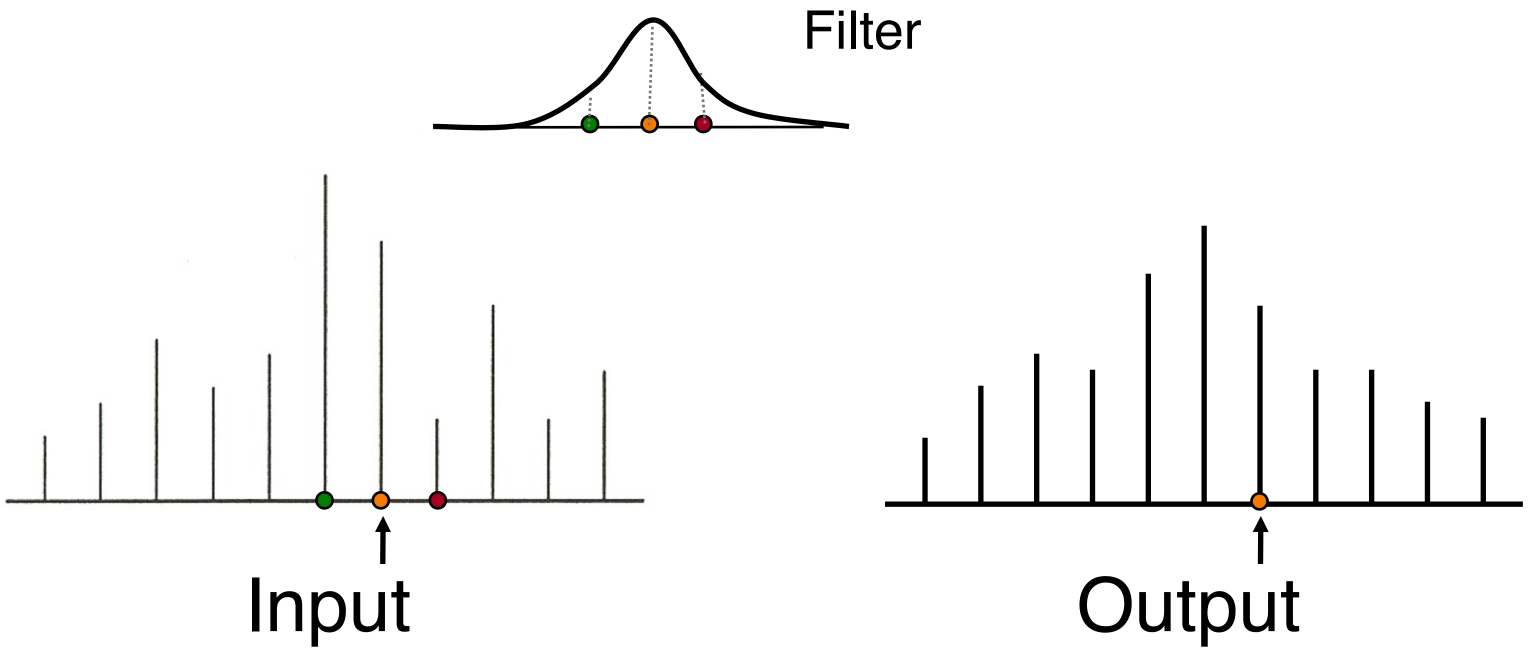
$$G(x,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$



Convolution with a Gaussian Filter

- Output is weighted sum of values in neighborhood of input image

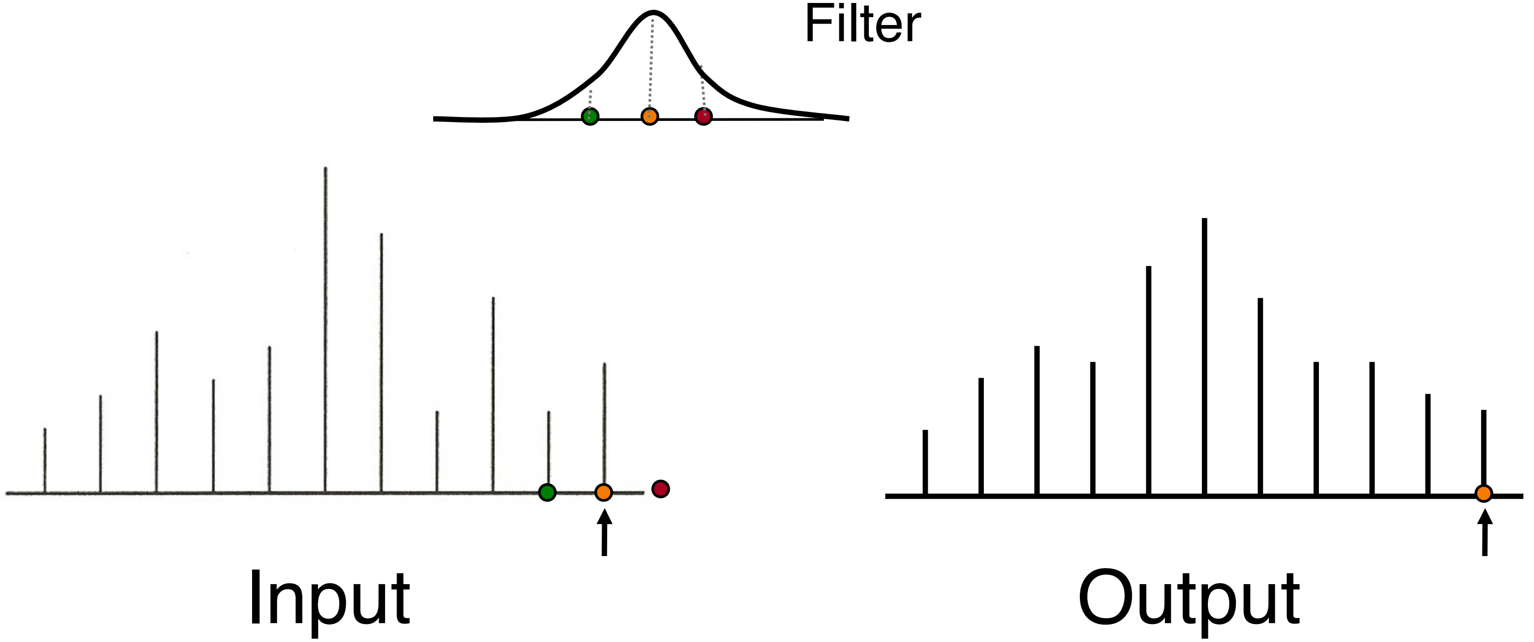
$$G(x,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$



Convolution with a Gaussian Filter

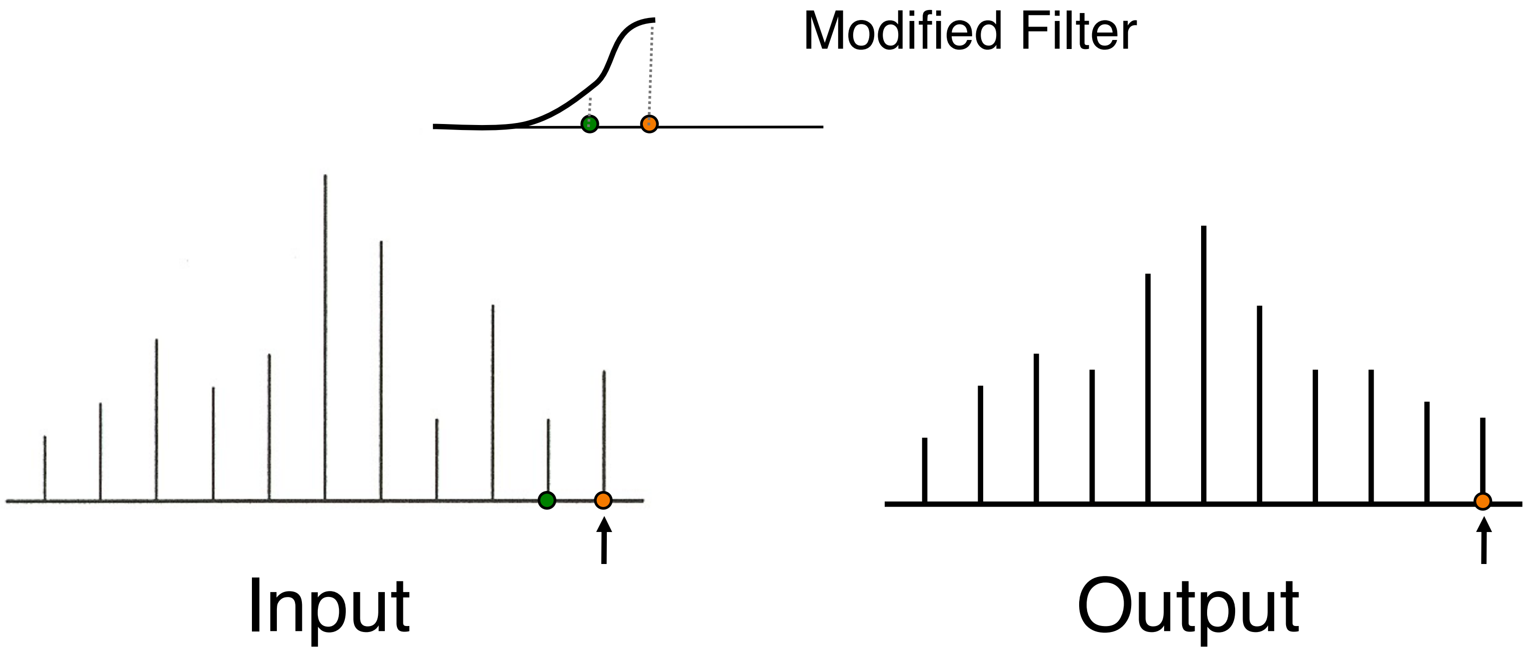
- What if filter extends beyond boundary?

