

Image Filtering



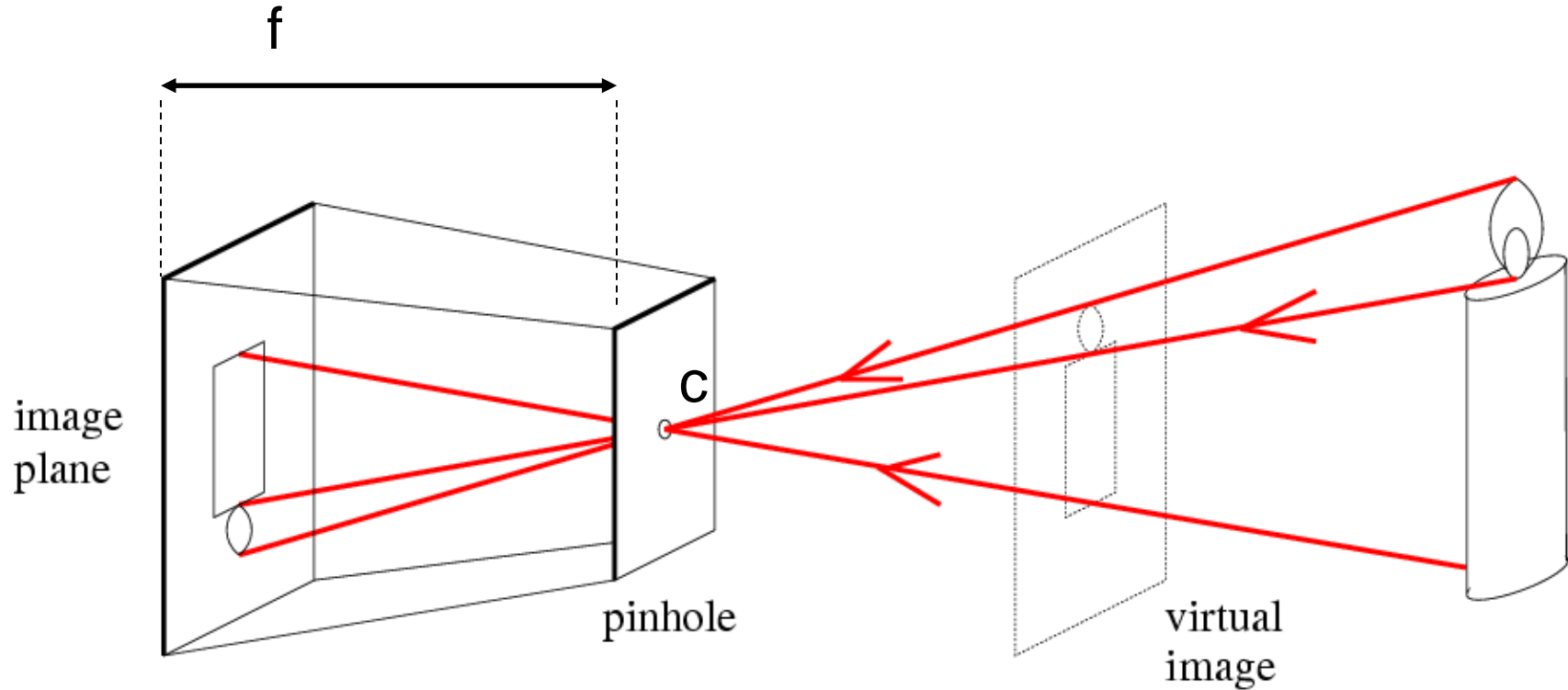
Computer Vision

James Hays

Many slides by Derek Hoiem

Recap: Light and Sensors

Pinhole camera

