

Image Filtering

Jianping Fan

**Department of Computer Science
UNC-Charlotte**

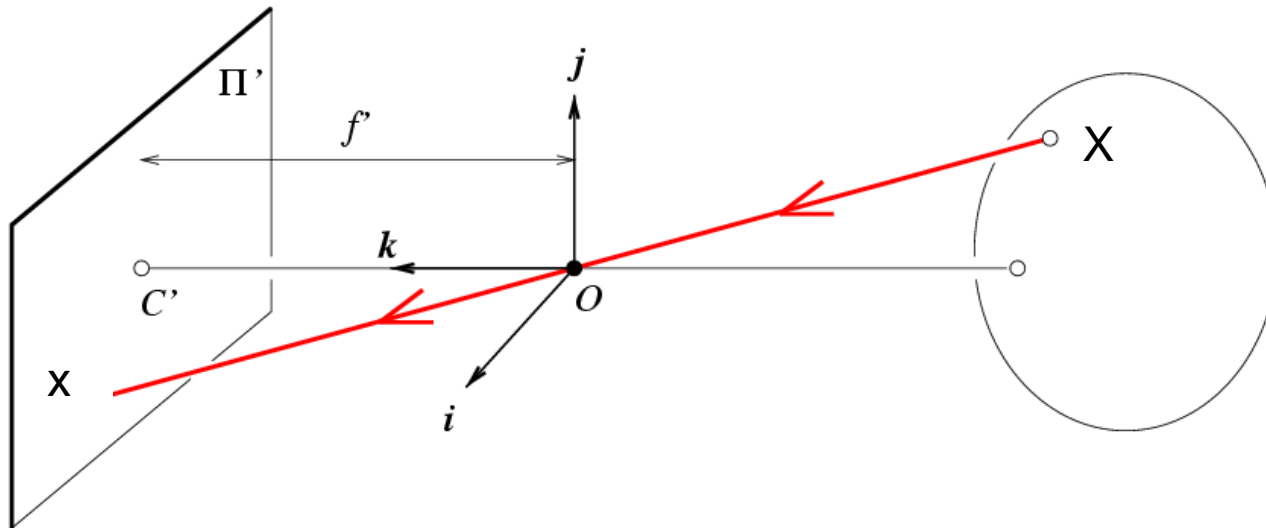
Course Website:

<http://webpages.uncc.edu/jfan/itcs5152.html>

Previous classes

- Computer vision overview
- Mathematics of pinhole camera (Geometry)

Recap: projection



$$\mathbf{x} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}] \mathbf{X}$$

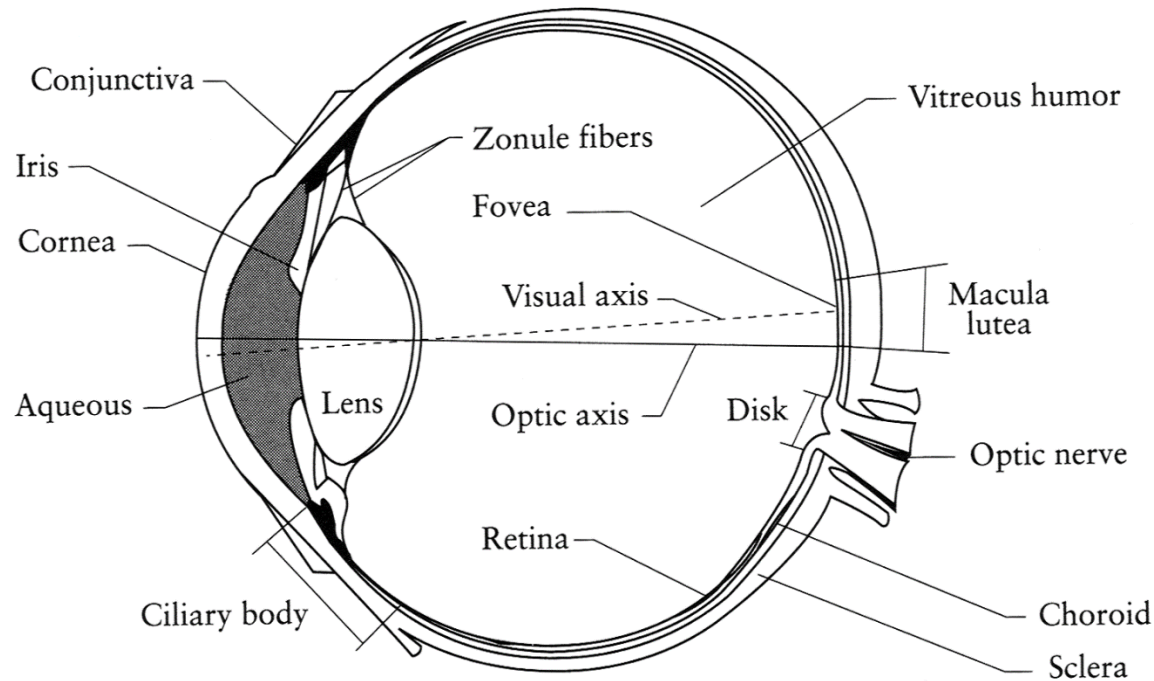


$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Today's class

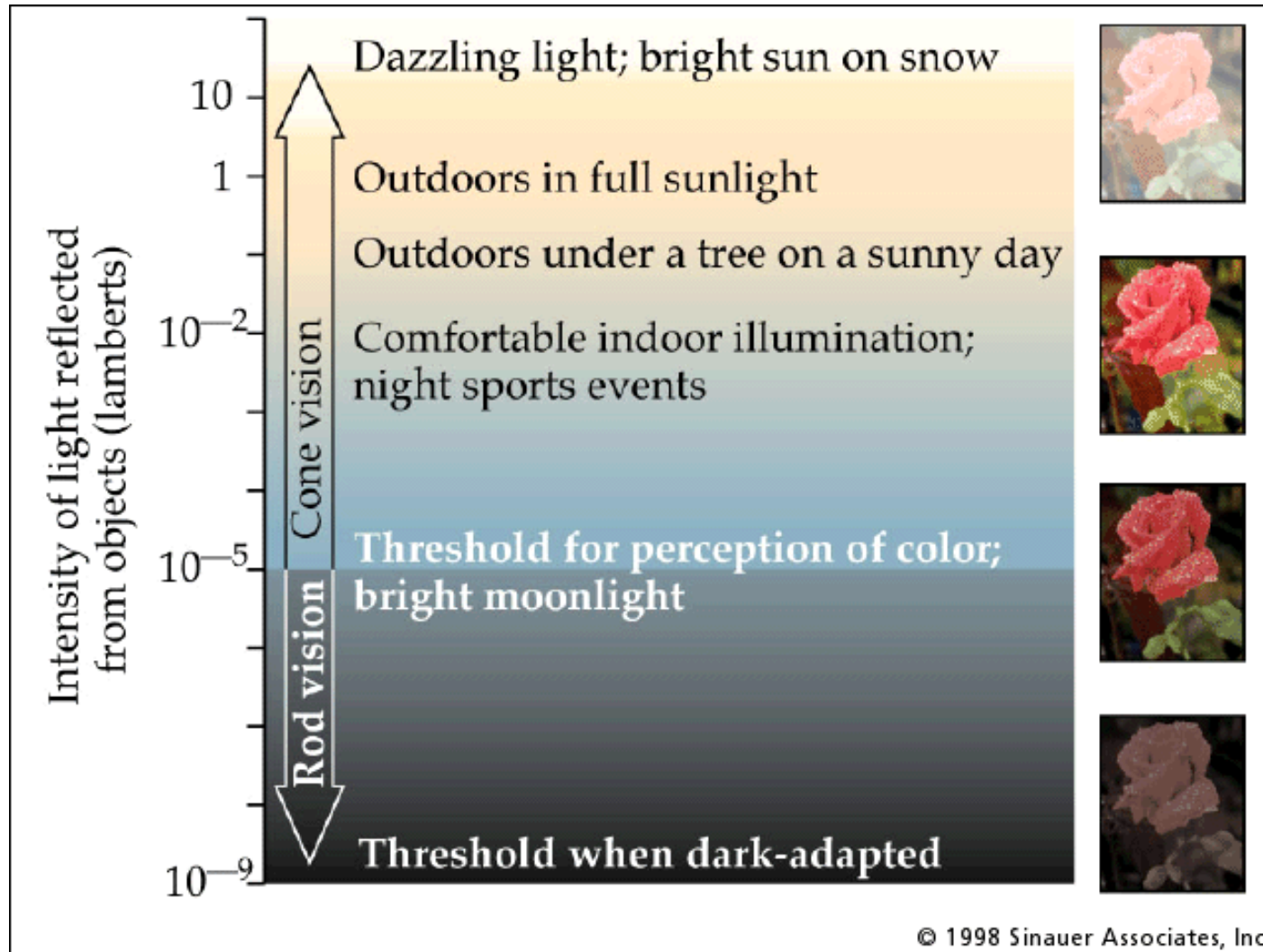
- Biological vision and color
- Image filtering

The Eye

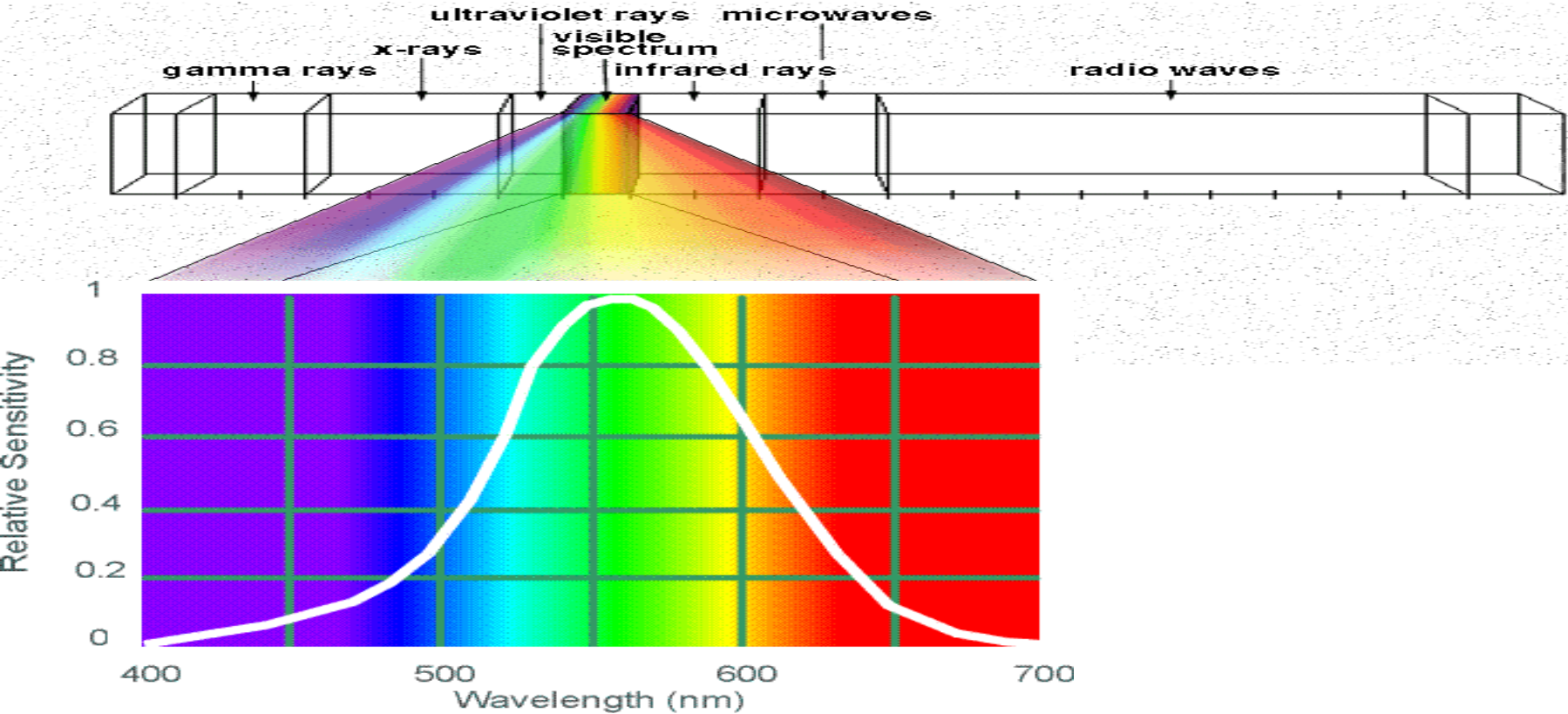


- The human eye is a camera!
 - **Iris** - colored annulus with radial muscles
 - **Pupil** - the hole (aperture) whose size is controlled by the iris
 - What's the "film"?
 - photoreceptor cells (rods and cones) in the **retina**

