NANJING UNIVERSITY

计算机视觉表征与识别

Chapter 2: Images and Filtering

王利民

媒体计算课题组

http://mcg.nju.edu.cn/

Overview

- What is an image?
- Image formation: light and color
- Image transformation
- Image noise and image smoothing
- Convolution operation
- Media filter

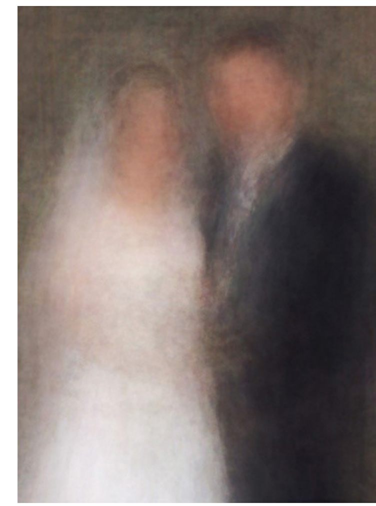
Images as matrices

Result of averaging 100 similar snapshots

Little Leaguer

Kids with Santa

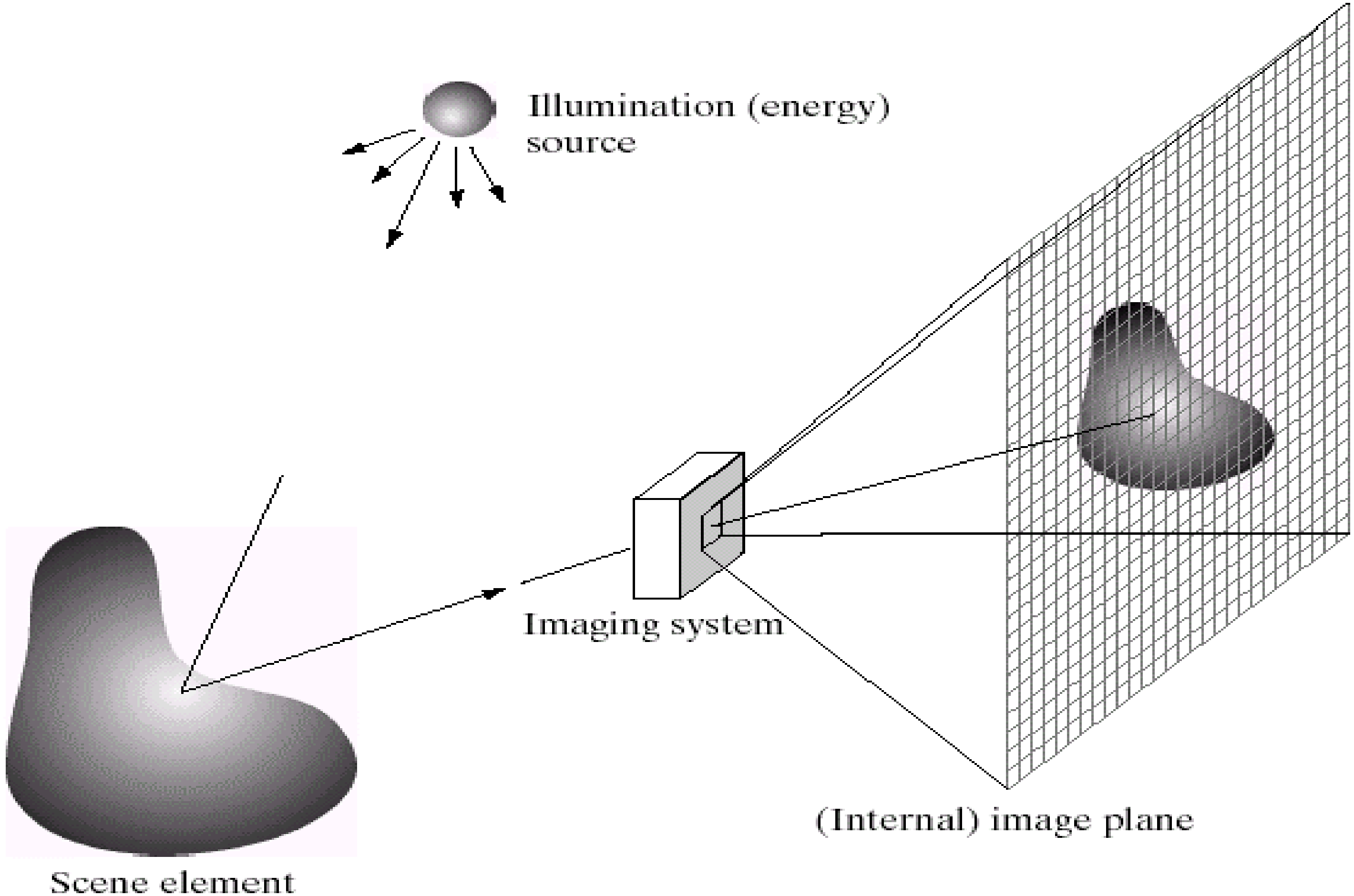
The Graduate



Newlyweds

From: *100 Special Moments*, by Jason Salavon (2004)
http://salavon.com/SpecialMoments/SpecialMoments.shtml

