CEE598 - Visual Sensing for Civil Infrastructure Eng. & Mgmt.

Session 7 – Pixels and Image Filtering

Mani Golparvar-Fard
Department of Civil and Environmental Engineering
3129D, Newmark Civil Engineering Lab
e-mail: mgolpar@illinois.edu

Outline

What we have learned so far:

- Image formation
- Camera Calibration and Single View Metrology

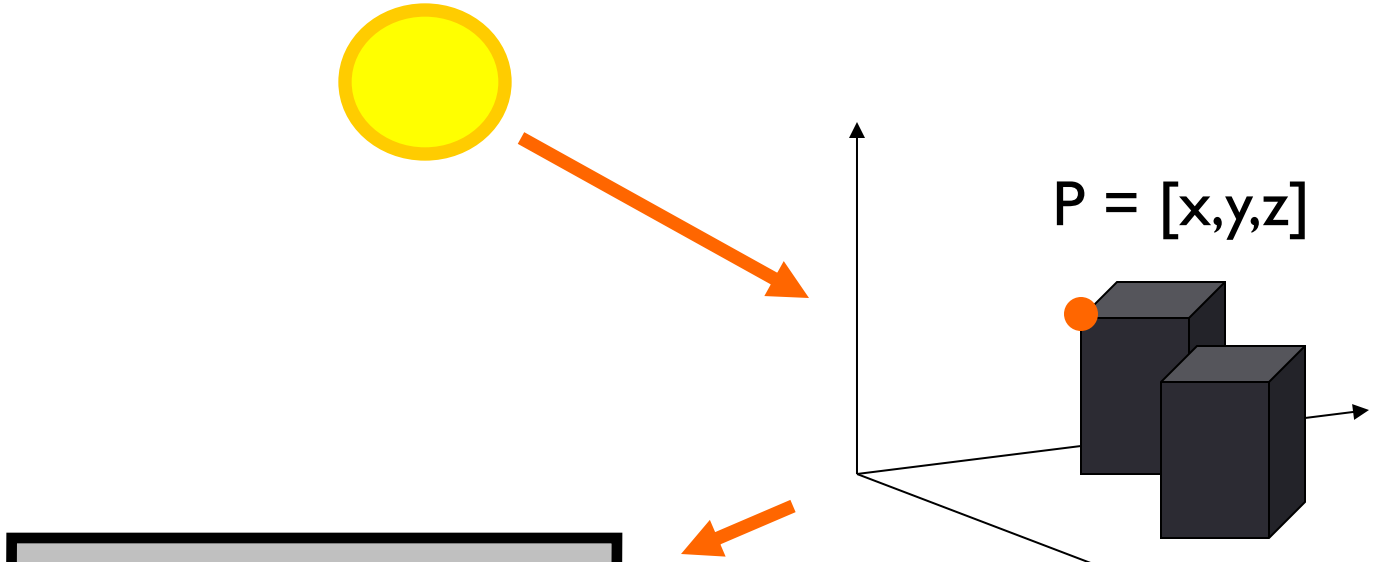
Introduction to Pixels and Image Filtering

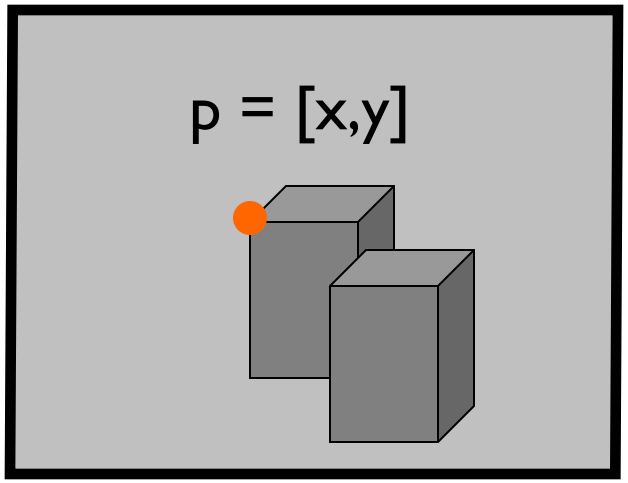
- Review of lighting
- Reflection and absorption
- What is image filtering and how do we do it?
- Color models (if time allows)

Reading: [FP] Chapters 7,8

Some slides in this lecture are courtesy to Prof. S. Savarese, Prof. D. Hoiem, prof F. Li, prof S. Lazebnik, and various other lecturers

From the 3D to 2D

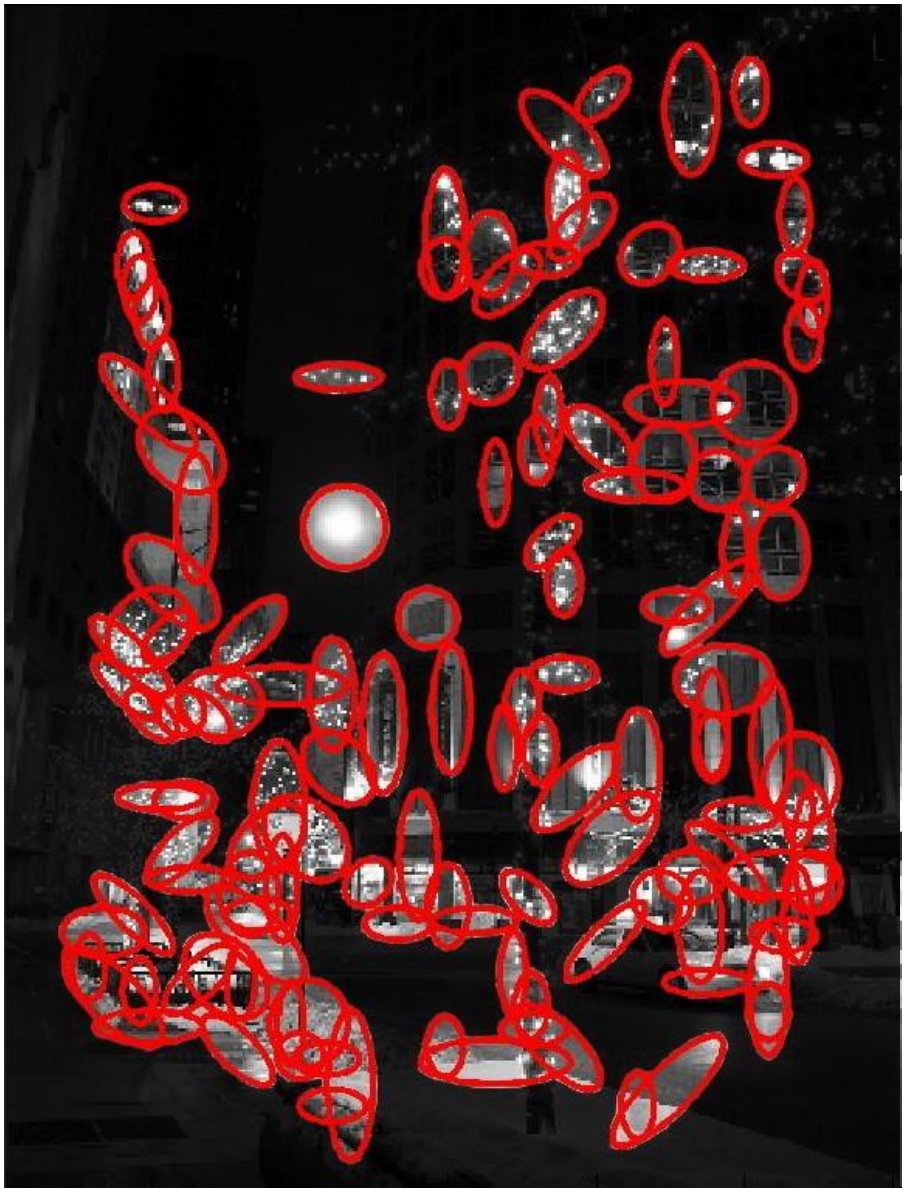




Image

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

Extract useful building blocks



The big picture…

Feature Detection

e.g. DoG

Feature Description

e.g. SIFT

database of local descriptors

Matching / Indexing / Detection

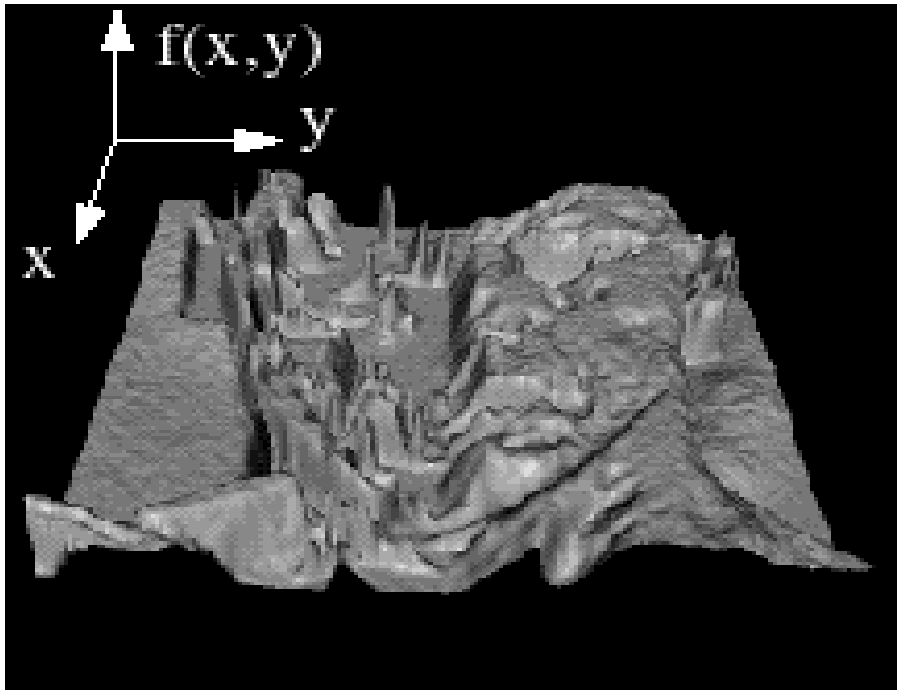
Images as functions

- We can think of an **image** as a function f, from R^2 to R:
  - Defined over a rectangle, with a finite range:
    - f: $[a,b] \times [c,d] \rightarrow [0,255]$
  - $f(x,y)$ gives the **intensity** at position (x,y)
- A color image:

$$f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$

Source: S. Seitz

Images as functions

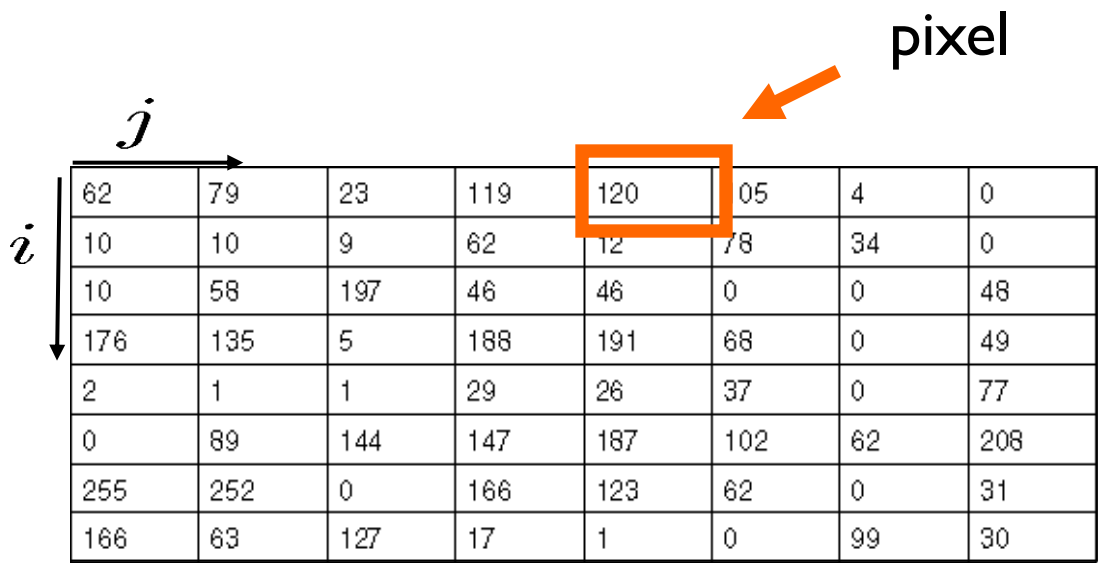


Source: S. Seitz

Images as functions

- Images are usually **digital** (**discrete**):
  - **Sample** the $2D$ space on a regular grid