Rod / Cone sensitivity



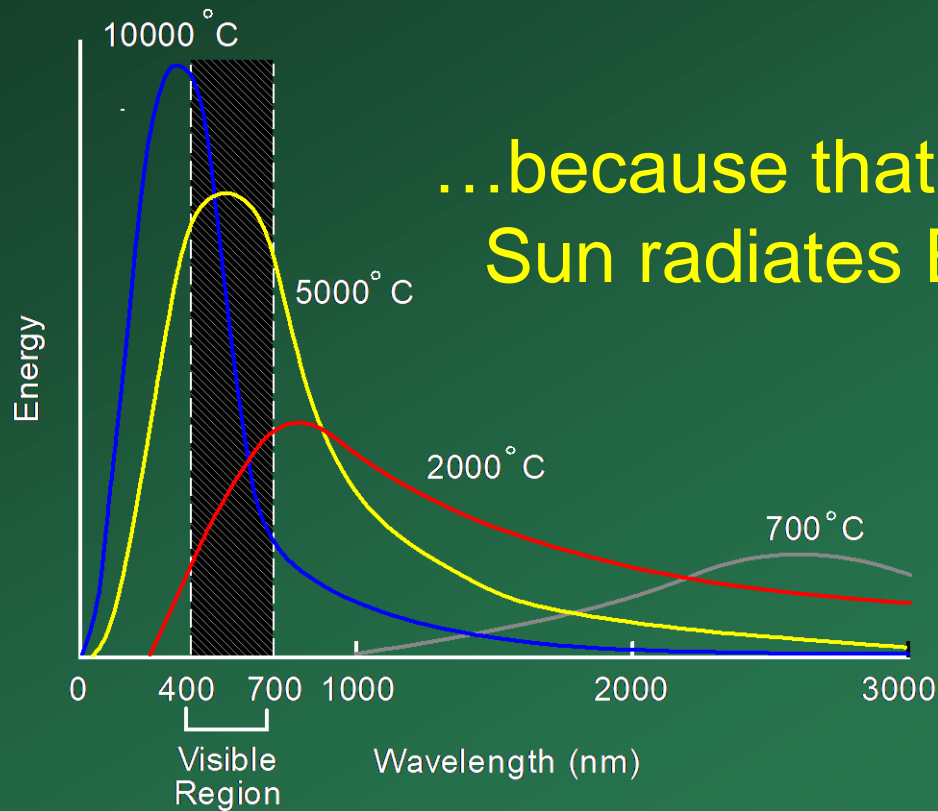
Electromagnetic Spectrum



Human Luminance Sensitivity Function

Visible Light

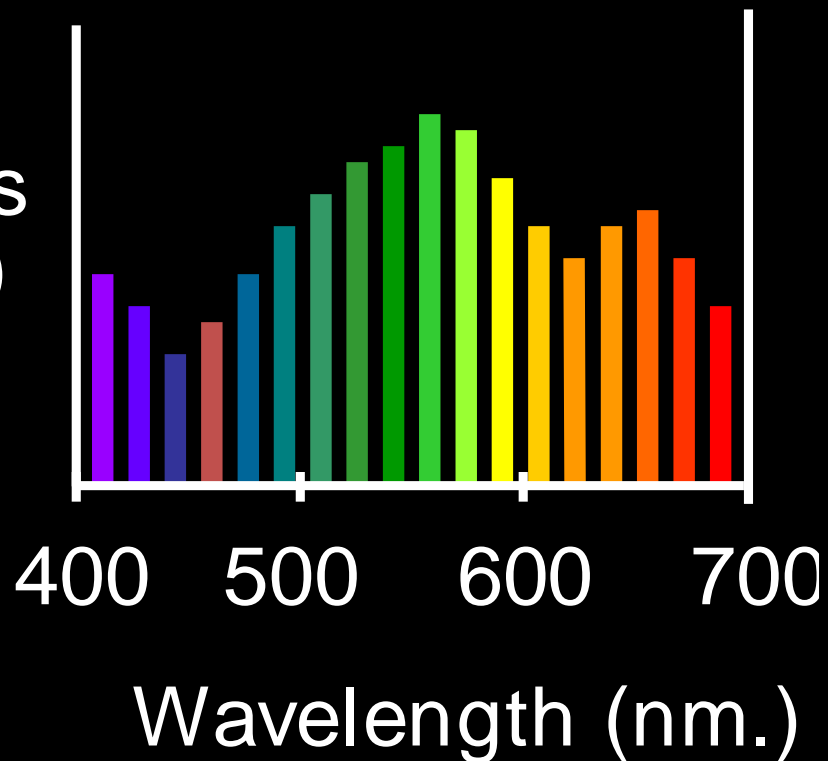
Why do we see light of these wavelengths?



The Physics of Light

Any patch of light can be completely described physically by its spectrum: the number of photons (per time unit) at each wavelength 400 - 700 nm.

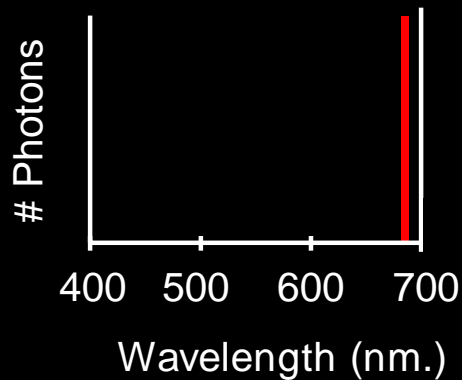
Photons
(per ms.)



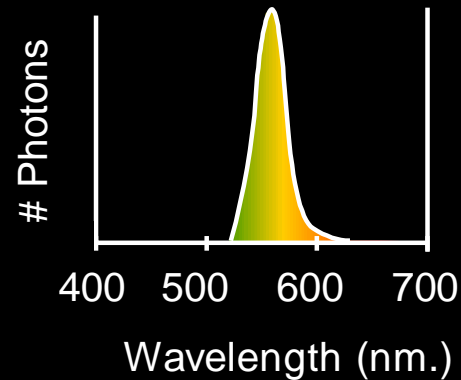
The Physics of Light

Some examples of the spectra of light sources

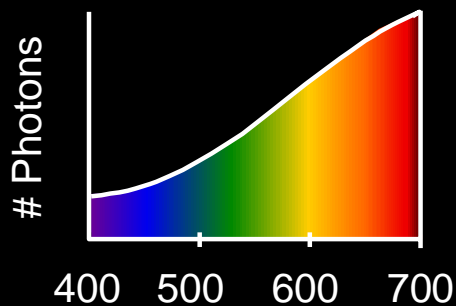
A. Ruby Laser



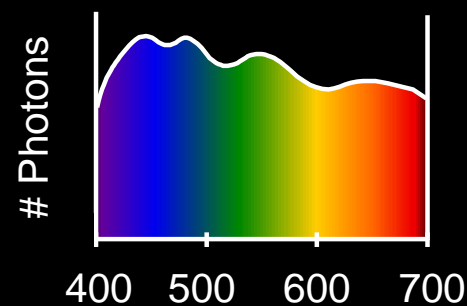
B. Gallium Phosphide Crystal



C. Tungsten Lightbulb



D. Normal Daylight

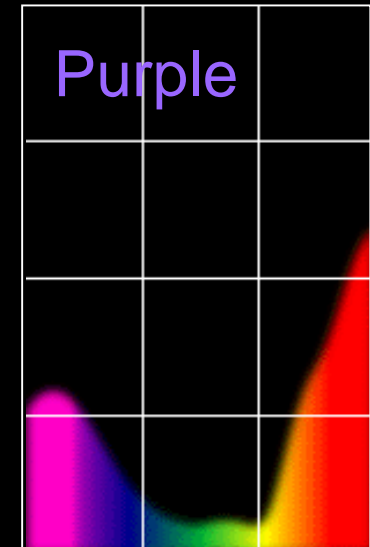
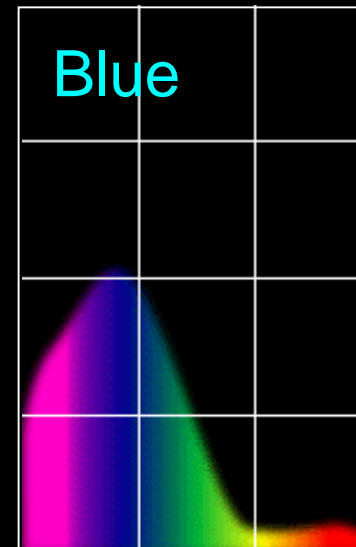
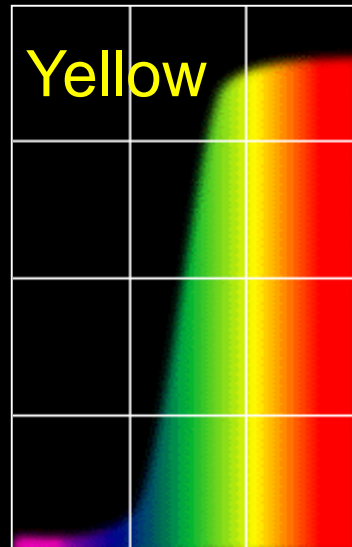
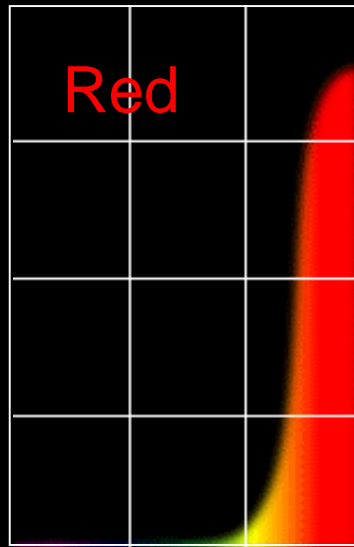