How light is recorded



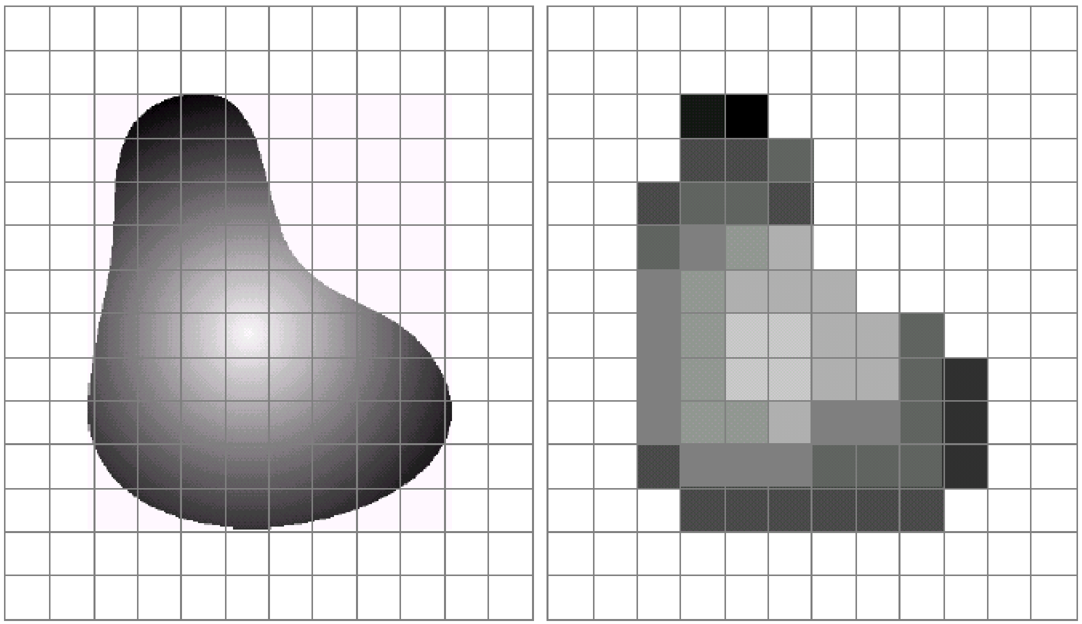
Digital camera

A digital camera replaces film with a sensor array

- Each cell in the array is light-sensitive diode that converts photons to electrons
- http://electronics.howstuffworks.com/digital-camera.htm