- The image can now be represented as a matrix of integer values



Source: S. Seitz

Pixels and Reflection Models

Quick Overview

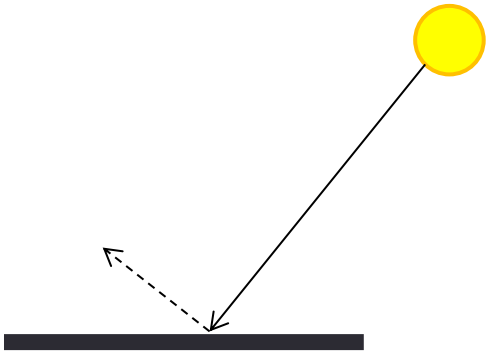
Reflection models

- Albedo: fraction of light that is reflected
  - Determines color (amount reflected at each wavelength)

Credit: D. Hoiem

Reflection models

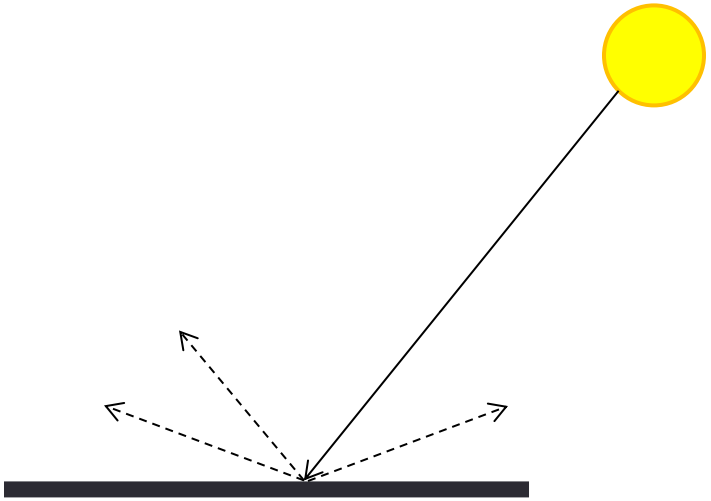
- Specular reflection: mirror-like
  - Light reflects at incident angle
  - Reflection color = incoming light color

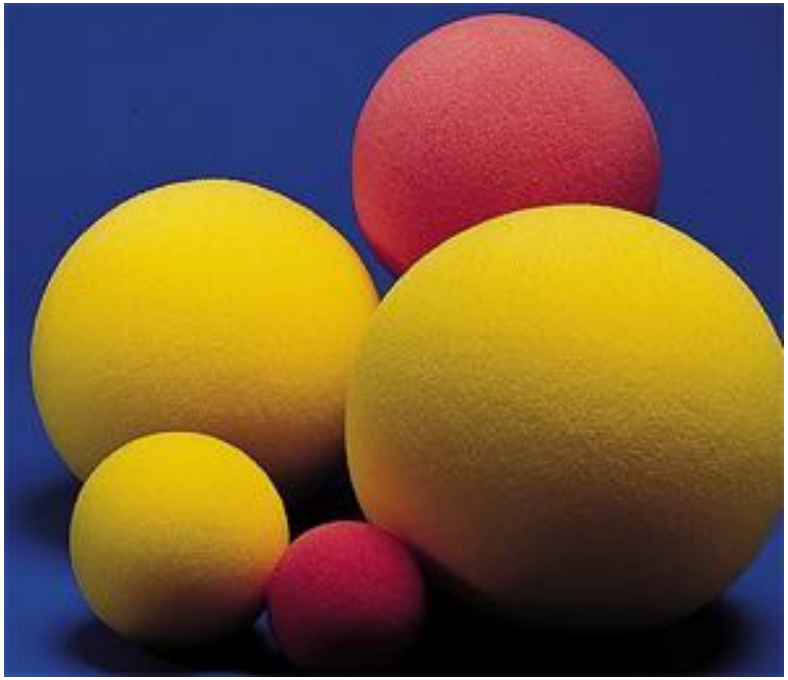


Credit: D. Hoiem

Reflection models

- Diffuse reflection
  - Light scatters in all directions (proportional to cosine with surface normal)
  - Observed intensity is independent of viewing direction
  - Reflection color depends on light color and albedo

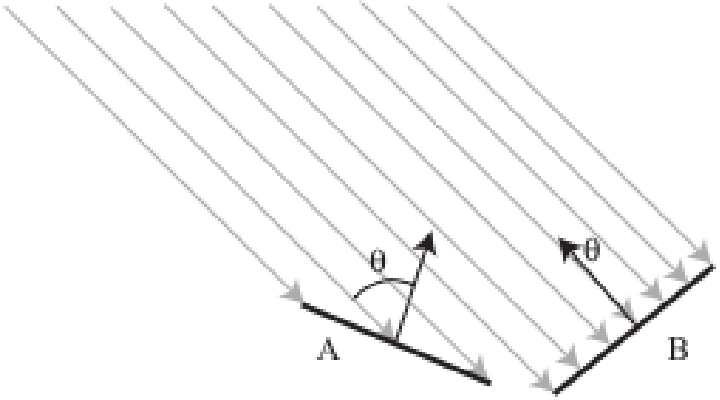




Credit: D. Hoiem

Surface orientation and light intensity

- Amount of light that hits surface from distant point source depends on angle between surface normal and source



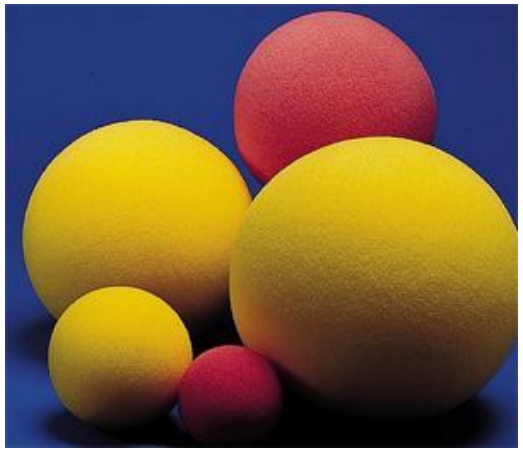
$$I(x) = \rho(x)(\boldsymbol{S} \cdot \boldsymbol{N}(x))$$

prop to cosine of relative angle

Credit: D. Hoiem

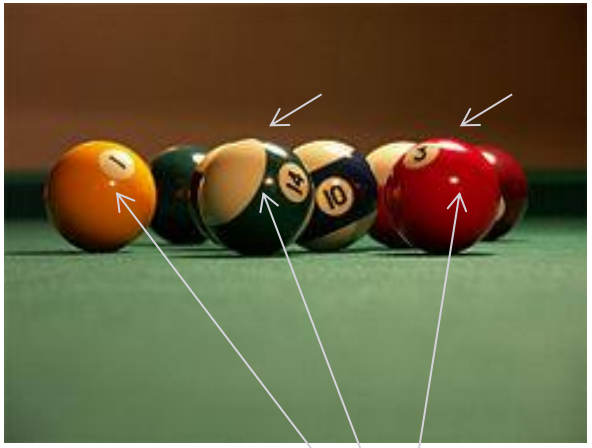
Reflection models

Lambertian: reflection all diffuse



Mirrored: reflection all specular

Glossy: reflection mostly diffuse, some specular



Specularities

Credit: D. Hoiem

Questions

- How many light sources are in the scene?
- How could I estimate the color of the camera's flash?

Credit: D. Hoiem

The plight of the poor pixel

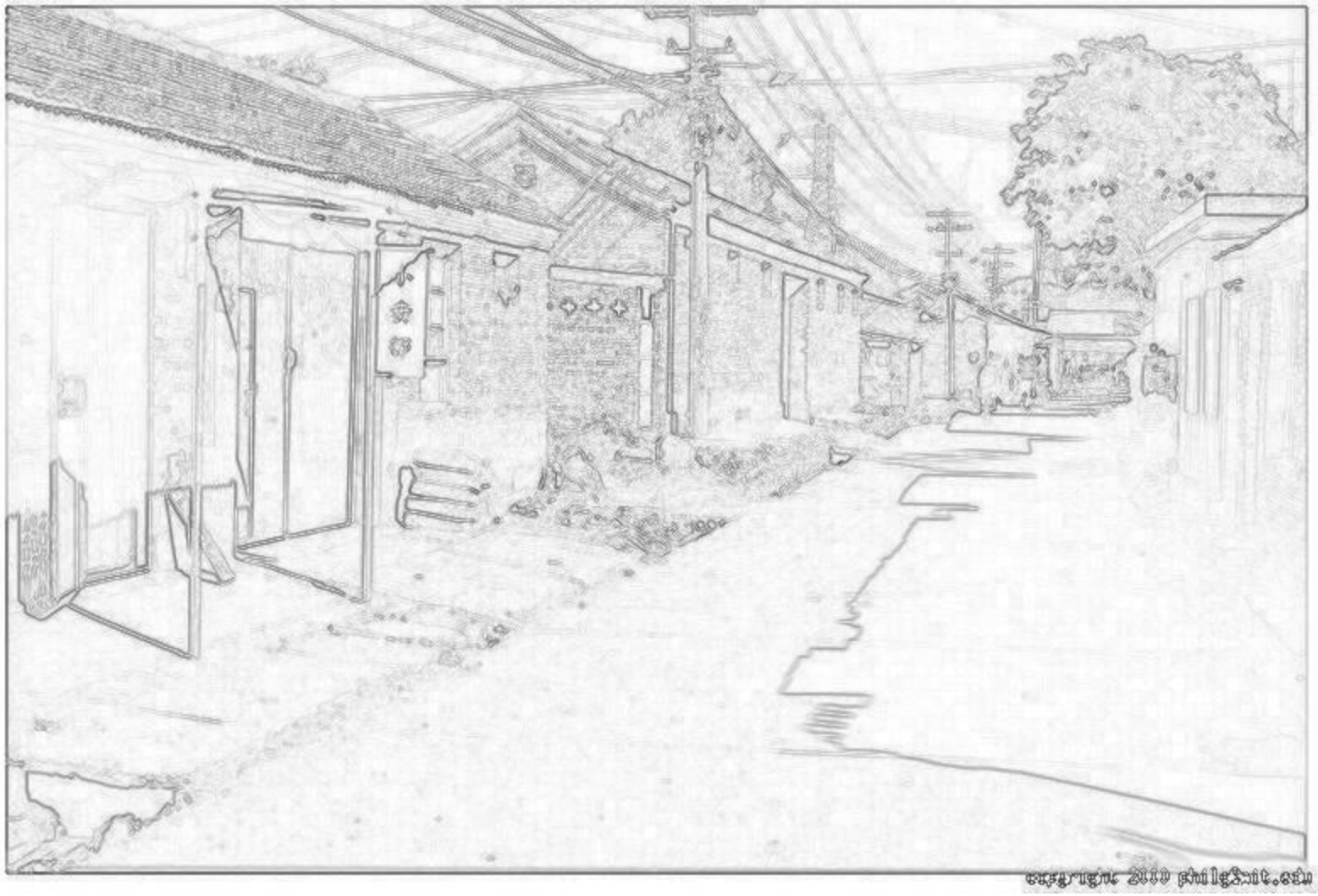
- A pixel's brightness is determined by
  - Light source (strength, direction, color)
  - Surface orientation
  - Surface material and albedo
  - Reflected light and shadows from surrounding surfaces
  - Gain on the sensor