The Physics of Light

Some examples of the reflectance spectra of surfaces



% Photons Reflected



400

700

400

700

400

700

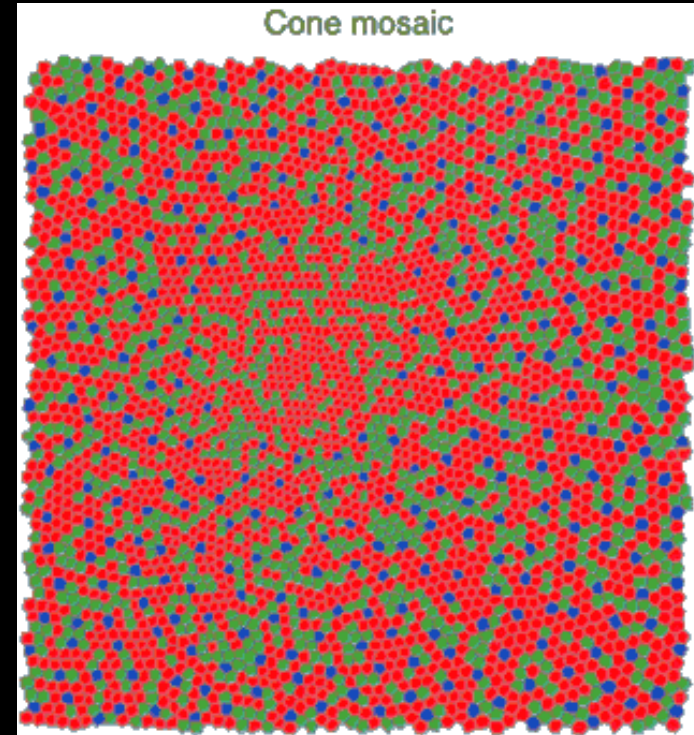
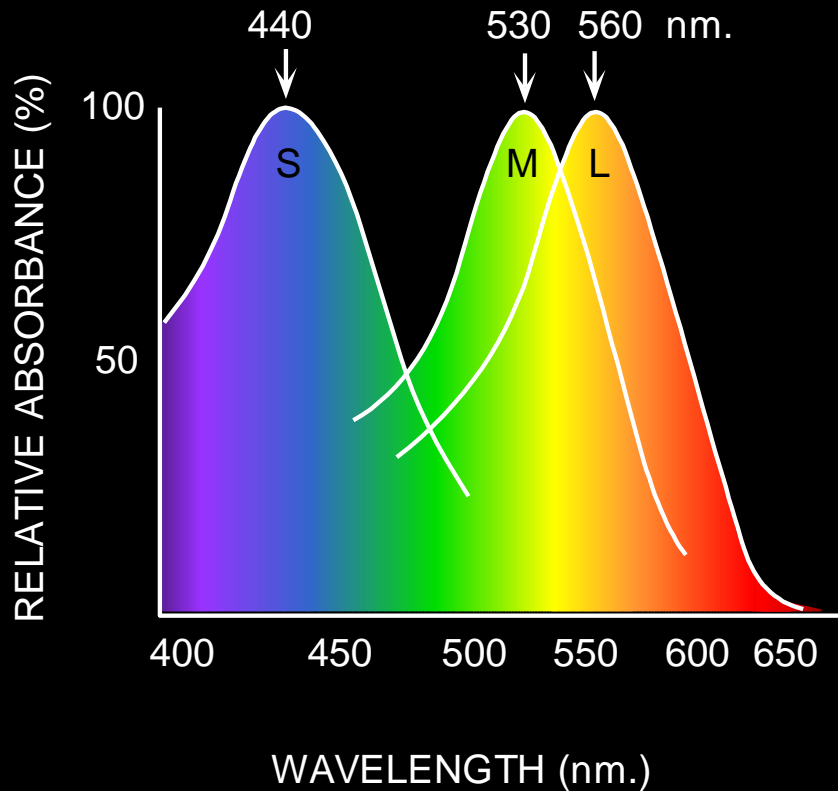
400

700

Wavelength (nm)

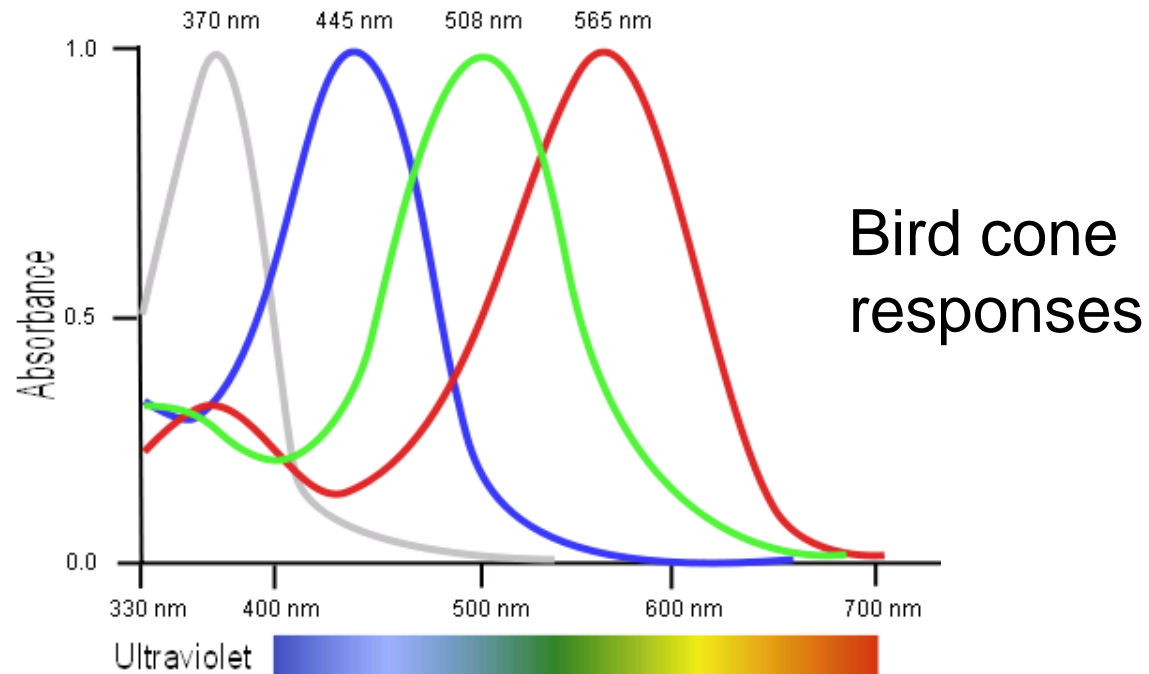
Physiology of Color Vision

Three kinds of cones:



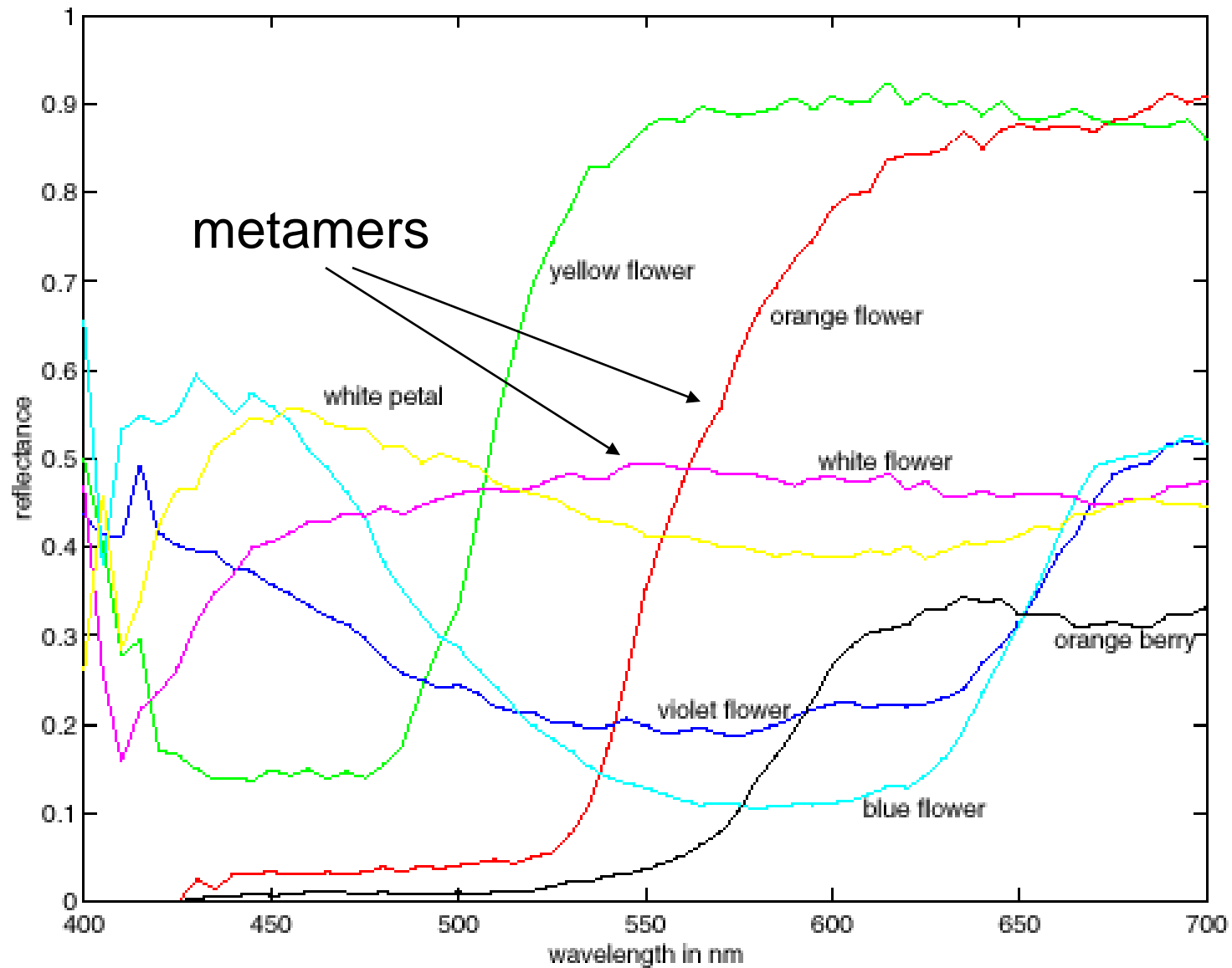
- Why are M and L cones so close?
- Why are there 3?

Tetrachromatism

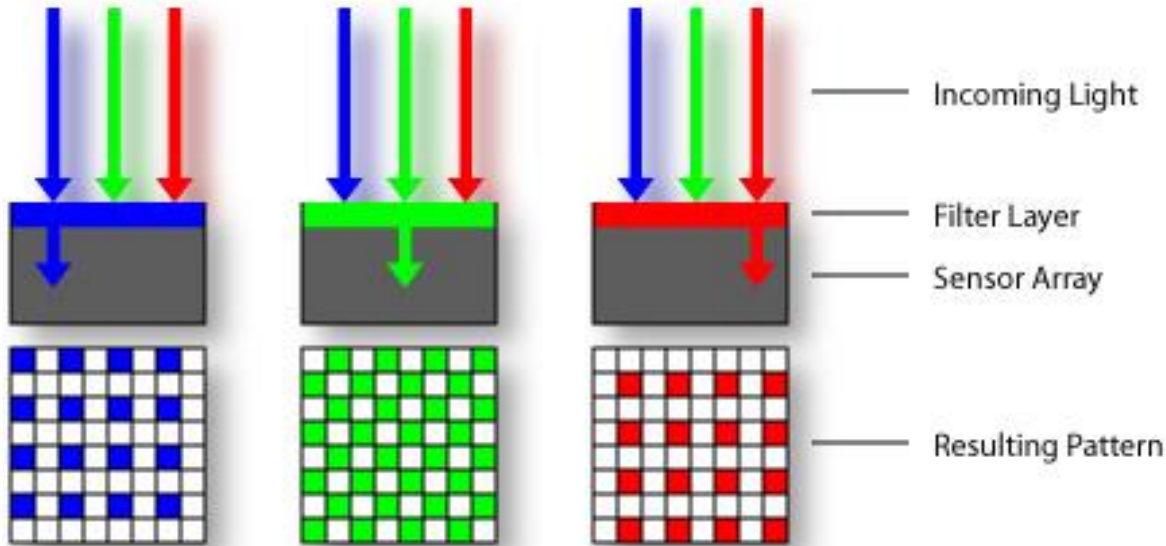
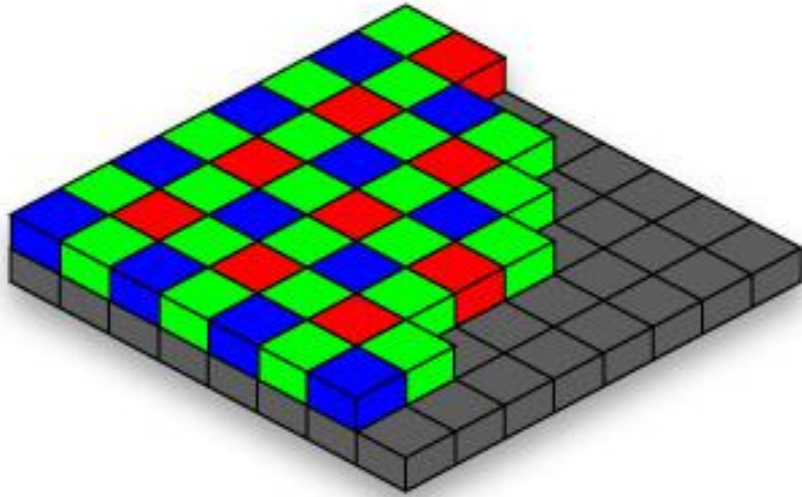


- Most birds, and many other animals, have cones for ultraviolet light.
- Some humans, mostly female, seem to have slight tetrachromatism.