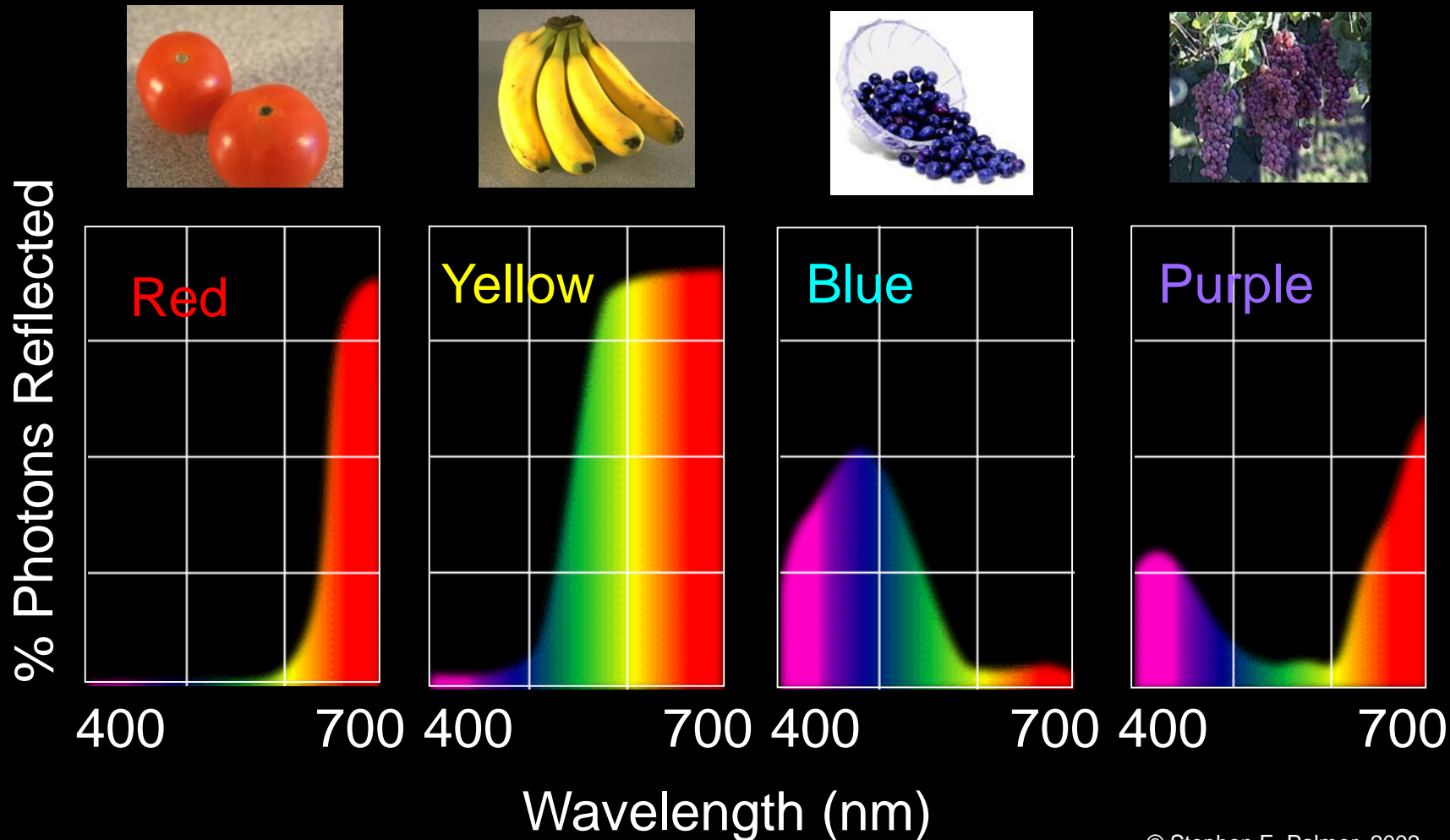


f = focal length

c = center of the camera

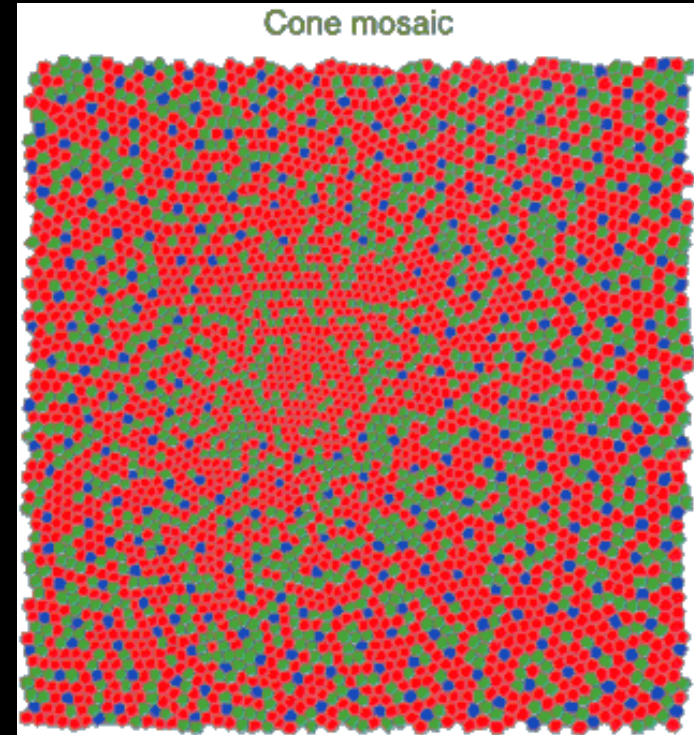
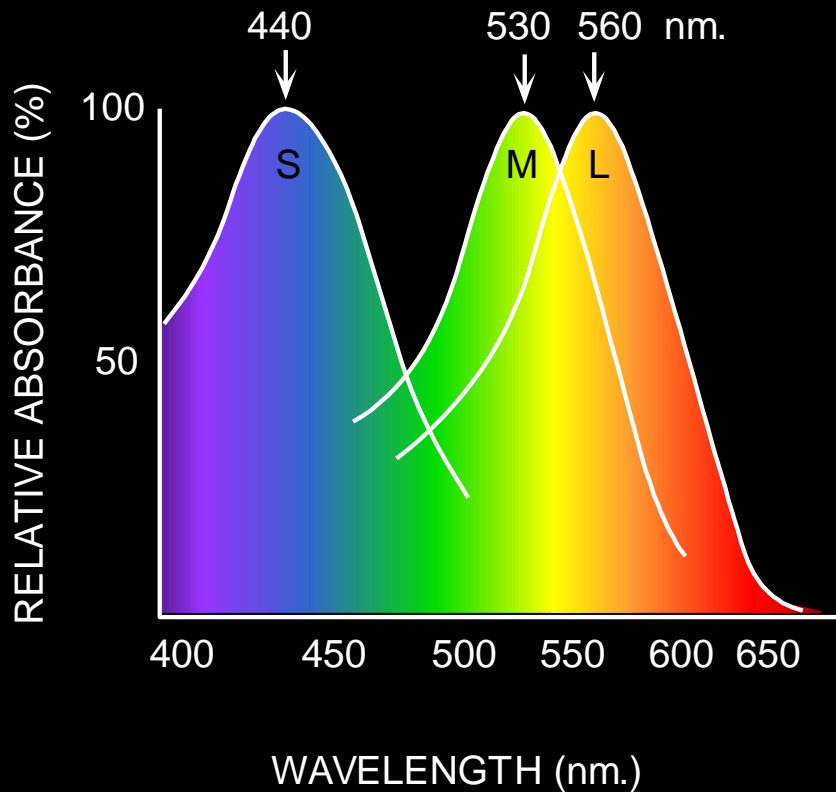
The Physics of Light