$$G(x,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$



Convolution with a Gaussian Filter

- What if filter extends beyond boundary?



Convolution with a Gaussian Filter

- Output contains samples from smoothed input

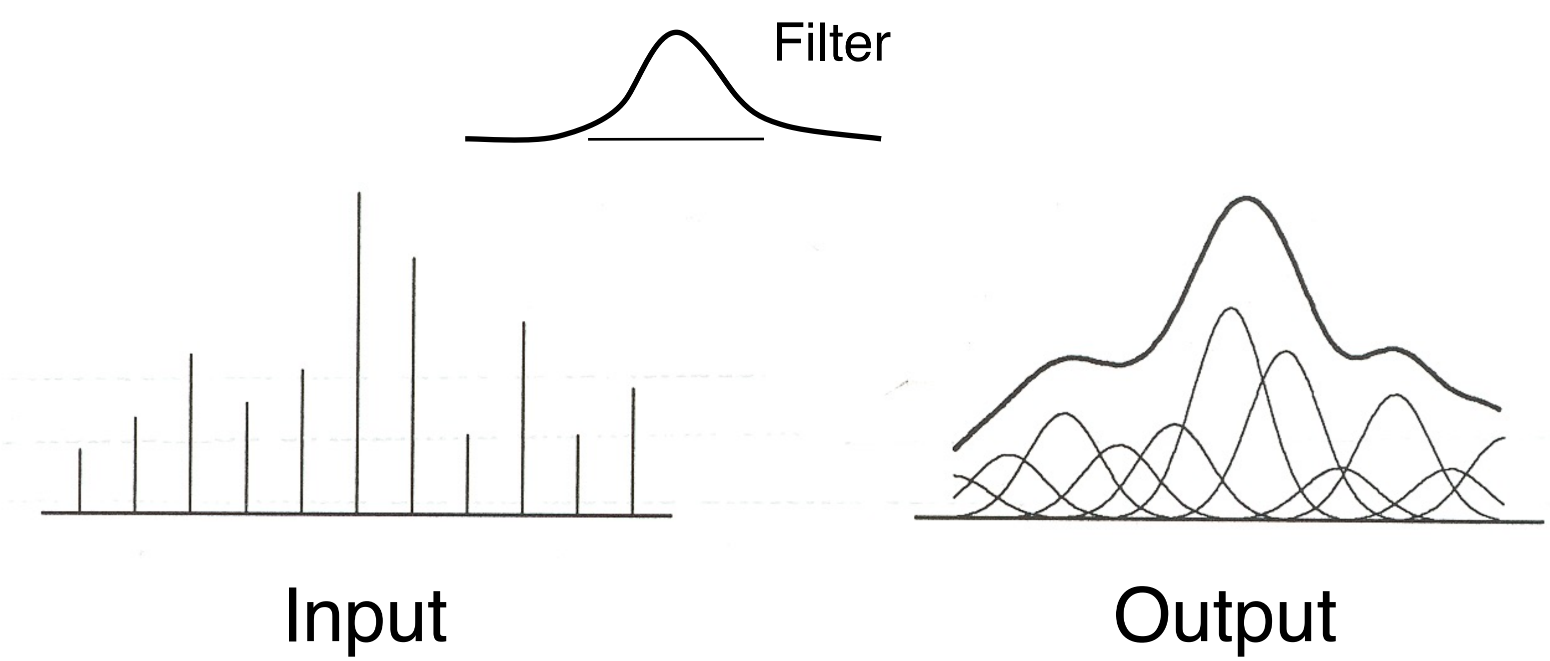
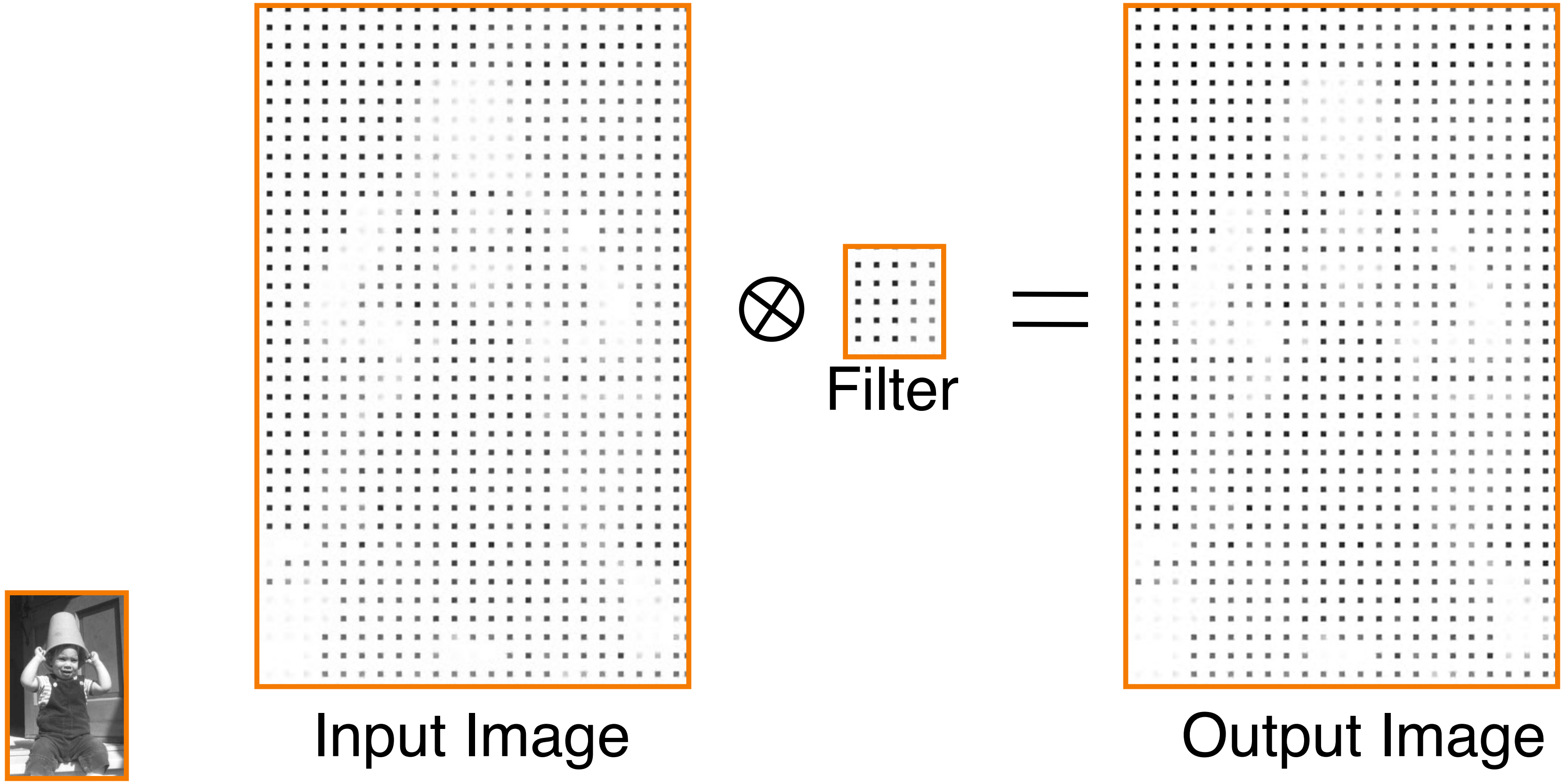


