



Image Processing

COS 426, Spring 2015
Adam Finkelstein



Image Processing Operations

- Luminance
 - Brightness
 - Contrast
 - Gamma
 - Histogram equalization
- Color
 - Grayscale
 - Saturation
 - White balance
- Linear filtering
 - Blur & sharpen
 - Edge detect
 - Convolution
- Non-linear filtering
 - Median
 - Bilateral filter
- Dithering
 - Quantization
 - Ordered dither
 - Floyd-Steinberg



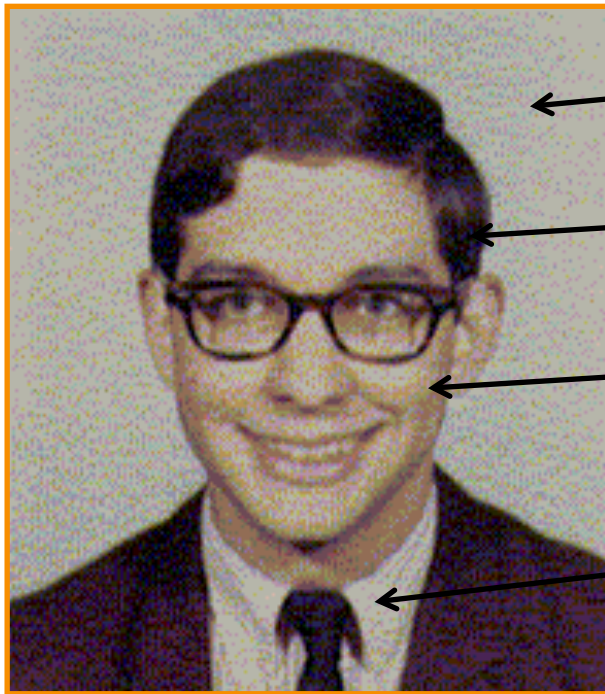
Image Processing Operations

- **Luminance**
 - Brightness
 - Contrast
 - Gamma
 - Histogram equalization
- **Color**
 - Grayscale
 - Saturation
 - White balance
- **Linear filtering**
 - Blur & sharpen
 - Edge detect
 - Convolution
- **Non-linear filtering**
 - Median
 - Bilateral filter
- **Dithering**
 - Quantization
 - Ordered dither
 - Floyd-Steinberg

What is Luminance?

Measures perceived “gray-level” of pixel

$$L = 0.30 * \text{red} + 0.59 * \text{green} + 0.11 * \text{blue}$$



← 0.5

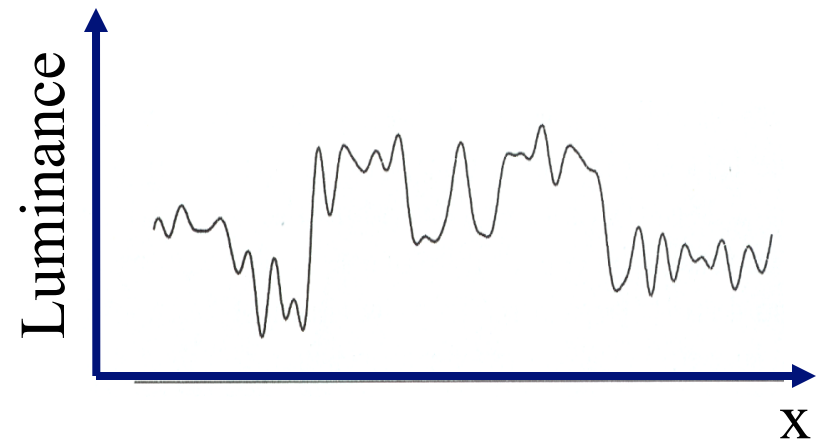
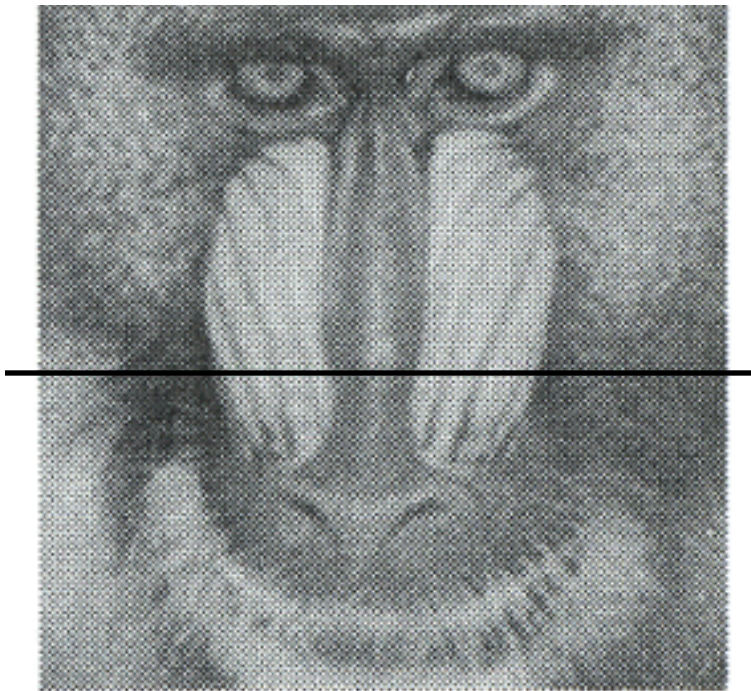
← 0.0

← 0.7

← 0.9

Luminance

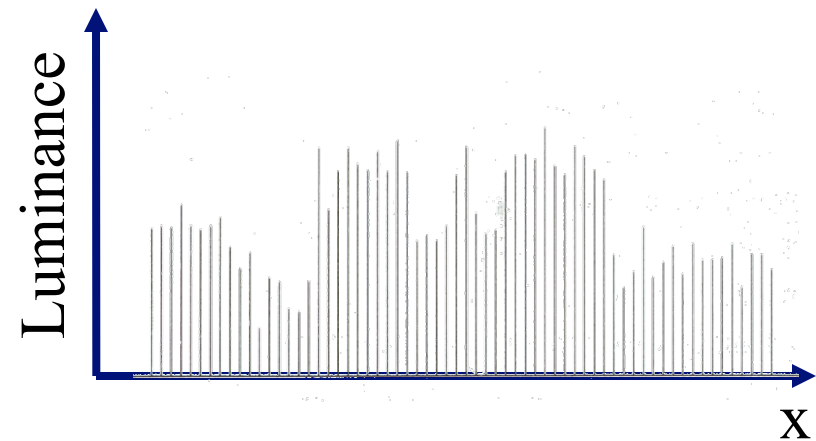
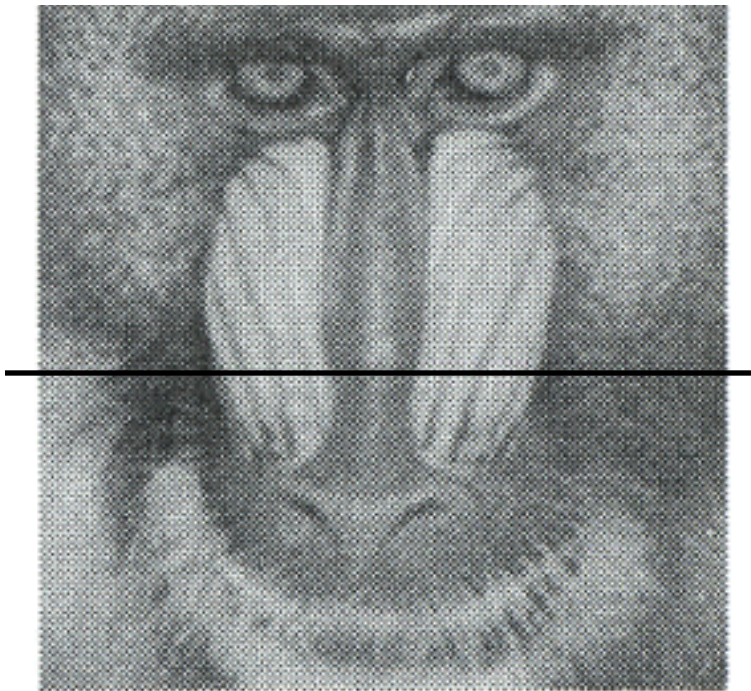
Measures perceived “gray-level” of pixel



Values of luminance for positions
on one horizontal scanline

Luminance

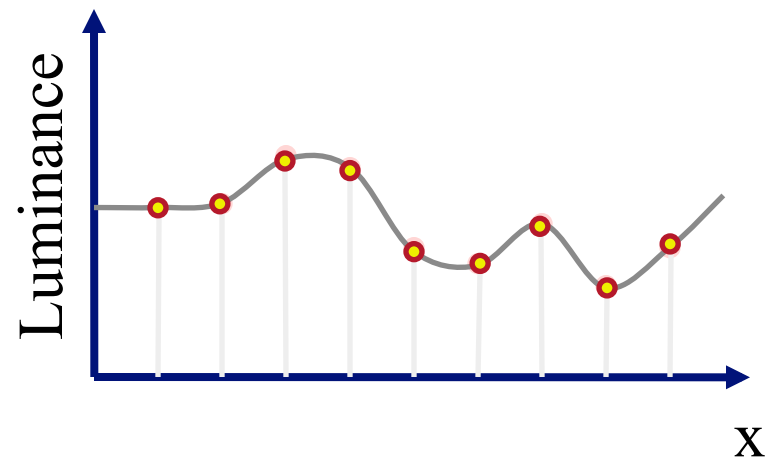
Measures perceived “gray-level” of pixel



Samples of luminance for pixels
on one horizontal scanline

Adjusting Brightness

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





Adjusting Brightness

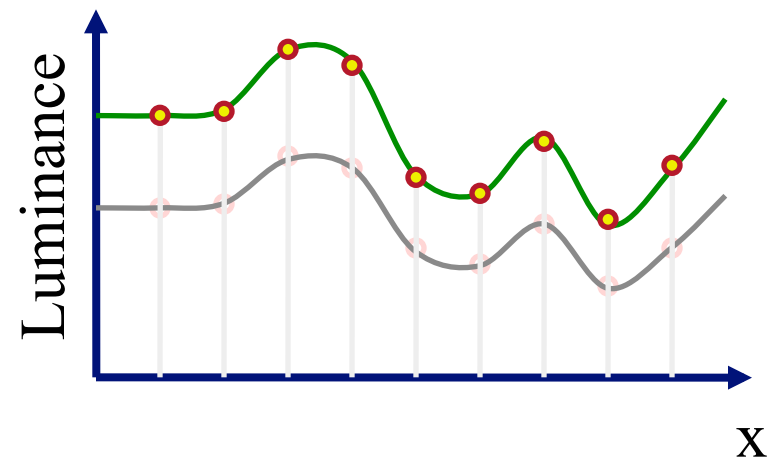
- Method 1: Convert to HSL, scale L, convert back (more on this shortly...)
- Method 2: Scale R, G, and B directly
 - Multiply each of red, green, and blue by a factor
 - Must clamp to $[0..1]$... always
($[0..1]$ in floating point but often $[0,255]$ for fixed point)



Original



Brighter



Adjusting Contrast

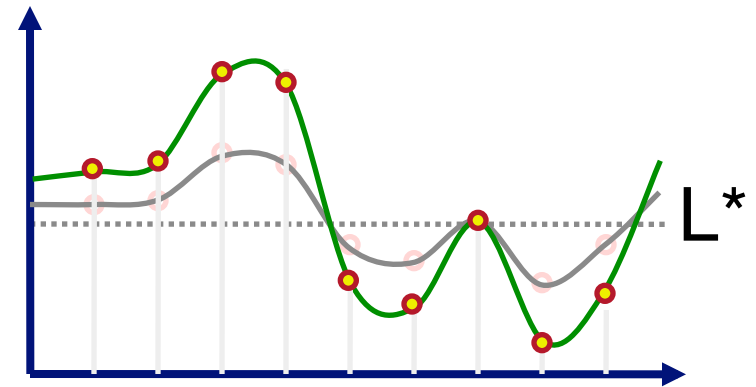
- Compute mean luminance L^* over whole image
Scale deviation from L^* for each pixel



Original



More Contrast

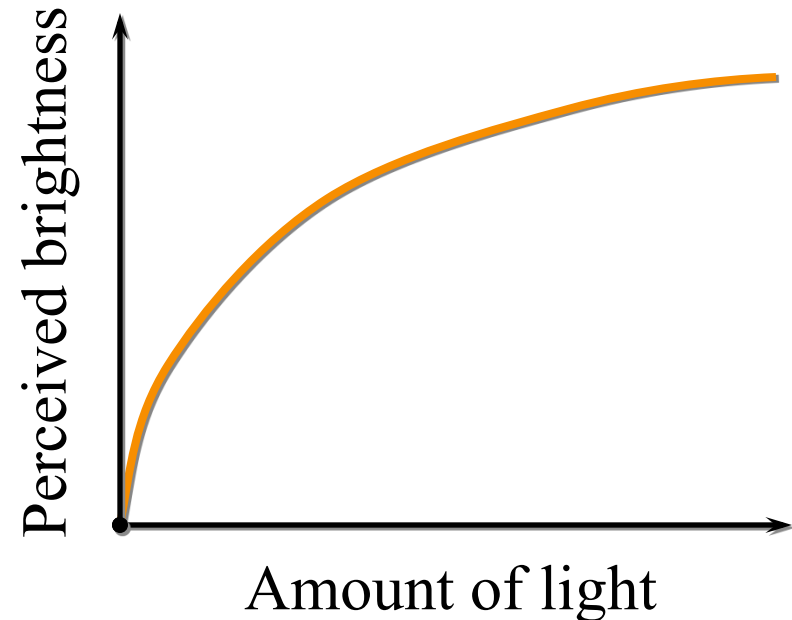


Adjusting Gamma



Apply non-linear function to account for difference between brightness and perceived brightness of display