Some examples of the reflectance spectra of surfaces



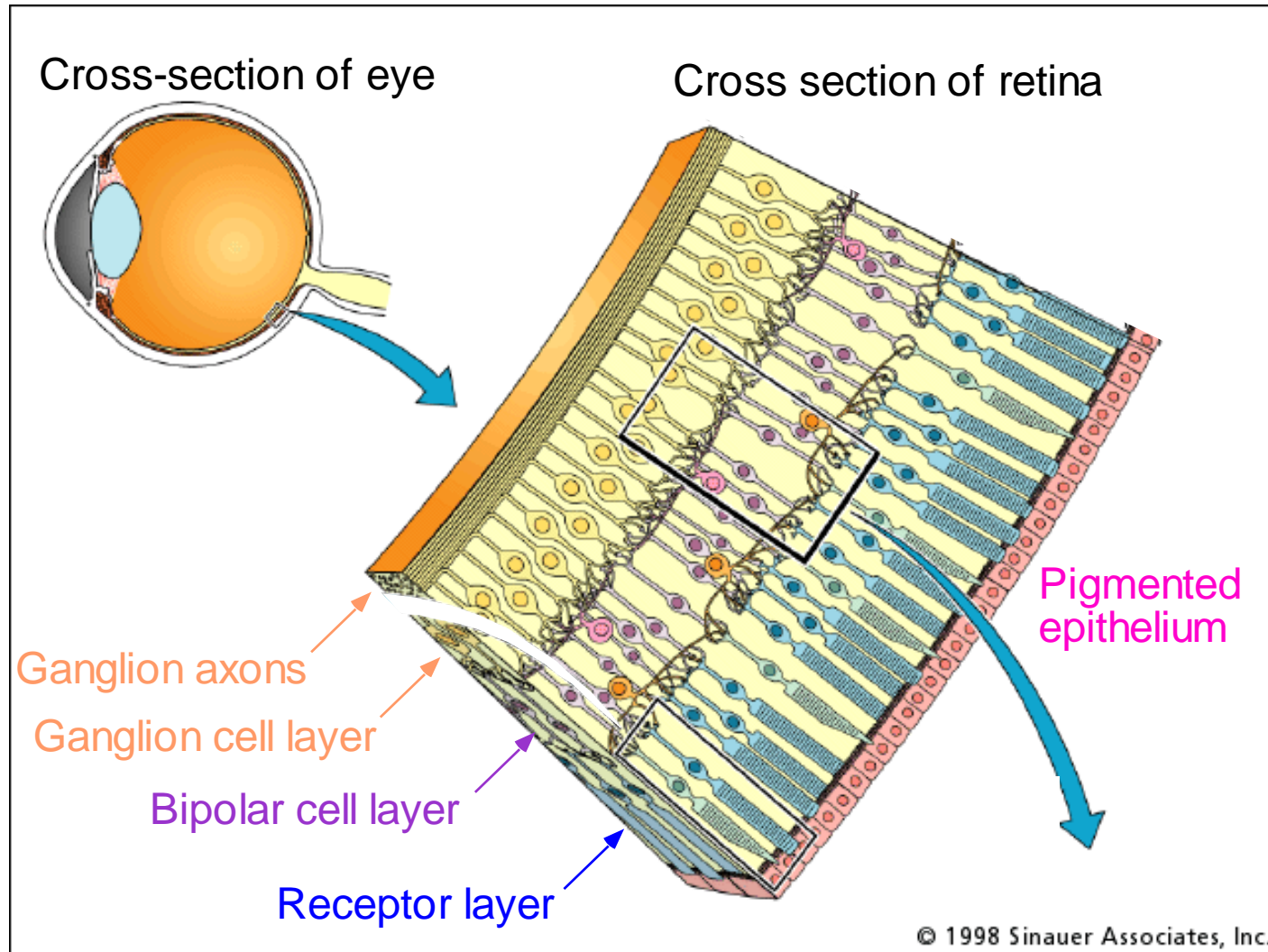
Physiology of Color Vision

Three kinds of cones:

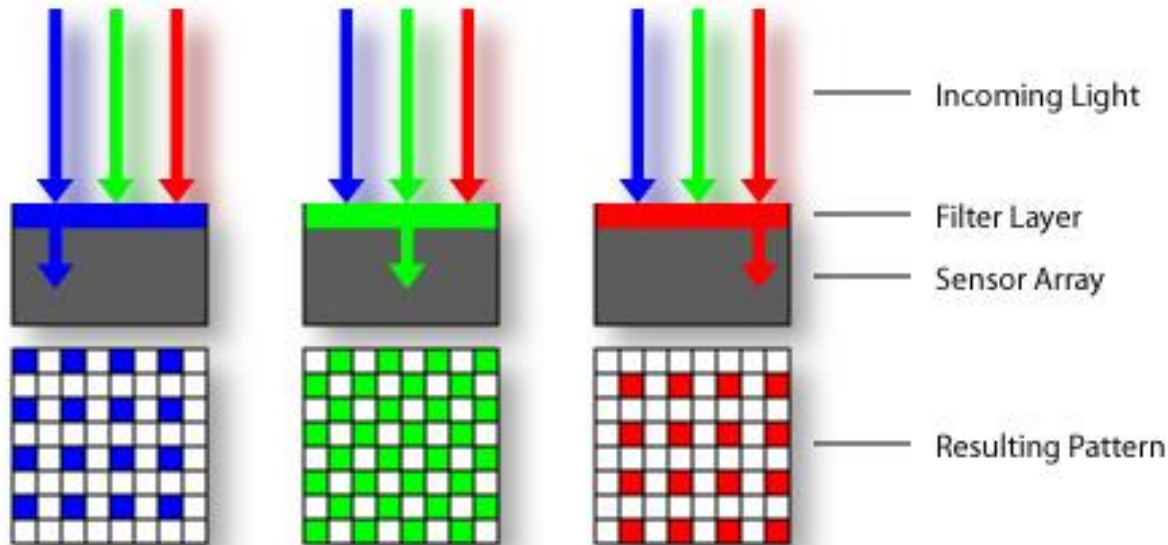
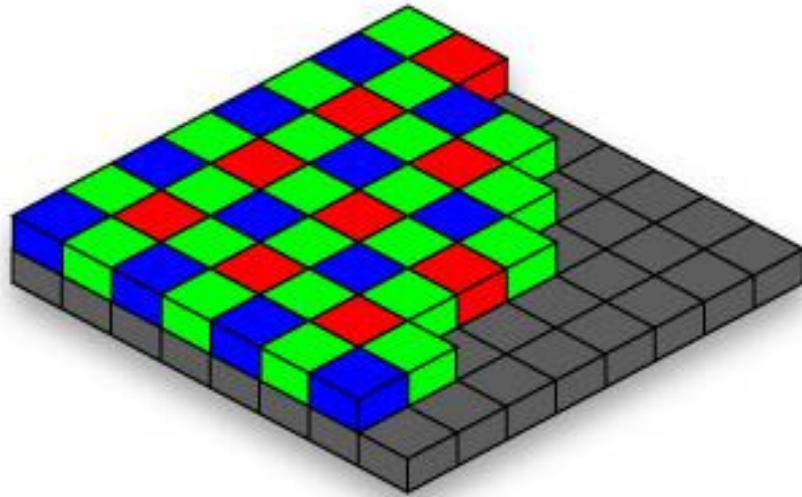


- Why are M and L cones so close?
- Why are there 3?