- A pixel's brightness tells us nothing by itself

Basis for interpreting intensity images

- Key idea: for nearby scene points, most factors do not change much
- The information is mainly contained in *local differences* of brightness

Darkness = Large Difference in Neighboring Pixels



Next few classes: different 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

The raster image (pixel matrix)

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

Credit: D. Hoiem

Image filtering

- Image filtering: compute function of local neighborhood at each position
- Linear filtering: function is a weighted sum/difference of pixel values
- Really important!
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching

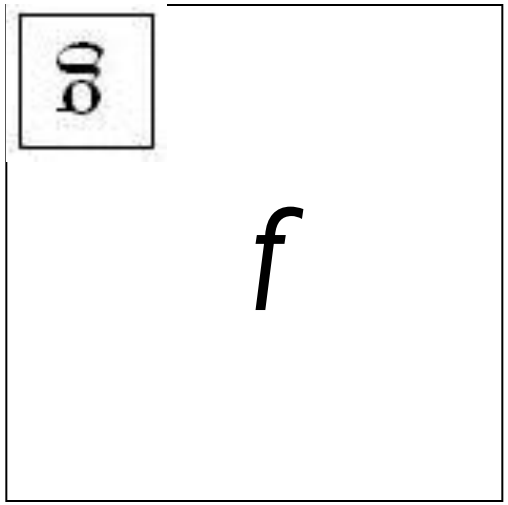
Credit: D. Hoiem

Convolution

- Let *f* be the image and *g* be the kernel. The output of convolving *f* with *g* is denoted by $f * g$.

$$(f * g)[m,n] = \sum_{k,l} f[k,l]\, g[m-k, n-l]$$

Weighted product of f(k,l) by g(-(k,l))) computed at different locations m,n



•MATLAB: conv2 vs. filter2 (also imfilter)

Example: box filter

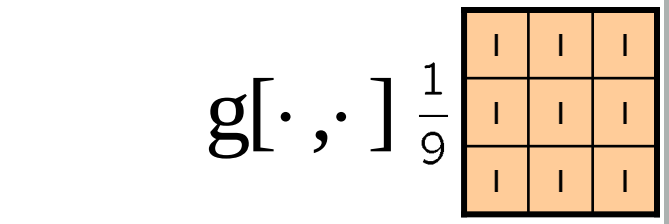
$g[\cdot,\cdot]$

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

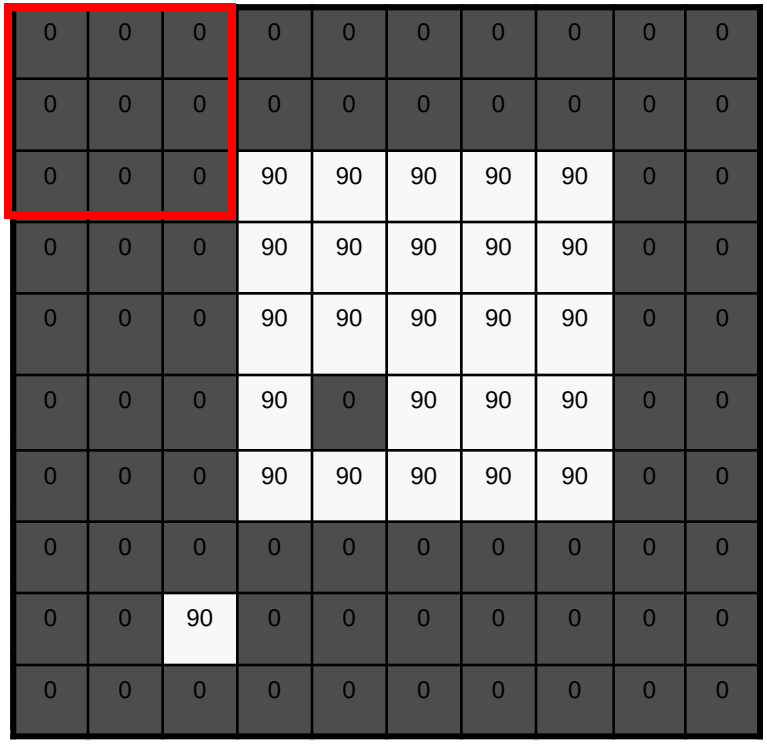
Slide credit: David Lowe (UBC)

Image filtering

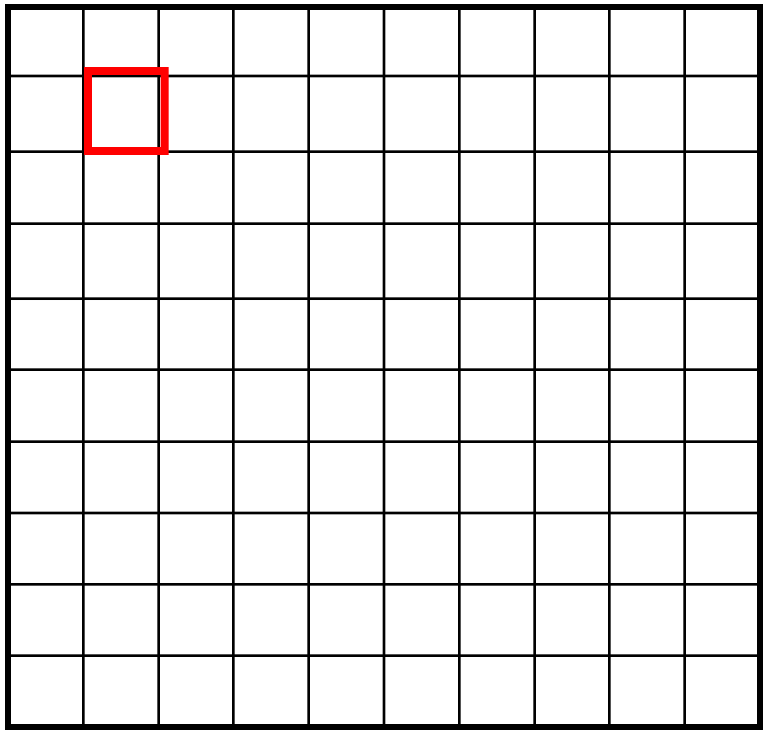
$g[\cdot,\cdot]$ $\frac{1}{9}$



$f[.,.]$



$h[.,.]$

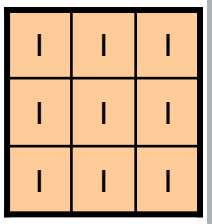


$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

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



$f[.,.]$

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

$h[.,.]$

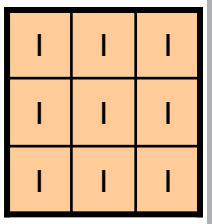
	0	10							

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

Credit: S. Seitz

Image filtering

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



$f[.,.]$

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

$h[.,.]$

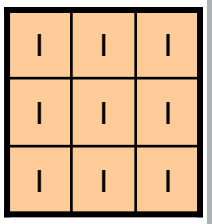
	0	10	20						

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

Credit: S. Seitz

Image filtering

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



$f[.,.]$

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

$h[.,.]$

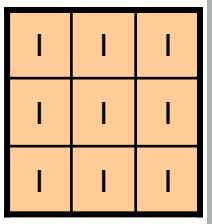
	0	10	20	30					

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

Credit: S. Seitz

Image filtering

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



$f[.,.]$

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

$h[.,.]$

	0	10	20	30	30				

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

Credit: S. Seitz

Image filtering

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

$f[.,.]$

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

$h[.,.]$

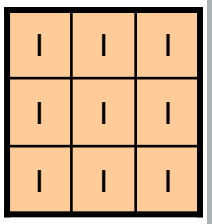
	0	10	20	30	30				
				?					

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

Credit: S. Seitz

Image filtering

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



$f[.,.]$

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

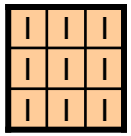
$h[.,.]$

	0	10	20	30	30				
						?			
				50					

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

Credit: S. Seitz

Image filtering

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

$f[.,.]$

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

$h[.,.]$

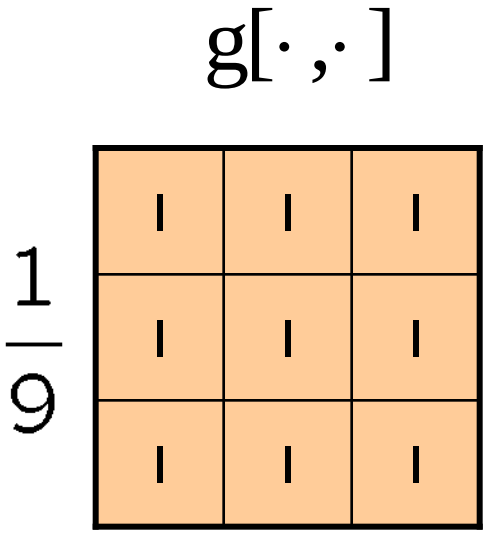
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

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

Credit: S. Seitz

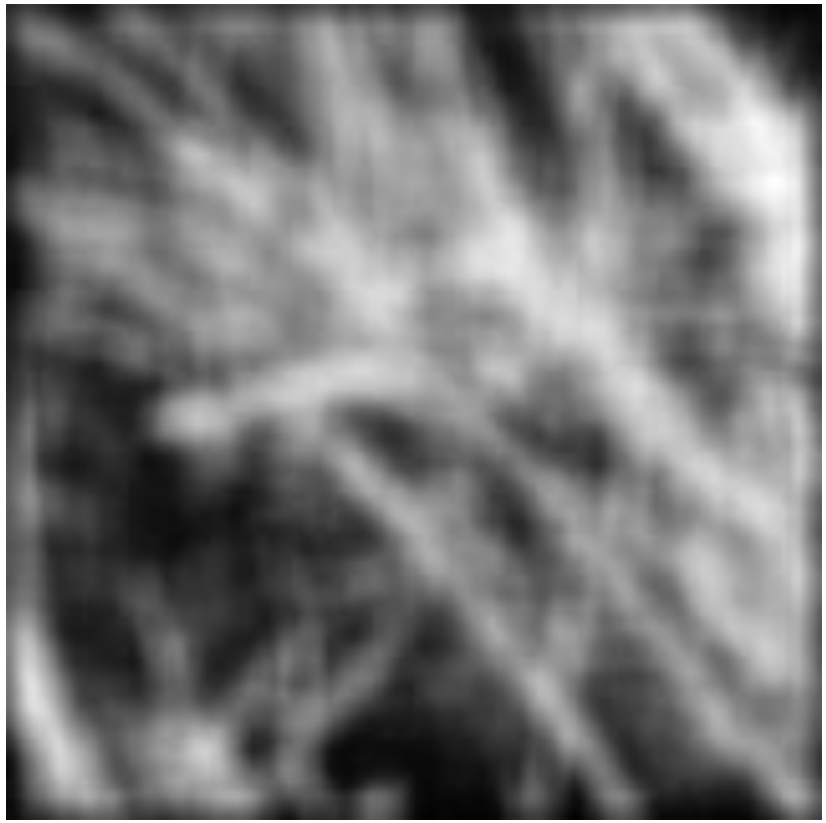
Example: box filter

- Kernel k with positive entries, that sum to 1.
- Notice: all weights are equal



Slide credit: David Lowe (UBC)

Smoothing with box filter



Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

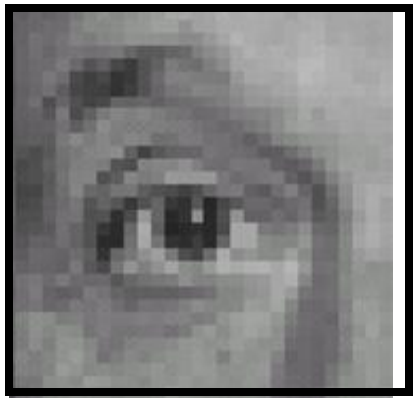
?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Source: D. Lowe