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



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

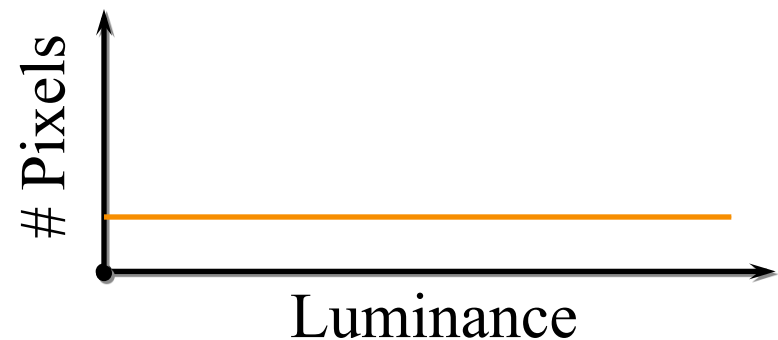
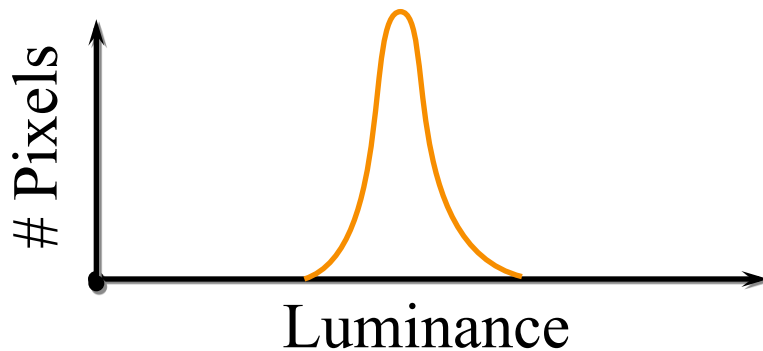




Image Processing Operations

- Luminance
 - Brightness
 - Contrast
 - Gamma
 - Histogram equalization
- Color
 - Grayscale
 - Saturation
 - White balance
- Linear filtering
 - Blur & sharpen
 - Edge detect
 - Convolution
- Non-linear filtering
 - Median
 - Bilateral filter
- Dithering
 - Quantization
 - Ordered dither
 - Floyd-Steinberg

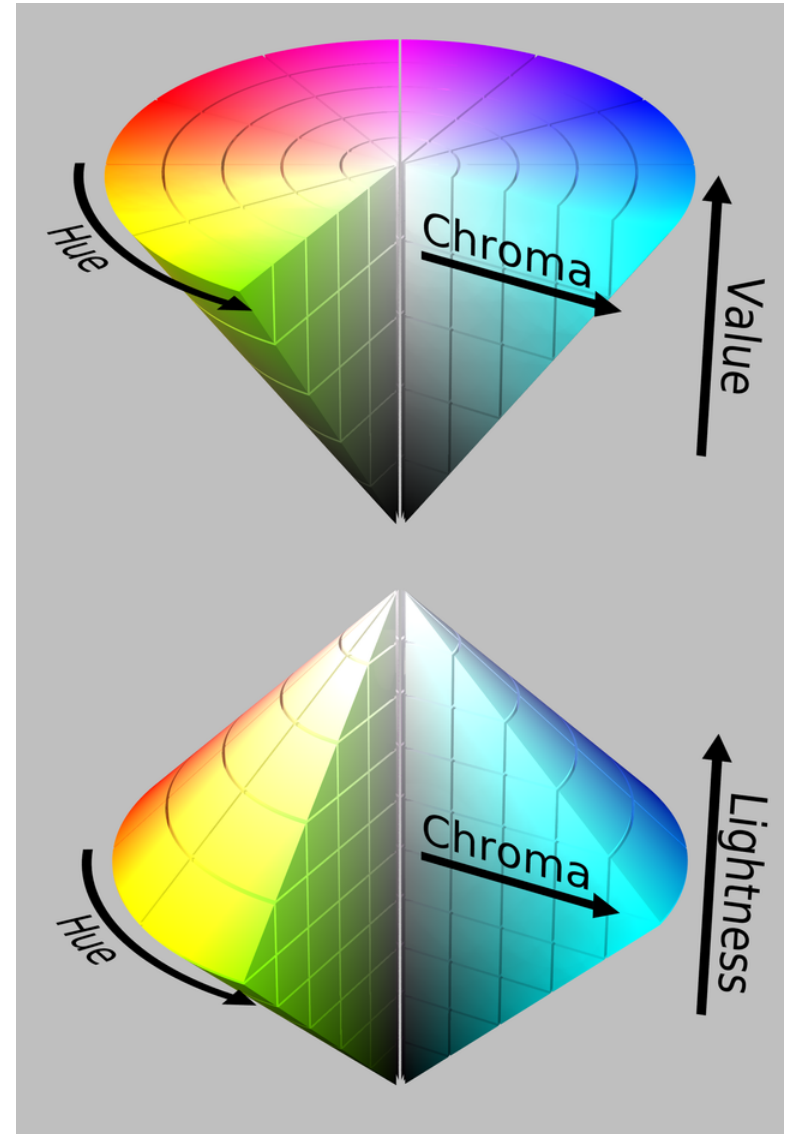
Color processing

- Color models (last lec.)

- RGB
- CMY
- HSV
- XYZ
- La^*b^*
- Etc.

→ HSV

→ HSL



[http://commons.wikimedia.org/wiki/
File:HSV_color_solid_cone_chroma_gray.png](http://commons.wikimedia.org/wiki/File:HSV_color_solid_cone_chroma_gray.png)



Grayscale

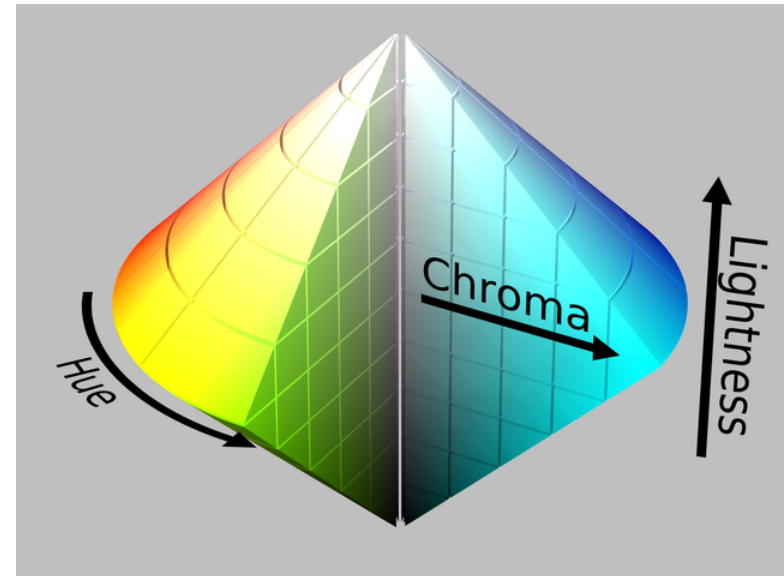
Convert from color to gray-levels



Original



Grayscale
(“black&white” photo)





Grayscale

Convert from color to gray-levels

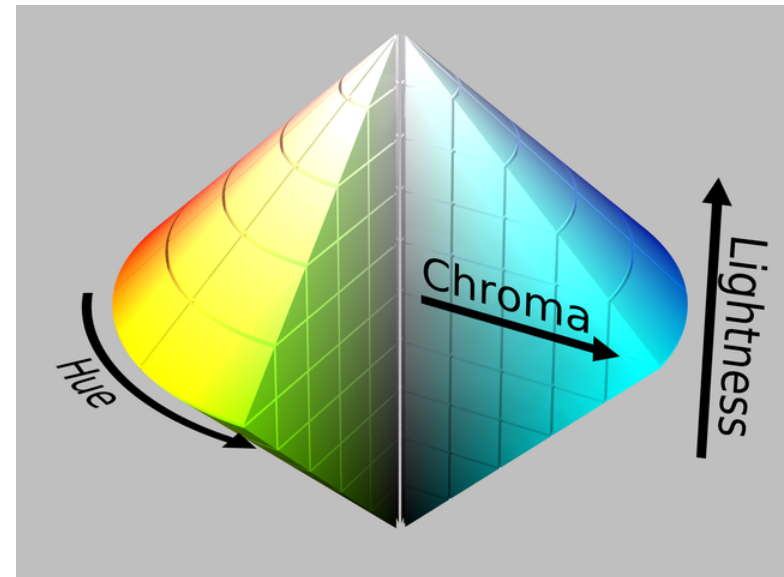


Original



Grayscale

(“black&white” photo)



Method 1: Convert to HSL, set $S=0$, convert back to RGB

Method 2: Set RGB of every pixel to (L,L,L)

Adjusting Saturation



Increase/decrease color saturation of every pixel



-1.0



0.0



0.5



1.0



2.5

Adjusting Saturation



Increase/decrease color saturation of every pixel



-1.0

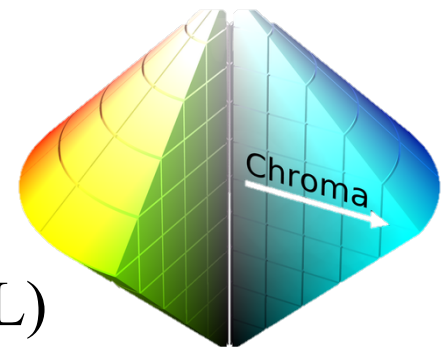
0.0

0.5

1.0

2.5

Method 1: Convert to HSL, scale S, convert back
Method 2: Set each pixel to factor * (R-L, G-L, B-L)



White Balance



Adjust colors so that a given RGB value is mapped to a neutral color

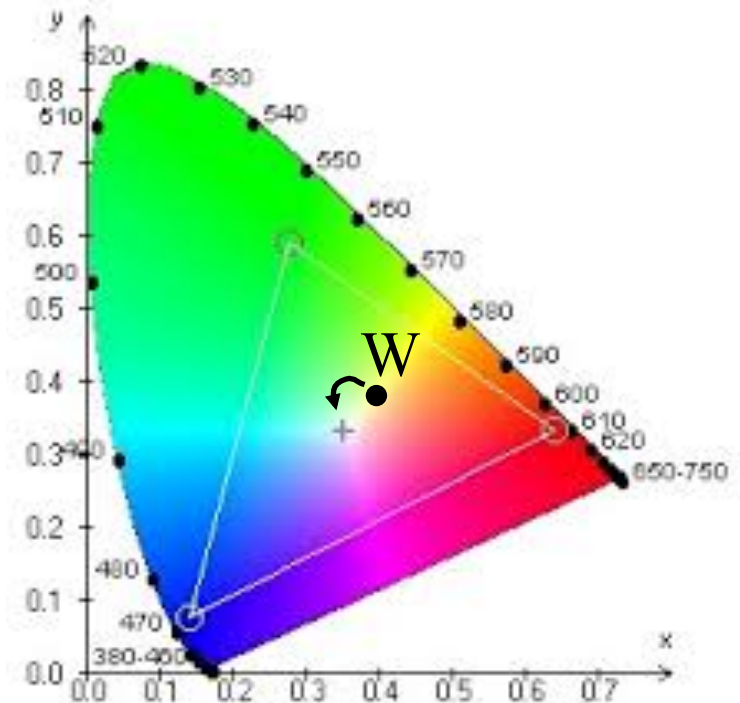


White Balance



Conceptually:

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



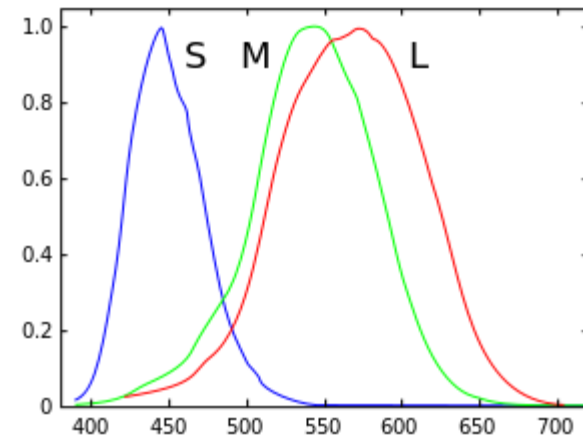
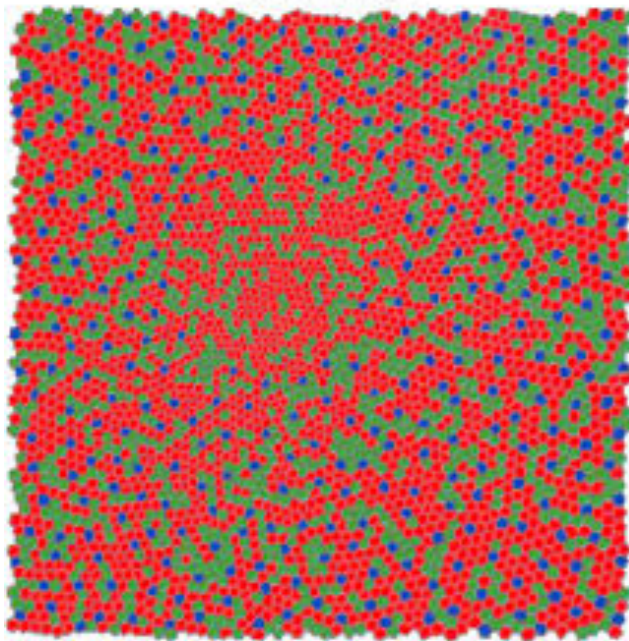
White Balance