The Retina

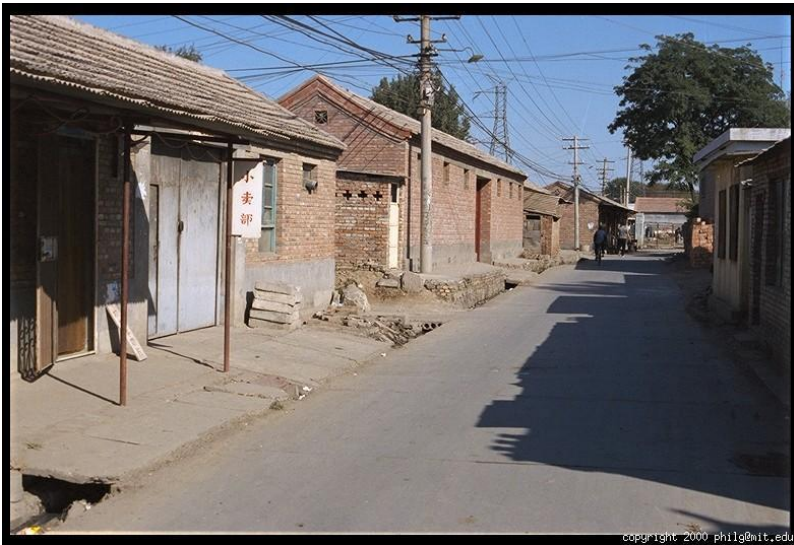


Practical Color Sensing: Bayer Grid



- Estimate RGB at 'G' cells from neighboring values

Color Image

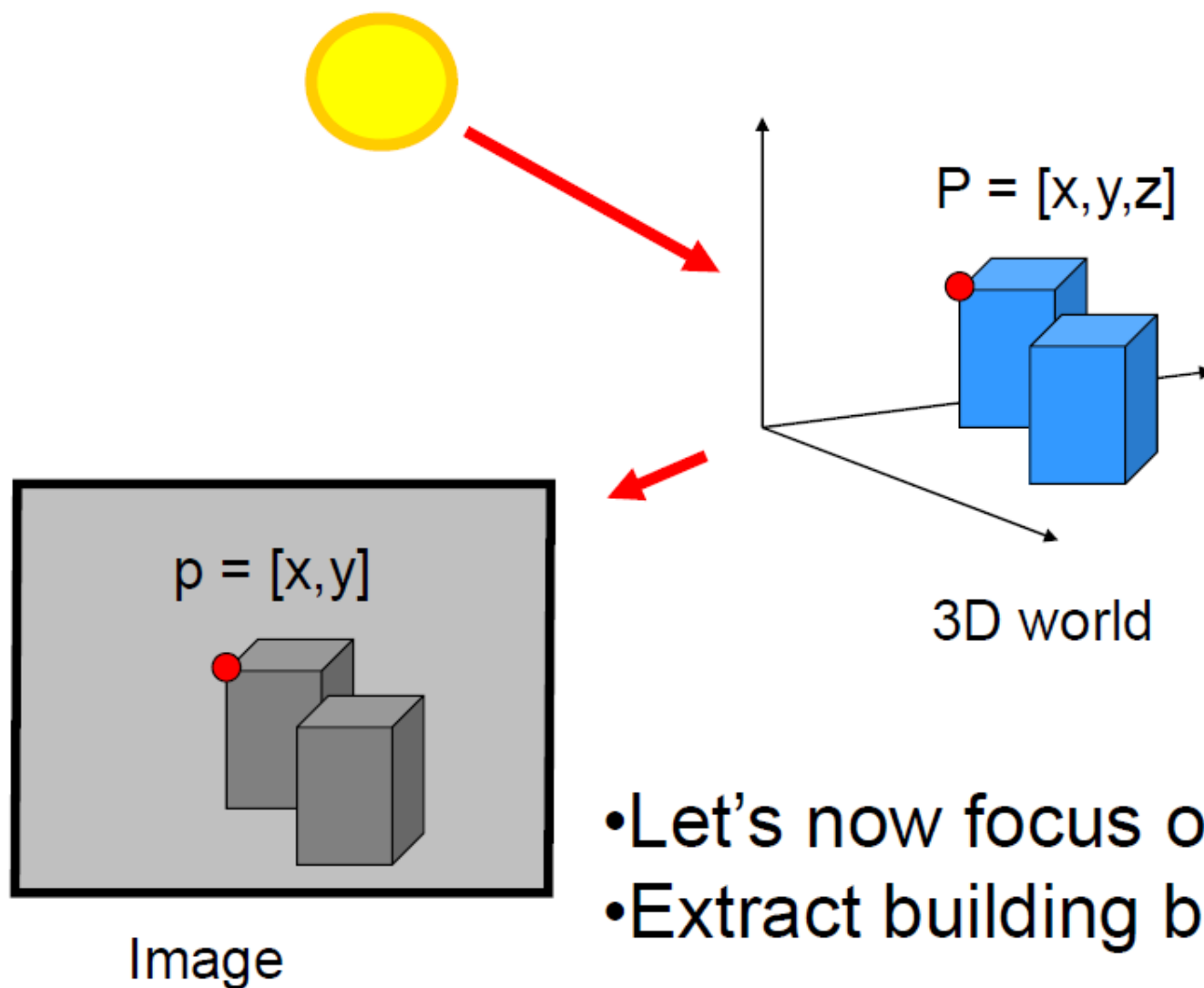


Images in Matlab

- Images represented as a matrix
- Suppose we have a NxM RGB image called “im”
 - $\text{im}(1,1,1)$ = top-left pixel value in R-channel
 - $\text{im}(y, x, b)$ = y pixels down, x pixels to right in the b^{th} channel
 - $\text{im}(N, M, 3)$ = bottom-right pixel in B-channel
- `imread(filename)` returns a uint8 image (values 0 to 255)
 - Convert to double format (values 0 to 1) with `im2double`