Slide by S. Seitz

Sensor array



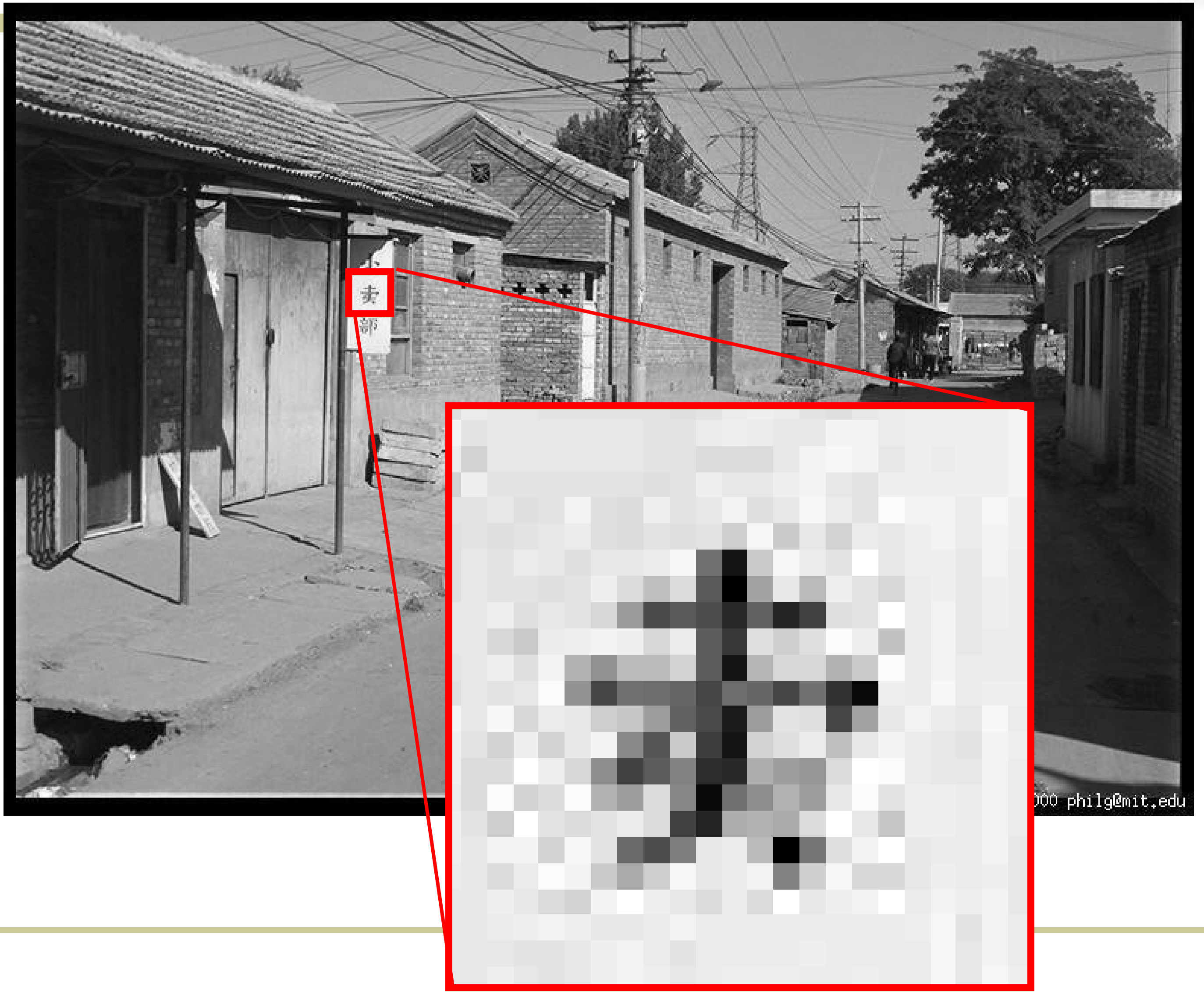
a b

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

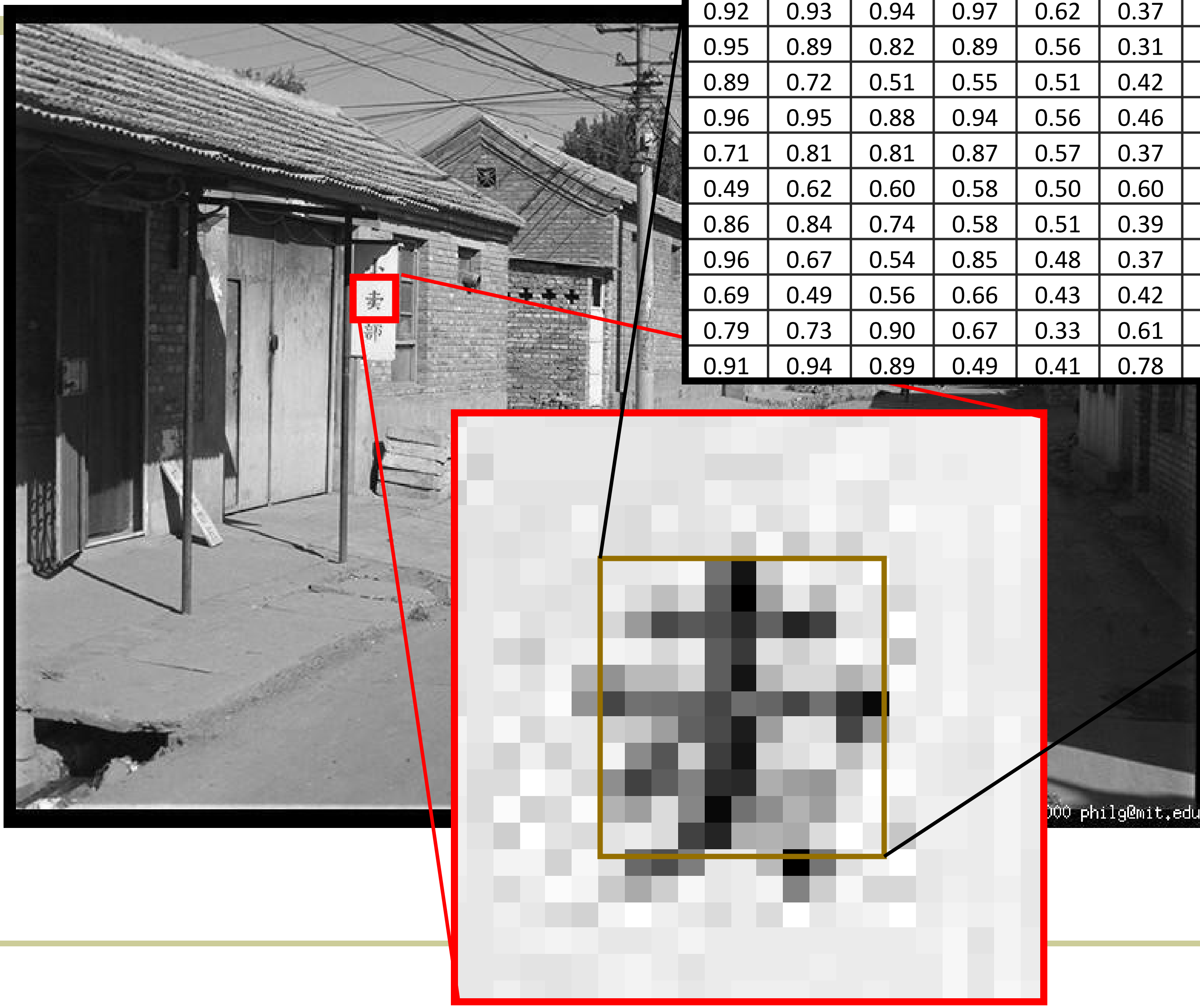
CMOS sensor

Each sensor cell records amount of light coming in at a small range of orientations

The raster image (pixel matrix)

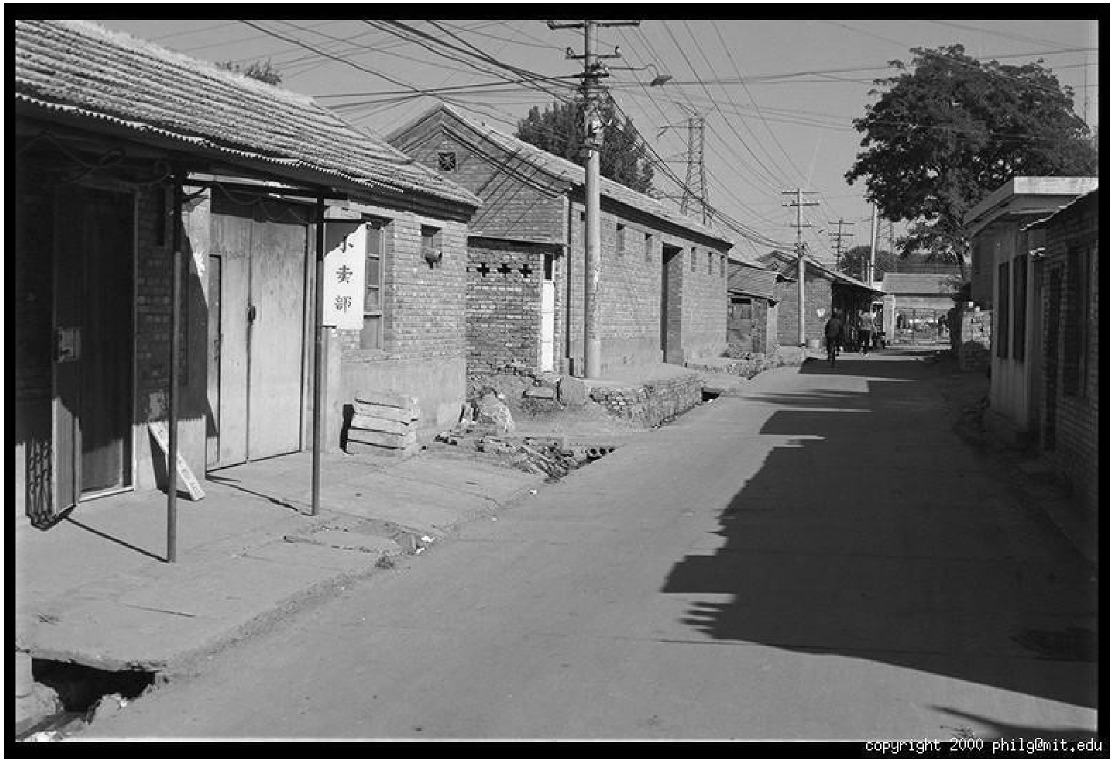


The raster image (pixel matrix)