Von Kries method: adjust colors in LMS color space

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

Cone mosaic





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

Image Processing Operations



- Luminance
 - Brightness
 - Contrast.
 - Gamma
 - Histogram equalization
- Color
 - Grayscale
 - Saturation
 - White balance
- Linear filtering
 - Blur & sharpen
 - Edge detect
 - Convolution
- Non-linear filtering
 - Median
 - Bilateral filter
- Dithering
 - Quantization
 - Ordered dither
 - Floyd-Steinberg

Blur



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

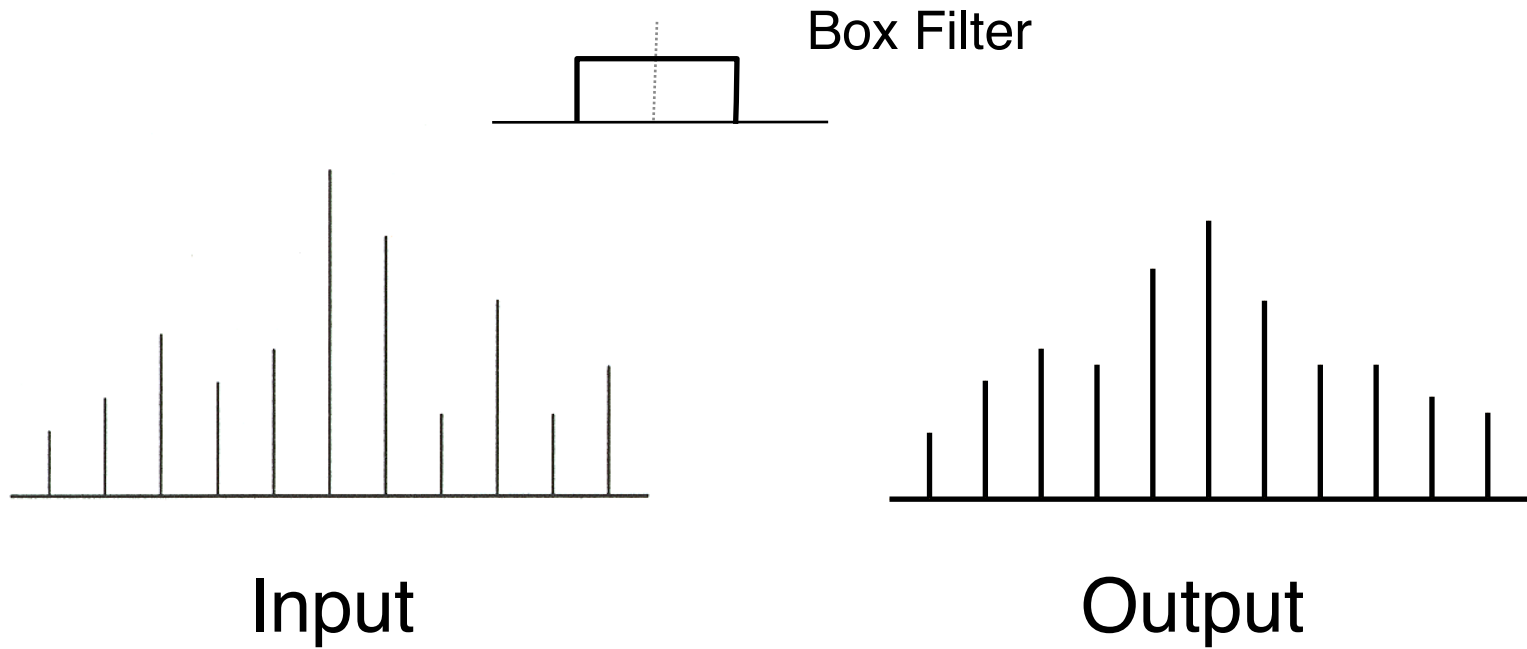




Basic Operation: Convolution

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

- Pattern of weights is the “filter” or “kernel”

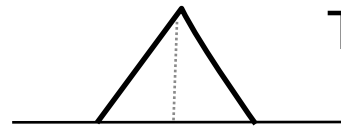




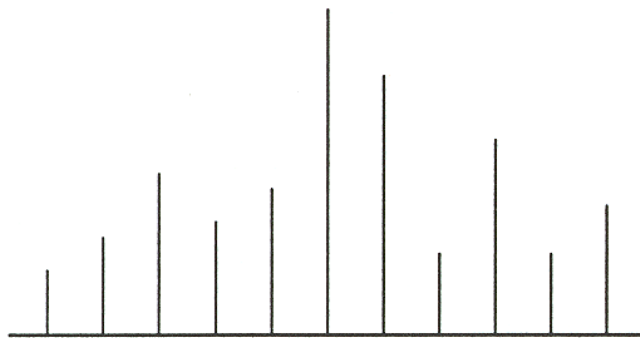
Basic Operation: Convolution

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

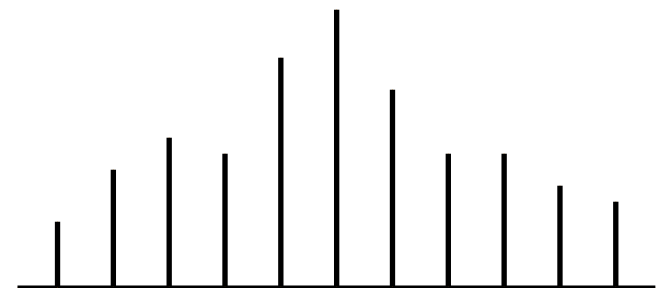
- Pattern of weights is the “filter” or “kernel”



Triangle Filter



Input



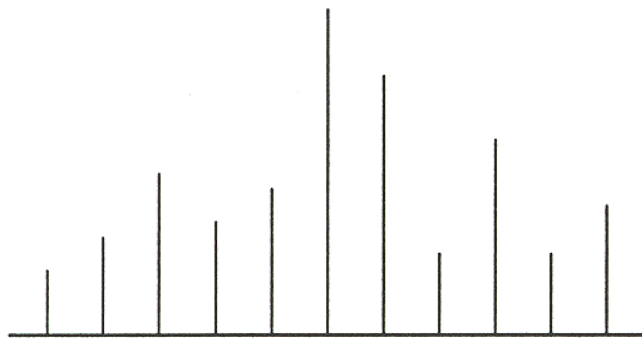
Output



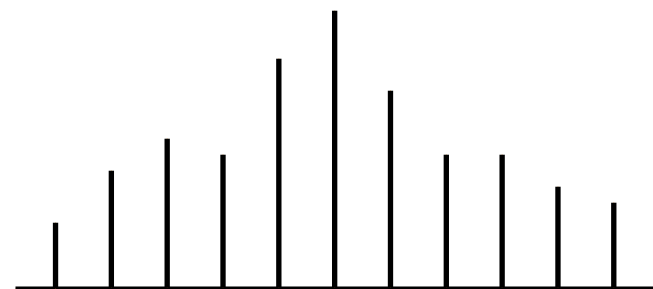
Basic Operation: Convolution

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

- Pattern of weights is the “filter” or “kernel”