Practice with linear filters

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad ?$$

(Note that filter sums to 1)

Source: D. Lowe

Practice with linear filters

Original

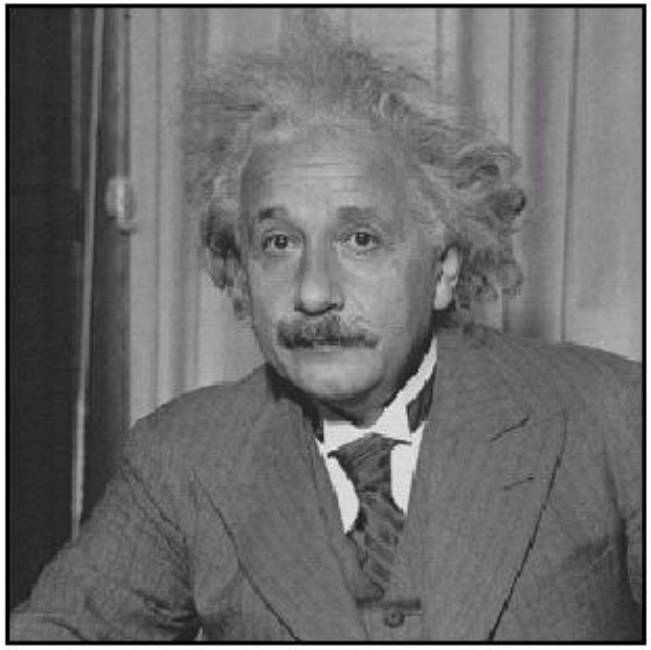
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter

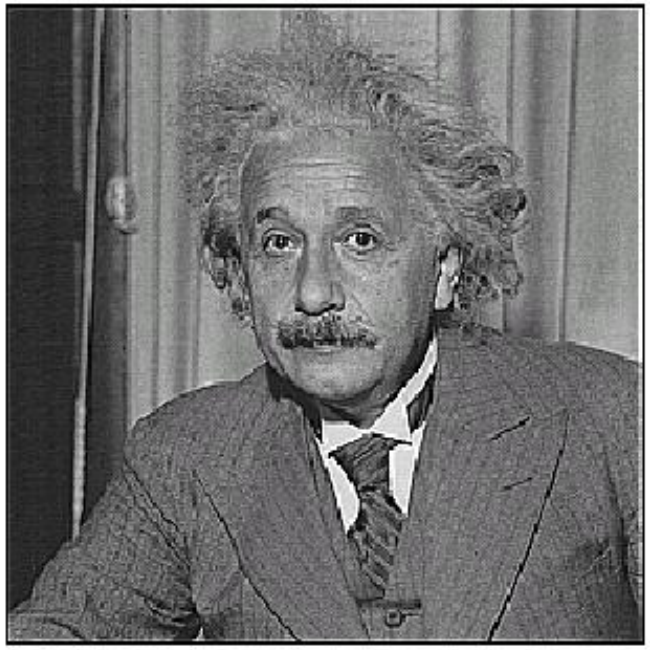
- Accentuates differences with local average

Source: D. Lowe

Sharpening



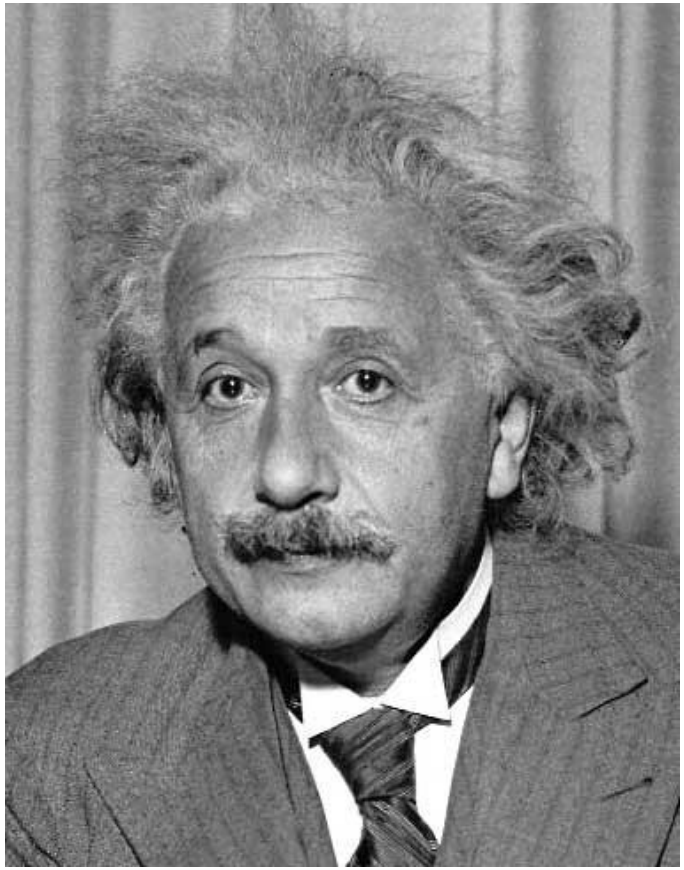
before



after

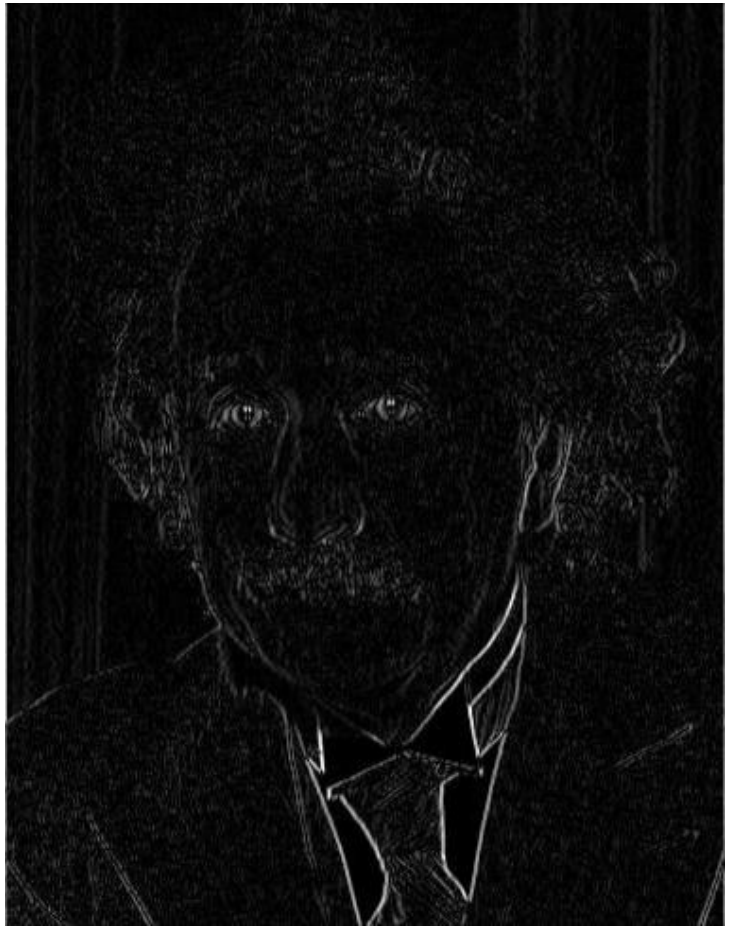
Source: D. Lowe

Other filters



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

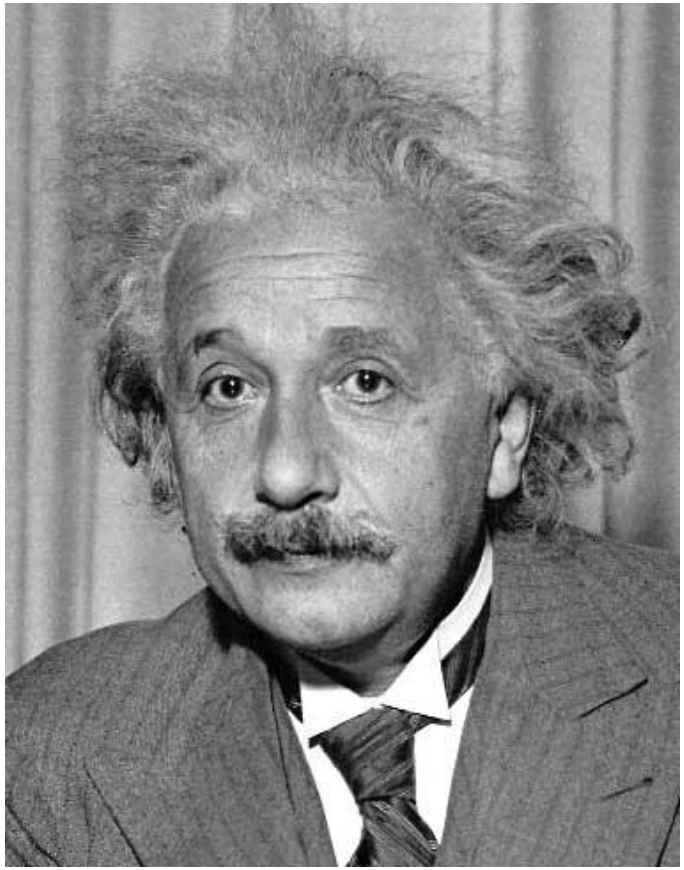
Sobel



Vertical Edge (absolute value)

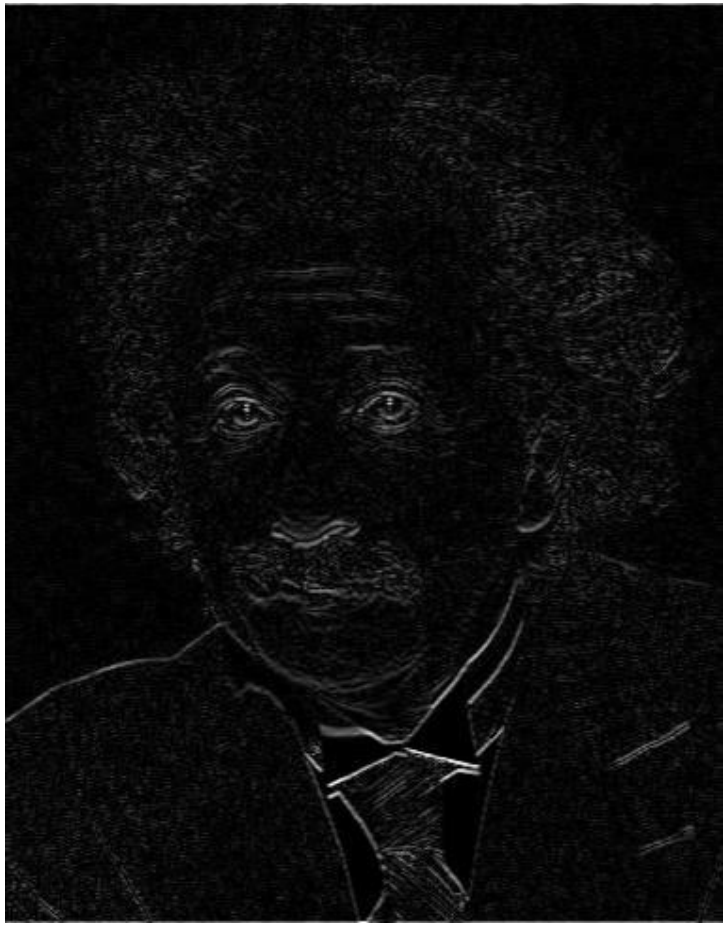
Source: D. Hoiem

Other filters



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

Sobel



Horizontal Edge (absolute value)