More Spectra

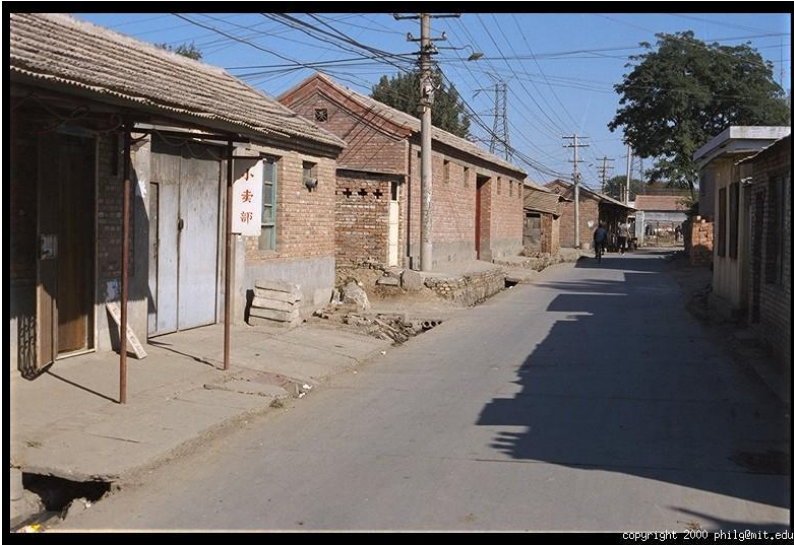


Practical Color Sensing: Bayer Grid



- Estimate RGB at 'G' cells from neighboring values

Color Image



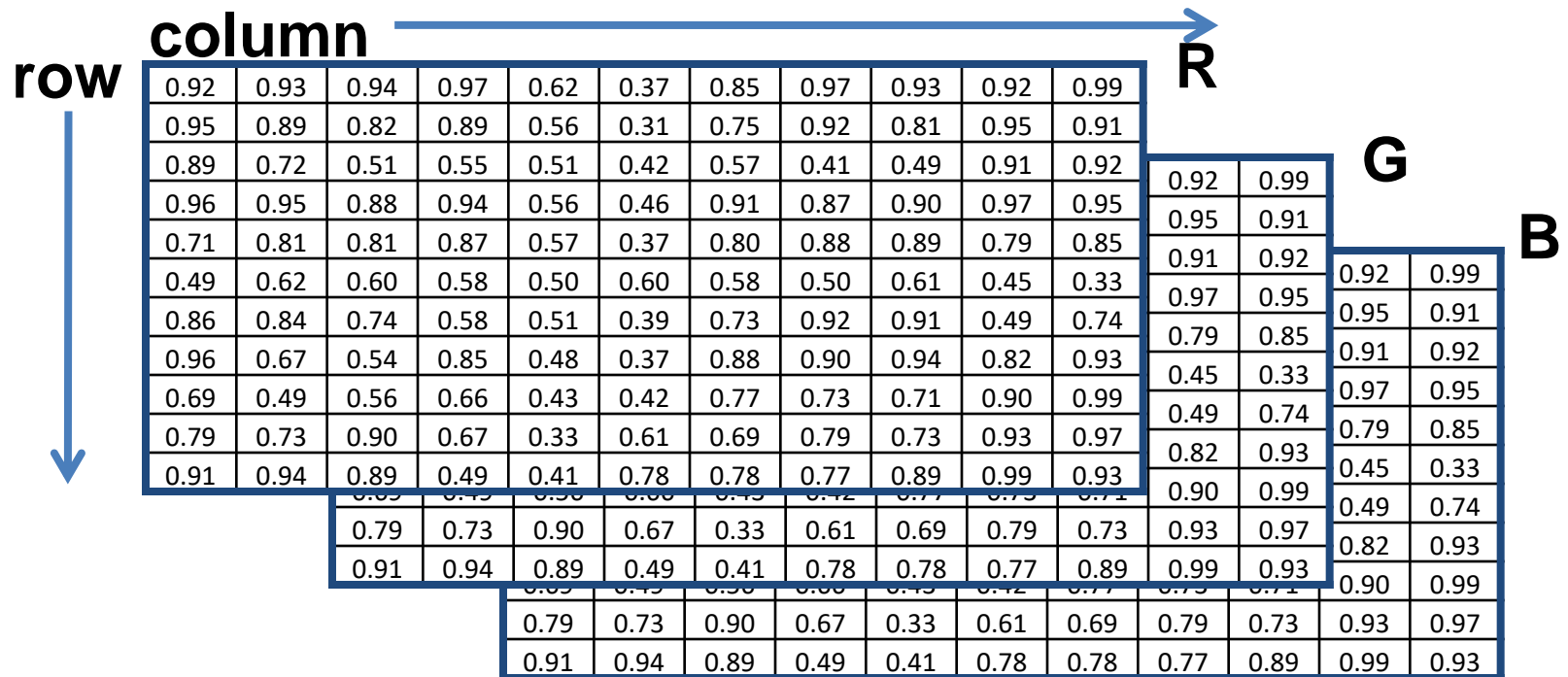
Images in Matlab

- Images represented as a matrix
- Suppose we have a NxM RGB image called “im”
 - $im(1,1,1)$ = top-left pixel value in R-channel
 - $im(y, x, b)$ = y pixels down, x pixels to right in the b^{th} channel
 - $im(N, M, 3)$ = bottom-right pixel in B-channel

row	column										G		B		
1	0.92	0.93	0.94	0.97	0.62	0.37	0.85	0.97	0.93	0.92	0.99	0.92	0.99	0.92	0.99
2	0.95	0.89	0.82	0.89	0.56	0.31	0.75	0.92	0.81	0.95	0.91	0.95	0.91	0.95	0.91
3	0.89	0.72	0.51	0.55	0.51	0.42	0.57	0.41	0.49	0.91	0.92	0.91	0.92	0.91	0.92
4	0.96	0.95	0.88	0.94	0.56	0.46	0.91	0.87	0.90	0.97	0.95	0.97	0.95	0.97	0.95
5	0.71	0.81	0.81	0.87	0.57	0.37	0.80	0.88	0.89	0.79	0.85	0.79	0.85	0.79	0.85
6	0.49	0.62	0.60	0.58	0.50	0.60	0.58	0.50	0.61	0.45	0.33	0.45	0.33	0.45	0.33
7	0.86	0.84	0.74	0.58	0.51	0.39	0.73	0.92	0.91	0.49	0.74	0.49	0.74	0.49	0.74
8	0.96	0.67	0.54	0.85	0.48	0.37	0.88	0.90	0.94	0.82	0.93	0.82	0.93	0.82	0.93
9	0.69	0.49	0.56	0.66	0.43	0.42	0.77	0.73	0.71	0.90	0.99	0.90	0.99	0.90	0.99
10	0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97	0.93	0.97	0.93	0.97
	0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93	0.99	0.93	0.99	0.93
	0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97	0.93	0.97	0.93	0.97
	0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93	0.99	0.93	0.99	0.93
	0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97	0.93	0.97	0.93	0.97
	0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93	0.99	0.93	0.99	0.93

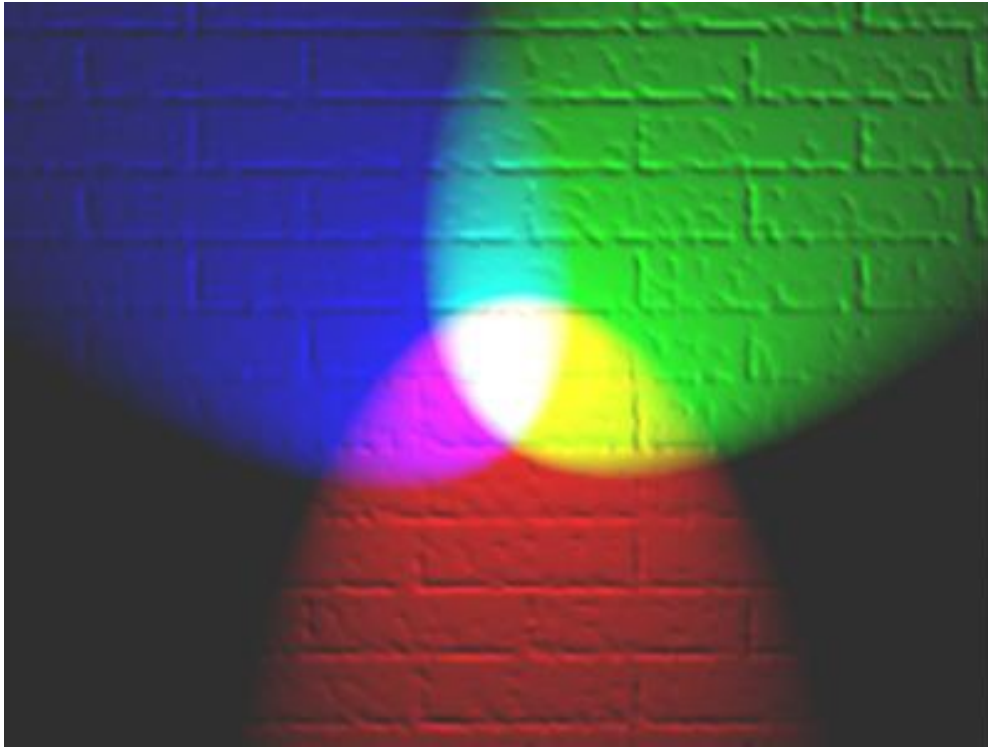
Images in ~~Matlab~~ Python

- Images represented as a matrix
- Suppose we have a NxM RGB image called “im”
 - $im(0,0,0)$ = top-left pixel value in R-channel
 - $im(y, x, b)$ = y pixels down, x pixels to right in the b^{th} channel
 - $im(N-1, M-1, 2)$ = bottom-right pixel in B-channel