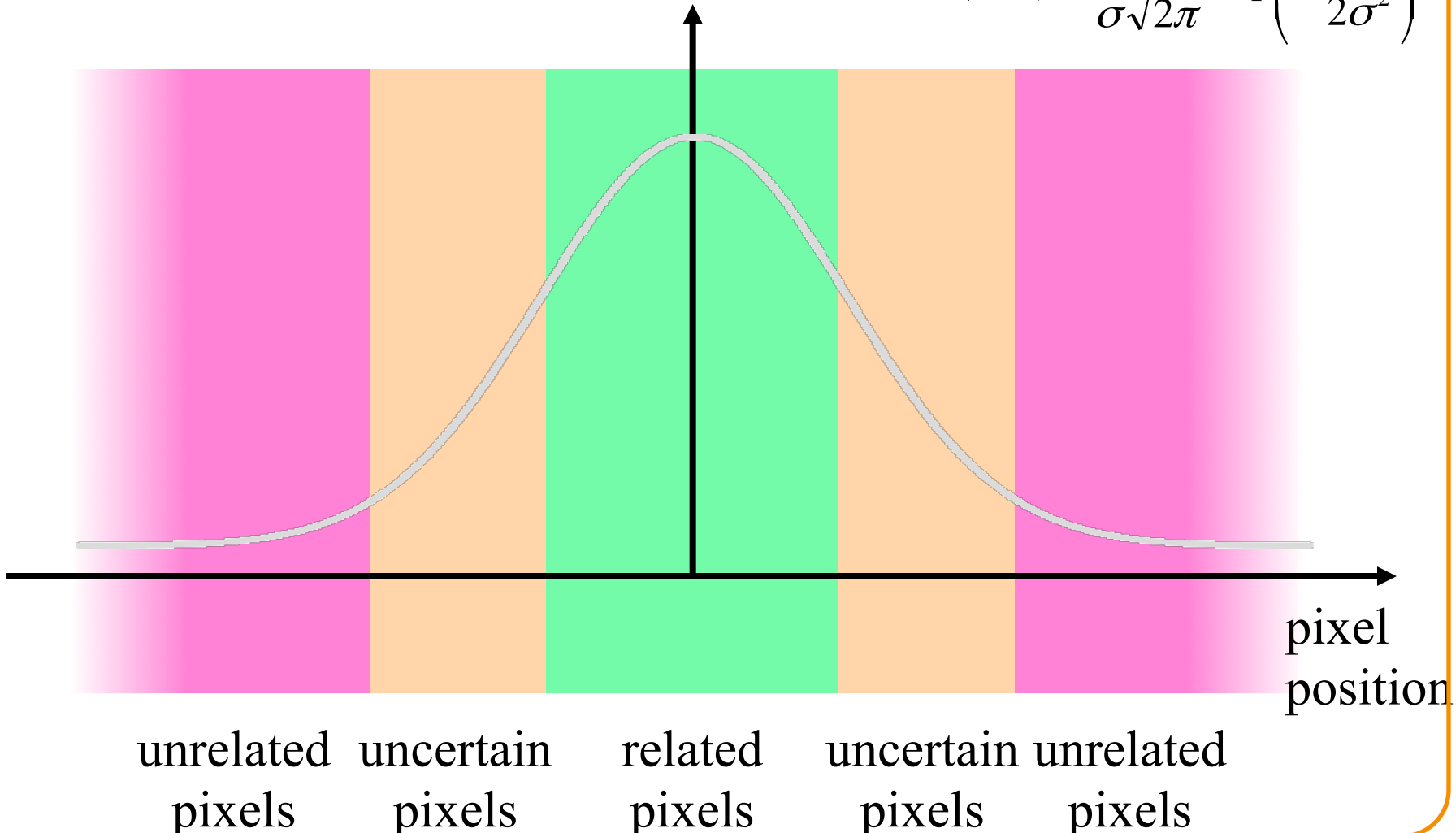
Input



Output

Convolution with a Gaussian Filter

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

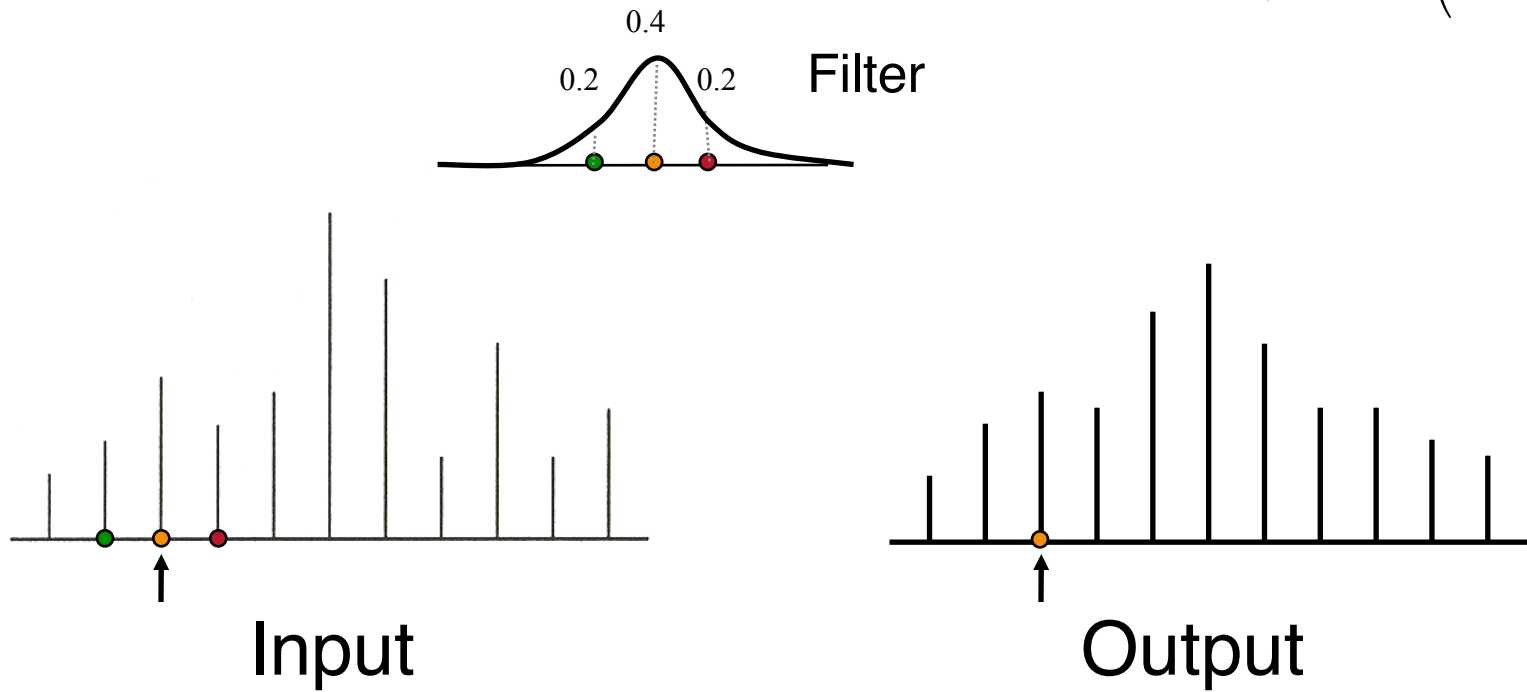


Convolution with a Gaussian Filter



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

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

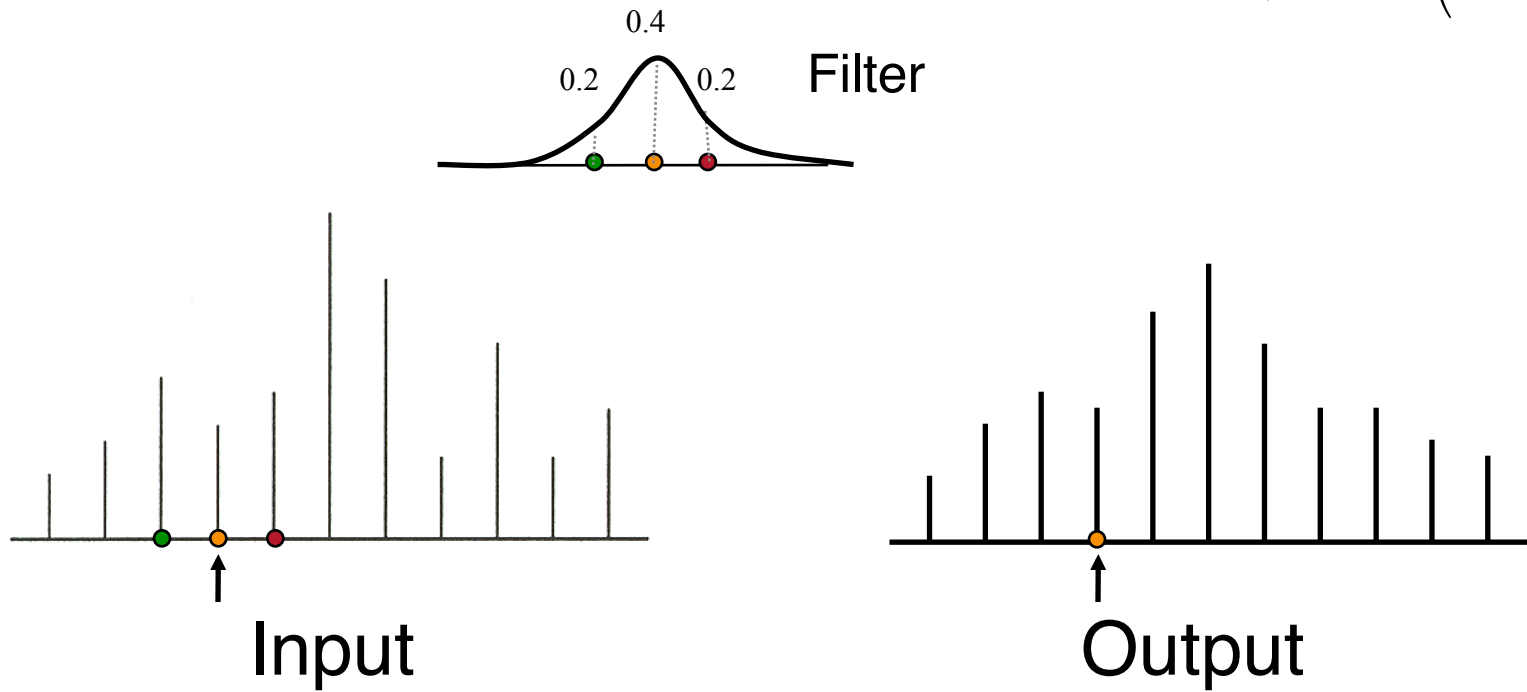


Convolution with a Gaussian Filter



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

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

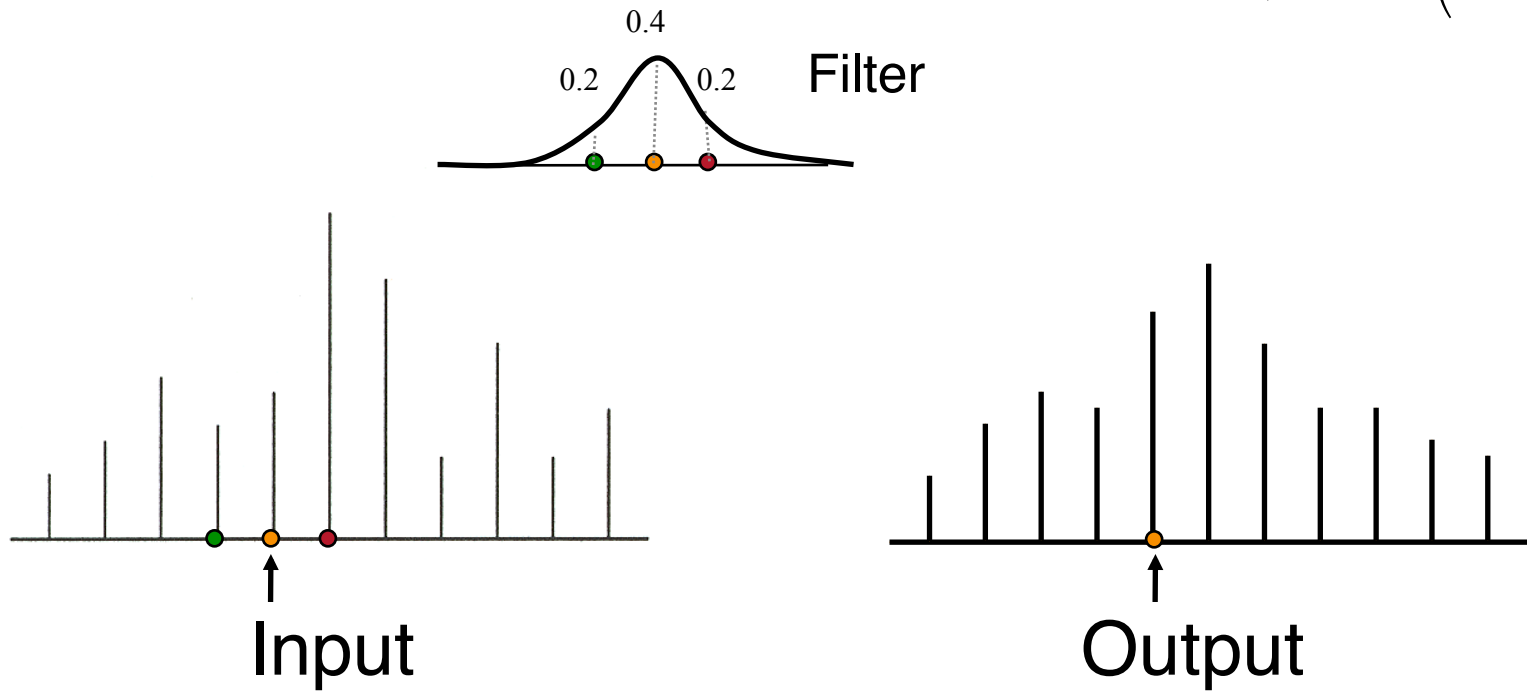


Convolution with a Gaussian Filter



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

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

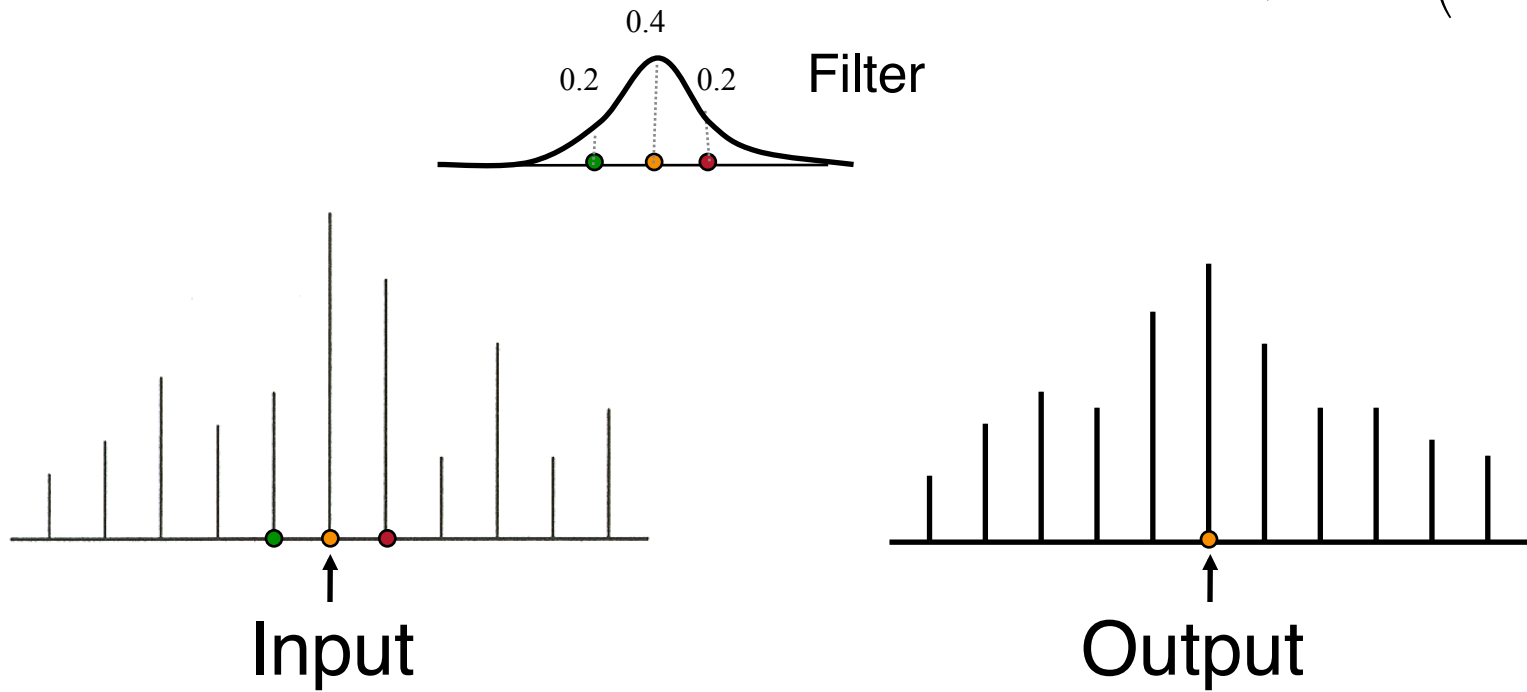


Convolution with a Gaussian Filter