Figure 2.4 Wolberg

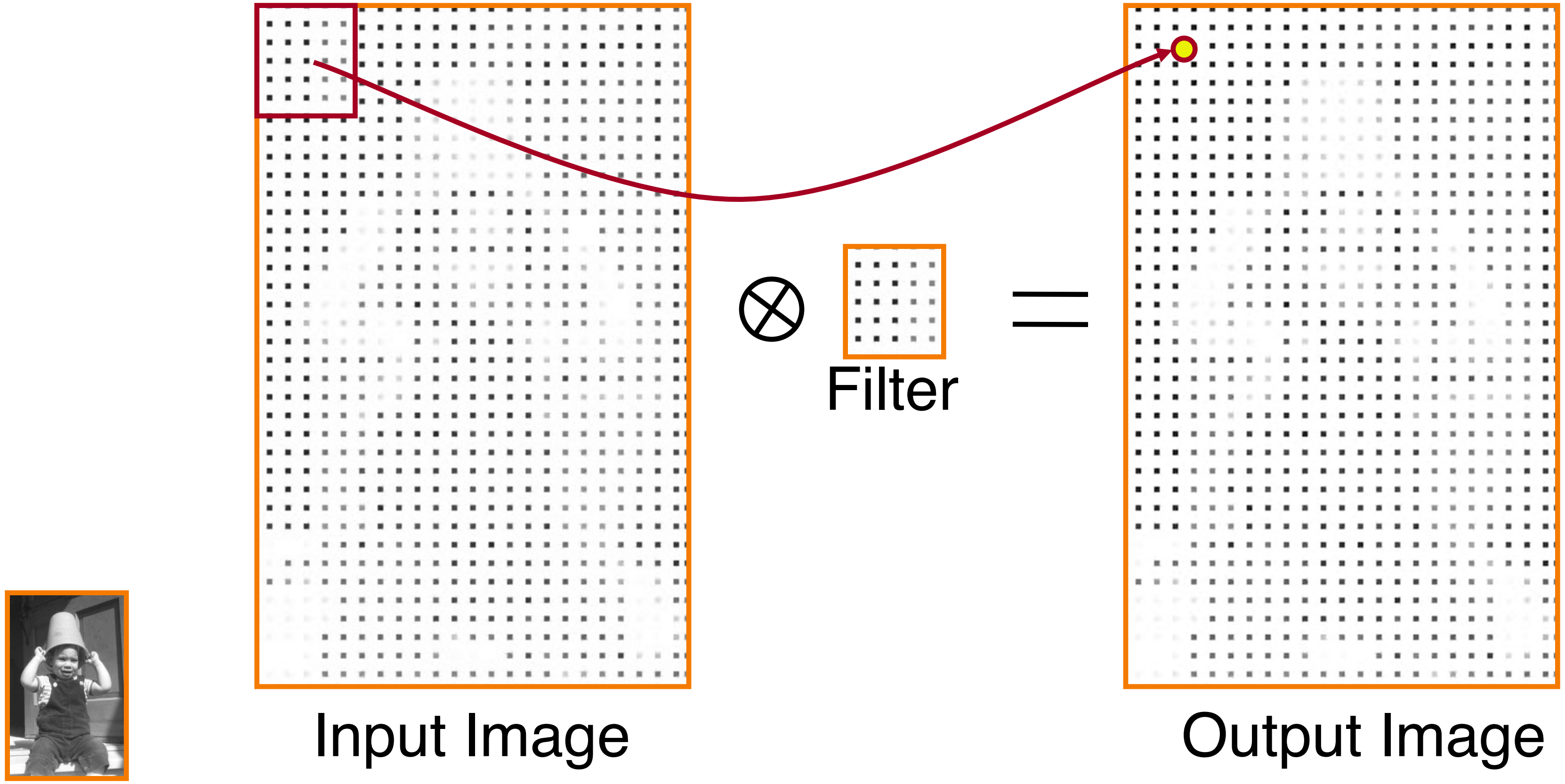
Linear Filtering

- 2D Convolution
  - Each output pixel is a linear combination of input pixels in 2D neighborhood with weights prescribed by a filter



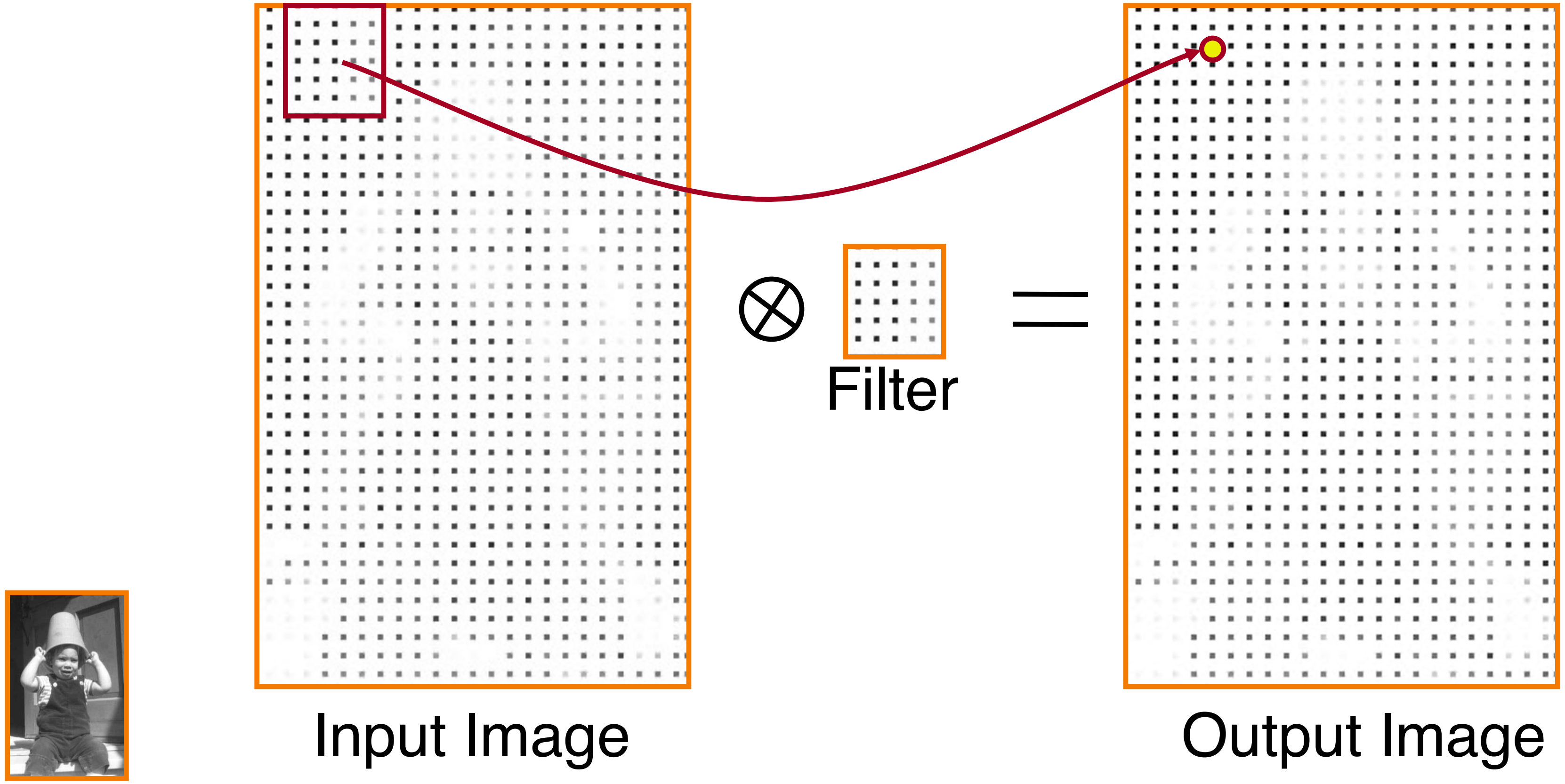
Linear Filtering

- 2D Convolution
  - Each output pixel is a linear combination of input pixels in 2D neighborhood with weights prescribed by a filter