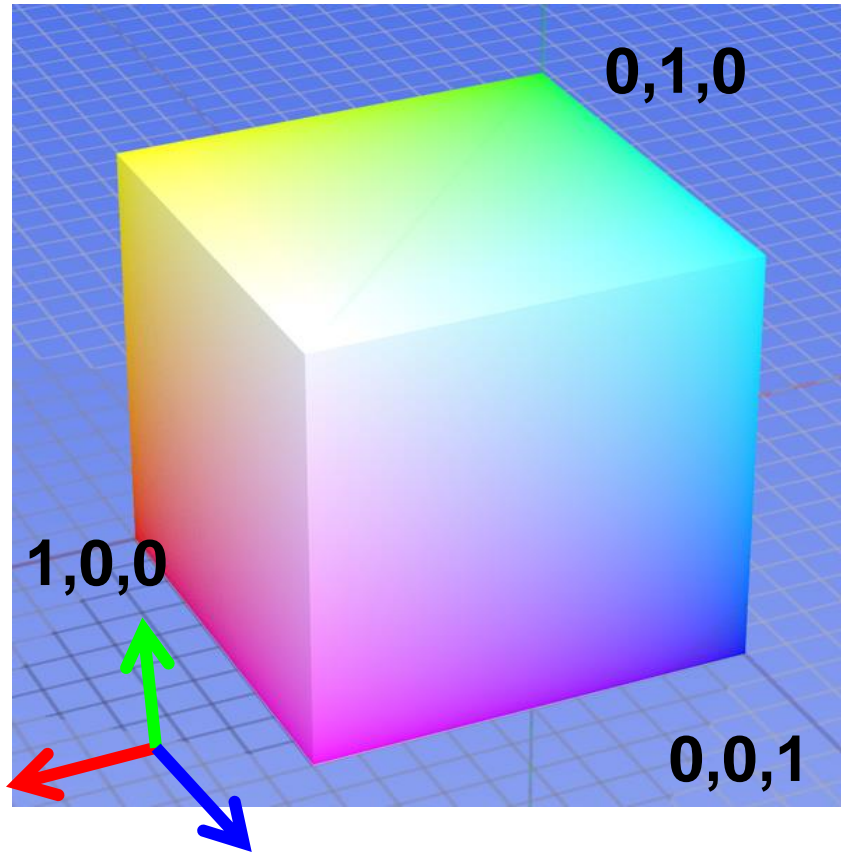
Color spaces

- How can we represent color?



Color spaces: RGB

Default color space



R
(G=0,B=0)



G
(R=0,B=0)



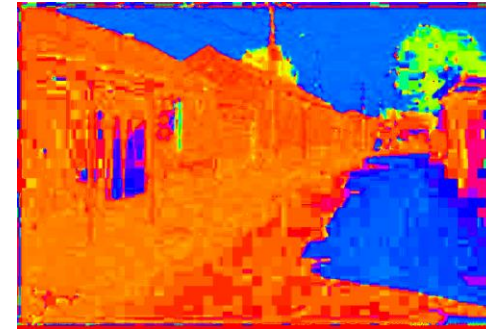
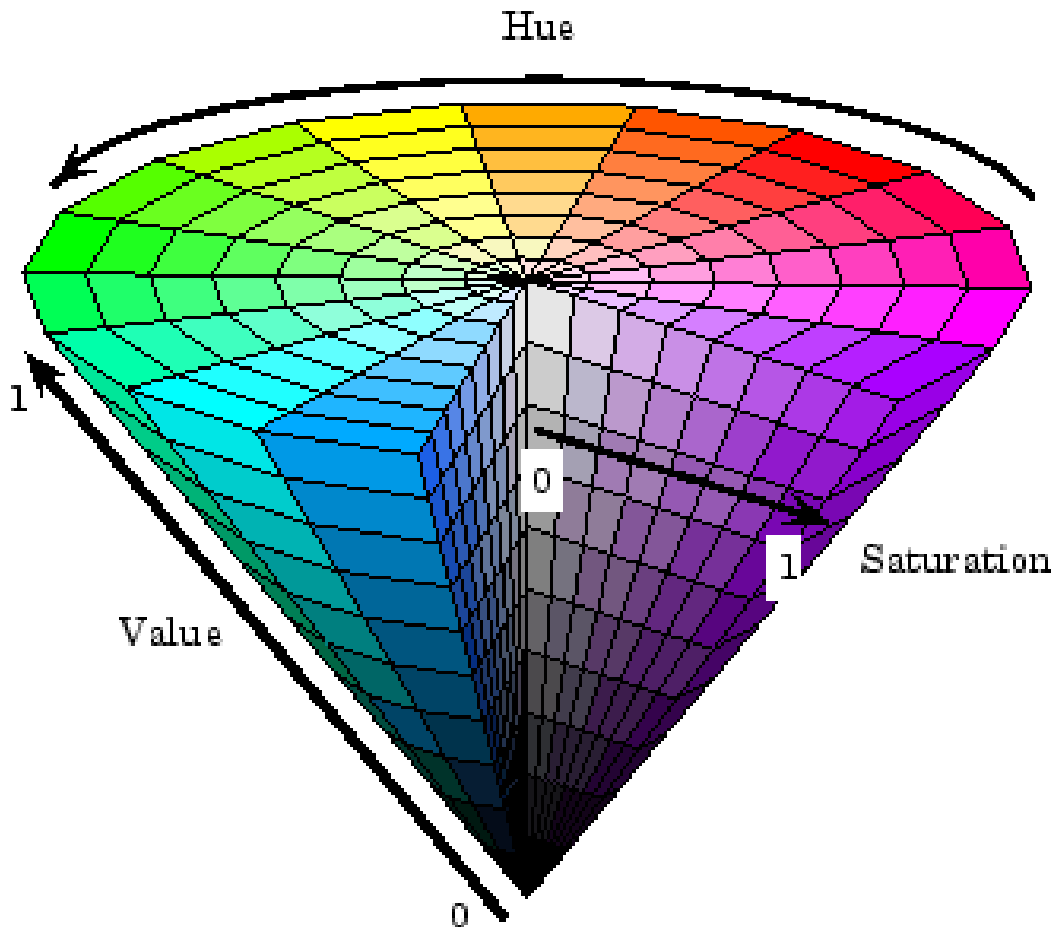
B
(R=0,G=0)

Some drawbacks

- Strongly correlated channels
- Non-perceptual

Color spaces: HSV

Intuitive color space



H
(S=1,V=1)



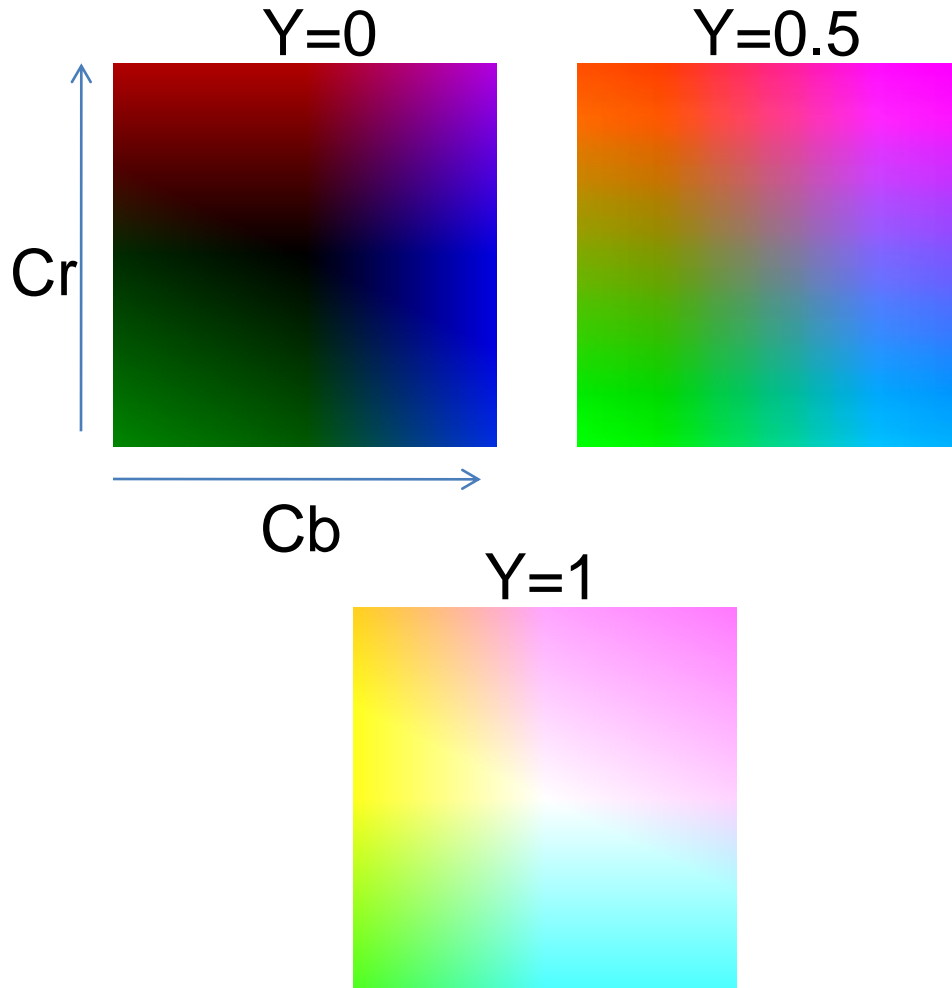
S
(H=1,V=1)



V
(H=1,S=0)

Color spaces: YCbCr

Fast to compute, good for compression, used by TV



Y
(Cb=0.5,Cr=0.5)



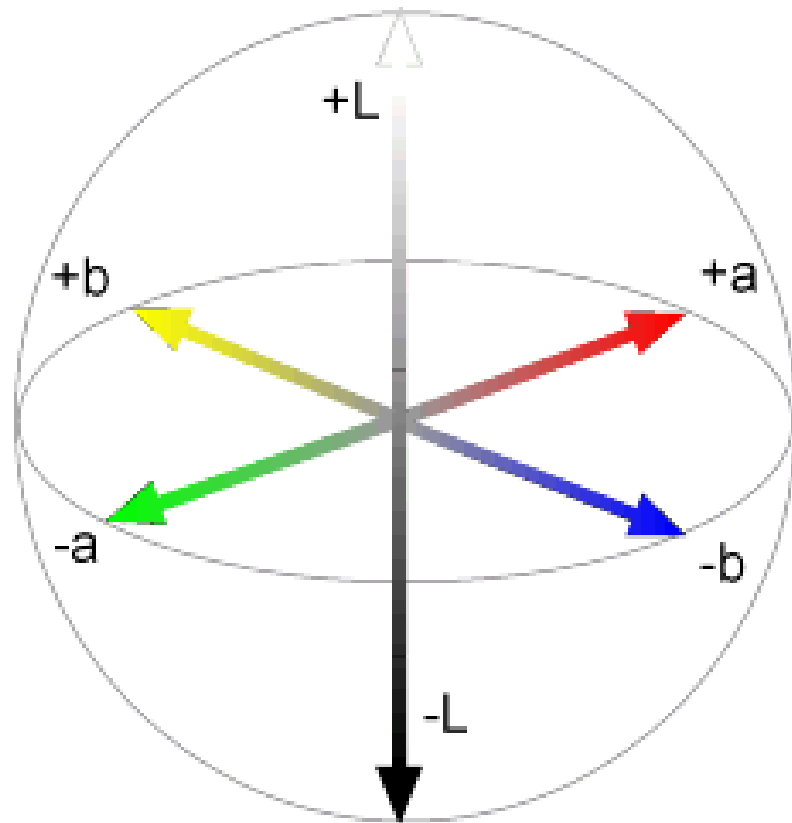
Cb
(Y=0.5,Cr=0.5)



Cr
(Y=0.5,Cb=0.5)

Color spaces: L*a*b*

“Perceptually uniform”* color space



L
(a=0,b=0)



a
(L=65,b=0)



b
(L=65,a=0)

If you had to choose, would you rather go without luminance or chrominance?

If you had to choose, would you rather go
without **luminance** or chrominance?