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

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

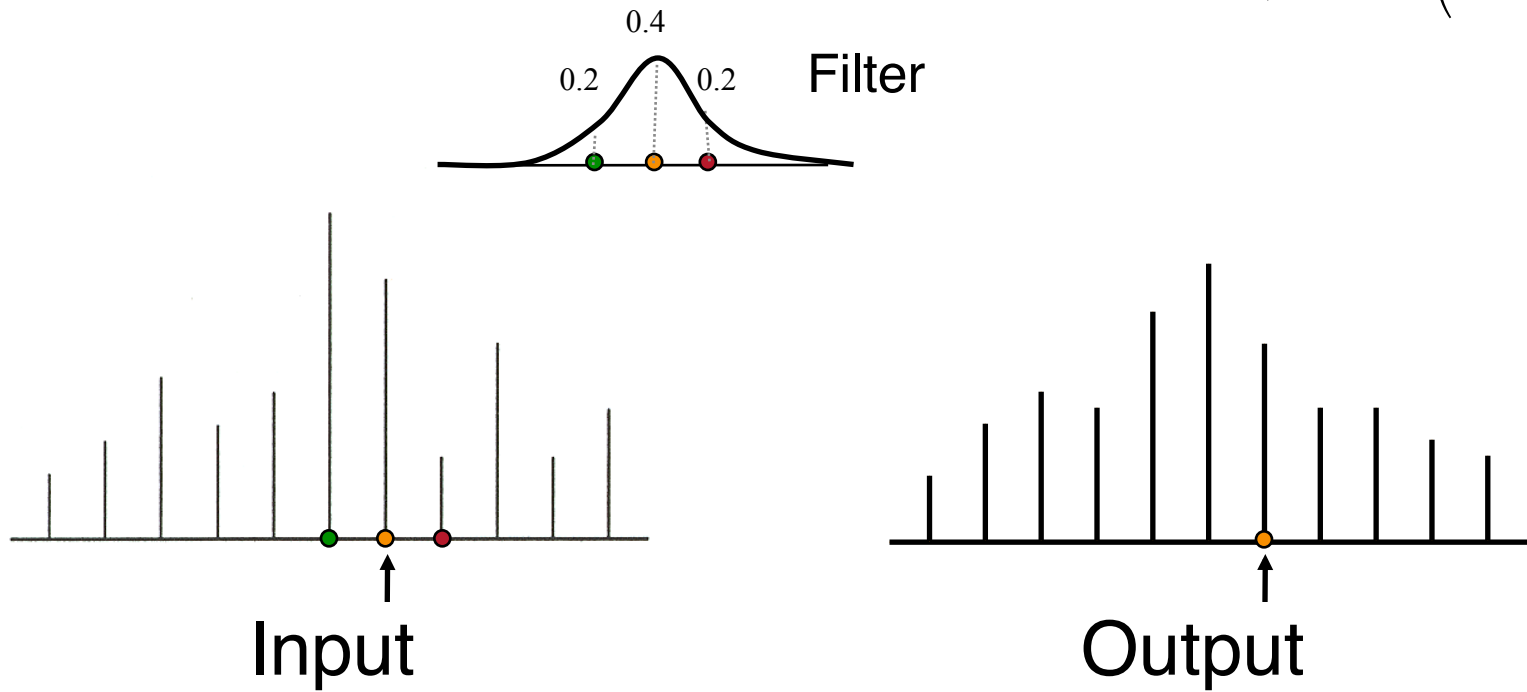


Convolution with a Gaussian Filter



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

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

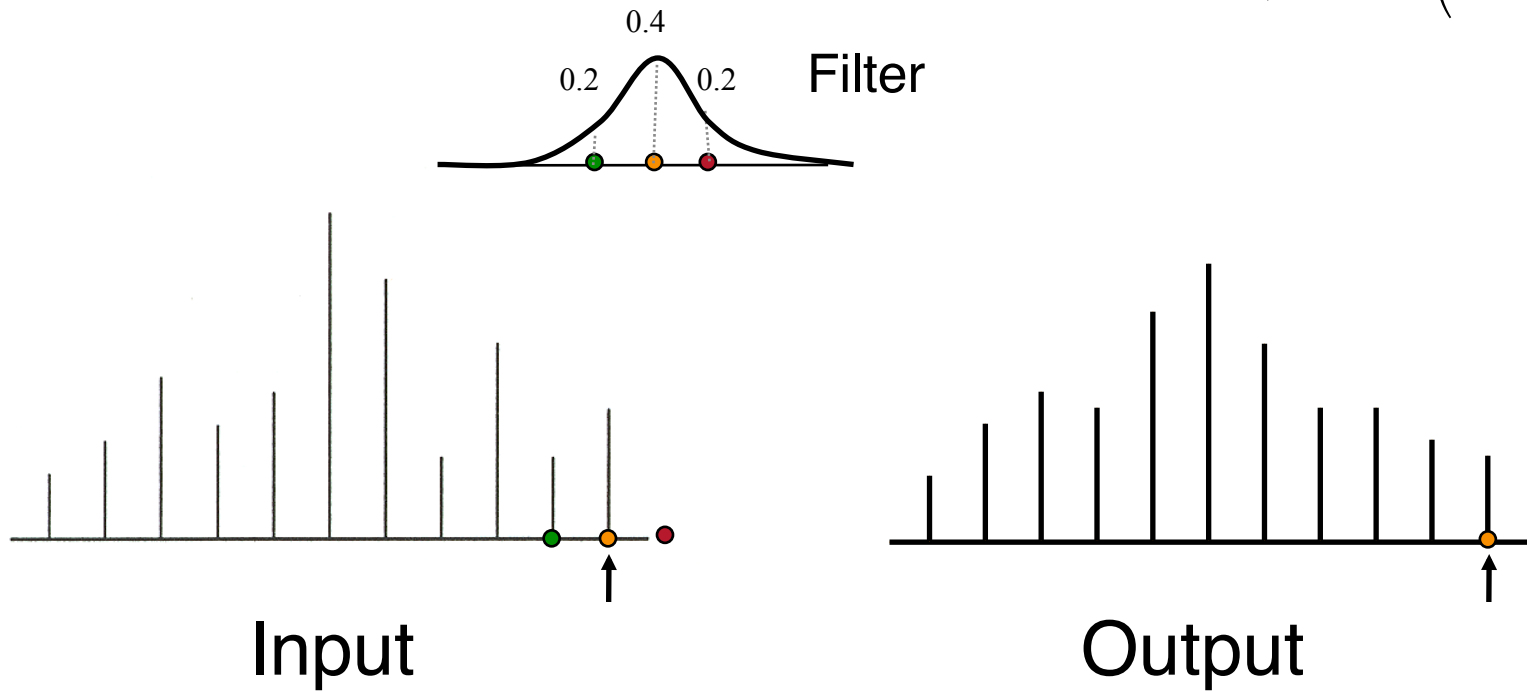


Convolution with a Gaussian Filter



What if filter extends beyond boundary?

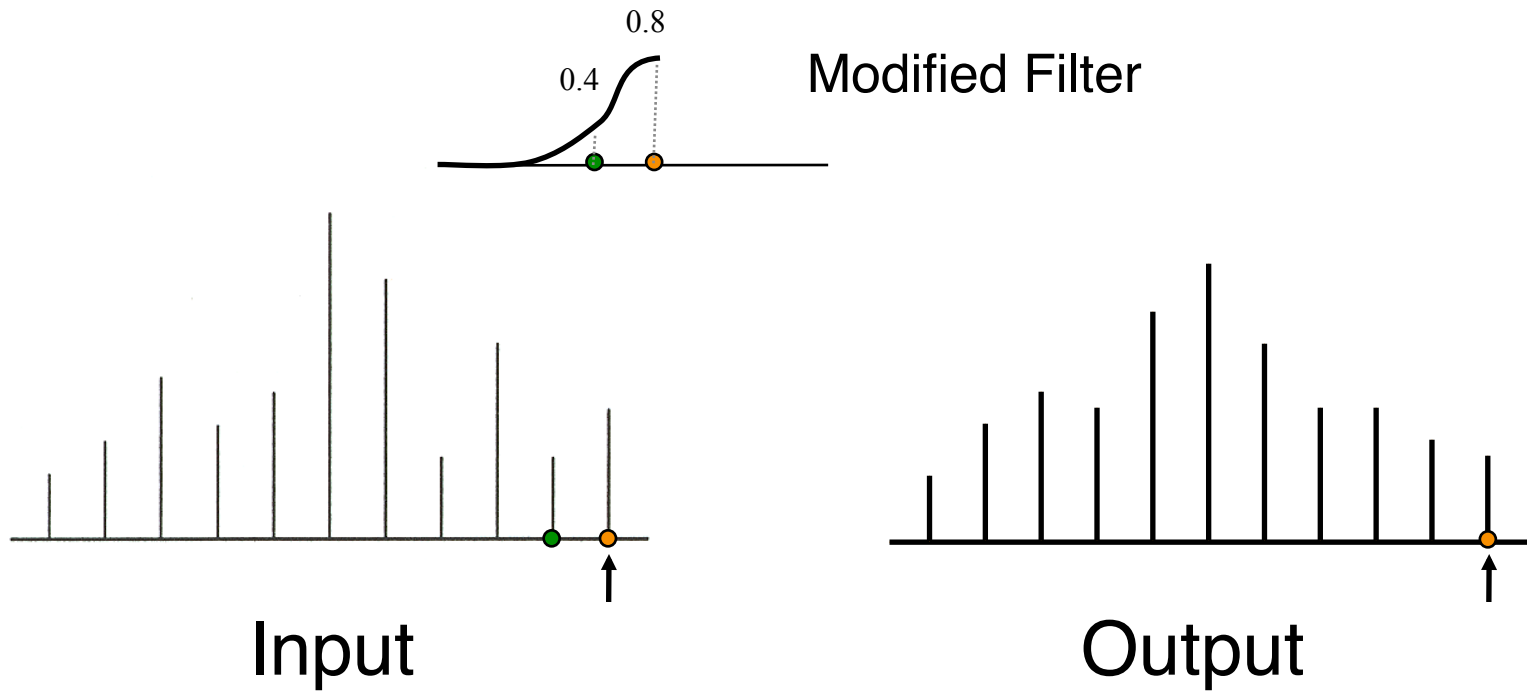
$$G(x, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$



Convolution with a Gaussian Filter



What if filter extends beyond boundary?



Convolution with a Gaussian Filter



Output contains samples from smoothed input

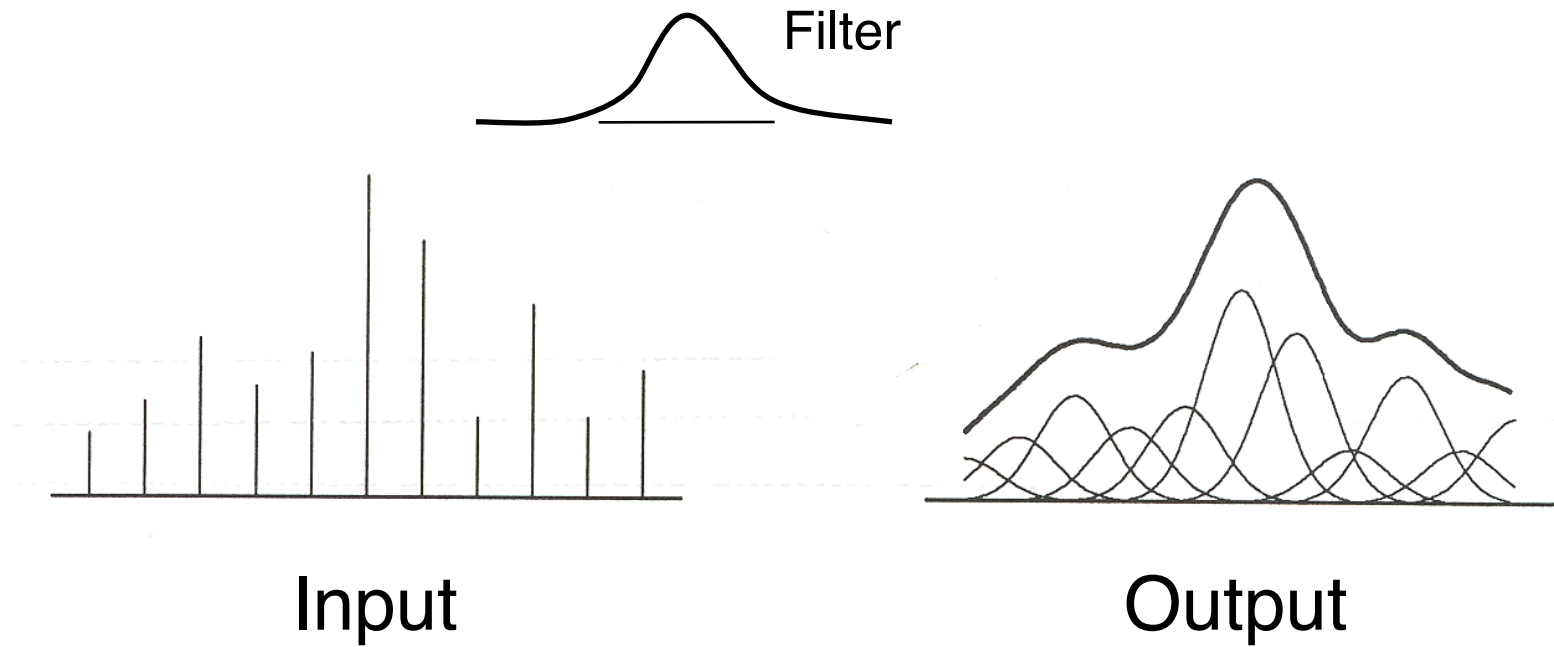
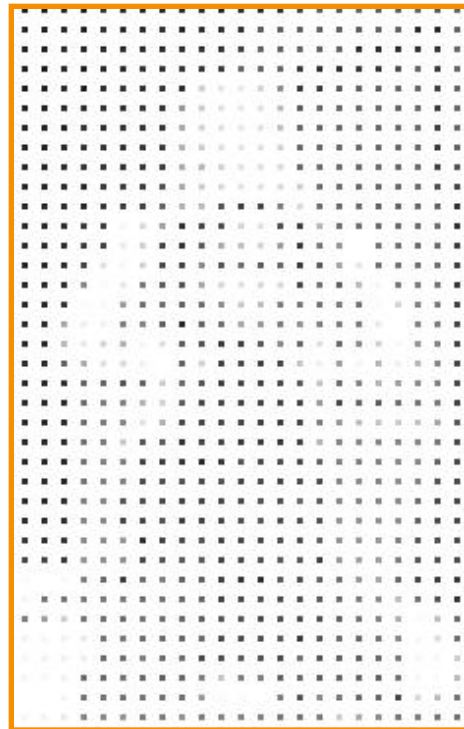