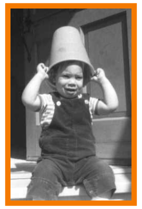
Linear Filtering

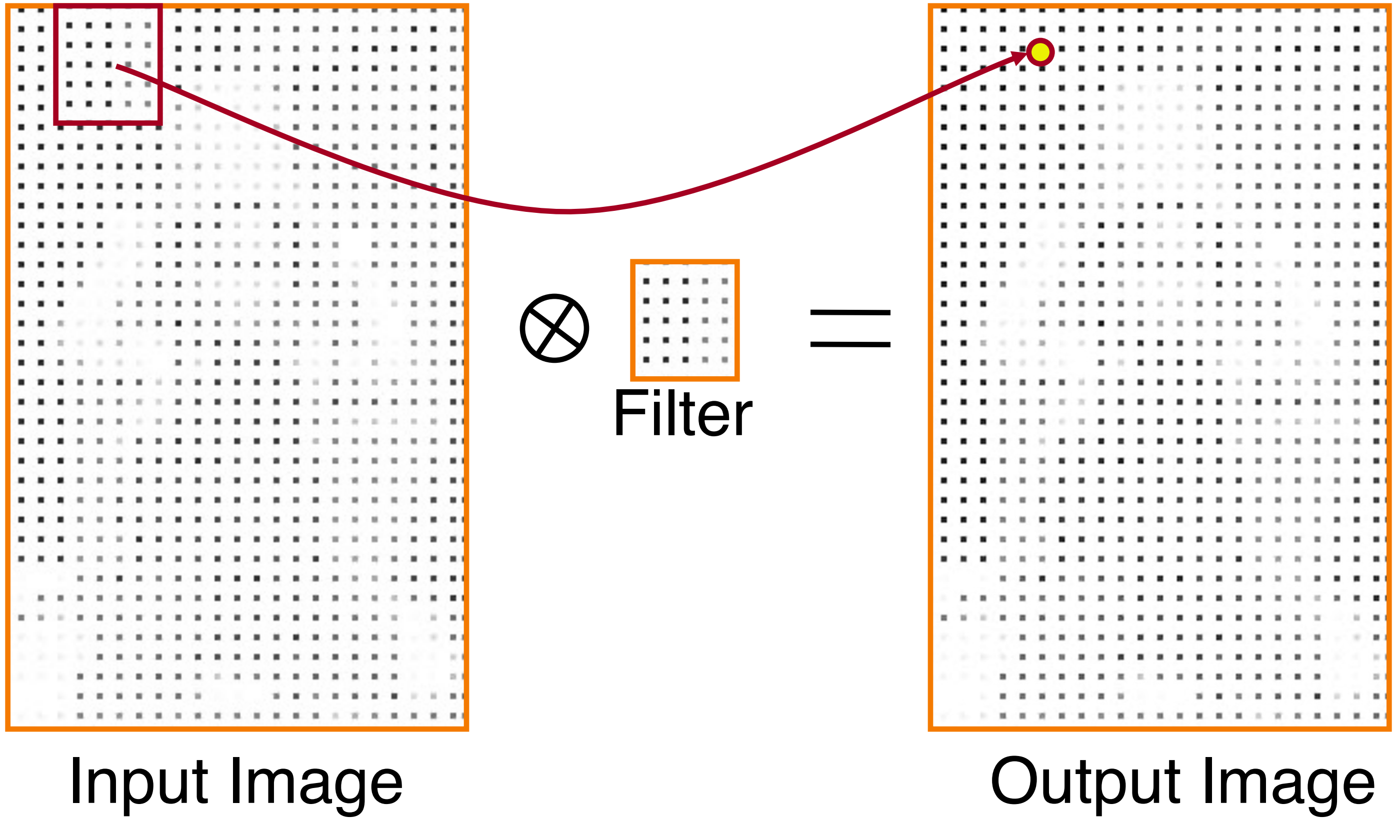
- 2D Convolution
  - Each output pixel is a linear combination of input pixels in 2D neighborhood with weights prescribed by a filter



Linear Filtering

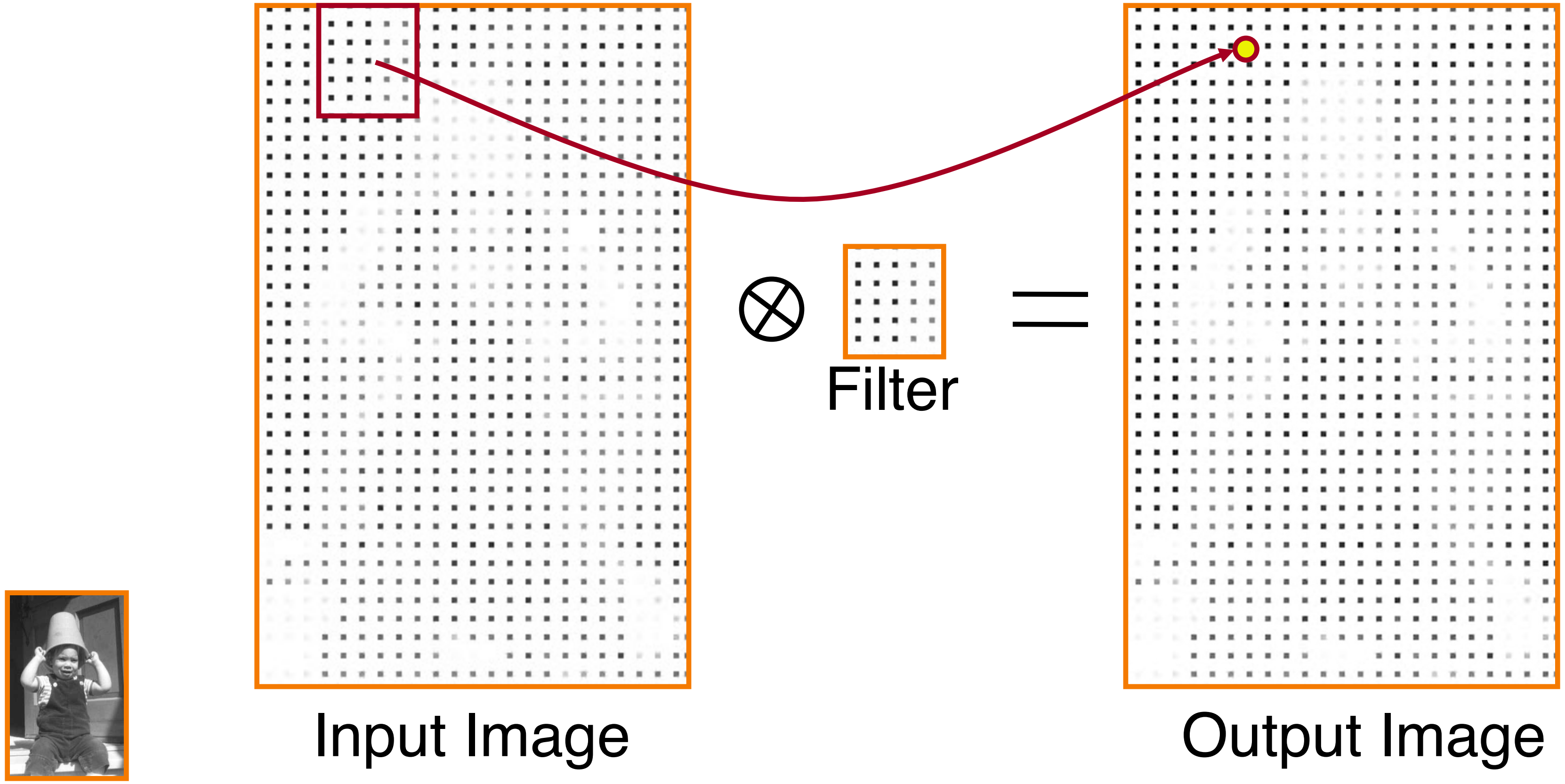
- 2D Convolution
  - Each output pixel is a linear combination of input pixels in 2D neighborhood with weights prescribed by a filter