Most information in intensity



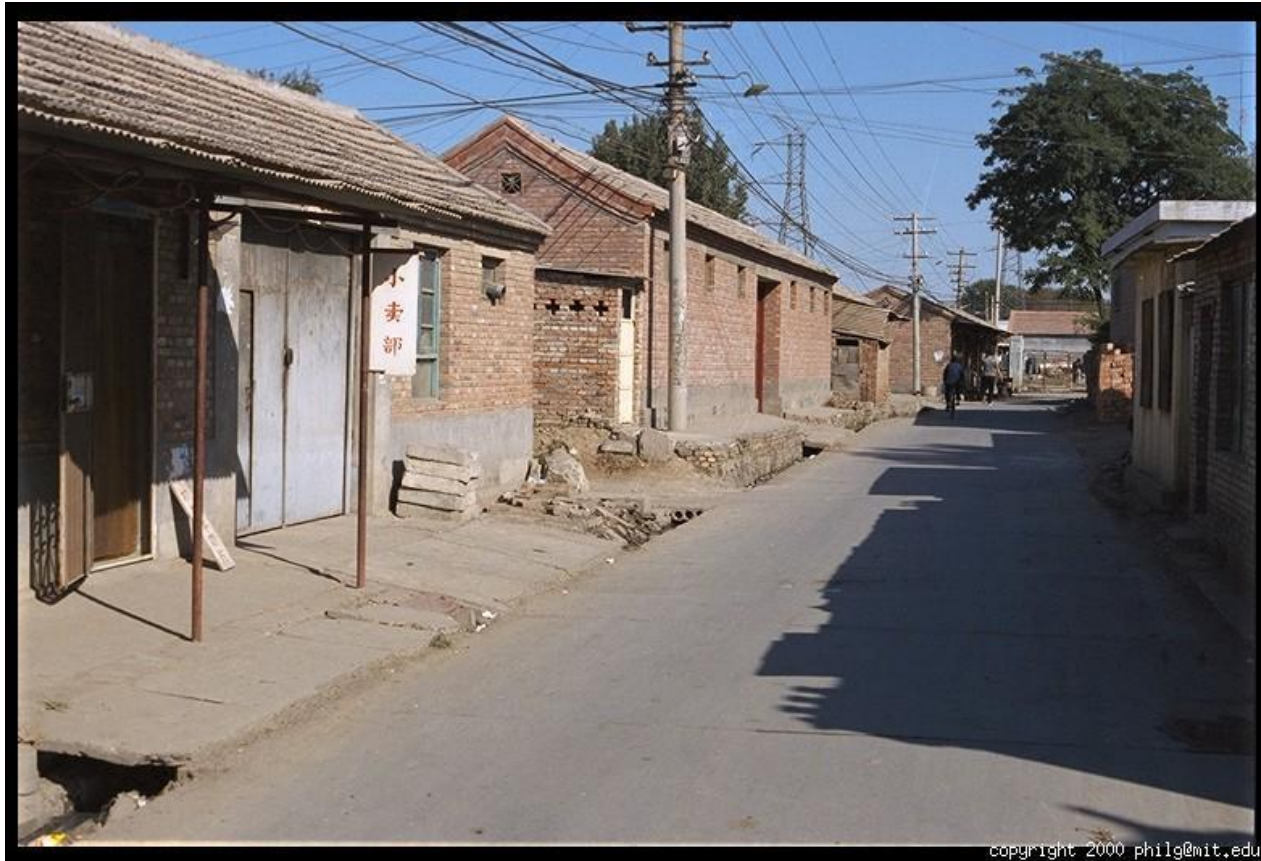
Only color shown – constant intensity

Most information in intensity



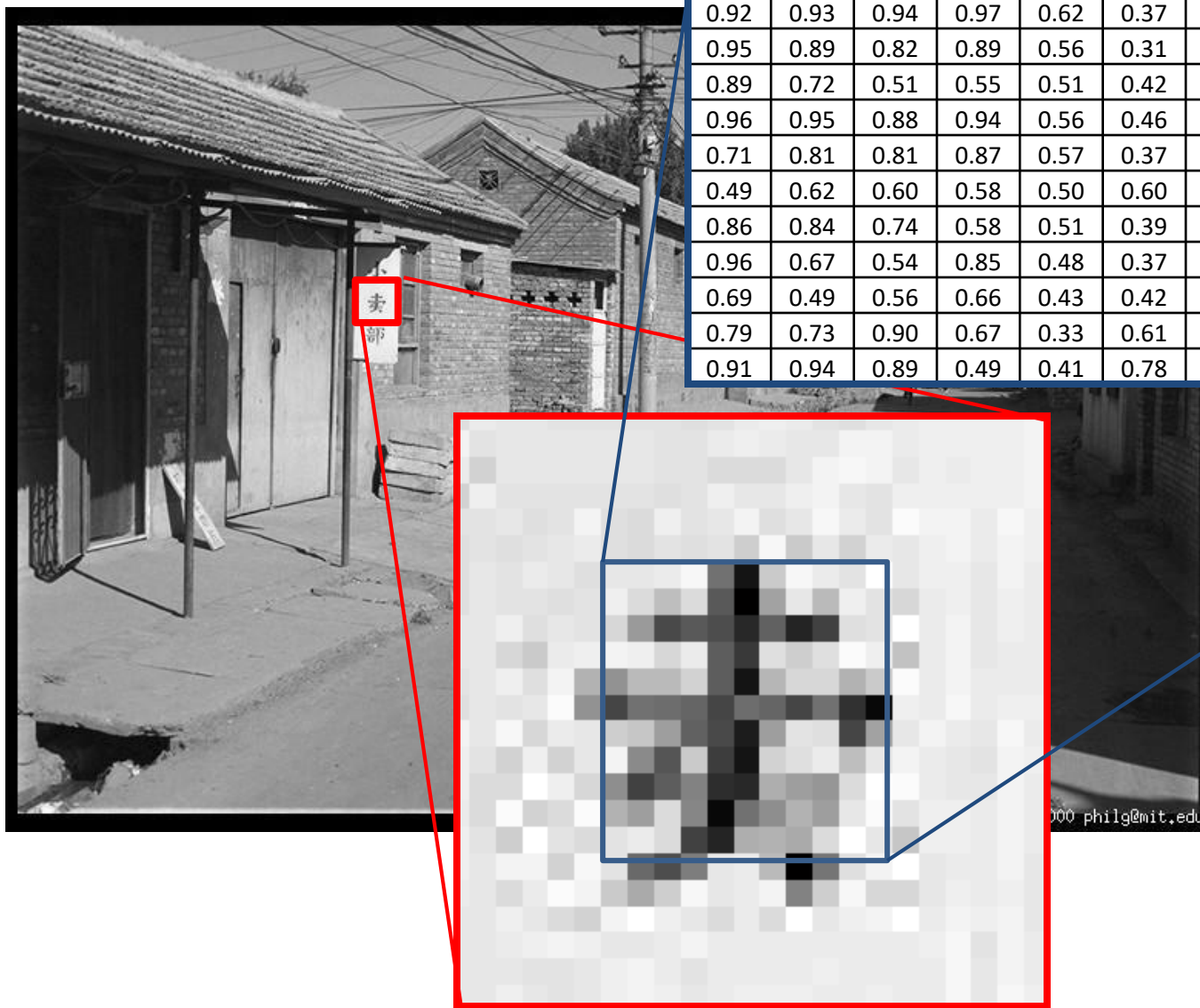
Only intensity shown – constant color

Most information in intensity



Original image

Back to grayscale intensity



0.92	0.93	0.94	0.97	0.62	0.37	0.85	0.97	0.93	0.92	0.99
0.95	0.89	0.82	0.89	0.56	0.31	0.75	0.92	0.81	0.95	0.91
0.89	0.72	0.51	0.55	0.51	0.42	0.57	0.41	0.49	0.91	0.92
0.96	0.95	0.88	0.94	0.56	0.46	0.91	0.87	0.90	0.97	0.95
0.71	0.81	0.81	0.87	0.57	0.37	0.80	0.88	0.89	0.79	0.85
0.49	0.62	0.60	0.58	0.50	0.60	0.58	0.50	0.61	0.45	0.33
0.86	0.84	0.74	0.58	0.51	0.39	0.73	0.92	0.91	0.49	0.74
0.96	0.67	0.54	0.85	0.48	0.37	0.88	0.90	0.94	0.82	0.93
0.69	0.49	0.56	0.66	0.43	0.42	0.77	0.73	0.71	0.90	0.99
0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97
0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93

Upcoming classes: three views of filtering

- Image filters in spatial domain
 - Filter is a mathematical operation of a grid of numbers
 - Smoothing, sharpening, measuring texture
- Image filters in the frequency domain
 - Filtering is a way to modify the frequencies of images
 - Denoising, sampling, image compression
- Templates and Image Pyramids
 - Filtering is a way to match a template to the image
 - Detection, coarse-to-fine registration

Image filtering

- Image filtering: compute function of local neighborhood at each position
- Really important!
 - Enhance images
 - Denoise, resize, increase contrast, etc.
 - Extract information from images
 - Texture, edges, distinctive points, etc.
 - Detect patterns
 - Template matching
 - Deep Convolutional Networks

Example: box filter

$g[\cdot, \cdot]$

	1	1	1
1	1	1	1
9	1	1	1

Image filtering

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	0	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

$h[\cdot, \cdot]$

$$h[m, n] = \sum_{k, l} g[k, l] f[m+k, n+l]$$

Image filtering

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot, \cdot]$

	0	10							

$$h[m, n] = \sum_{k, l} g[k, l] f[m+k, n+l]$$

Image filtering

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot, \cdot]$

	0	10	20						

$$h[m, n] = \sum_{k, l} g[k, l] f[m+k, n+l]$$

Image filtering

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot, \cdot]$

	0	10	20	30					

$$h[m, n] = \sum_{k, l} g[k, l] f[m+k, n+l]$$

Image filtering

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot, \cdot]$

	0	10	20	30	30				

$$h[m, n] = \sum_{k, l} g[k, l] f[m+k, n+l]$$

Image filtering

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$$f[\cdot, \cdot]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h[\cdot, \cdot]$$

	0	10	20	30	30				

$$h[m, n] = \sum_{k, l} g[k, l] f[m+k, n+l]$$

Image filtering

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot, \cdot]$

	0	10	20	30	30				
							?		
					50				

$$h[m, n] = \sum_{k, l} g[k, l] f[m+k, n+l]$$

Image filtering

$$g[\cdot, \cdot] = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$f[\cdot, \cdot]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h[\cdot, \cdot]$$

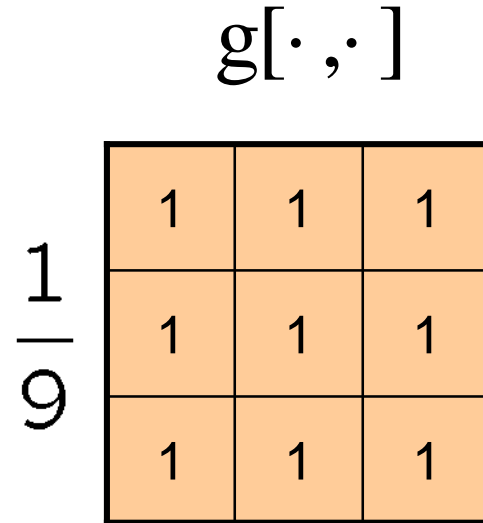
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$h[m, n] = \sum_{k, l} g[k, l] f[m+k, n+l]$$