Source: D. Hoiem

Basic gradient filters

Horizontal Gradient

0	0	0
-1	0	1
0	0	0

or

-1	0	1

Vertical Gradient

0	1	0
0	0	0
0	-1	0

or

-1
0
1

How could we synthesize motion blur?

```
theta = 30; len = 20;
fil = imrotate(ones(1, len), theta, 'bilinear');
fil = fil / sum(fil(:));
figure(2), imshow(imfilter(im, fil));
```

Filtering vs. Convolution

- 2d filtering

g=filter f=image

  - `h=filter2(g,f);` or
    `h=imfilter(f,g);`

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

- 2d convolution
  - `h=conv2(g,f);`

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

Key properties of linear filters

Linearity:

$\texttt{filter}(f_1 + f_2) = \texttt{filter}(f_1) + \texttt{filter}(f_2)$

Shift invariance: same behavior regardless of pixel location

$\texttt{filter}(\texttt{shift}(f)) = \texttt{shift}(\texttt{filter}(f))$

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

Source: S. Lazebnik

More properties

- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal

- 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: $a * (b + c) = (a * b) + (a * c)$

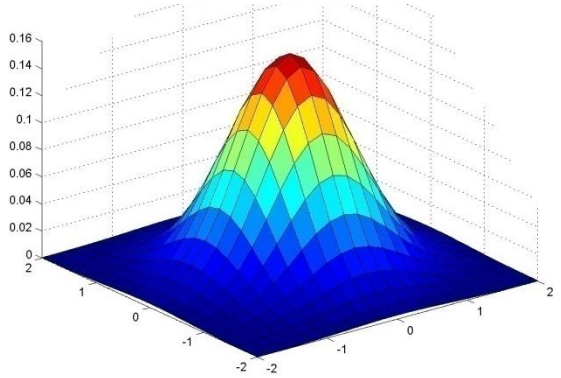
- Scalars factor out: $ka * b = a * kb = k (a * b)$

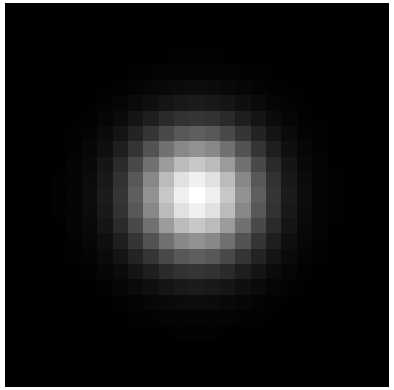
- Identity: unit impulse $e = [0, 0, 1, 0, 0]$,
  $a * e = a$

Source: S. Lazebnik

Important filter: Gaussian

- Spatially-weighted average





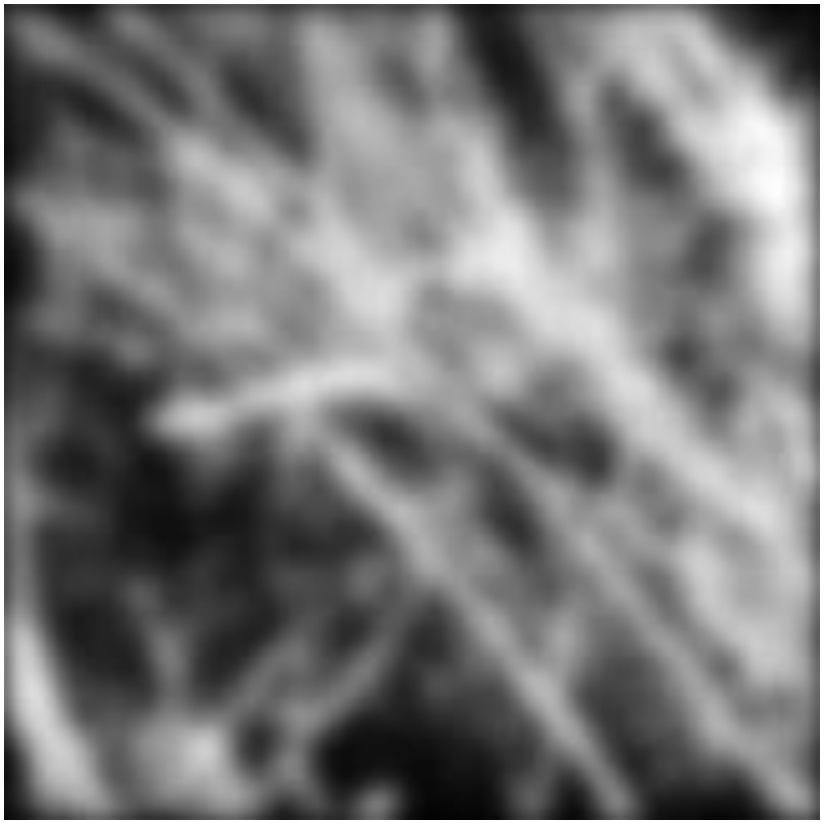
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

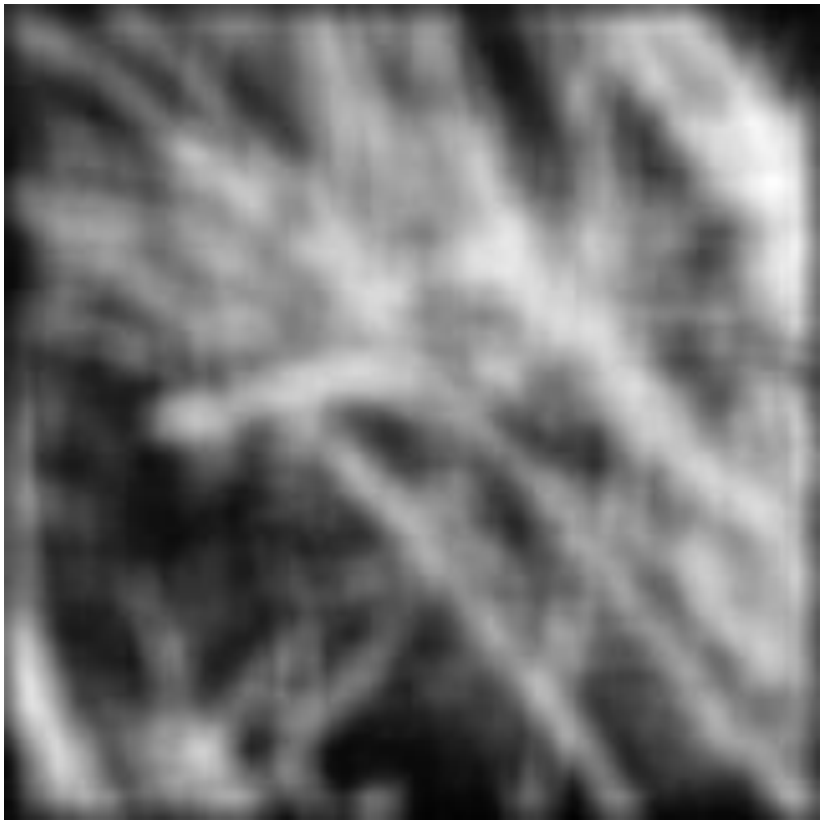
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Slide credit: Christopher Rasmussen

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
  - Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$

- *Separable* kernel
  - Factors into product of two 1D Gaussians

Source: K. Grauman

Separability of the Gaussian filter

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

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

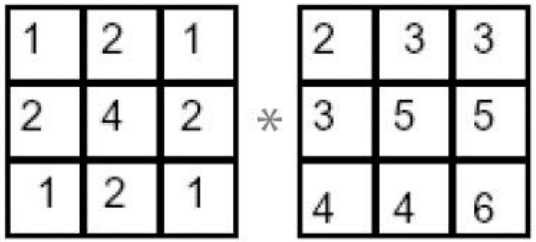
The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

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

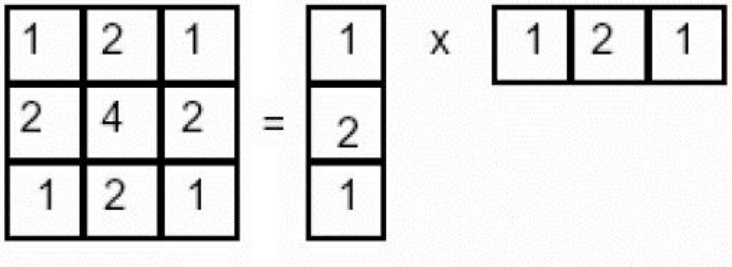
Source: D. Lowe

Separability example

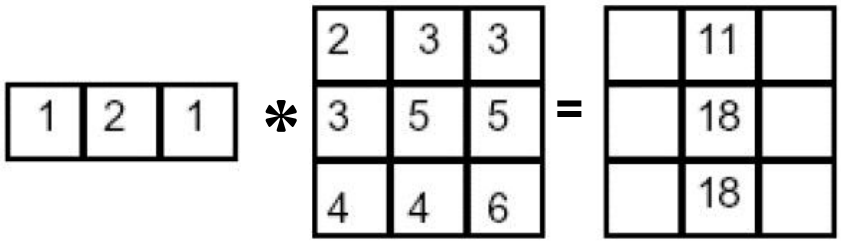
2D filtering
(center location only)



The filter factors into a product of 1D filters:



Perform filtering along rows:



Followed by filtering along the remaining column:

Source: K. Grauman

Separability

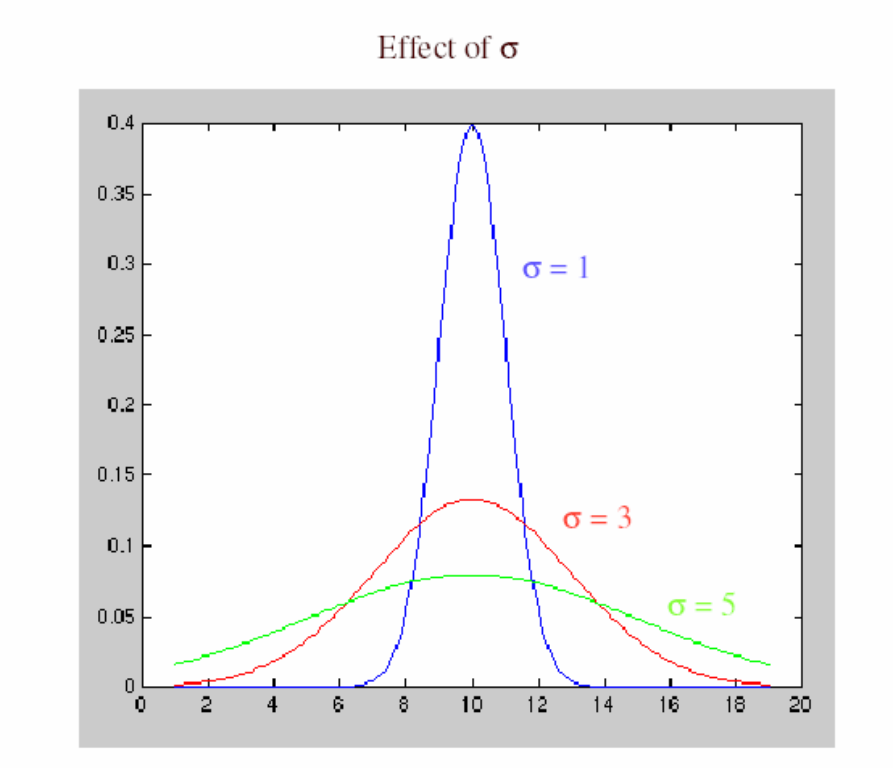
- Why is separability useful in practice?

Some practical matters

Practical matters

How big should the filter be?