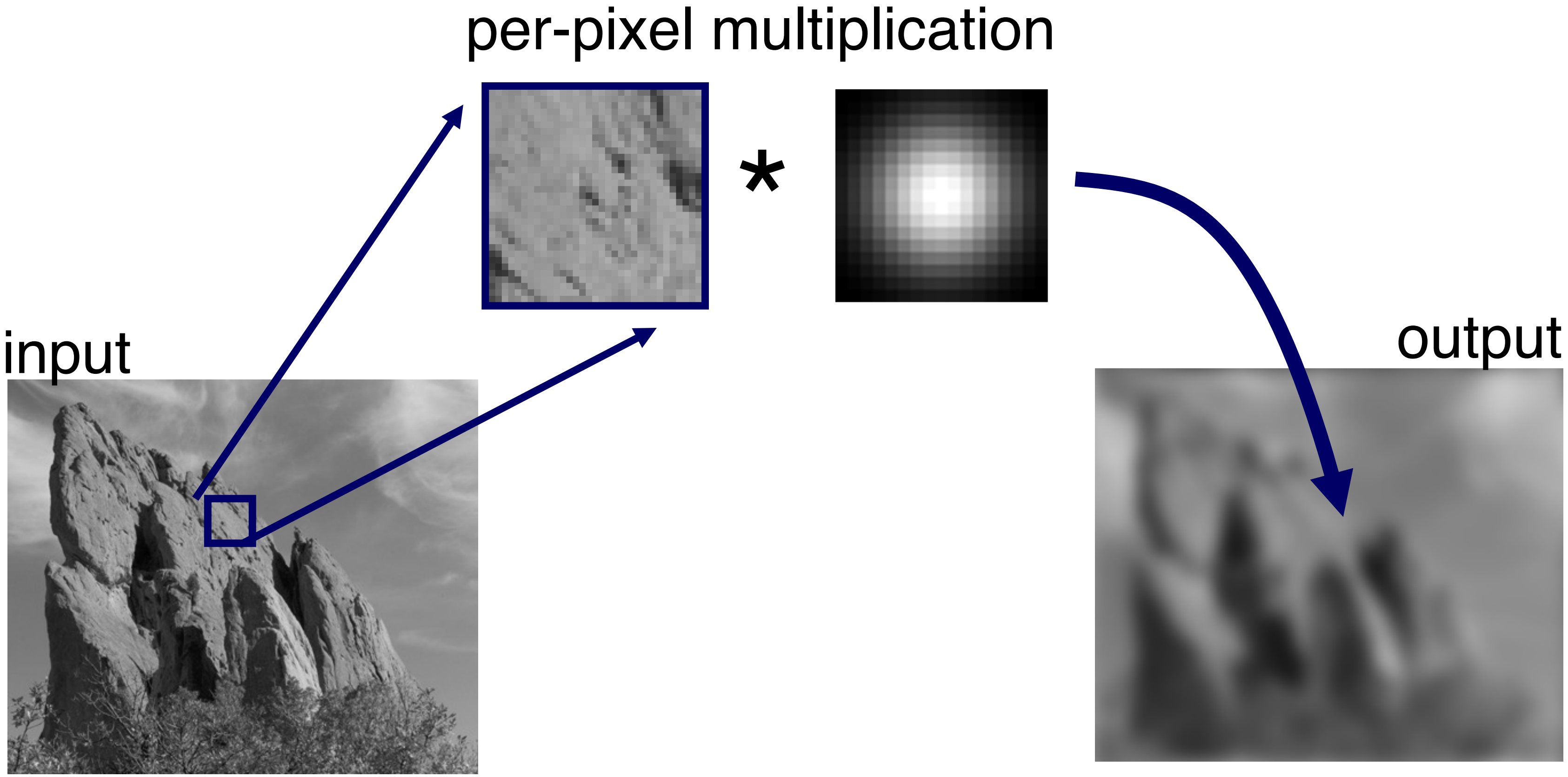


Linear Filtering

- 2D Convolution
  - Each output pixel is a linear combination of input pixels in 2D neighborhood with weights prescribed by a filter



Gaussian Blur

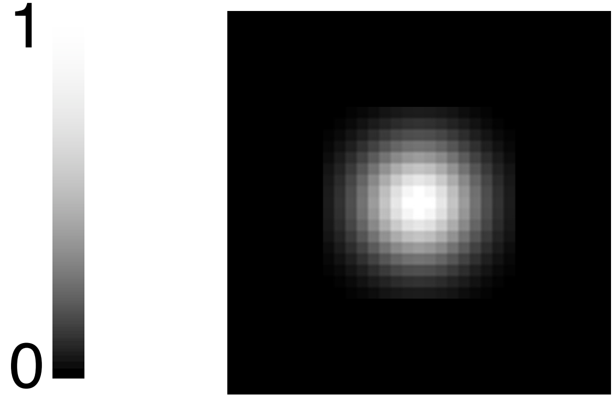


Gaussian Blur

- Output value is weighted sum of values in neighborhood of input image

$$Blur(I_{\mathbf{p}}, \sigma) = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G(\|\mathbf{p} - \mathbf{q}\|, \sigma)\, I_{\mathbf{q}}$$

normalized
Gaussian function




input


Gaussian blur

Linear Filtering

- Many interesting linear filters
  - Blur
  - Edge detect
  - Sharpen
  - Emboss
  - etc.

Filter = ?

Edge Detection

- Convolve with a 2D Laplacian filter that finds differences between neighbor pixels

Original

Detect edges

$$\text{Filter} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & +8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Sharpen

- Sum detected edges with original image

Original

Sharpened

$$\text{Filter} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & +9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Emboss

- Convolve with a filter that highlights gradients in particular directions

Original

Embossed

$$\text{Filter} = \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Side Note: Separable Filters

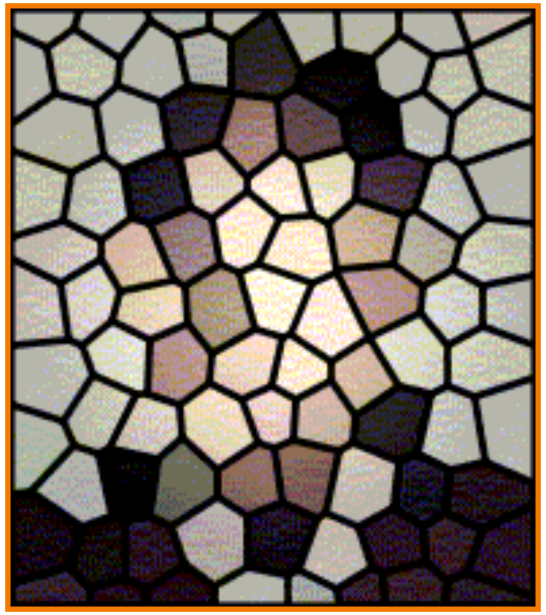
- Some filters are separable (e.g., Gaussian)
  - First, apply 1-D convolution across every row
  - Then, apply 1-D convolution across every column
  - HUGE impact on performance (when kernel is big)

Non-Linear Filtering

- Each output pixel is a non-linear function of input pixels in neighborhood (filter depends on input)