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

What determines a pixel's intensity

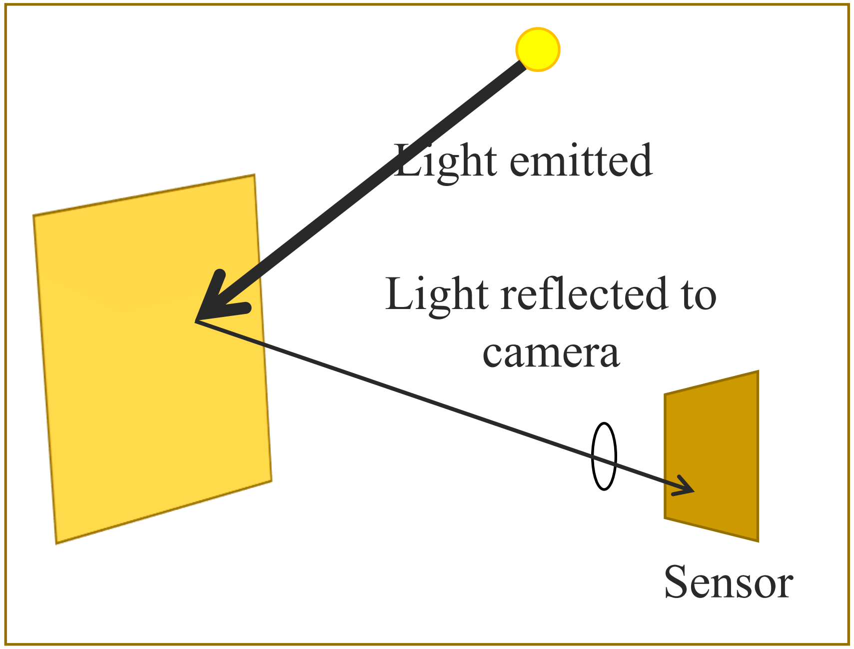


Overview

- What is an image?
- **Image formation: light and color**
- Image transformation
- Image noise and image smoothing
- Convolution operation
- Media filter

How much light is recorded

- Major factors
  - Illumination strength and direction
  - Surface geometry
  - Surface material
  - Nearby surfaces
  - Camera gain/exposure



Intensity and Surface Orientation

Intensity depends on illumination angle because less light comes in at oblique angles.