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



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

Source: S. Marschner

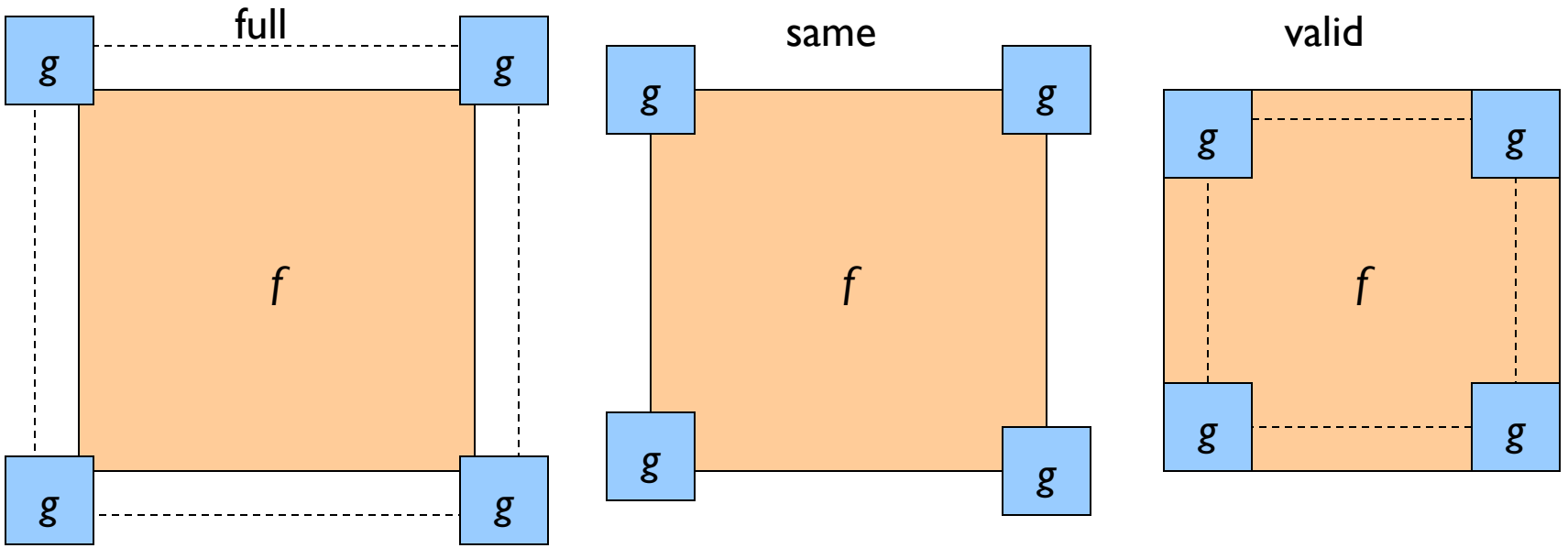
Practical matters

- methods (MATLAB):
  - clip filter (black): imfilter(f, g, 0)
  - wrap around: imfilter(f, g, 'circular')
  - copy edge: imfilter(f, g, 'replicate')
  - reflect across edge: imfilter(f, g, 'symmetric')

Source: S. Marschner

Practical matters

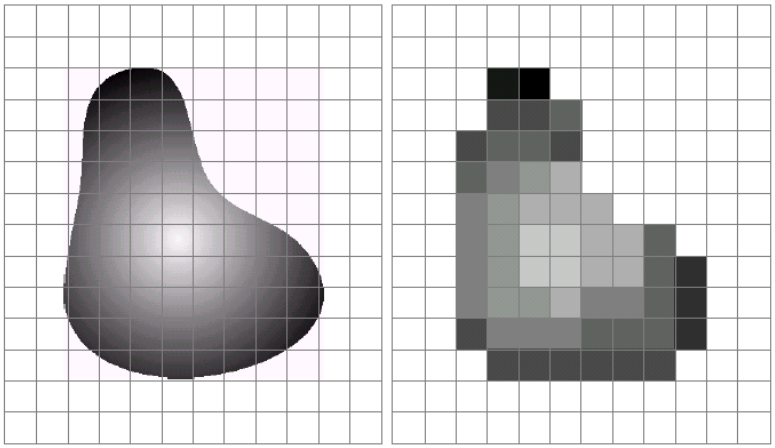
- What is the size of the output?
- MATLAB: filter2(g, f, *shape*)
  - *shape* = 'full': output size is sum of sizes of f and g
  - *shape* = 'same': output size is same as f
  - *shape* = 'valid': output size is difference of sizes of f and g



Source: S. Lazebnik

A little more about color…

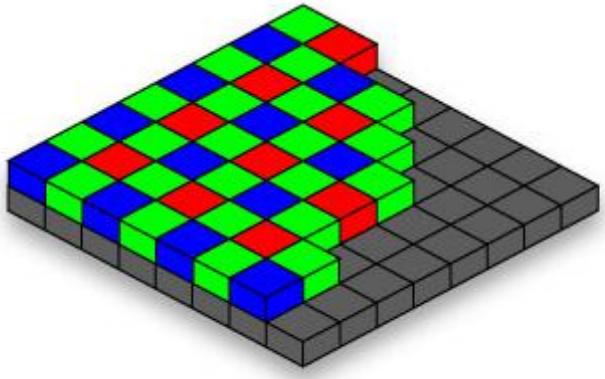
Digital Color Images

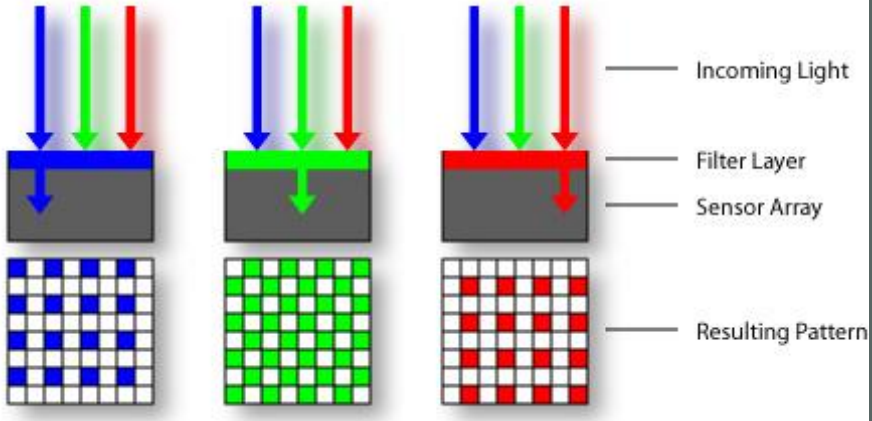


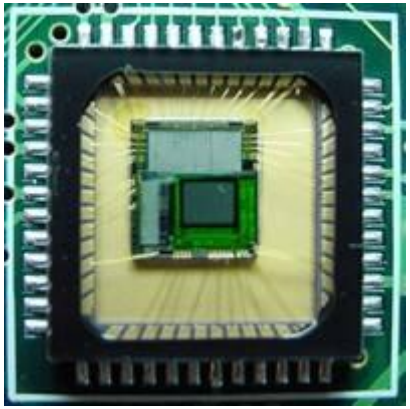
a b

FIGURE 2.17 (a) Continuos image projected onto a sensor array. (b) Result of image sampling and quantization.