Original

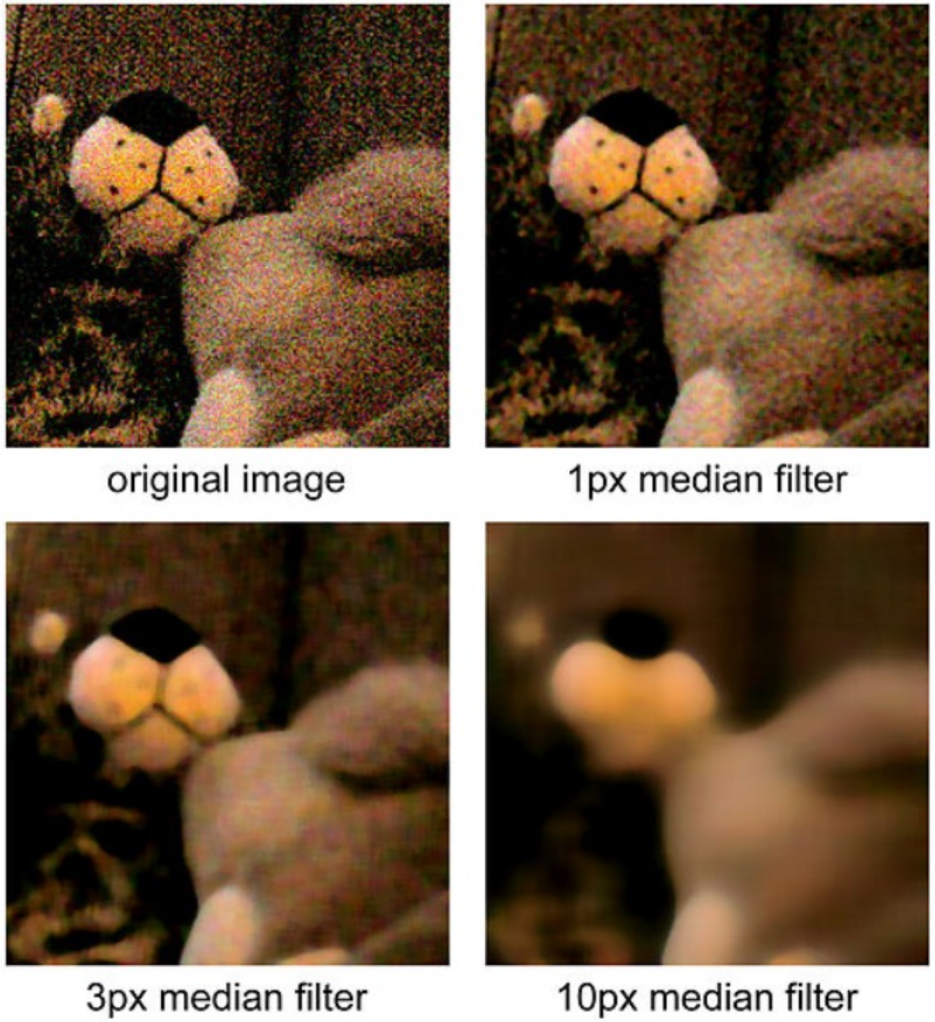
Paint



Stained Glass

Median or “Despeckling” Filter

- Each output pixel is median of input pixels in neighborhood



original image
1px median filter
3px median filter
10px median filter

Bilateral Filter

- Gaussian blur uses same filter for all pixels
  - Blurs *across* edges as much as in other areas

Original

Gaussian Blur

Bilateral Filter

- Gaussian blur uses same filter for all pixels
  - Prefer a filter that preserves edges (adapts to content)

Original

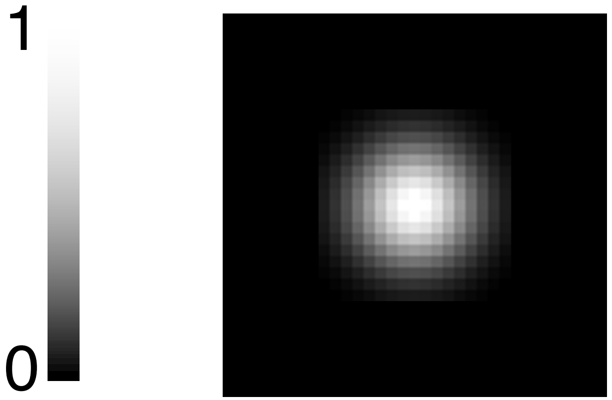
Bilateral Filter

Recall: Gaussian Blur

- Output value is weighted sum of values in neighborhood of input image

$$Blur(I_{\mathbf{p}},\sigma) = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in S} G(\|\mathbf{p}-\mathbf{q}\|,\sigma)\, I_{\mathbf{q}}$$

normalized
Gaussian function



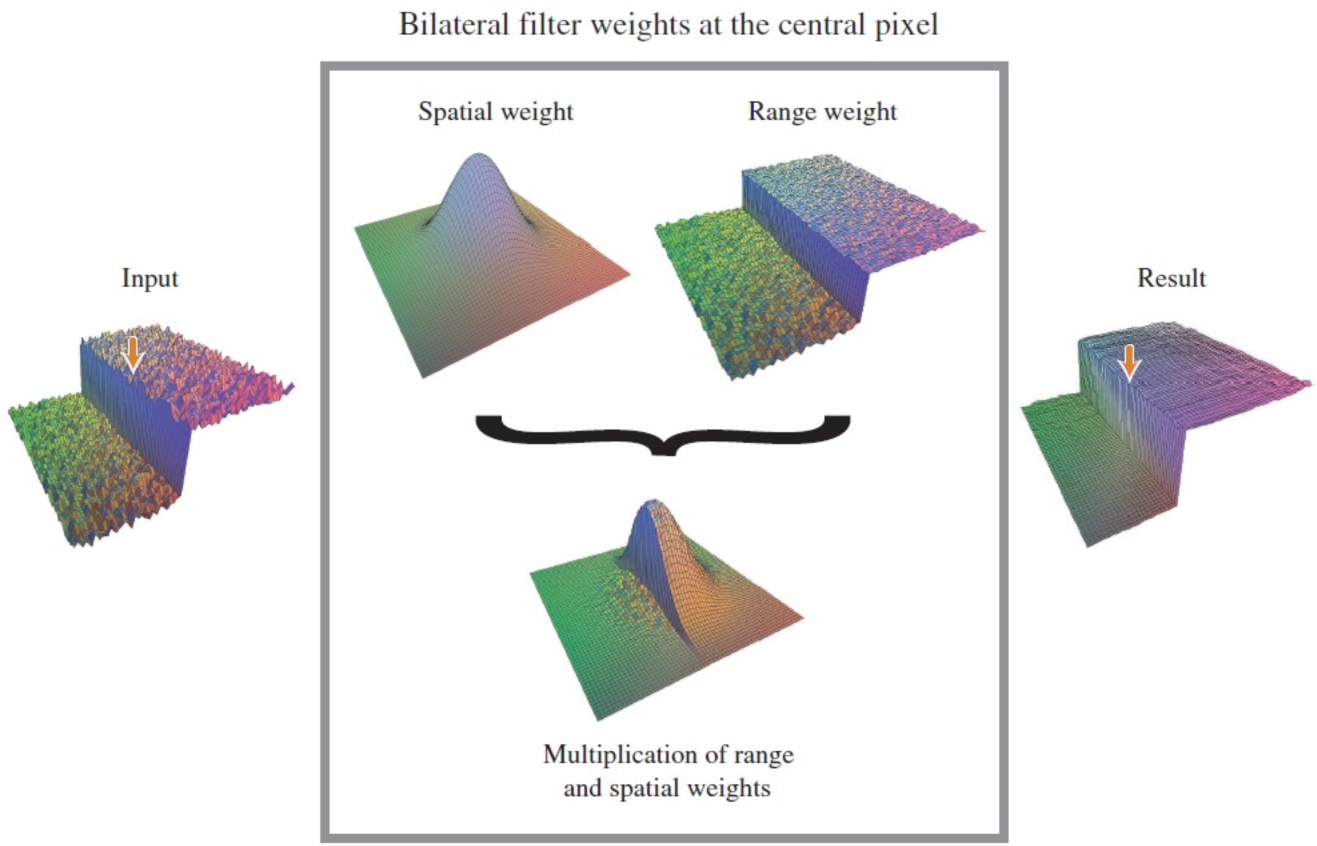
Bilateral Filter

- Combine Gaussian filtering in both spatial domain and color domain

$$Bilateral[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} \underbrace{G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)}_{\text{Spatial Proximity Weight}} \, \underbrace{G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|)}_{\text{Color Proximity Weight}} \, I_{\mathbf{q}}$$