Figure 2.4 Wolberg

Linear Filtering

2D Convolution

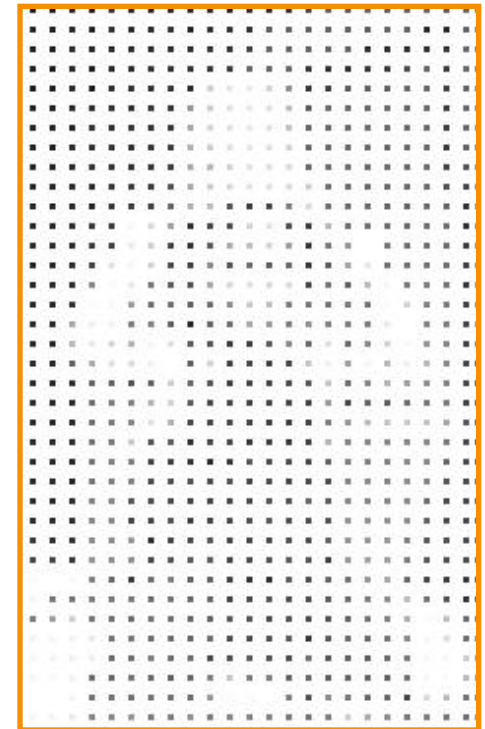
- o Each output pixel is a linear combination of input pixels in 2D neighborhood with weights prescribed by a filter



Input Image

$$\otimes \begin{array}{|c|} \hline \cdot \\ \hline \end{array} =$$

Filter

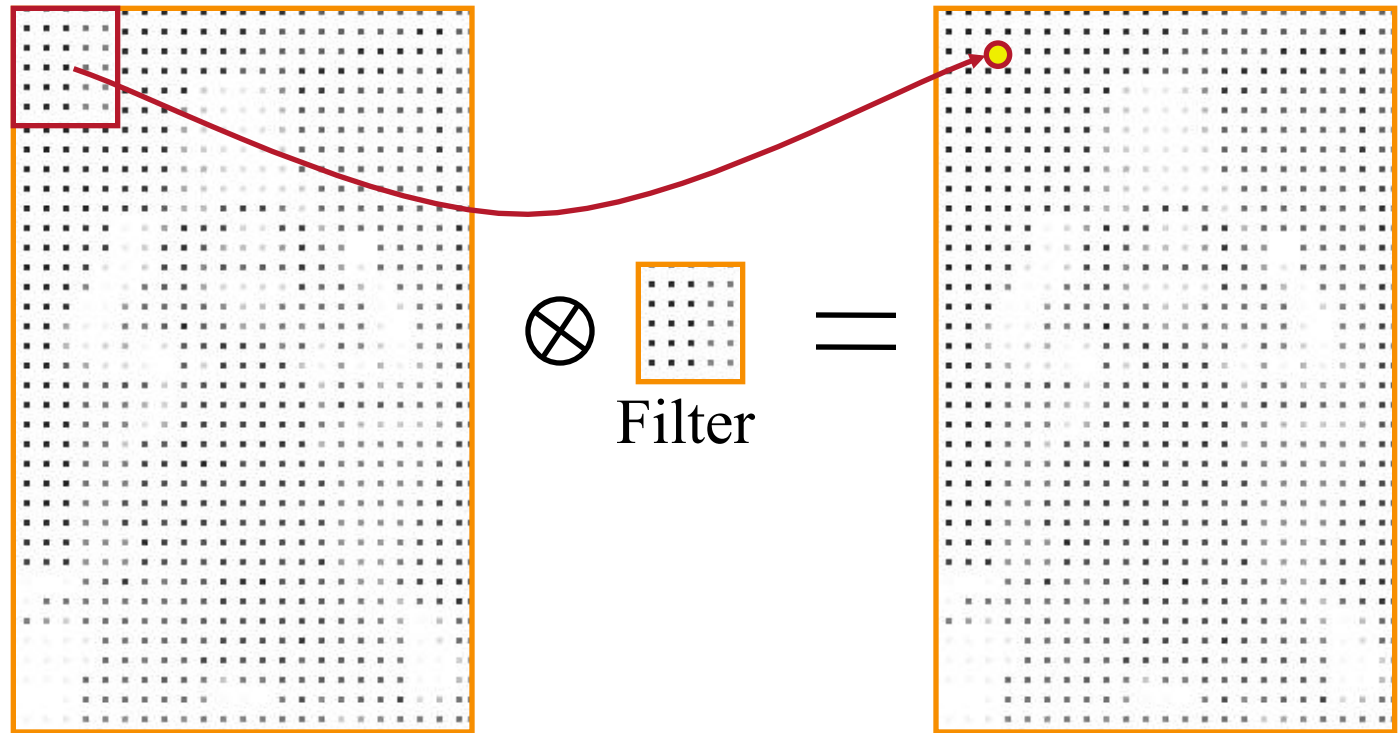


Output Image

Linear Filtering

2D Convolution

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



Input Image

Output Image

Linear Filtering

2D Convolution

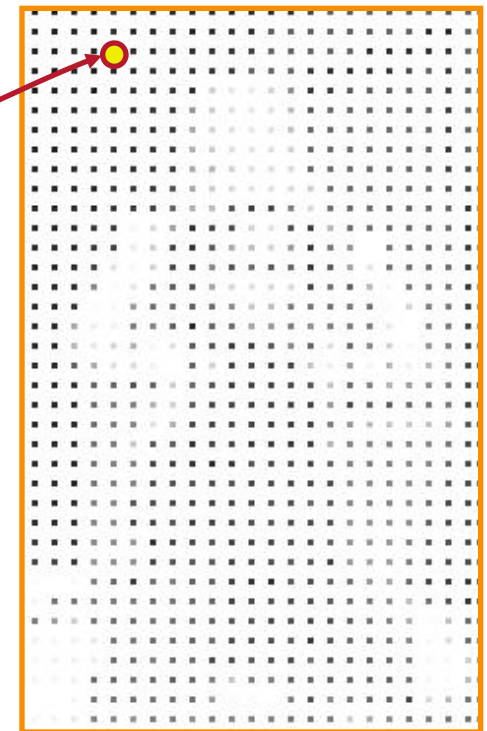
- o Each output pixel is a linear combination of input pixels in 2D neighborhood with weights prescribed by a filter



Input Image



Filter



Output Image

Linear Filtering

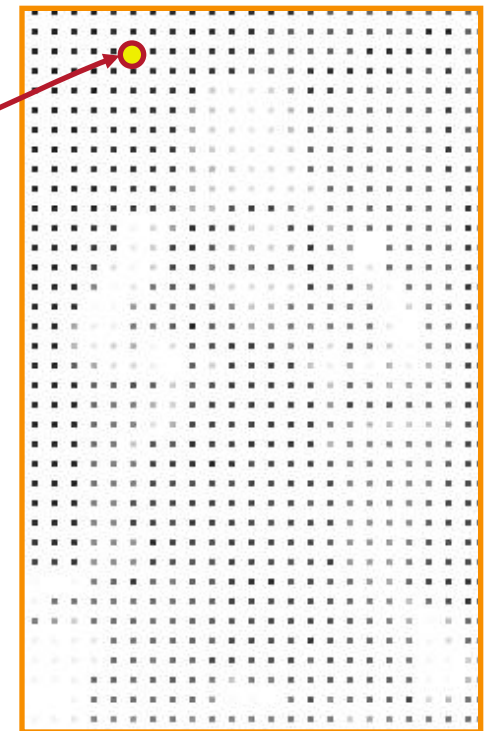
2D Convolution

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



Input Image

$$\otimes \begin{matrix} \square \\ \text{Filter} \end{matrix} =$$

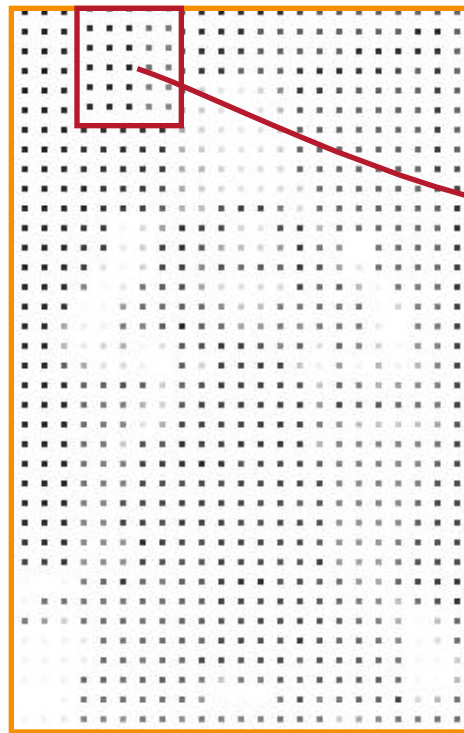


Output Image

Linear Filtering

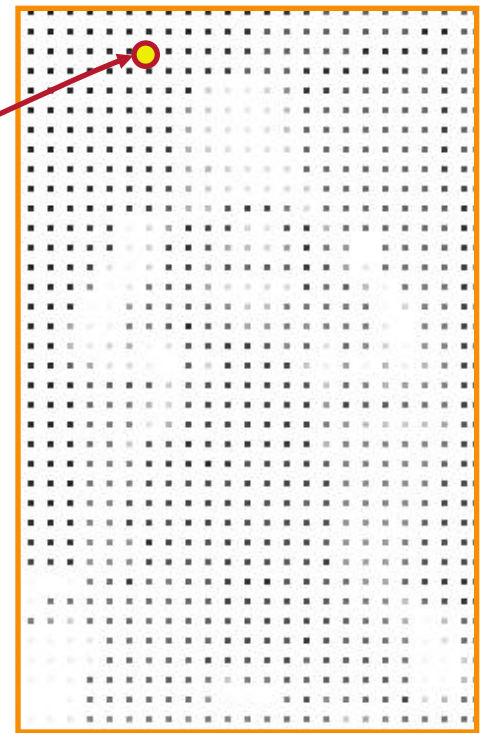
2D Convolution

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



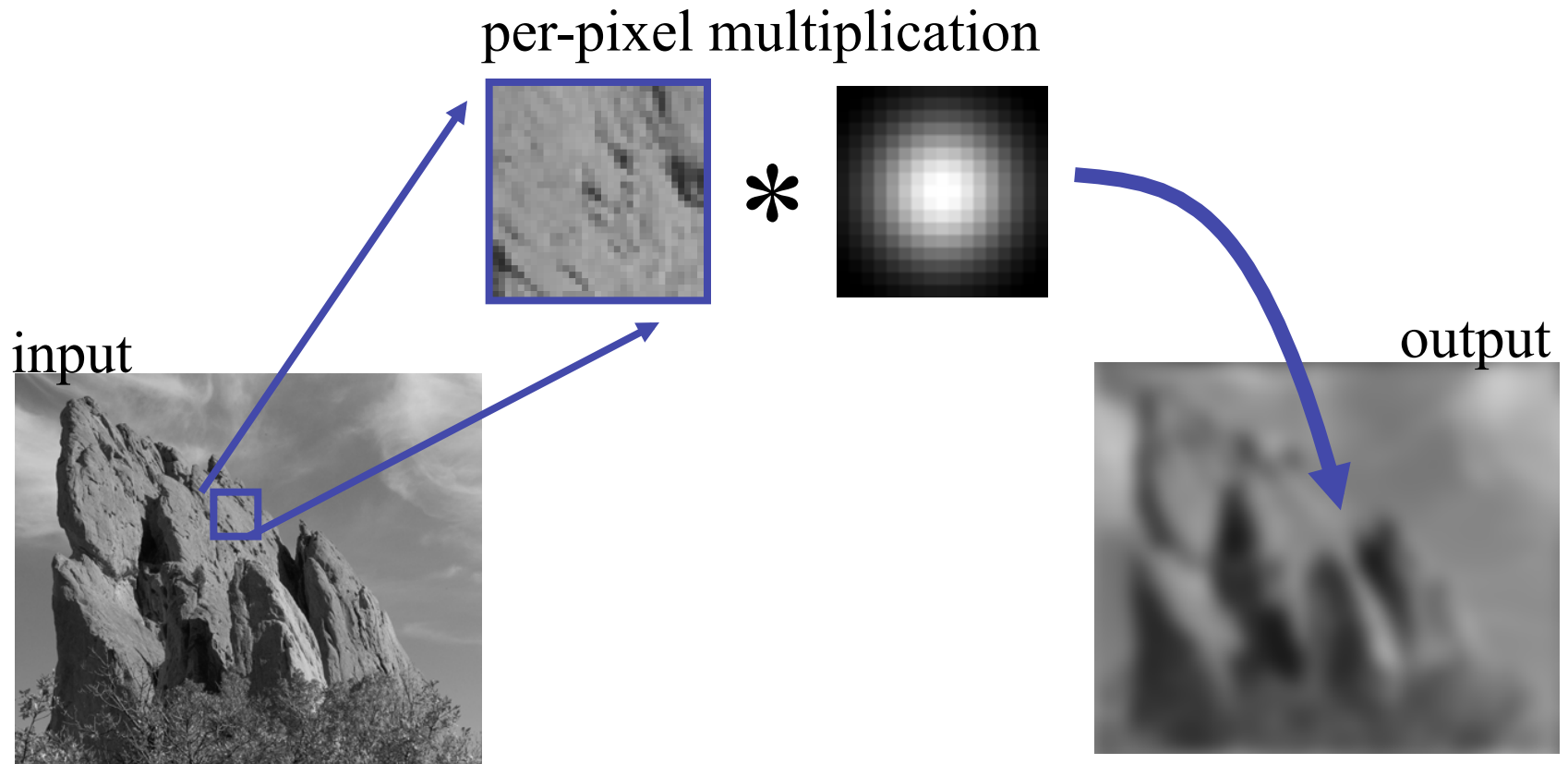
Input Image

$$\otimes \begin{matrix} \square \\ \text{Filter} \end{matrix} =$$



Output Image

Gaussian Blur



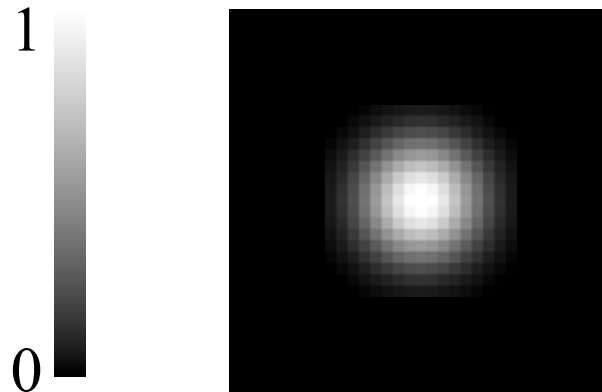
Gaussian Blur

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

$$\text{Blur}(I_p, \sigma) = \frac{1}{W_p} \sum_{q \in S} G(\|p - q\|, \sigma) I_q$$