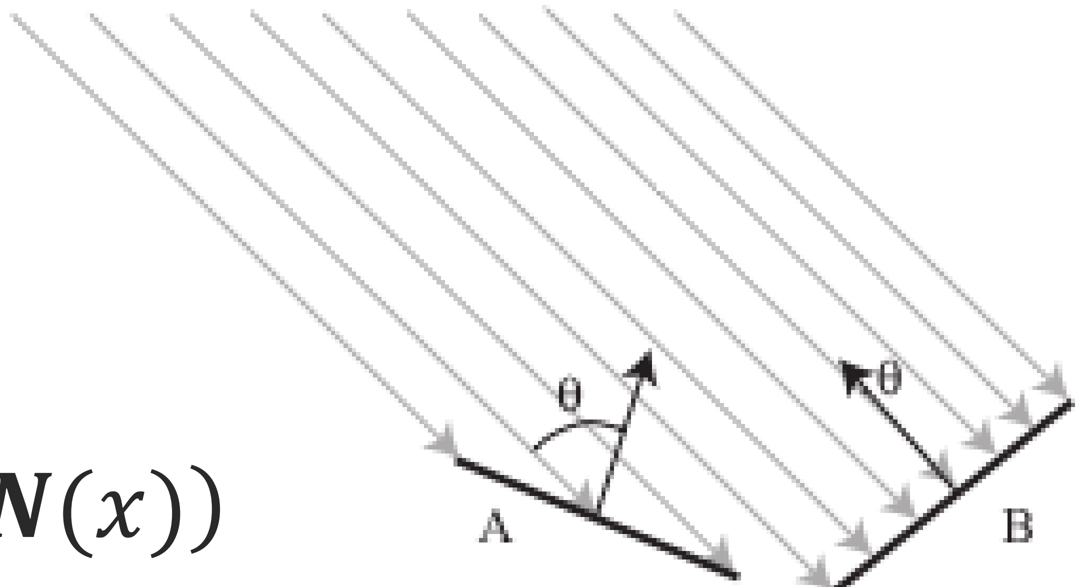
ρ = albedo

$\boldsymbol{S}$ = directional source

$\boldsymbol{N}$ = surface normal

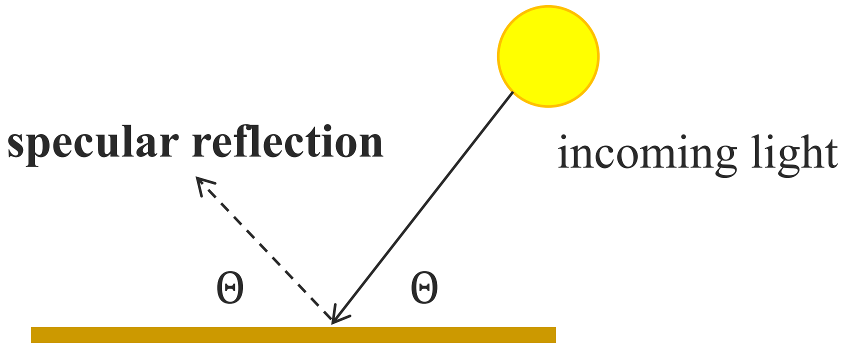
I = reflected intensity

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

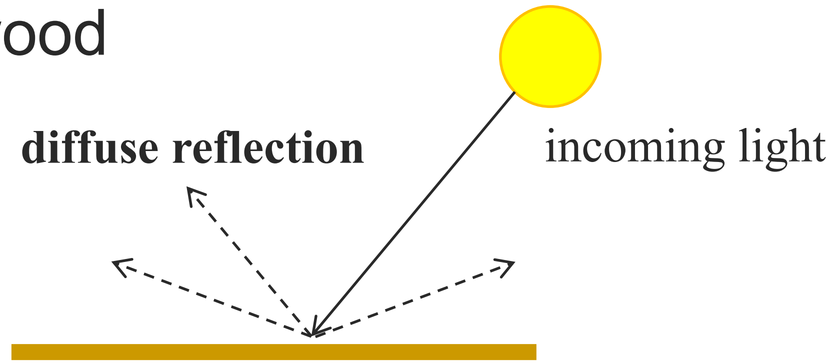


Basic models of reflection

- Specular: light bounces off at the incident angle
  - E.g., mirror

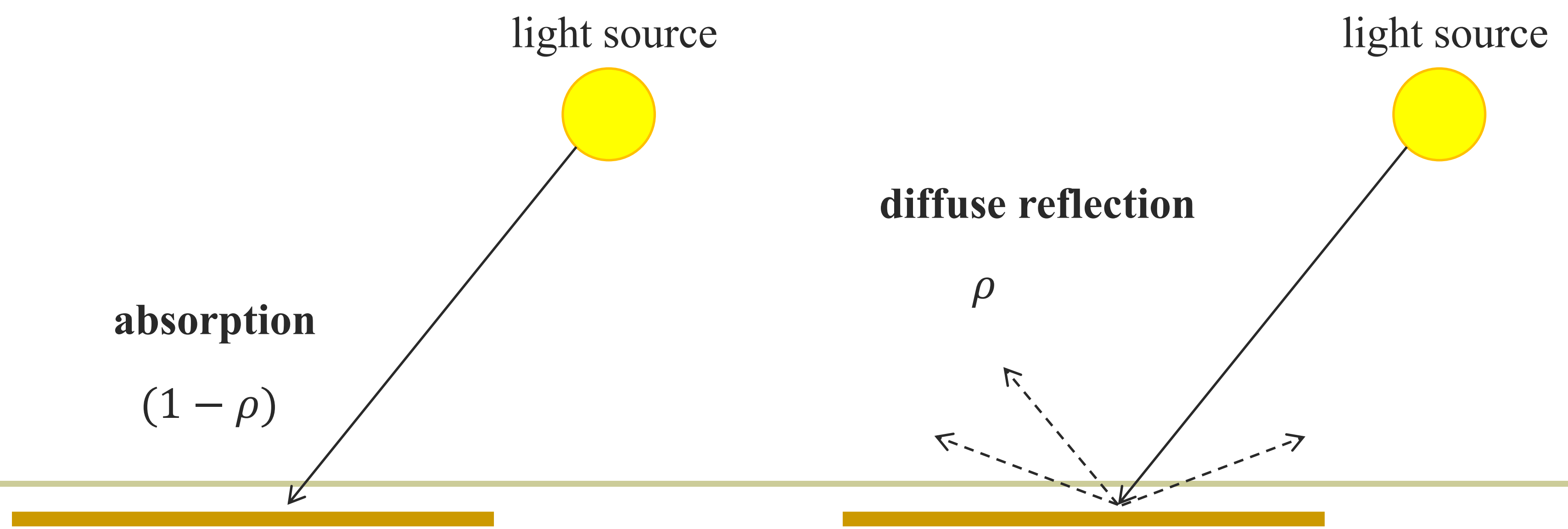


- Diffuse: light scatters in all directions
  - E.g., brick, cloth, rough wood



Lambertian reflectance model

- Some light is absorbed (function of albedo ρ)
- Remaining light is scattered (diffuse reflection)
- Examples: soft cloth, concrete, matte paints

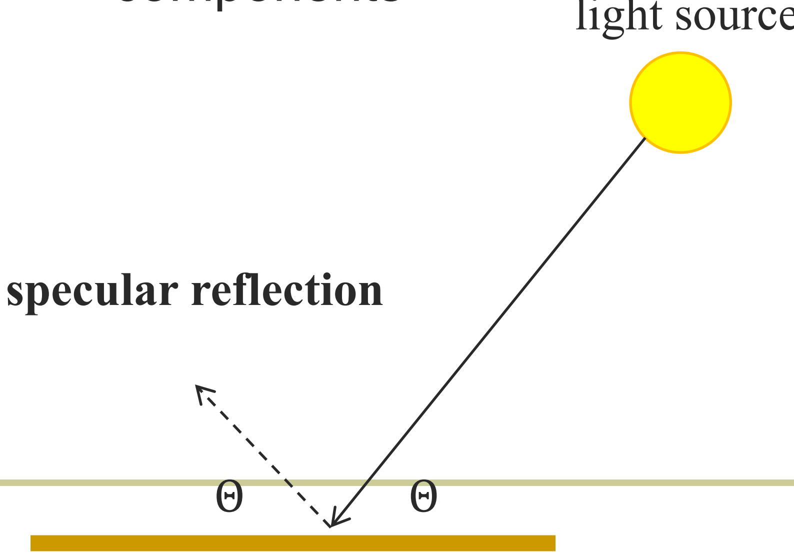


Specular Reflection

Flickr, by suzysputnik

- Reflected direction depends on light orientation and surface normal
  - E.g., mirrors are fully specular
  - Most surfaces can be modeled with a mixture of diffuse and specular components

Flickr, by piratejohnny



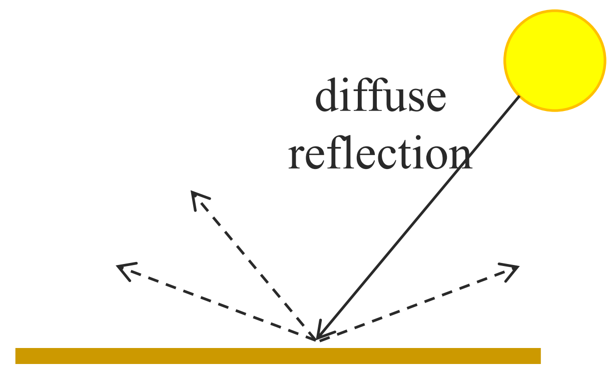
Most surfaces have both specular and diffuse components

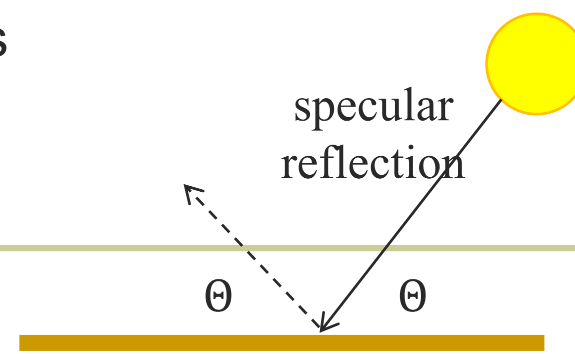
- Specularity = spot where specular reflection dominates (typically reflects light source)