Bayer Filter



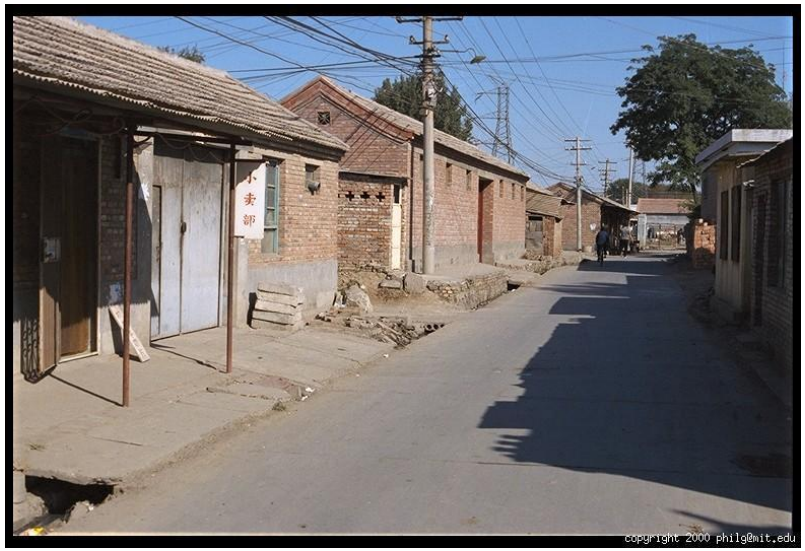




CMOS sensor

Source: D. Hoiem

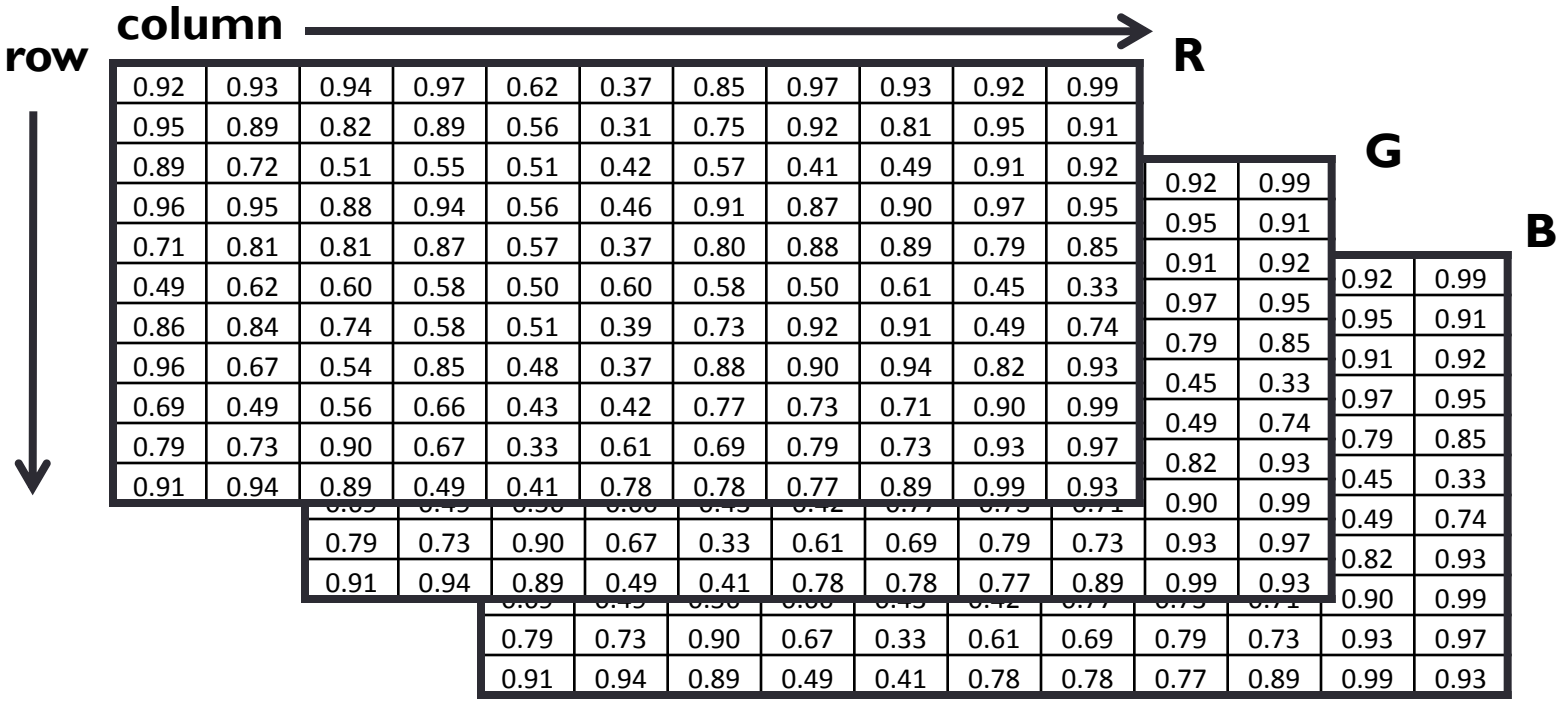
Color Image



Source: D. Hoiem

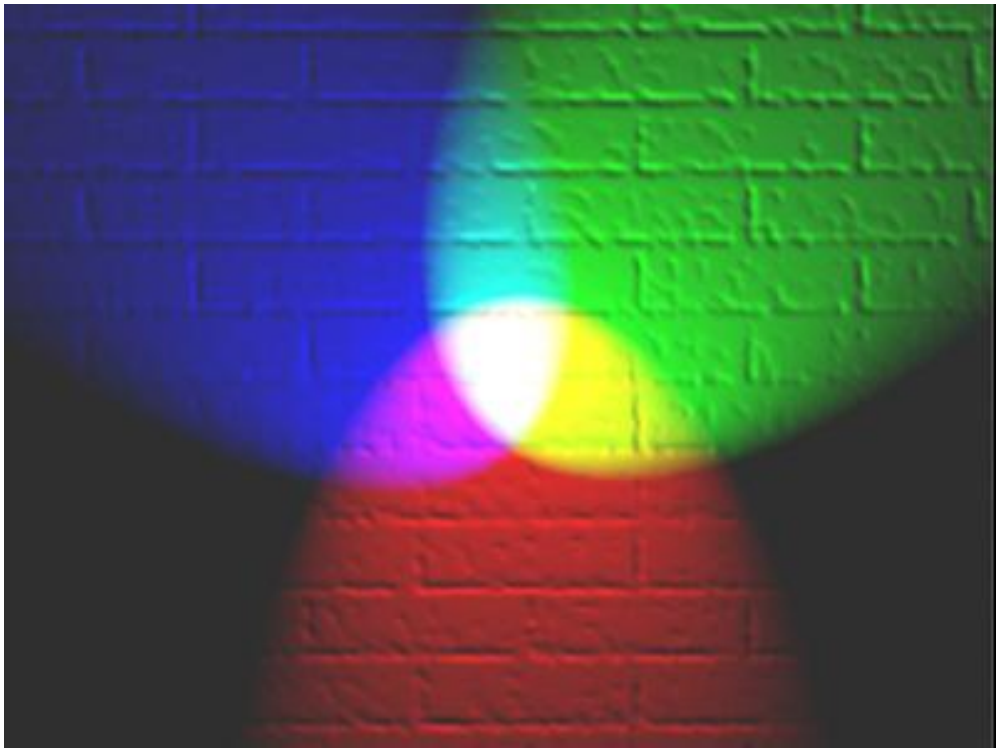
Images in Matlab

- 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 bth channel
  - 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



Color spaces

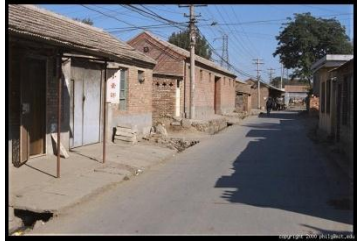
- How can we represent color?



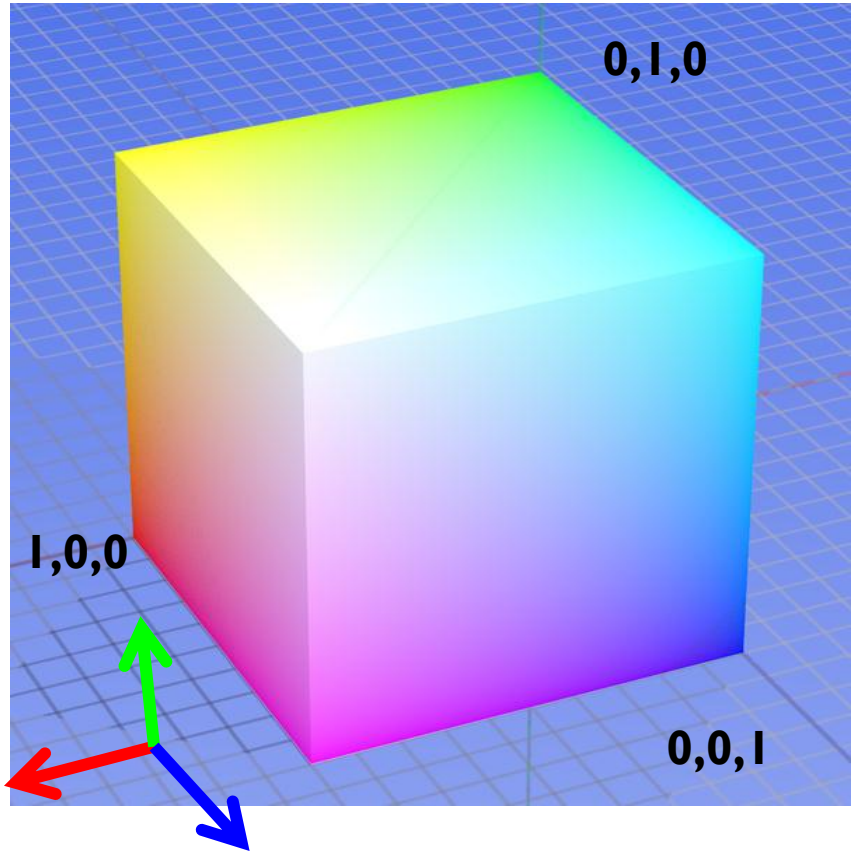
Source: D. Hoiem

http://en.wikipedia.org/wiki/File:RGB_illumination.jpg

Color spaces: RGB



Default color space



R
(G=0,B=0)

G
(R=0,B=0)

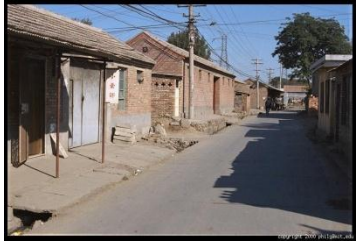
B
(R=0,G=0)

Some drawbacks

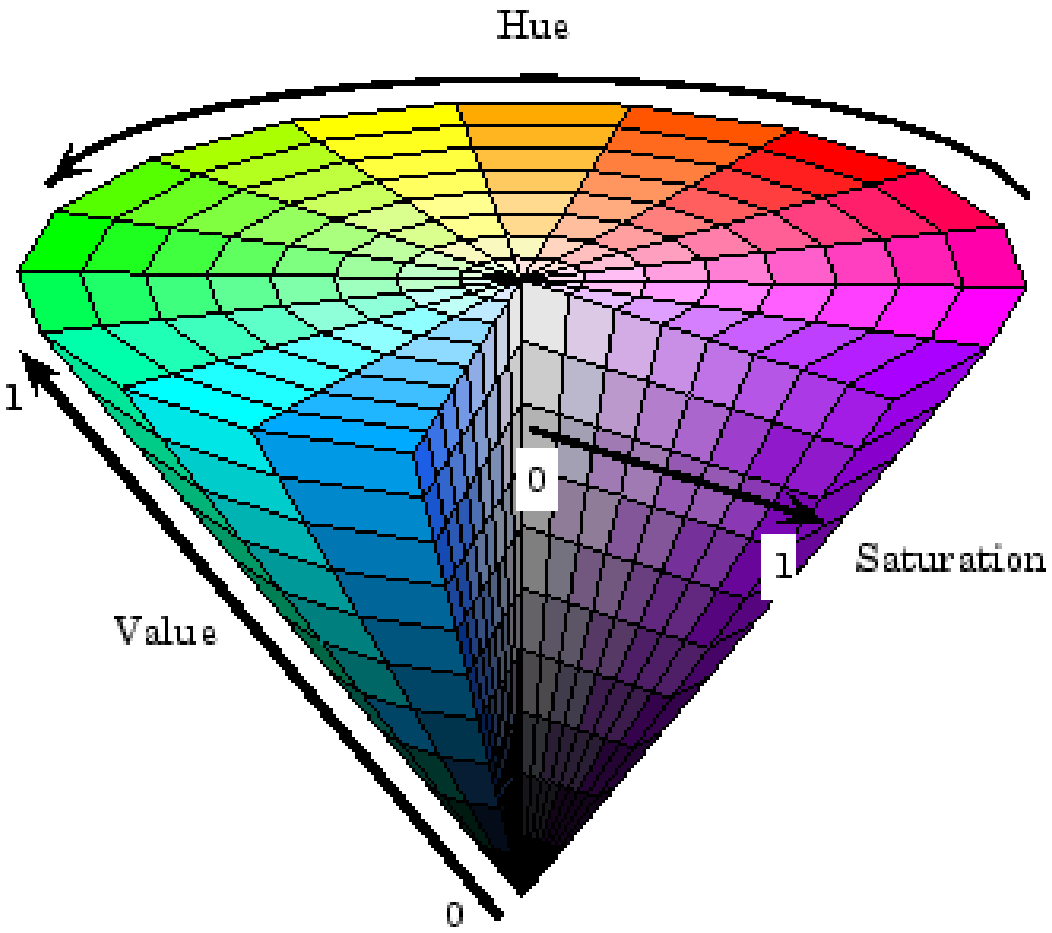
- Strongly correlated channels
- Non-perceptual

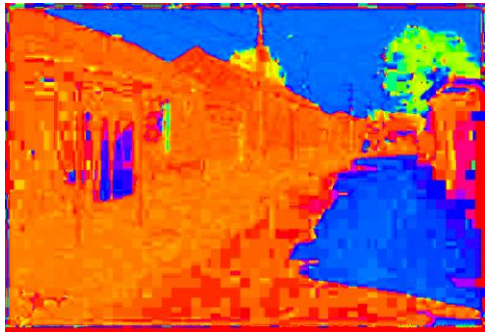
Image from: http://en.wikipedia.org/wiki/File:RGB_color_solid_cube.png

Color spaces: HSV



Intuitive color space





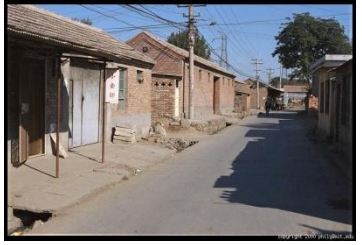
H
(S=1,V=1)

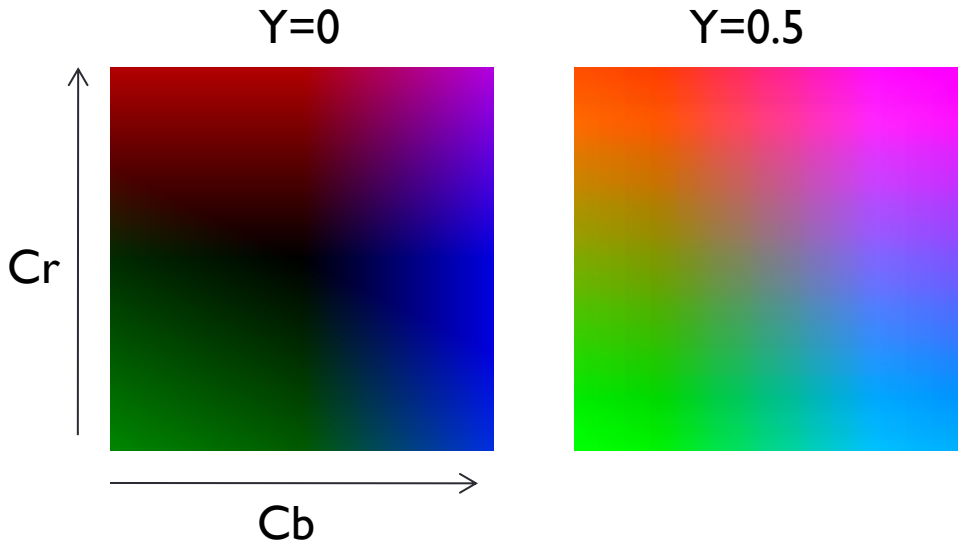
S
(H=1,V=1)

V
(H=1,S=0)

Color spaces: YCbCr

Fast to compute, good for compression, used by TV





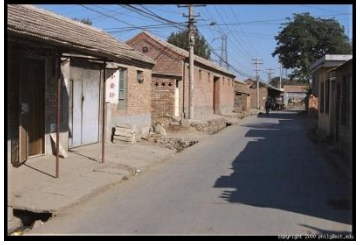
Y
(Cb=0.5,Cr=0.5)