normalized
Gaussian function



input



Gaussian blur



Linear Filtering

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

Filter = ?

Blur

Convolve with a 2D Gaussian filter



Original



Blur



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

$$\text{Filter} = \begin{bmatrix} 1/16 & 2/16 & 1/16 \\ 2/16 & 4/16 & 2/16 \\ 1/16 & 2/16 & 1/16 \end{bmatrix}$$



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)



Image Processing Operations

- Luminance
 - Brightness
 - Contrast.
 - Gamma
 - Histogram equalization
- Color
 - Grayscale
 - Saturation
 - White balance
- Linear filtering
 - Blur & sharpen
 - Edge detect
 - Convolution
- Non-linear filtering
 - Median
 - Bilateral filter
- Dithering
 - Quantization
 - Ordered dither
 - Floyd-Steinberg



Non-Linear Filtering

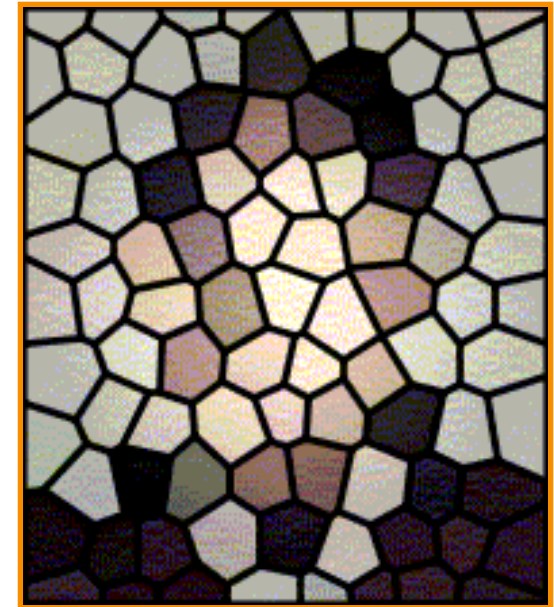
Each output pixel is a non-linear function of input pixels in neighborhood (filter depends on input)



Original



Paint



Stained Glass



Median 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 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

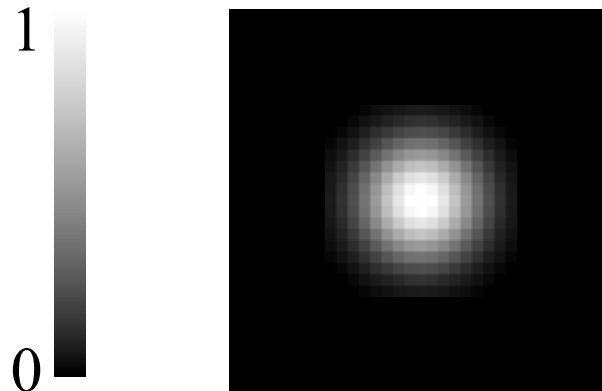
Gaussian Blur

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

$$\text{Blur}(I_p, \sigma) = \frac{1}{W_p} \sum_{q \in S} G(\|p - q\|, \sigma) I_q$$



normalized
Gaussian function





Bilateral Filter

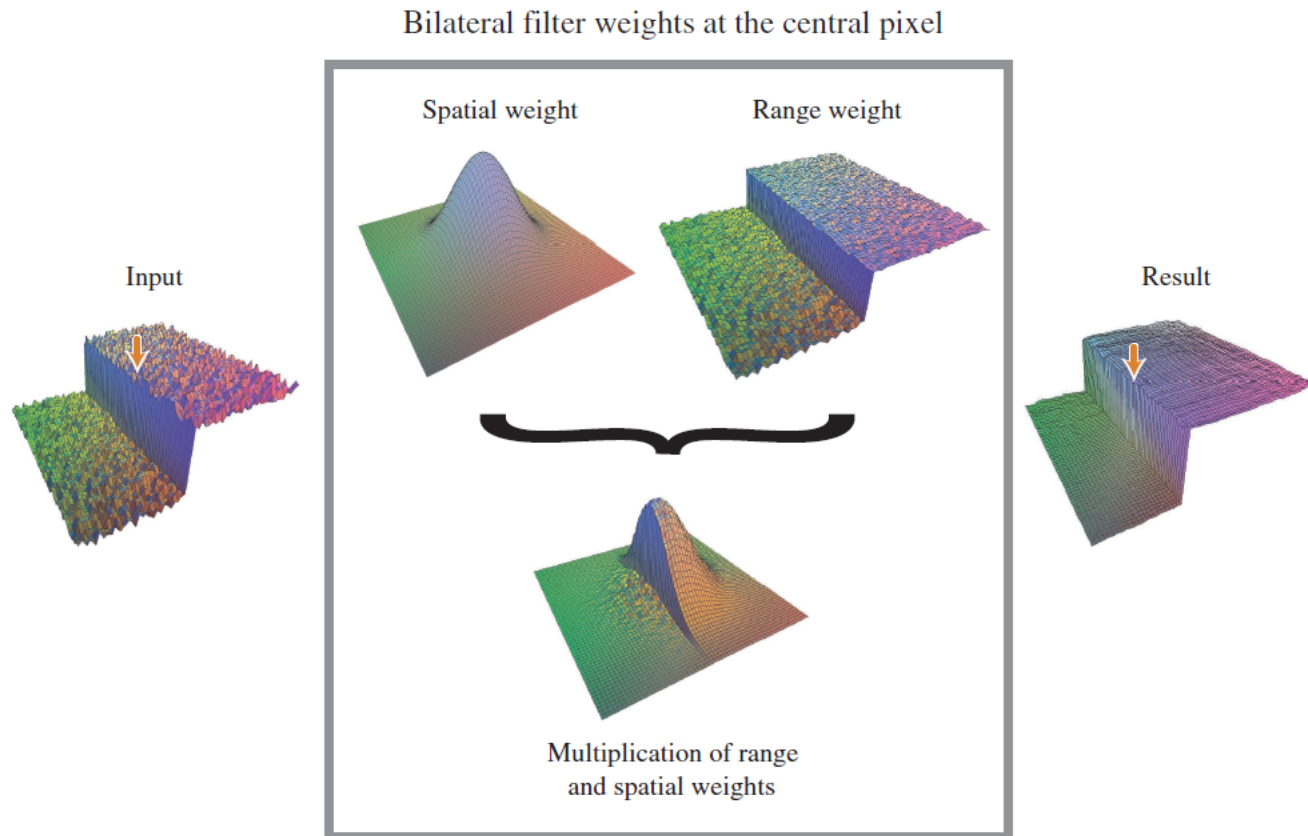
Combine Gaussian filtering in both spatial domain and color domain

$$Bilateral [I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s} (\|\mathbf{p} - \mathbf{q}\|) G_{\sigma_r} (|I_p - I_q|) I_q$$

\uparrow \uparrow
Spatial Color
Proximity Proximity
Weight Weight

Bilateral Filtering

Combine Gaussian filtering in both spatial domain and color domain





input

$\sigma_r = 0.1$



$\sigma_r = 0.25$



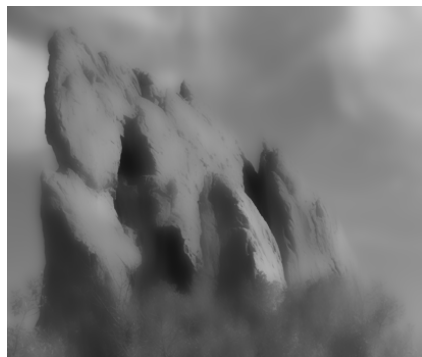
$\sigma_r = \infty$
(Gaussian blur)



$\sigma_s = 2$



$\sigma_s = 6$



$\sigma_s = 18$

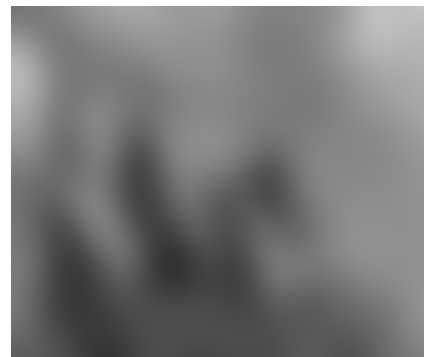
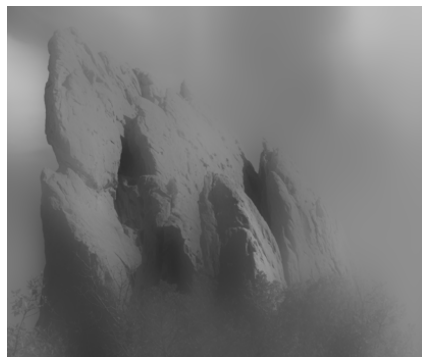




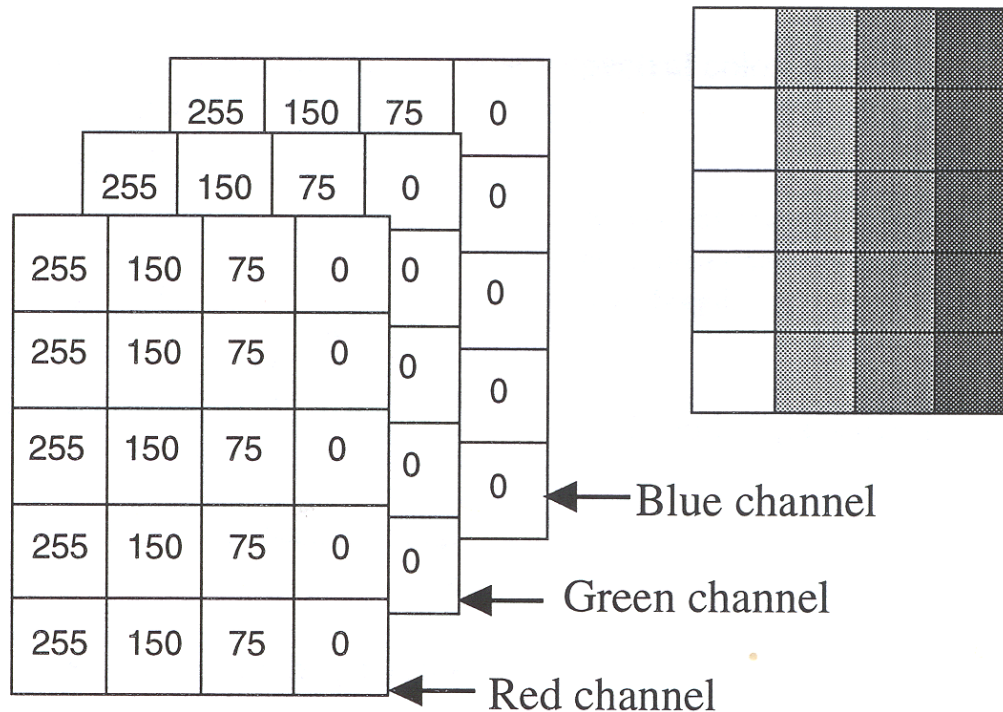
Image Processing Operations

- Luminance
 - Brightness
 - Contrast.
 - Gamma
 - Histogram equalization
- Color
 - Grayscale
 - Saturation
 - White balance
- Linear filtering
 - Blur & sharpen
 - Edge detect
 - Convolution
- Non-linear filtering
 - Median
 - Bilateral filter
- Dithering
 - Quantization
 - Ordered dither
 - Floyd-Steinberg