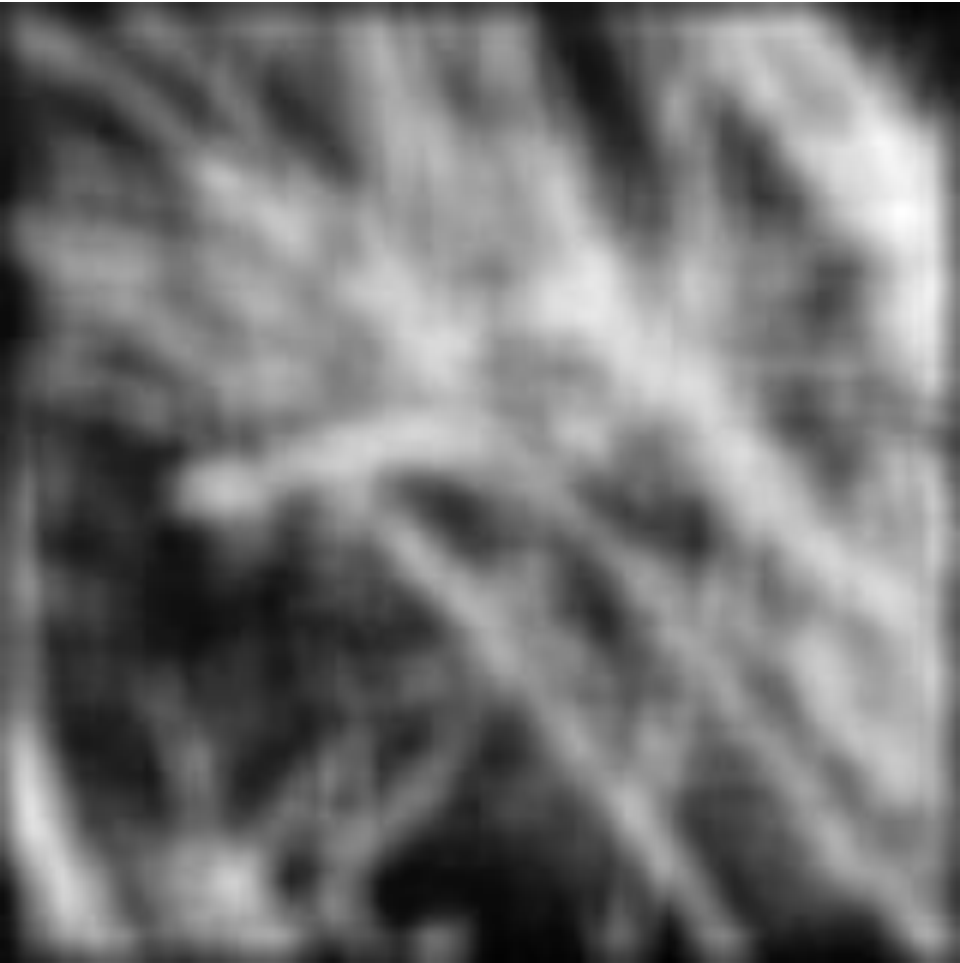
Box Filter

What does it do?

- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)



Smoothing with box filter



Practice with linear filters



Original

0	0	0
0	1	0
0	0	0

?

Practice with linear filters



Original

0	0	0
0	1	0
0	0	0



Filtered
(no change)

Practice with linear filters



Original

0	0	0
0	0	1
0	0	0

?

Practice with linear filters



Original

0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Practice with linear filters



Original

0	0	0
0	2	0
0	0	0

−

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

?

(Note that filter sums to 1)

Practice with linear filters



Original

0	0	0
0	2	0
0	0	0

−

$\frac{1}{9}$

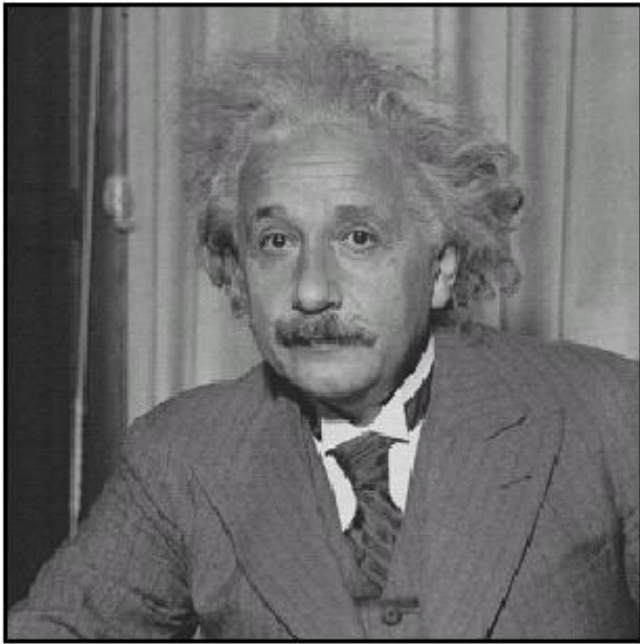
1	1	1
1	1	1
1	1	1



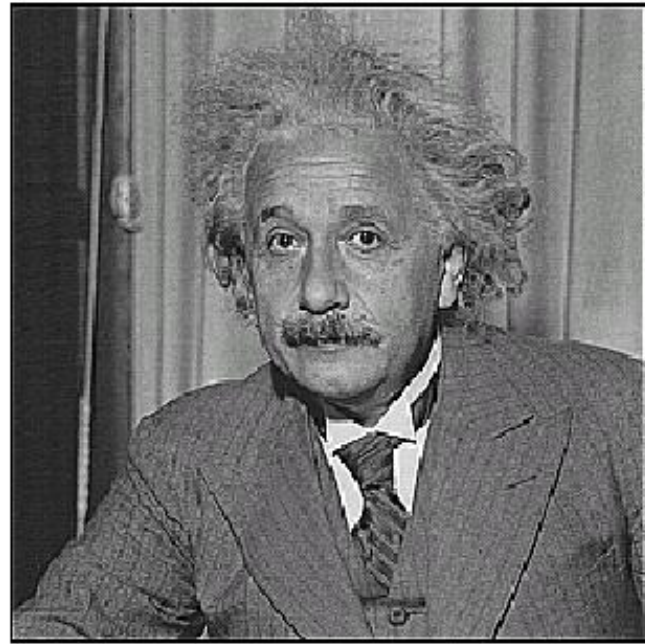
Sharpening filter

- Accentuates differences with local average

Sharpening

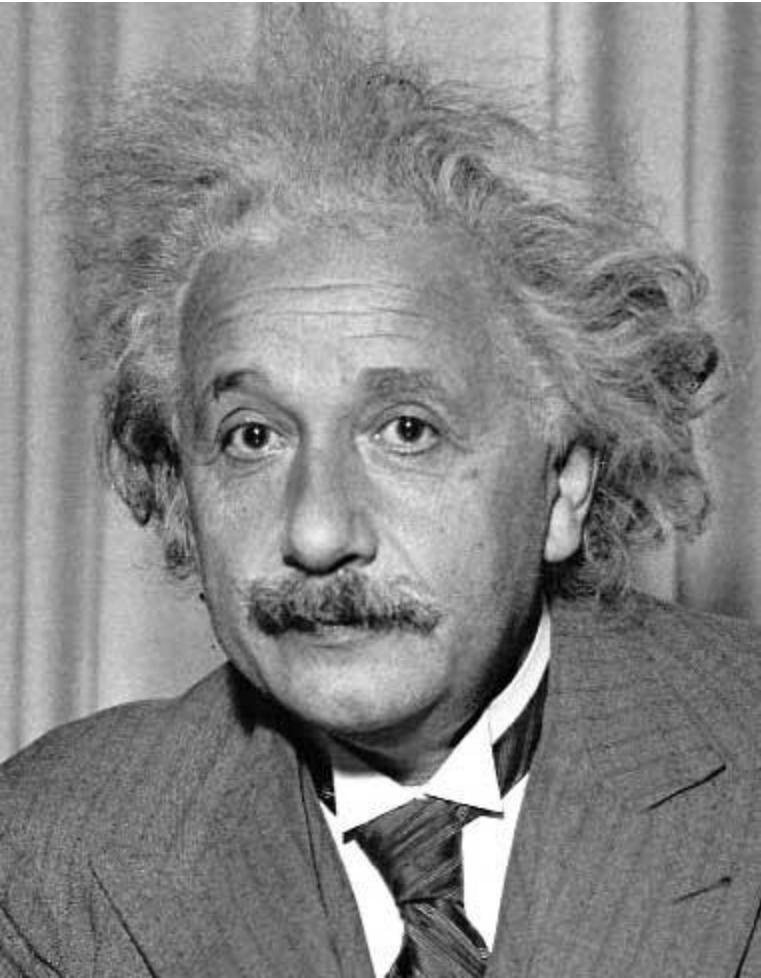


before



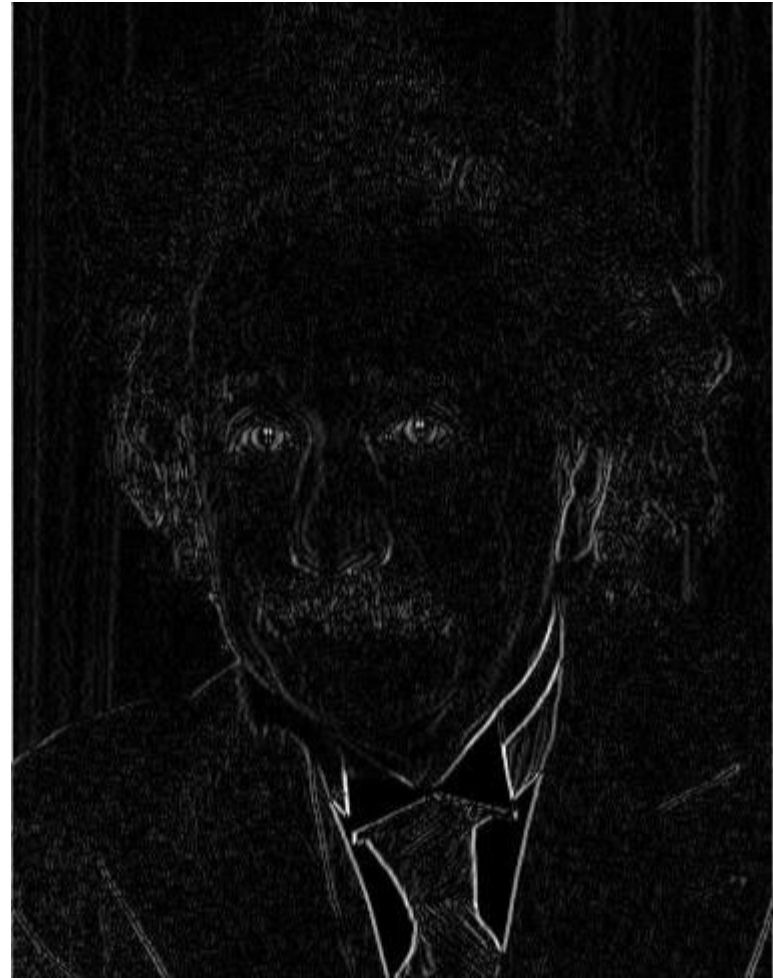
after

Other filters



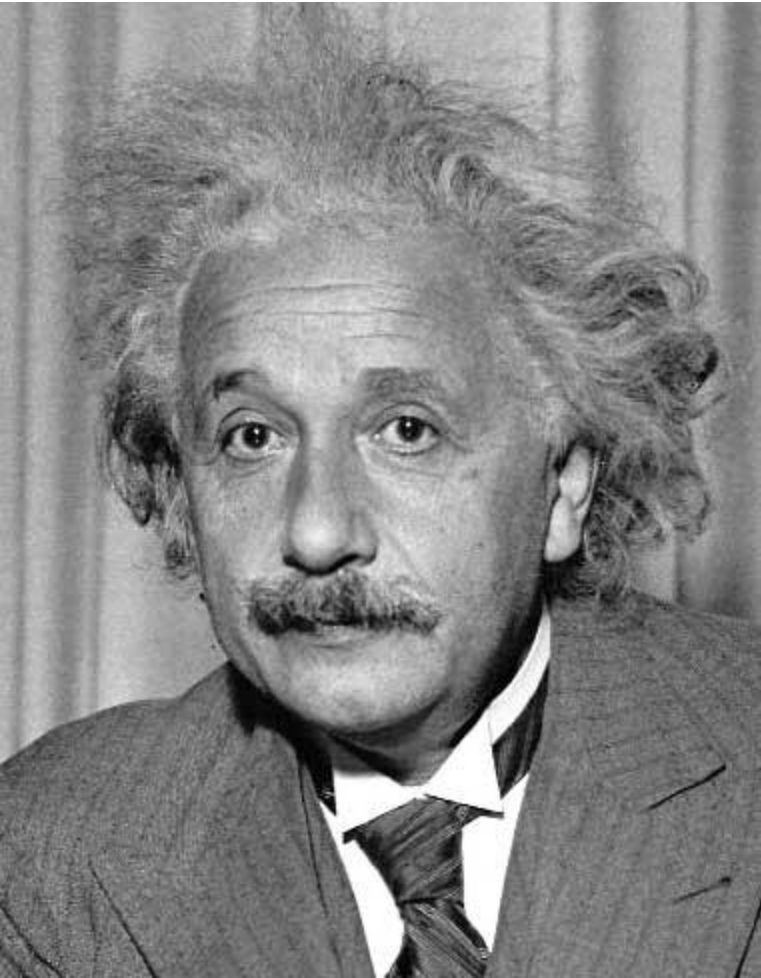
1	0	-1
2	0	-2
1	0	-1

Sobel



Vertical Edge
(absolute value)