Cb
(Y=0.5,Cr=0.5)

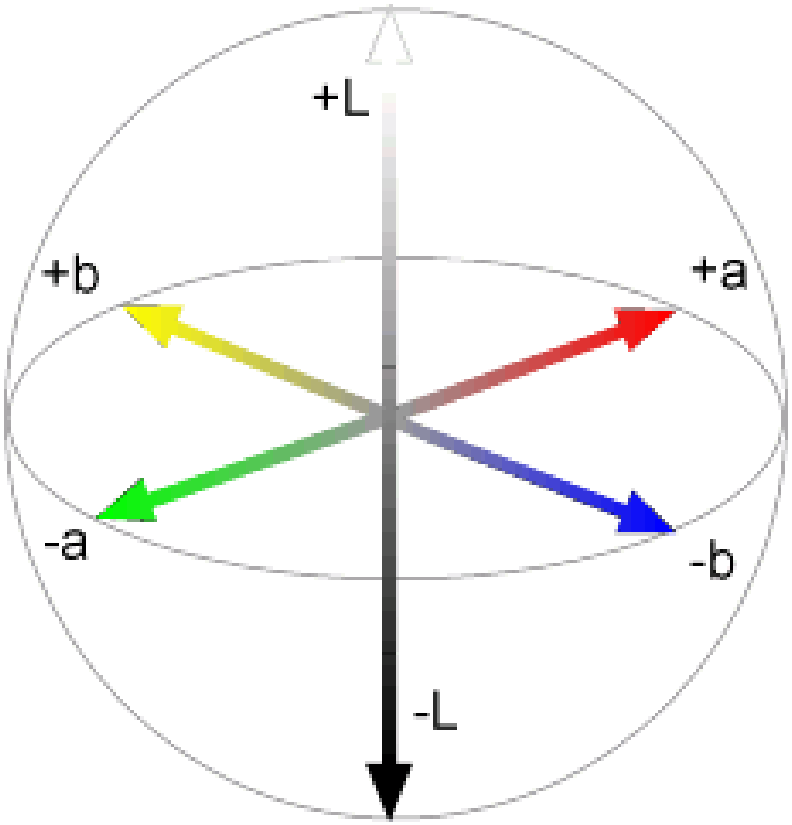
Cr
(Y=0.5,Cb=05)

$$
\begin{aligned}
Y' &= 16 + \frac{65.738 \cdot R'_D}{256} + \frac{129.057 \cdot G'_D}{256} + \frac{25.064 \cdot B'_D}{256} \\
C_B &= 128 + \frac{-37.945 \cdot R'_D}{256} - \frac{74.494 \cdot G'_D}{256} + \frac{112.439 \cdot B'_D}{256} \\
C_R &= 128 + \frac{112.439 \cdot R'_D}{256} - \frac{94.154 \cdot G'_D}{256} - \frac{18.285 \cdot B'_D}{256}
\end{aligned}
$$

Color spaces: CIE L*a*b*



"Perceptually uniform" color space



Luminance = brightness
Chrominance = color

L
(a=0,b=0)

a
(L=65,b=0)

b
(L=65,a=0)

Which contains more information?

(a) **intensity** (1 channel)

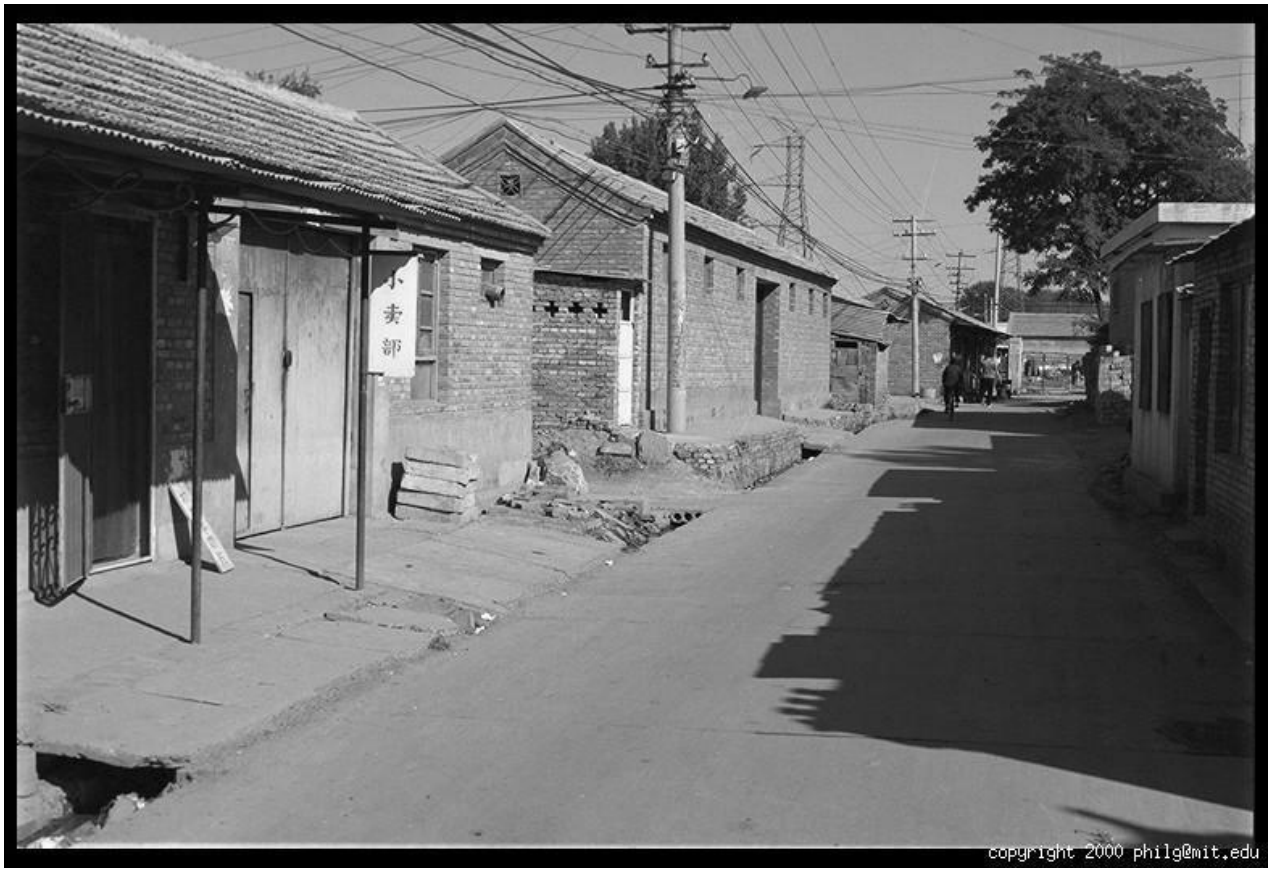
(b) **chrominance** (2 channels)

Most information in intensity

Only color shown – constant intensity

Source: D. Hoiem

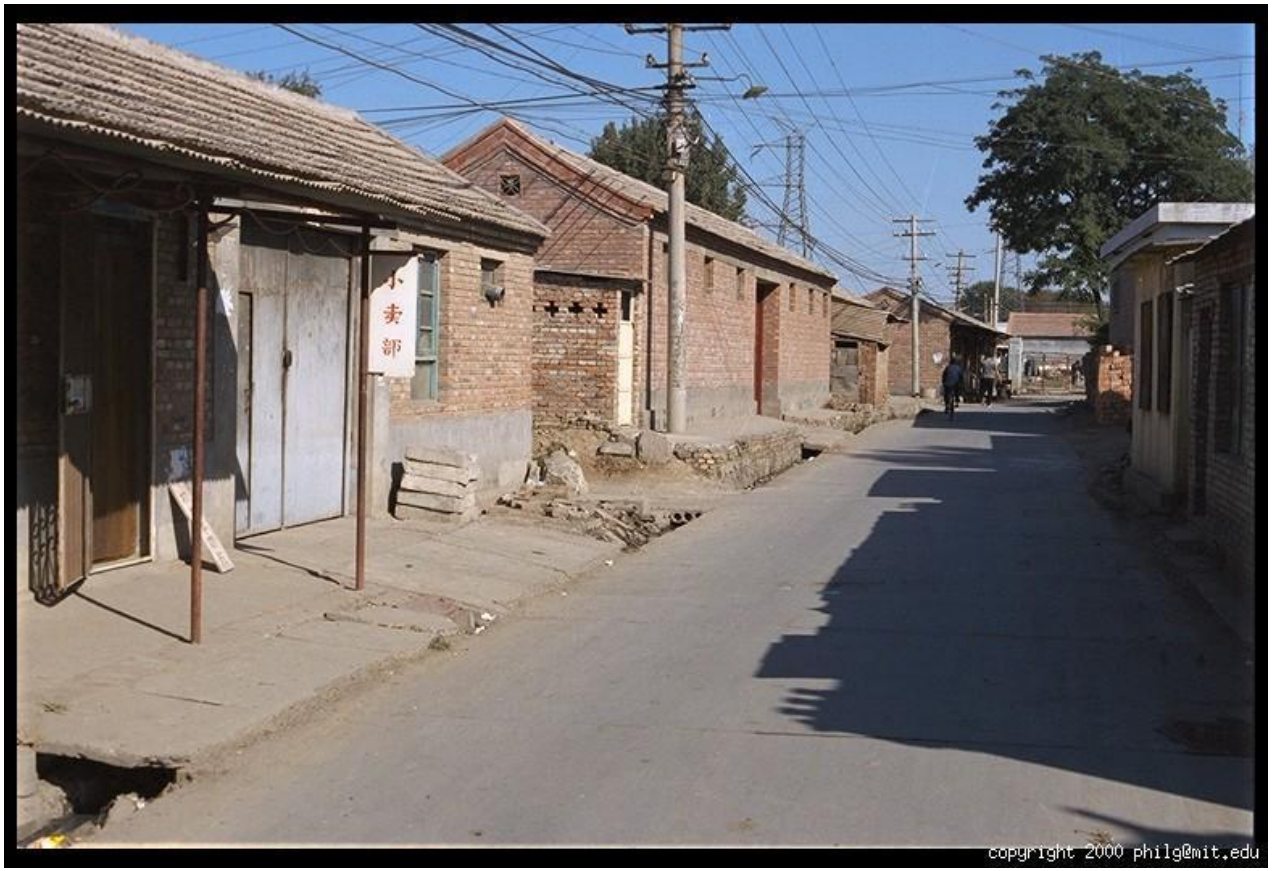
Most information in intensity



Only intensity shown – constant color

Source: D. Hoiem

Most information in intensity

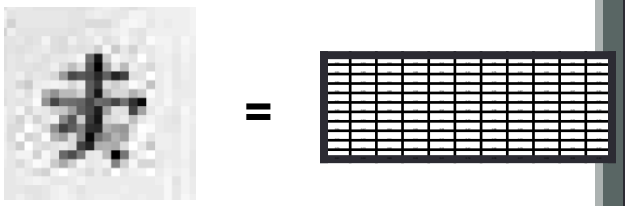


Original image

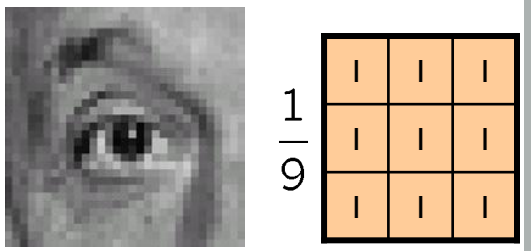
Source: D. Hoiem

Important Take Home Messages

- Image is a matrix of numbers (light intensities at different orientations)
  - Interpretted mainly through local comparisons



- Linear filtering is sum of dot product at each position
  - Can smooth, sharpen, translate (among many other uses)



- Attend to details: filter size, extrapolation, cropping

- Color spaces beyond RGB sometimes useful