Quantization

Reduce intensity resolution

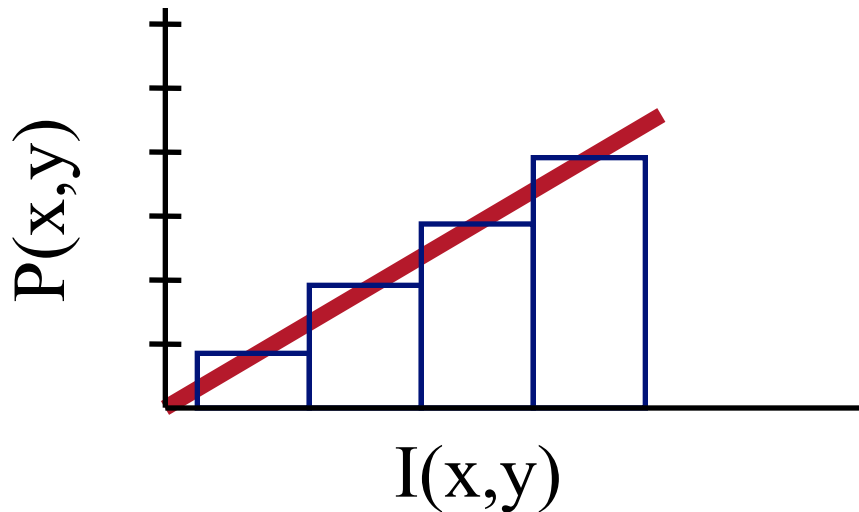
- o Frame buffers have limited number of bits per pixel
- o Physical devices have limited dynamic range



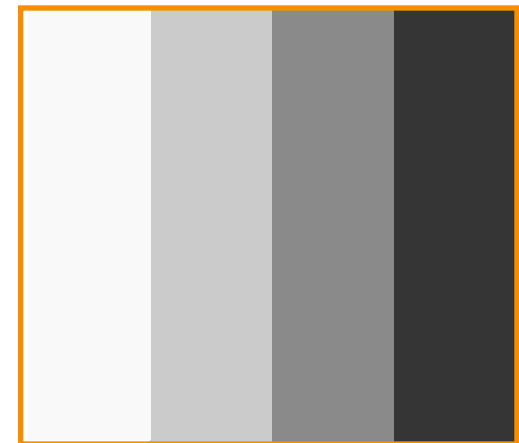


Uniform Quantization

$P(x, y) = \text{round}(I(x, y))$
where $\text{round}()$ chooses nearest value that can be represented.



$I(x,y)$



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

Uniform Quantization

Images with decreasing bits per pixel:



8 bits



4 bits



2 bits



1 bit

Notice contouring.

Reducing Effects of Quantization



- Intensity resolution / spatial resolution tradeoff
- Dithering
 - Random dither
 - Ordered dither
 - Error diffusion dither
- Halftoning
 - Classical halftoning

Dithering

Distribute errors among pixels

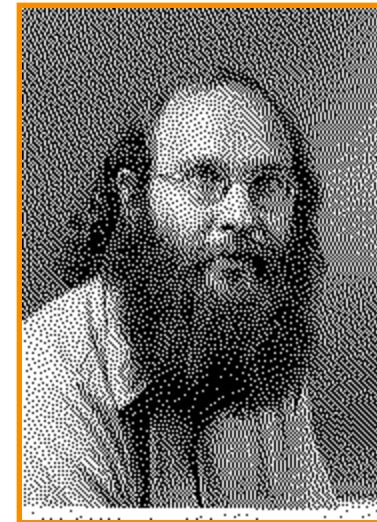
- o Exploit spatial integration in our eye
- o Display greater range of perceptible intensities



Original
(8 bits)



Uniform
Quantization
(1 bit)



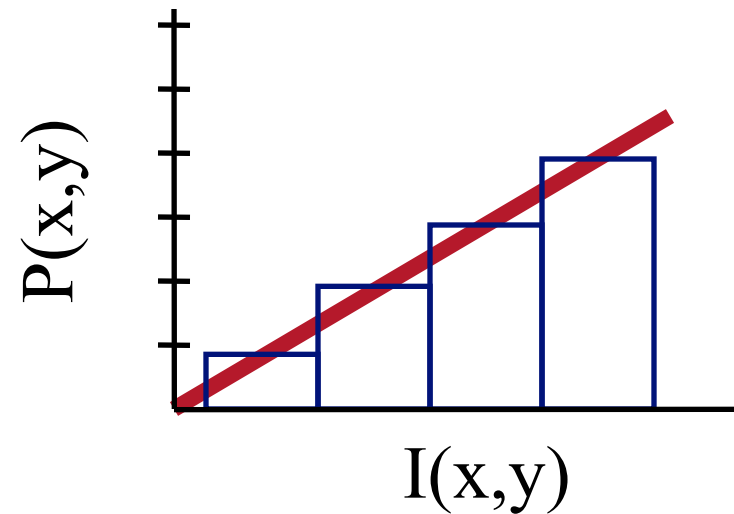
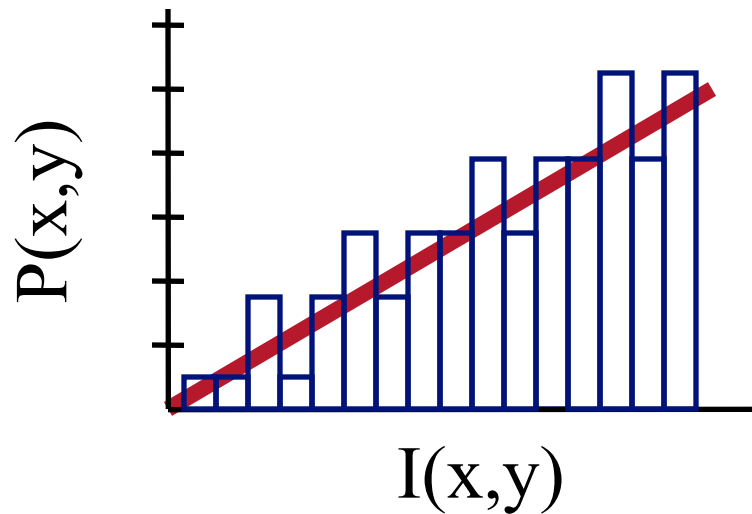
Floyd-Steinberg
Dither
(1 bit)



Random Dither

Randomize quantization errors

- o Errors appear as noise



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

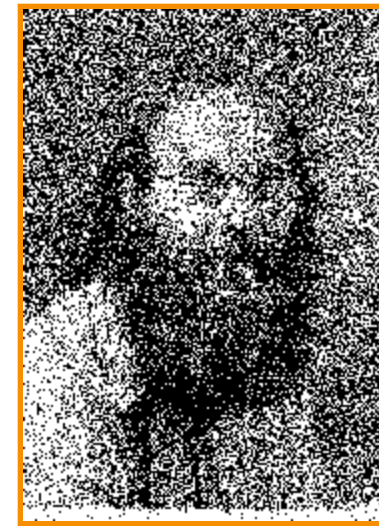
Random Dither



Original
(8 bits)



Uniform
Quantization
(1 bit)



Random
Dither
(1 bit)



Ordered Dither

Pseudo-random quantization errors

- o Matrix stores pattern of thresholds

```
i = x mod n
```

```
j = y mod n
```

```
e = I(x,y) - trunc(I(x,y))
```

```
threshold = (D(i,j)+1)/(n2+1)
```

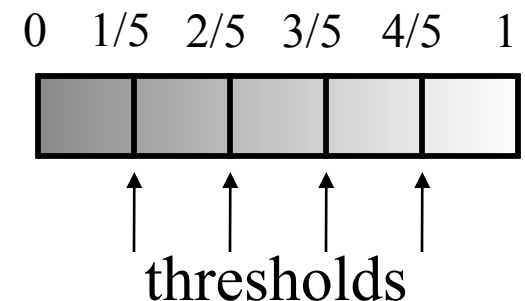
```
if (e > threshold)
```

```
    P(x,y) = ceil(I(x, y))
```

```
else
```

```
    P(x,y) = floor(I(x,y))
```

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





Ordered Dither

Bayer's ordered dither matrices

$$D_n = \begin{bmatrix} 4D_{n/2} + D_2(1,1)U_{n/2} & 4D_{n/2} + D_2(1,2)U_{n/2} \\ 4D_{n/2} + D_2(2,1)U_{n/2} & 4D_{n/2} + D_2(2,2)U_{n/2} \end{bmatrix}$$

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

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

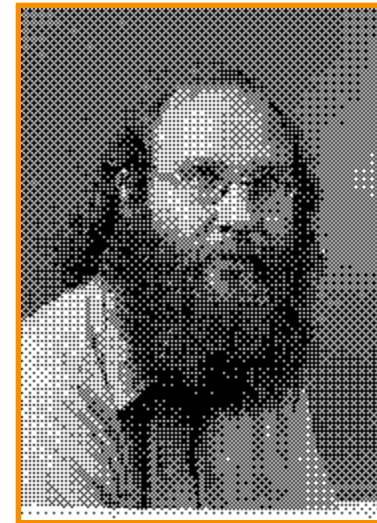
Ordered Dither



Original
(8 bits)



Random
Dither
(1 bit)



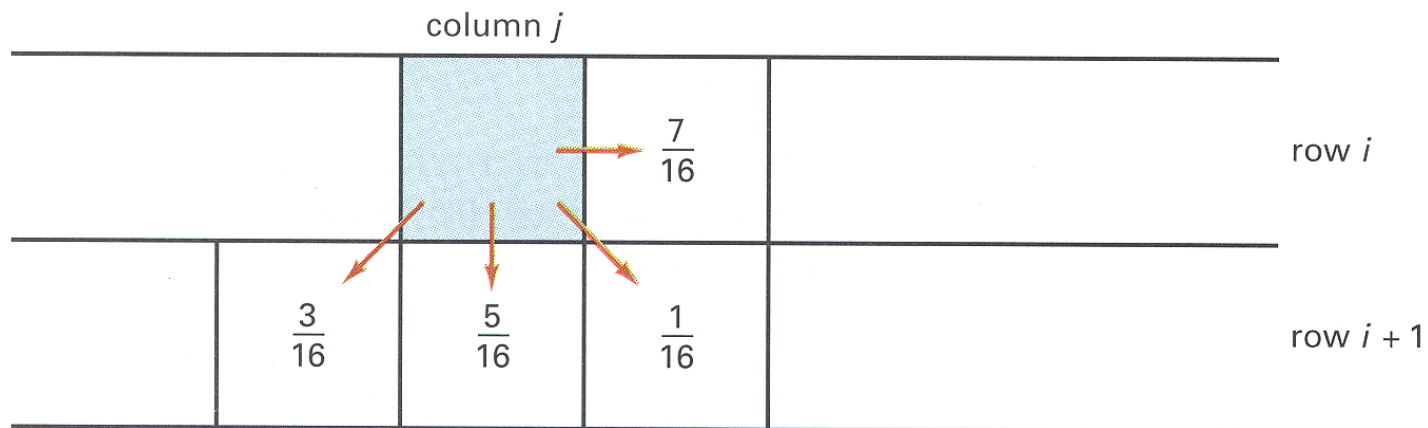
Ordered
Dither
(1 bit)



Error Diffusion Dither

Spread quantization error over neighbor pixels

- o Error dispersed to pixels right and below
- o Floyd-Steinberg weights:



$$\frac{3}{16} + \frac{5}{16} + \frac{1}{16} + \frac{7}{16} = 1.0$$

Figure 14.42 from H&B

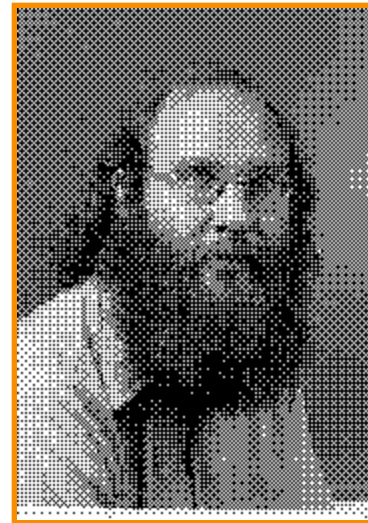
Error Diffusion Dither



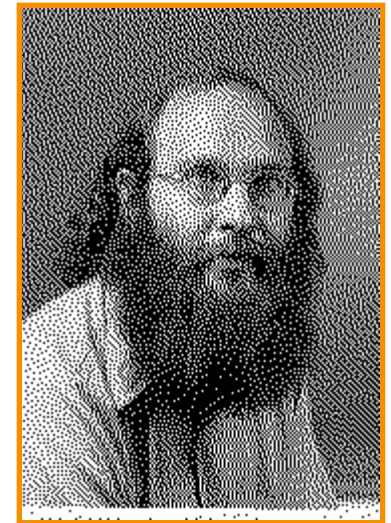
Original
(8 bits)



Random
Dither
(1 bit)



Ordered
Dither
(1 bit)



Floyd-Steinberg
Dither
(1 bit)

Summary



- Color transformations
 - Different color spaces useful for different operations
- Filtering
 - Compute new values for image pixels based on function of old values in neighborhood
- Dithering
 - Reduce visual artifacts due to quantization
 - Distribute errors among pixels
 - Exploit spatial integration in our eye