row	column	R	G	B
1	1	0.92	0.92	0.92
1	2	0.93	0.99	0.99
1	3	0.94	0.95	0.91
1	4	0.97	0.91	0.92
1	5	0.62	0.97	0.85
1	6	0.37	0.95	0.91
1	7	0.85	0.90	0.99
1	8	0.97	0.97	0.93
1	9	0.93	0.92	0.99
1	10	0.92	0.99	0.91
2	1	0.95	0.95	0.91
2	2	0.89	0.91	0.92
2	3	0.82	0.97	0.95
2	4	0.89	0.95	0.91
2	5	0.56	0.90	0.97
2	6	0.31	0.97	0.95
2	7	0.75	0.99	0.93
2	8	0.92	0.99	0.91
2	9	0.81	0.92	0.92
2	10	0.95	0.95	0.91
3	1	0.89	0.91	0.92
3	2	0.72	0.97	0.95
3	3	0.51	0.95	0.91
3	4	0.55	0.90	0.97
3	5	0.51	0.97	0.95
3	6	0.42	0.99	0.93
3	7	0.57	0.99	0.91
3	8	0.41	0.92	0.92
3	9	0.49	0.97	0.95
3	10	0.91	0.99	0.91
4	1	0.96	0.95	0.91
4	2	0.95	0.97	0.95
4	3	0.88	0.90	0.97
4	4	0.94	0.97	0.95
4	5	0.56	0.99	0.93
4	6	0.46	0.99	0.91
4	7	0.91	0.92	0.92
4	8	0.87	0.97	0.95
4	9	0.90	0.99	0.91
4	10	0.97	0.95	0.91
5	1	0.71	0.91	0.92
5	2	0.81	0.97	0.95
5	3	0.81	0.95	0.91
5	4	0.87	0.90	0.97
5	5	0.57	0.97	0.95
5	6	0.37	0.99	0.93
5	7	0.80	0.99	0.91
5	8	0.88	0.92	0.92
5	9	0.89	0.97	0.95
5	10	0.79	0.99	0.91
6	1	0.49	0.91	0.92
6	2	0.62	0.97	0.95
6	3	0.60	0.95	0.91
6	4	0.58	0.90	0.97
6	5	0.50	0.97	0.95
6	6	0.60	0.99	0.93
6	7	0.58	0.99	0.91
6	8	0.50	0.92	0.92
6	9	0.61	0.97	0.95
6	10	0.45	0.99	0.91
7	1	0.86	0.95	0.91
7	2	0.84	0.97	0.95
7	3	0.74	0.95	0.91
7	4	0.58	0.90	0.97
7	5	0.51	0.97	0.95
7	6	0.39	0.99	0.93
7	7	0.73	0.99	0.91
7	8	0.92	0.92	0.92
7	9	0.91	0.97	0.95
7	10	0.49	0.99	0.91
8	1	0.96	0.91	0.92
8	2	0.67	0.97	0.95
8	3	0.54	0.95	0.91
8	4	0.85	0.90	0.97
8	5	0.48	0.97	0.95
8	6	0.37	0.99	0.93
8	7	0.88	0.99	0.91
8	8	0.90	0.92	0.92
8	9	0.94	0.97	0.95
8	10	0.82	0.99	0.91
9	1	0.69	0.95	0.91
9	2	0.49	0.97	0.95
9	3	0.56	0.95	0.91
9	4	0.66	0.90	0.97
9	5	0.43	0.97	0.95
9	6	0.42	0.99	0.93
9	7	0.77	0.99	0.91
9	8	0.73	0.92	0.92
9	9	0.71	0.97	0.95
9	10	0.90	0.99	0.91
10	1	0.79	0.95	0.91
10	2	0.73	0.97	0.95
10	3	0.90	0.95	0.91
10	4	0.67	0.90	0.97
10	5	0.33	0.97	0.95
10	6	0.61	0.99	0.93
10	7	0.69	0.99	0.91
10	8	0.79	0.92	0.92
10	9	0.73	0.97	0.95
10	10	0.93	0.99	0.91
11	1	0.91	0.95	0.91
11	2	0.94	0.97	0.95
11	3	0.89	0.95	0.91
11	4	0.49	0.90	0.97
11	5	0.41	0.97	0.95
11	6	0.78	0.99	0.93
11	7	0.78	0.99	0.91
11	8	0.77	0.92	0.92
11	9	0.89	0.97	0.95
11	10	0.99	0.99	0.91

From the 3D to 2D



- Let's now focus on 2D
- Extract building blocks

Extract useful building blocks



The big picture...