Typically, specular component is small

Recap

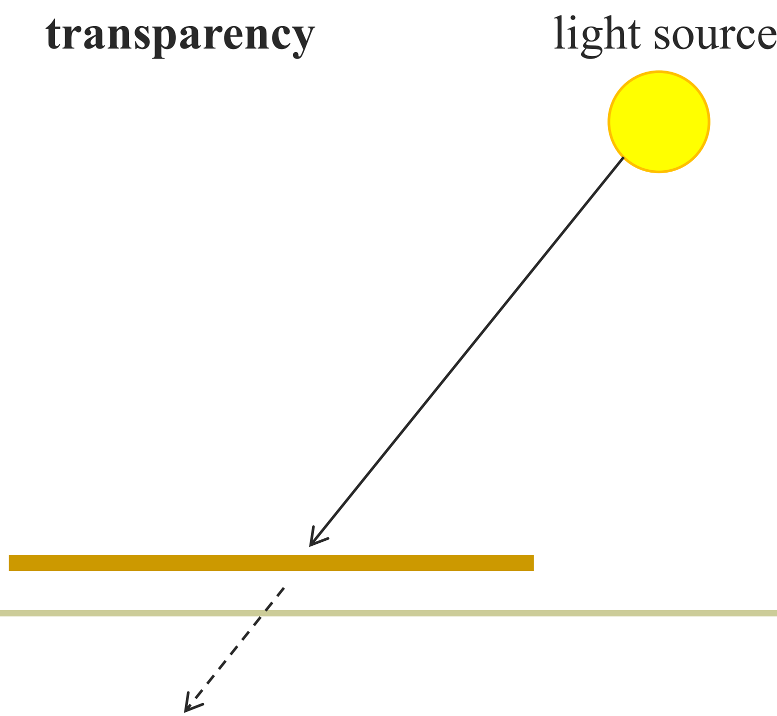
- When light hits a typical surface
  - Some light is absorbed (1-ρ)
    - More absorbed for low albedos
  - Some light is reflected diffusely
    - Independent of viewing direction
  - Some light is reflected specularly
    - Light bounces off (like a mirror), depends on viewing direction