Other filters



1	2	1
0	0	0
-1	-2	-1

Sobel



Horizontal Edge
(absolute value)

Filtering vs. Convolution

- 2d filtering

– `h=filter2(f, I);` or

`h=imfilter(I, f);`

$$h[m,n] = \sum_{k,l} f[k,l] I[m+k, n+l]$$

- 2d convolution

– `h=conv2(f, I);`

$$h[m,n] = \sum_{k,l} f[k,l] I[m-k, n-l]$$

Key properties of linear filters

Linearity:

$$\text{imfilter}(I, f_1 + f_2) = \text{imfilter}(I, f_1) + \text{imfilter}(I, f_2)$$

Shift invariance: same behavior regardless of pixel location

$$\text{imfilter}(I, \text{shift}(f)) = \text{shift}(\text{imfilter}(I, f))$$

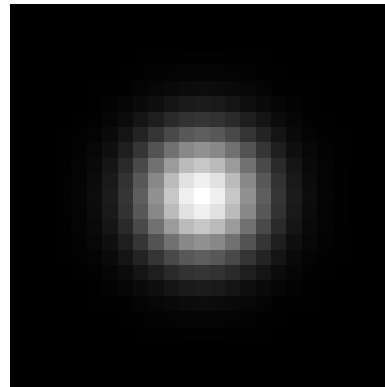
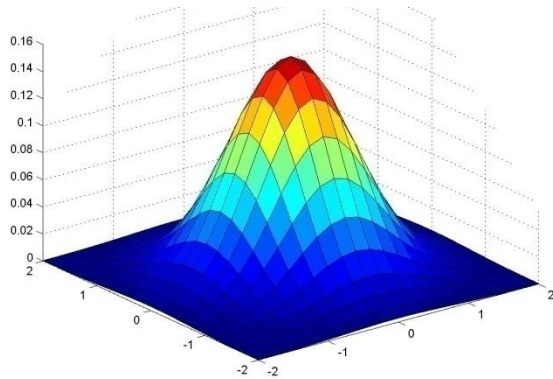
Any linear, shift-invariant operator can be represented as a convolution

More properties

- Commutative: $a * b = b * a$
 - Conceptually no difference between filter and signal
 - But particular filtering implementations might break this equality
- Associative: $a * (b * c) = (a * b) * c$
 - Often apply several filters one after another: $((a * b_1) * b_2) * b_3$
 - This is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$
- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$
- Scalars factor out: $ka * b = a * kb = k(a * b)$
- Identity: unit impulse $e = [0, 0, 1, 0, 0]$,
 $a * e = a$

Important filter: Gaussian

- Weight contributions of neighboring pixels by nearness

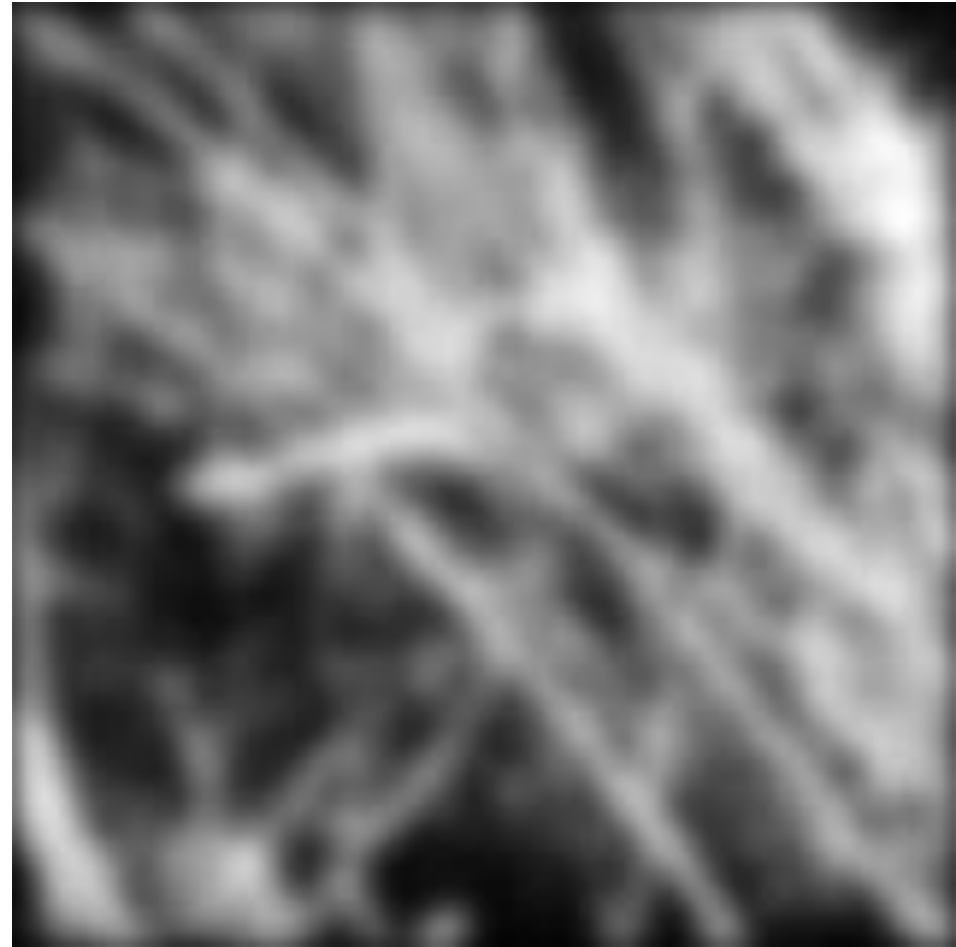


0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

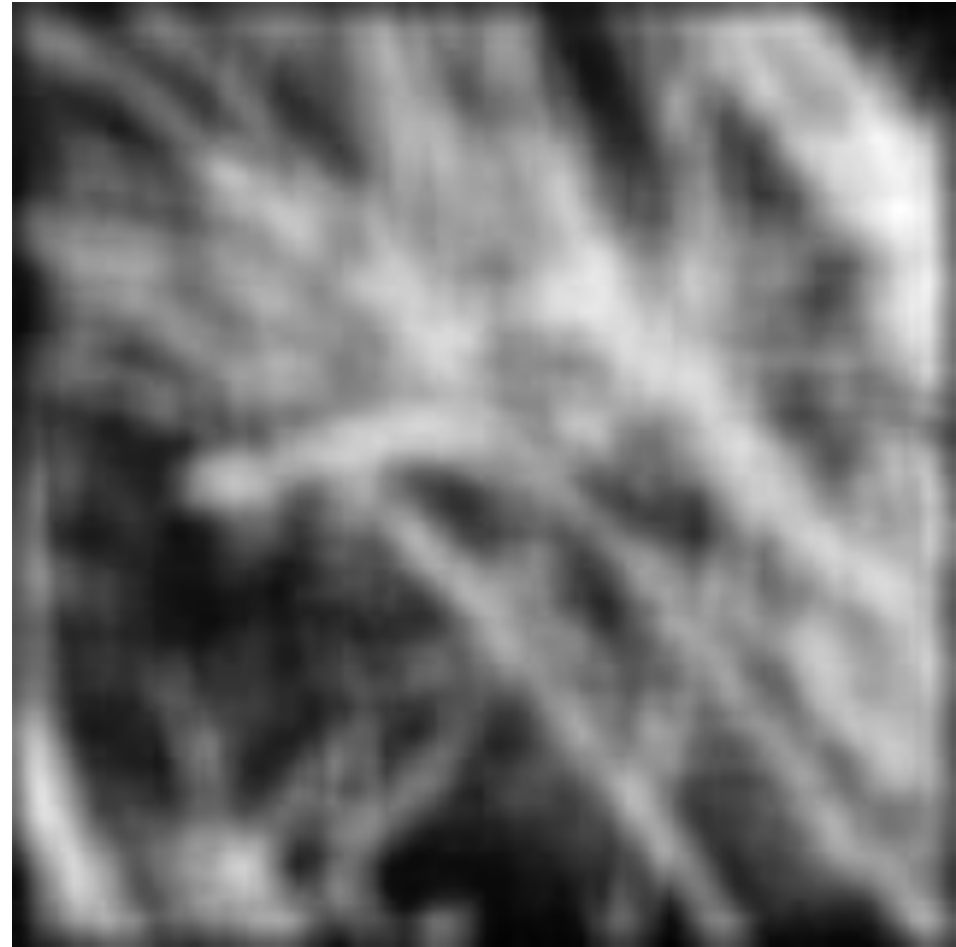
5 x 5, $\sigma = 1$

$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Smoothing with Gaussian filter



Smoothing with box filter



Gaussian filters

- Remove “high-frequency” components from the image (low-pass filter)
 - Images become more smooth
- Convolution with self is another Gaussian
 - So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
 - Convoluting two times with Gaussian kernel of width σ is same as convoluting once with kernel of width $\sigma\sqrt{2}$
- *Separable* kernel
 - Factors into product of two 1D Gaussians

Separability of the Gaussian filter

$$\begin{aligned} G_{\sigma}(x, y) &= \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right) \right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{y^2}{2\sigma^2}\right) \right) \end{aligned}$$

The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

In this case, the two functions are the (identical) 1D Gaussian

Separability example

2D convolution
(center location only)

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 2 & 3 & 3 \\ \hline 3 & 5 & 5 \\ \hline 4 & 4 & 6 \\ \hline \end{array}$$

The filter factors
into a product of 1D
filters:

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} = \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 1 \\ \hline \end{array} \times \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline \end{array}$$

Perform convolution
along rows:

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 2 & 3 & 3 \\ \hline 3 & 5 & 5 \\ \hline 4 & 4 & 6 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline & 11 & \\ \hline & 18 & \\ \hline & 18 & \\ \hline \end{array}$$

Followed by convolution
along the remaining column:

Separability

- Why is separability useful in practice?

Practical matters

How big should the filter be?

- Values at edges should be near zero
- Rule of thumb for Gaussian: set filter half-width to about 3σ

Practical matters

- What about near the edge?
 - the filter window falls off the edge of the image
 - need to extrapolate
 - methods:
 - clip filter (black)
 - wrap around
 - copy edge
 - reflect across edge