Bilateral Filtering

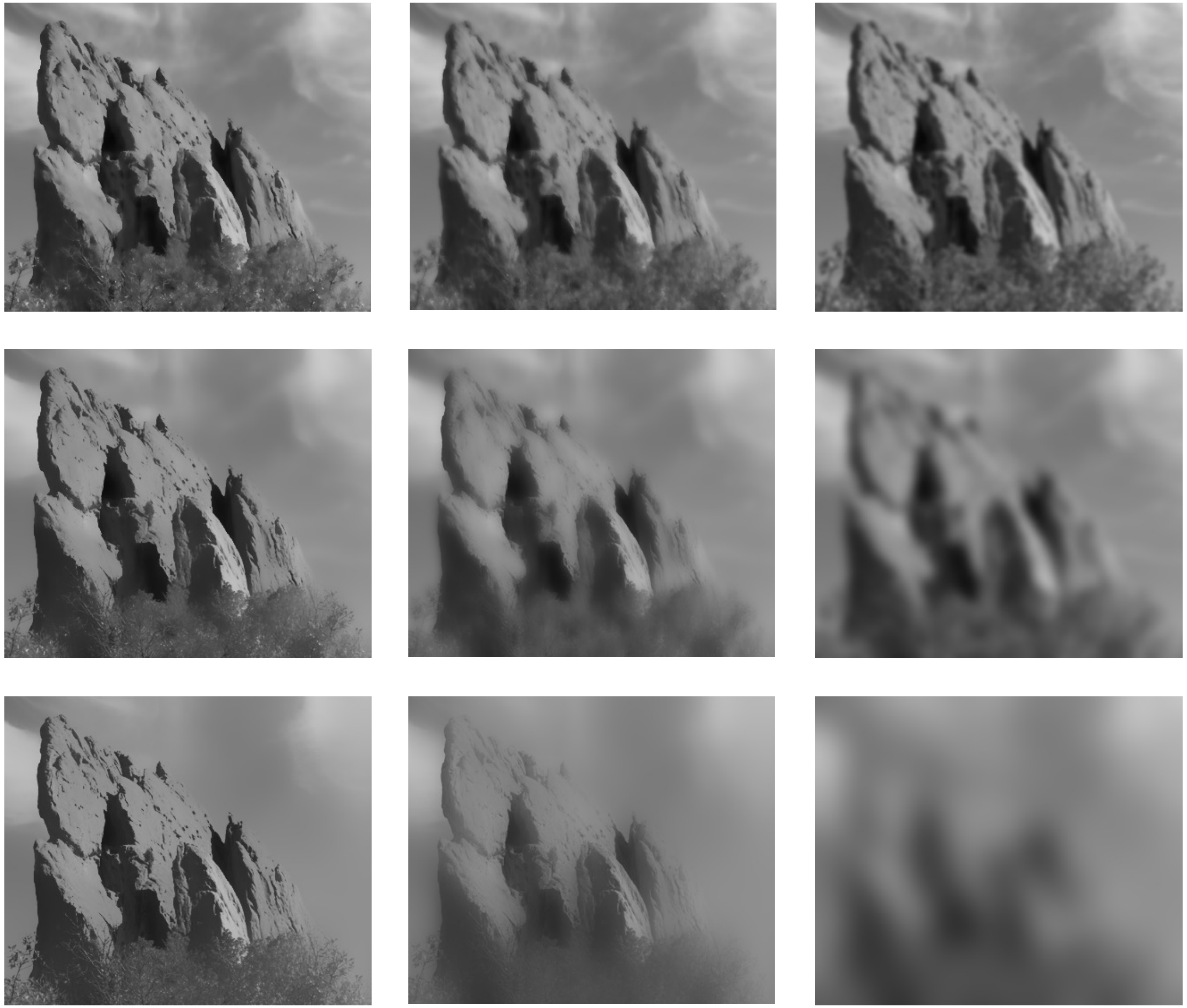
- Combine Gaussian filtering in both spatial domain and color domain



Bilateral Filtering

input

	$\sigma_r = 0.1$	$\sigma_r = 0.25$	$\sigma_r = \infty$ (Gaussian blur)
$\sigma_s = 2$			
$\sigma_s = 6$			
$\sigma_s = 18$			

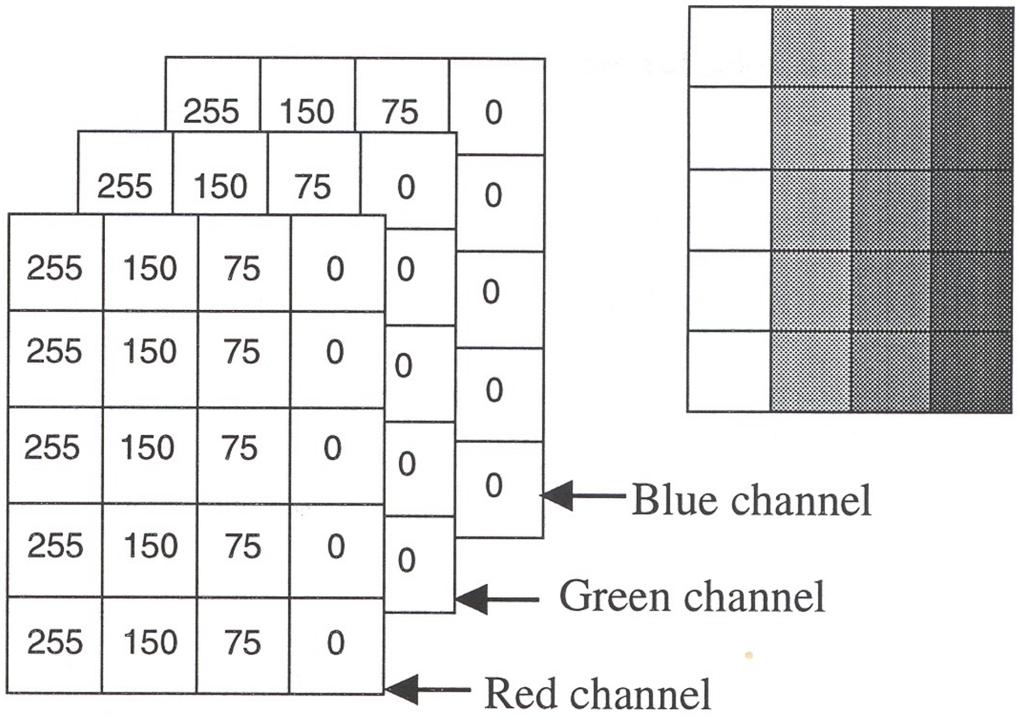


Digital Image Processing

- Changing pixel values
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Histogram equalization
- Filtering over neighborhoods
  - Blur & sharpen
  - Detect edges
  - Median
  - Bilateral filter
- Moving image locations
  - Scale
  - Rotate
  - Warp
- Combining images
  - Composite
  - Morph
- Quantization
- Spatial / intensity tradeoff
  - Dithering

Quantization

- Reduced intensity resolution
  - Frame buffers have limited number of bits per pixel
  - Physical devices have limited dynamic range



Effects of Quantization

8 bits / pixel / color

6 bits / pixel / color

Marc Levoy / Hanna-Barbera

Effects of Quantization

5 bits / pixel / color

4 bits / pixel / color

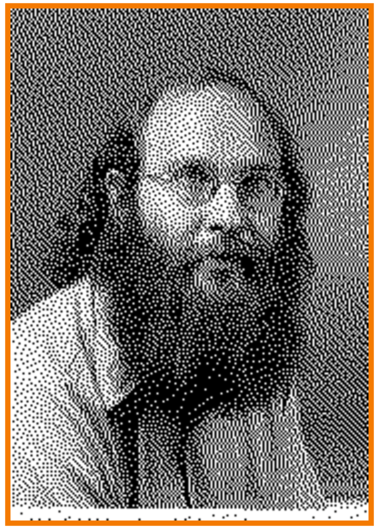
Marc Levoy / Hanna-Barbera

Dithering

- Distribute errors among pixels
  - Exploit spatial integration in our eye
  - Display greater range of perceptible intensities
  - Trade off spatial resolution for intensity resolution

Original
(8 bits)

Uniform
Quantization
(1 bit)

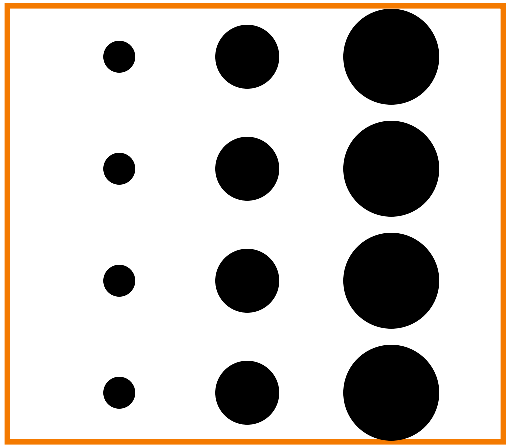


Floyd-Steinberg
Dither
(1 bit)