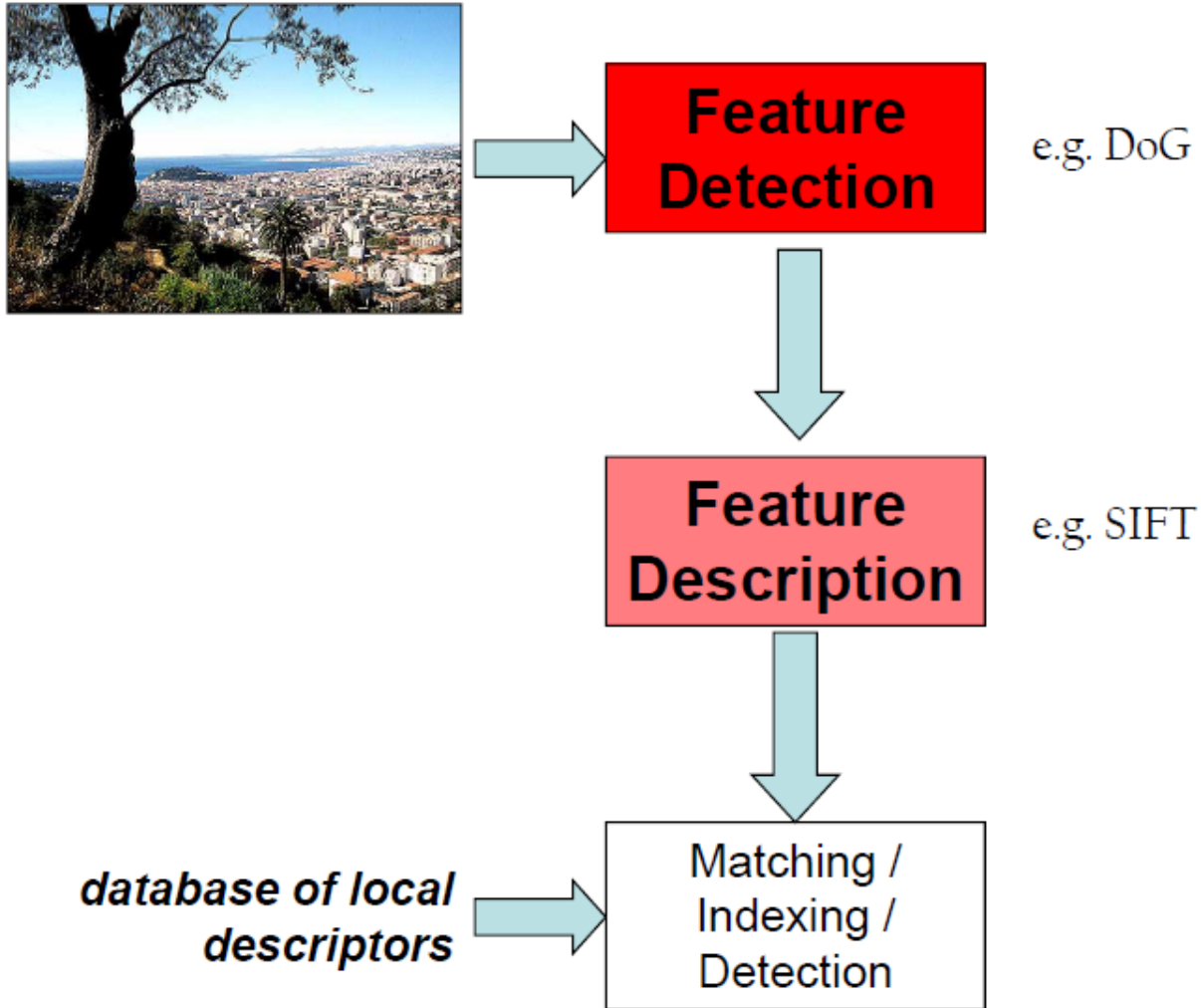


Image Filtering



Computer Vision

James Hays

Many slides by Derek Hoiem

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

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

1	1	1
1	1	1
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

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

$f[\cdot, \cdot]$

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

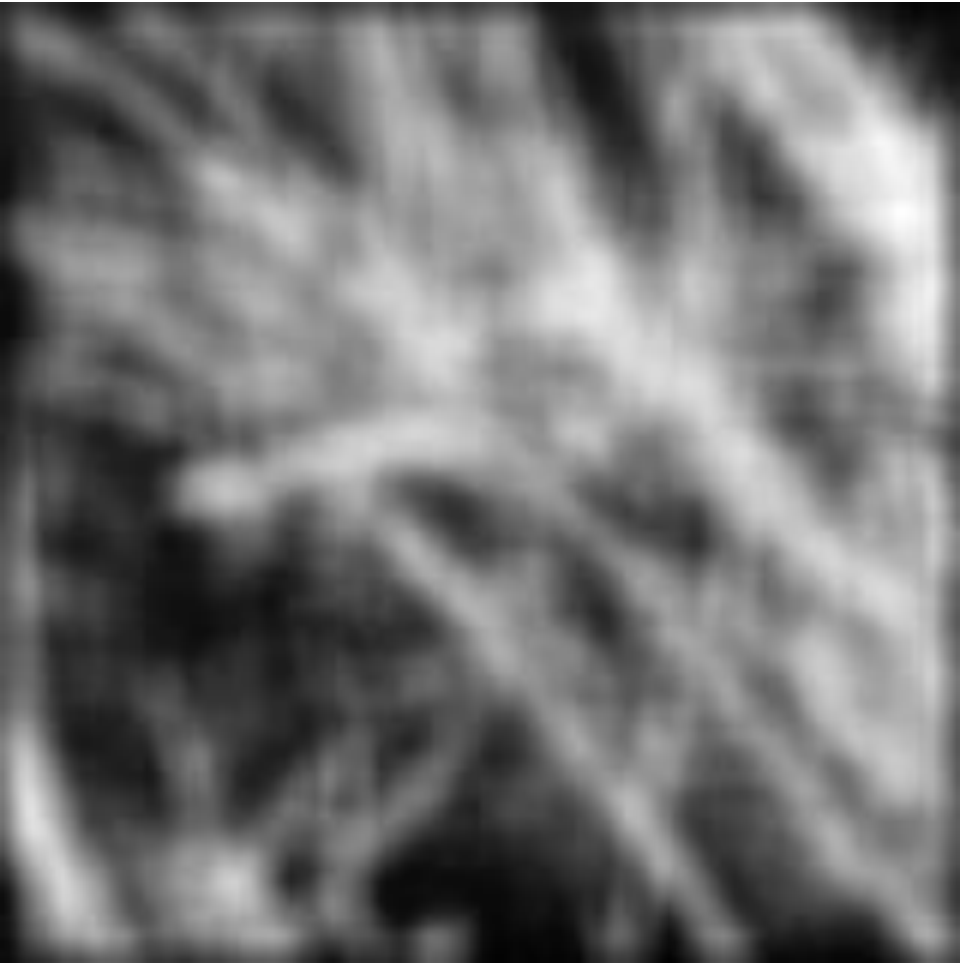
What does it do?