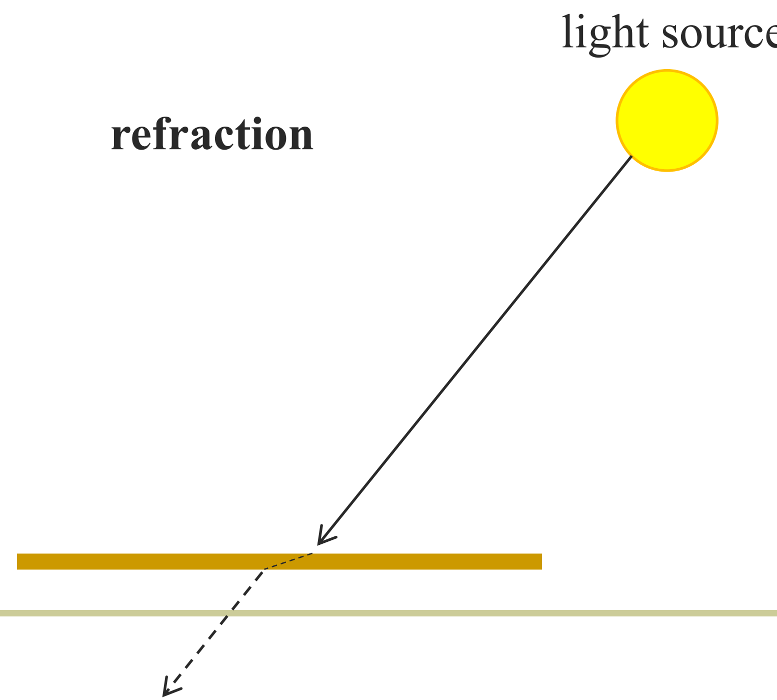


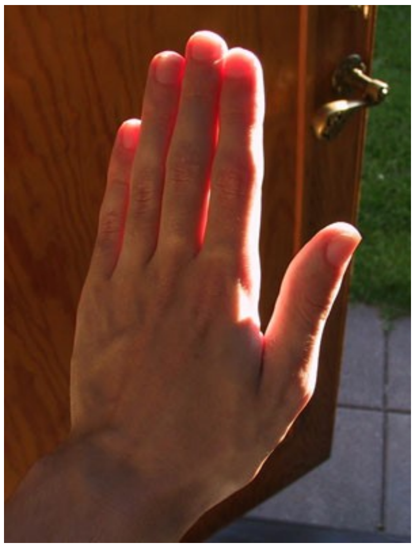
Other possible effects

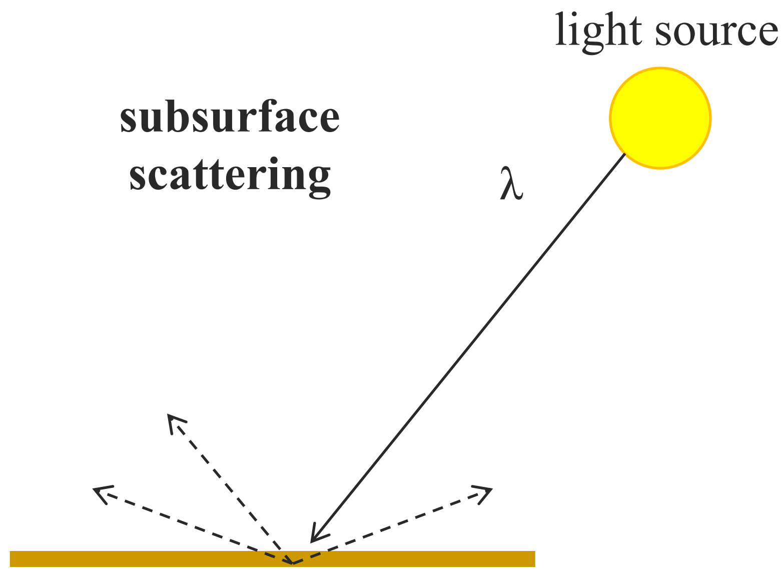
transparency



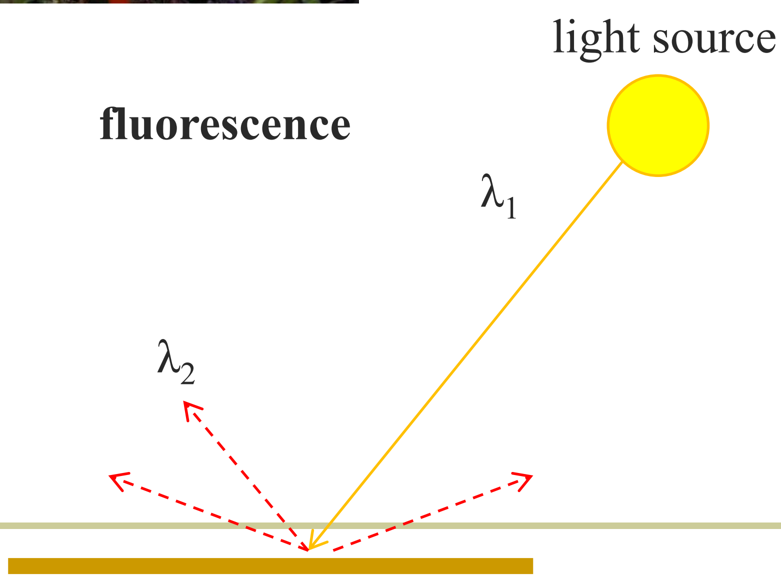
refraction



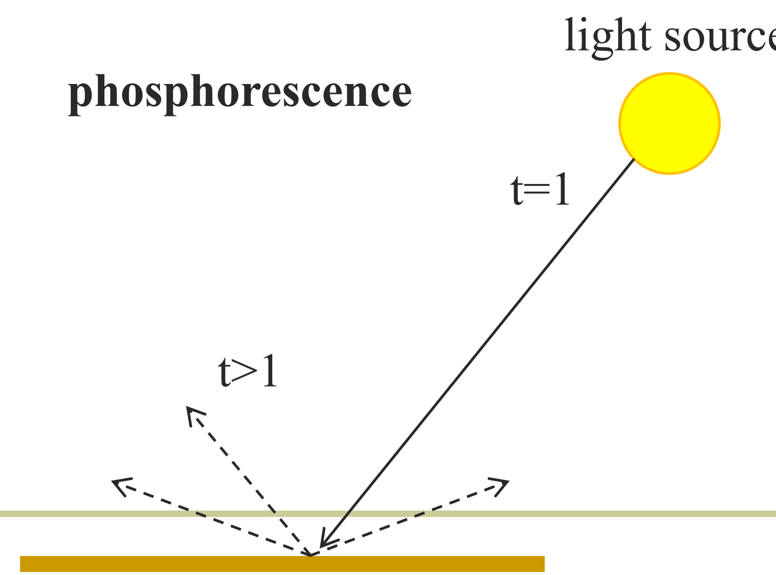



light source
subsurface
scattering
λ

fluorescence

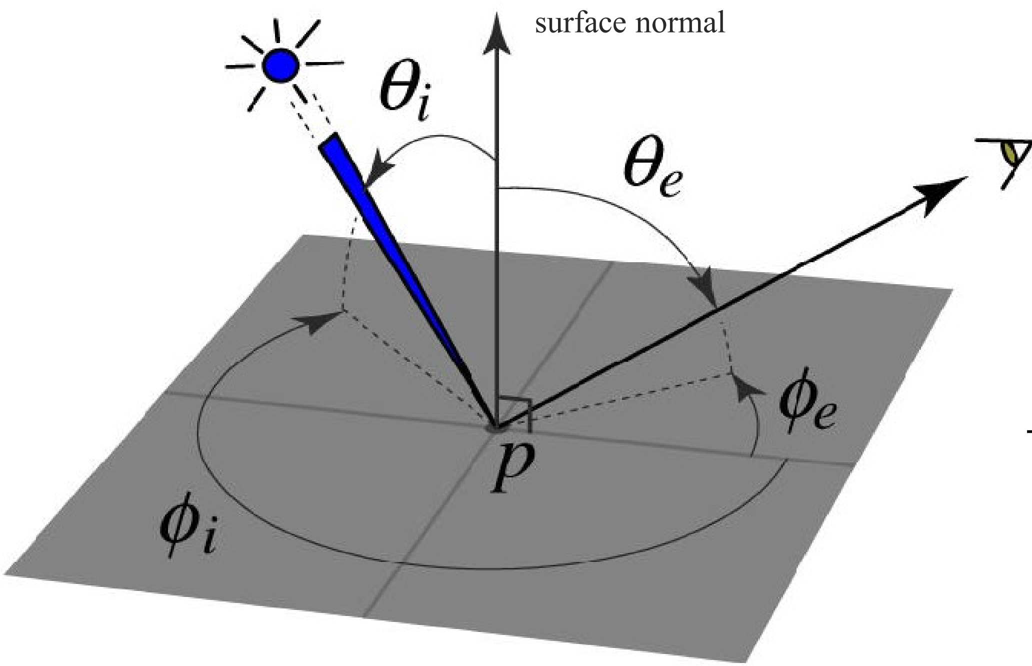


phosphorescence



BRDF: Bidirectional Reflectance Distribution Function

- Model of local reflection that tells how bright a surface appears when viewed from one direction when light falls on it from another



$$\rho(\theta_i, \phi_i, \theta_e, \phi_e; \lambda) =$$

$$\frac{L_e(\boldsymbol{\theta}_e, \boldsymbol{\phi}_e)}{E_i(\boldsymbol{\theta}_i, \boldsymbol{\phi}_i)} = \frac{L_e(\boldsymbol{\theta}_e, \boldsymbol{\phi}_e)}{L_i(\boldsymbol{\theta}_i, \boldsymbol{\phi}_i) \cos \boldsymbol{\theta}_i \, d\boldsymbol{\omega}}$$