Classical Halftoning

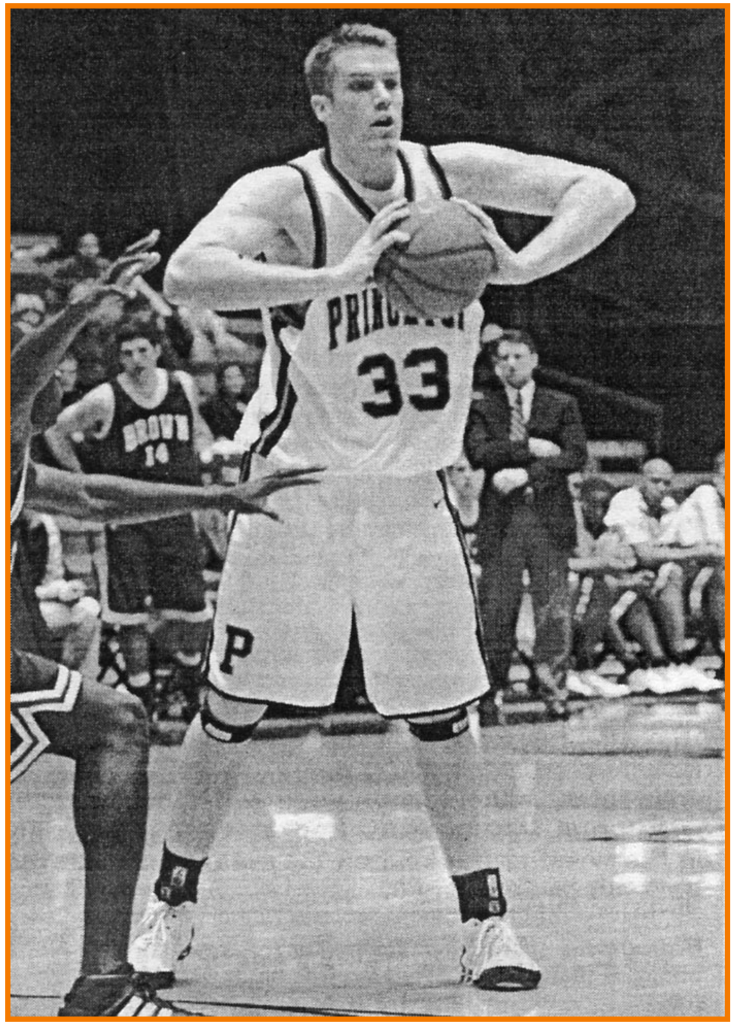
- Use dots of varying size to represent intensities
  - Area of dots proportional to intensity in image

I(x,y)

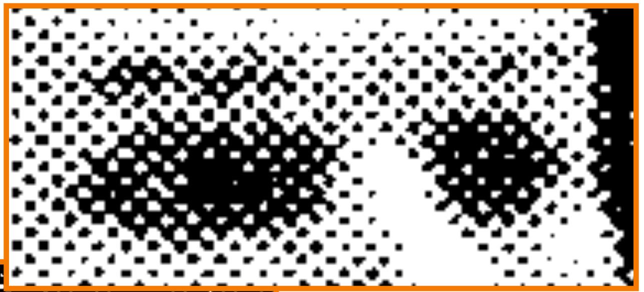


P(x,y)

Classical Halftoning



From Town Topics, Princeton



Digital Halftone Patterns

- Use cluster of pixels to represent intensity

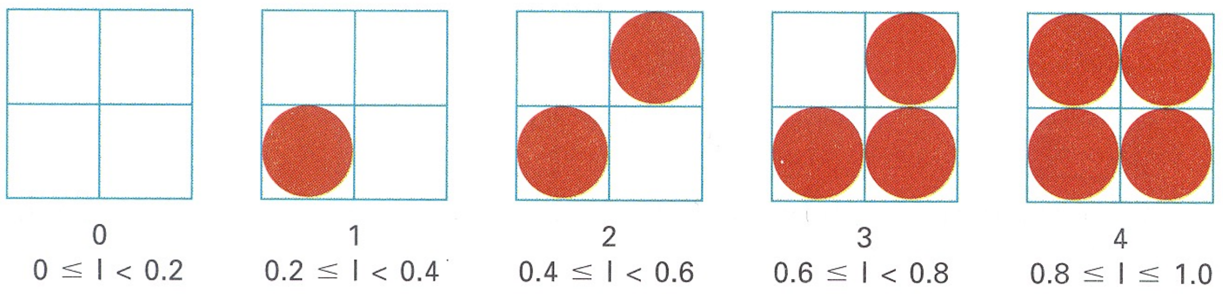
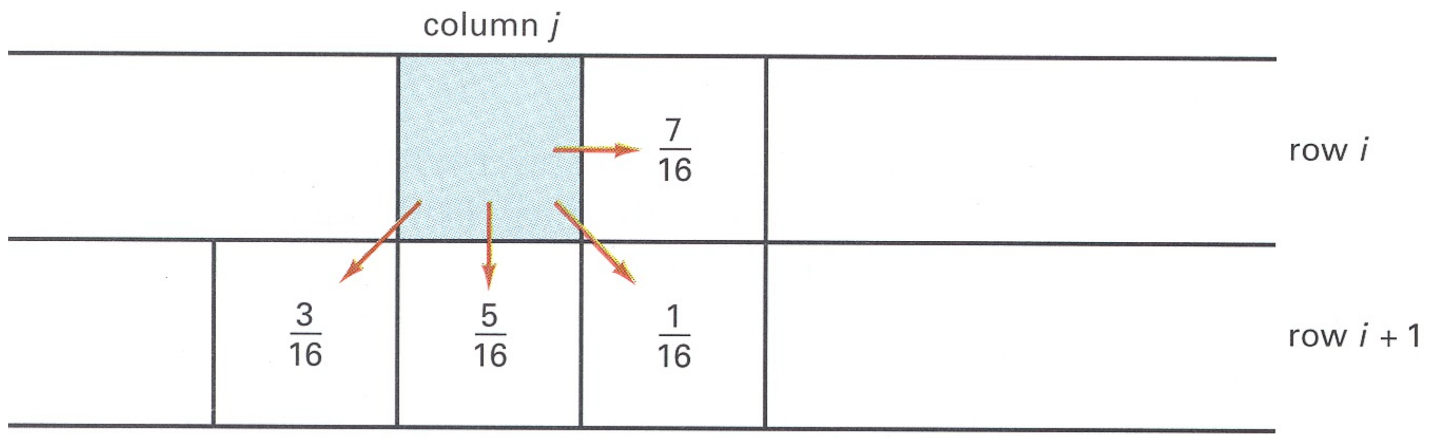


Figure 14.37 from H&B

Error Diffusion Dither

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



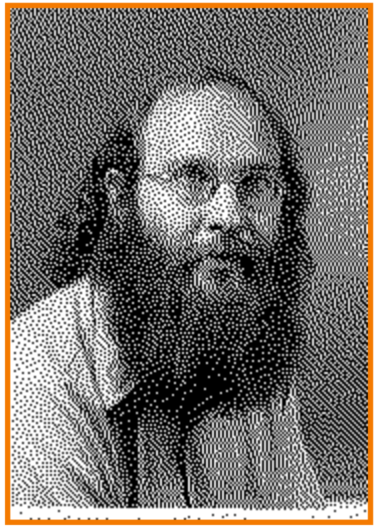
$$3/16 + 5/16 + 1/16 + 7/16 = 1.0$$

Figure 14.42 from H&B

Error Diffusion Dither

Original
(8 bits)

Uniform
Quantization
(1 bit)



Floyd-Steinberg
Dither
(1 bit)

Next Time…

- Changing pixel values
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Histogram equalization
- Filtering over neighborhoods
  - Blur & sharpen
  - Detect edges
  - Median
  - Bilateral filter
- Moving image locations
  - Scale
  - Rotate
  - Warp
- Combining images
  - Composite
  - Morph
- Quantization
- Spatial / intensity tradeoff
  - Dithering