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

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

1	1	1
1	1	1
1	1	1

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

—

$\frac{1}{9}$

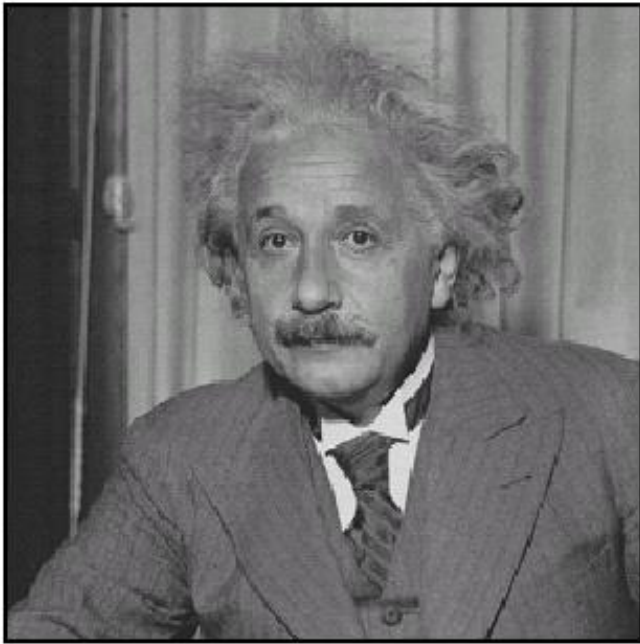
1	1	1
1	1	1
1	1	1



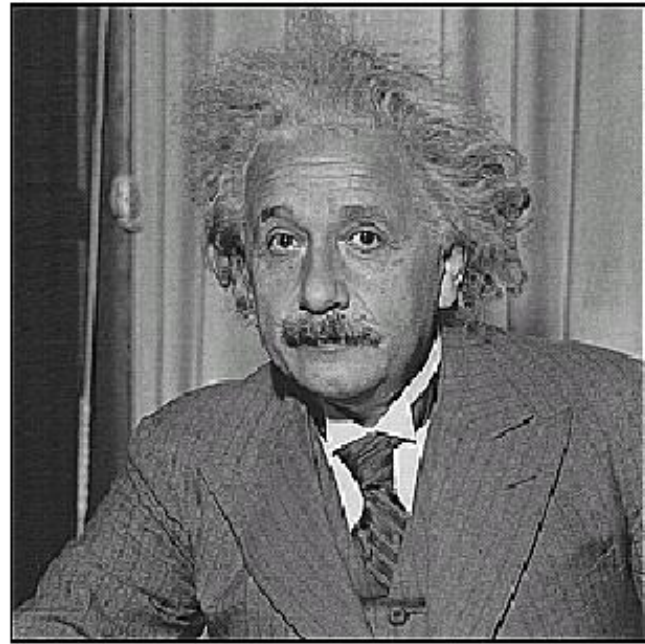
Sharpening filter

- Accentuates differences with local average

Sharpening

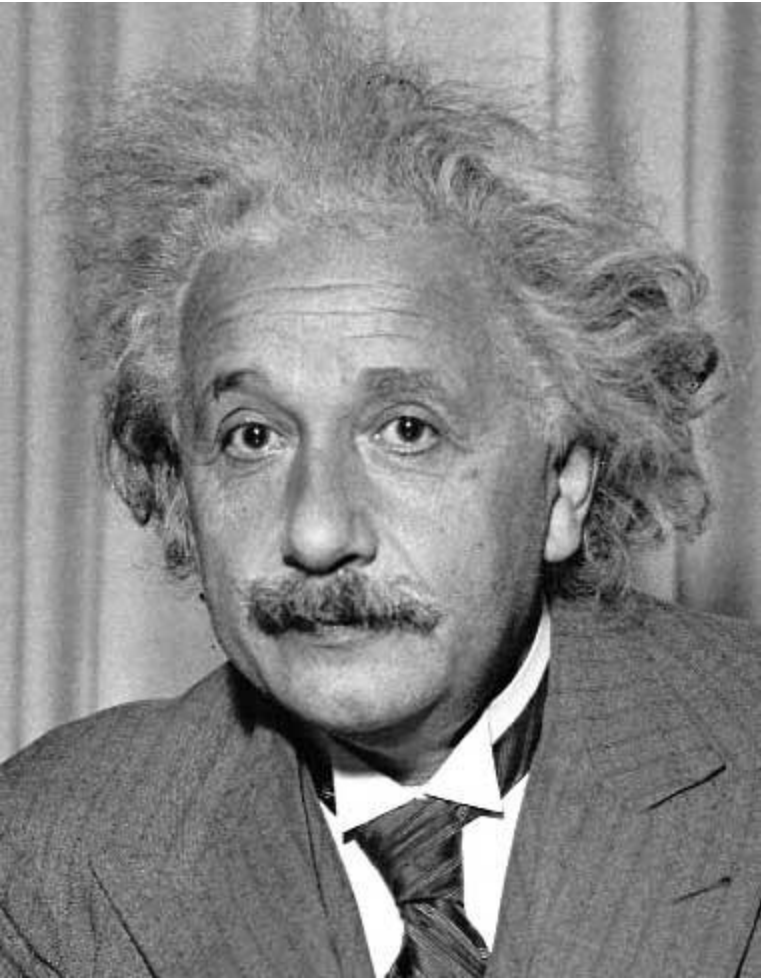


before



after

Other filters



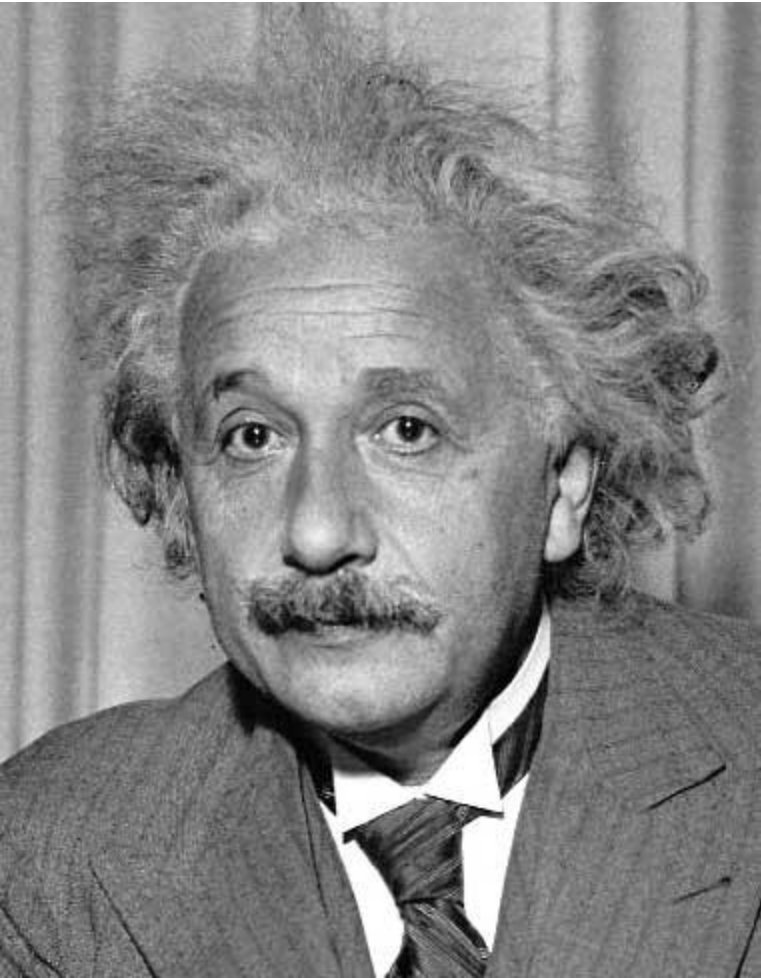
1	0	-1
2	0	-2
1	0	-1

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

$f = \text{filter}$ $I = \text{image}$
– `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

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

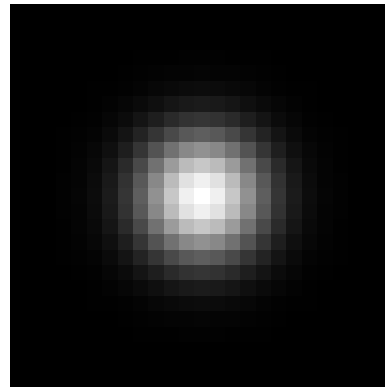
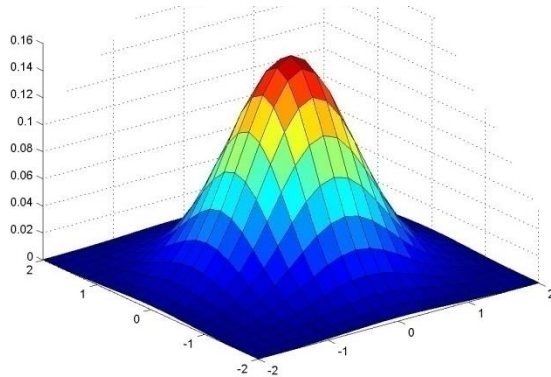
Any linear, shift-invariant operator can be represented as a convolution

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

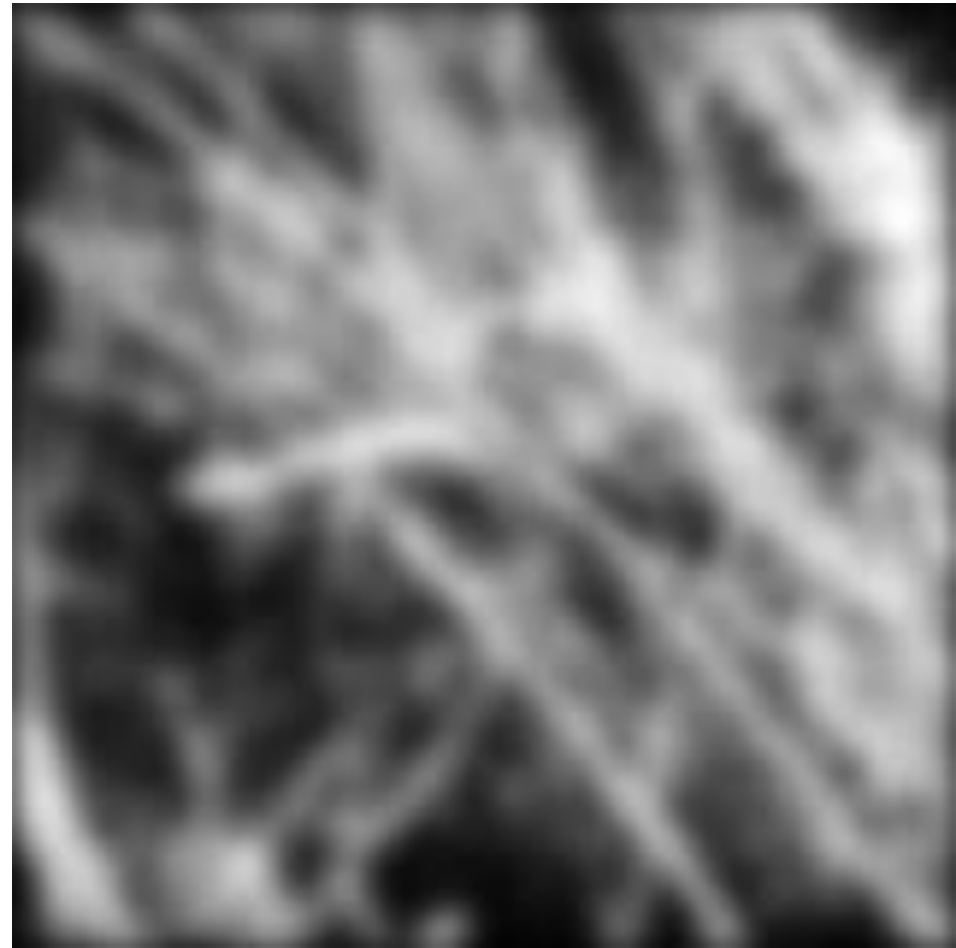


0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

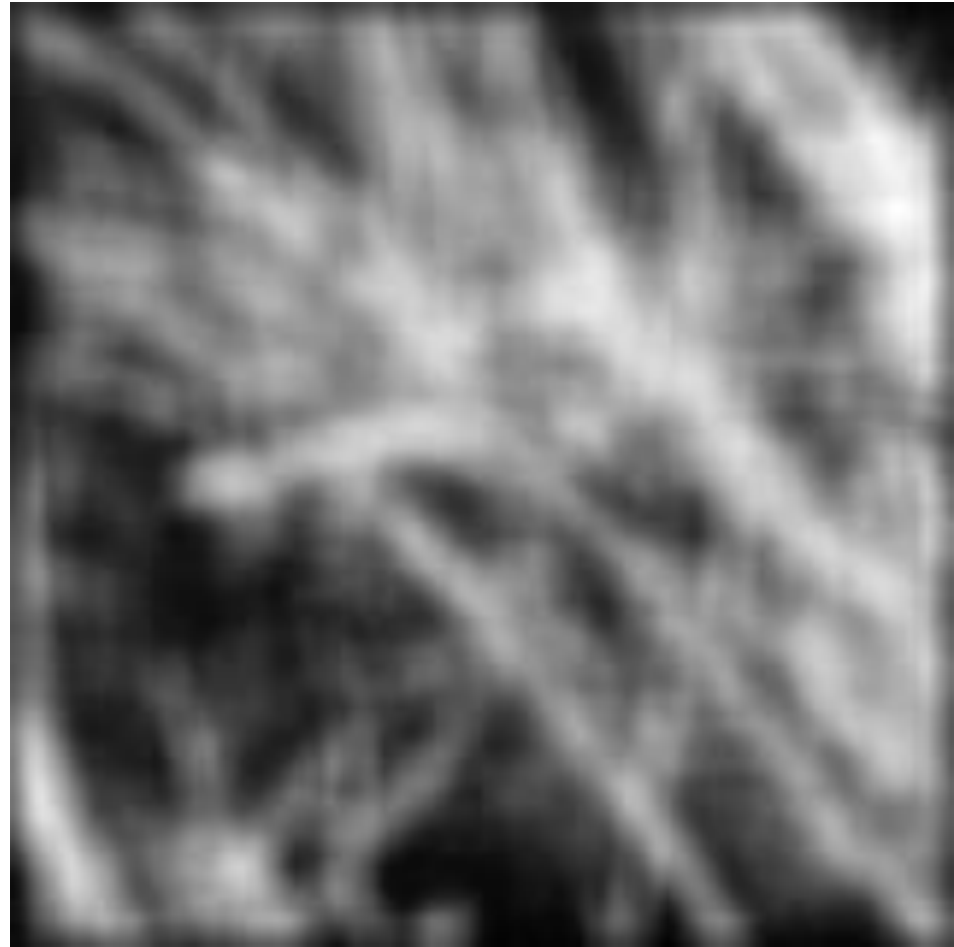
5 x 5, $\sigma = 1$

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

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

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

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)

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix}$$

The filter factors
into a product of 1D
filters:

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

Perform convolution
along rows:

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix} = \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix}$$

Followed by convolution
along the remaining column:

Separability

- Why is separability useful in practice?

Some practical matters

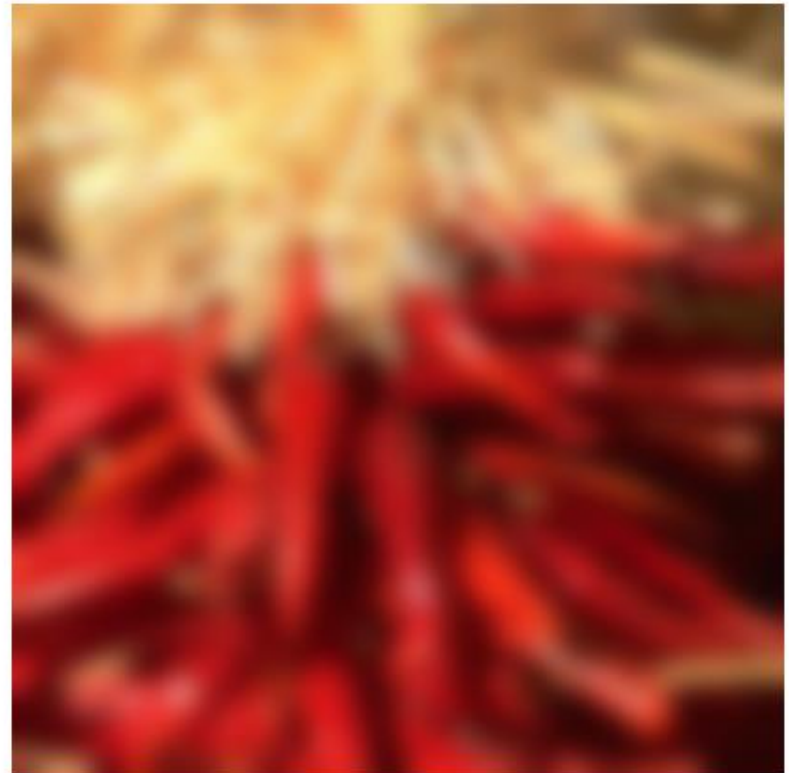
Practical matters

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



Next class: Thinking in Frequency