Slide credit: S. Savarese

BRDFs can be incredibly complicated…



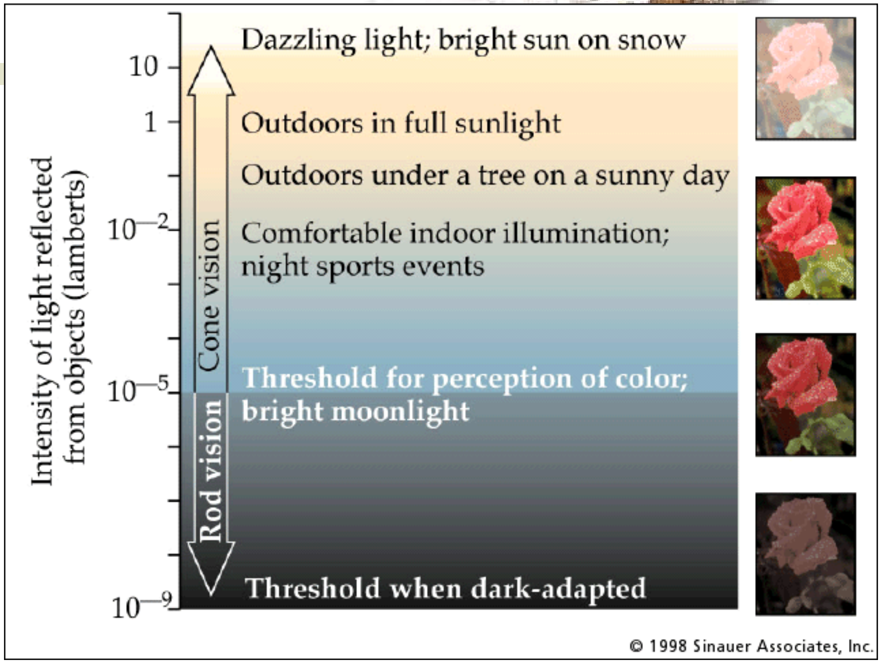


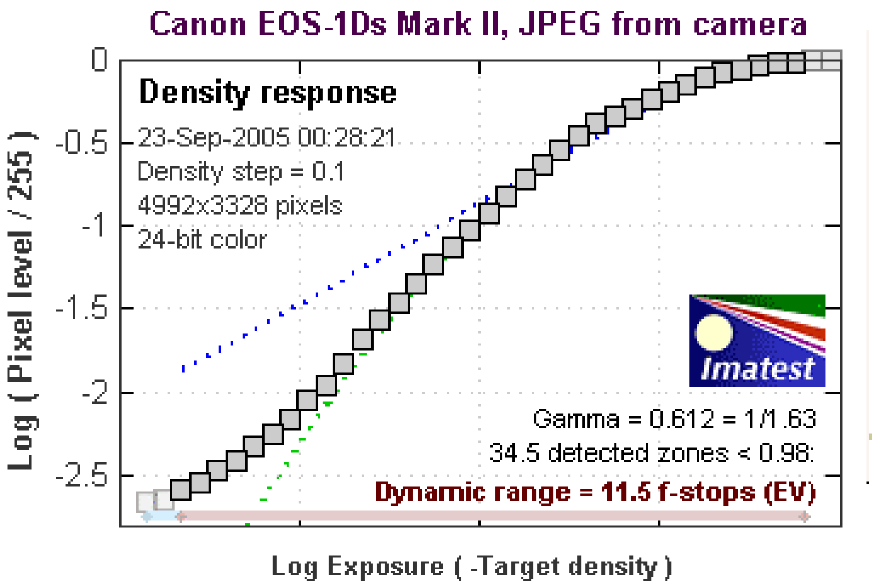



Slide Credit: L. Lazebnik

Dynamic range and camera response

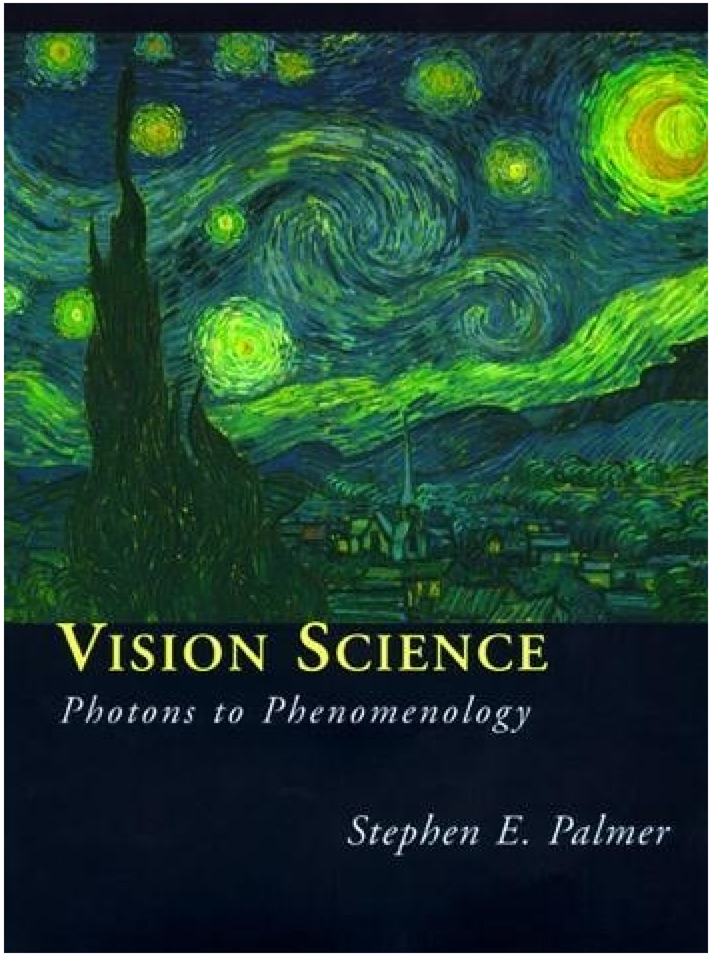
- Typical scenes have a huge dynamic range
- Camera response is roughly linear in the mid range (15 to 240) but non-linear at the extremes
  - called saturation or undersaturation





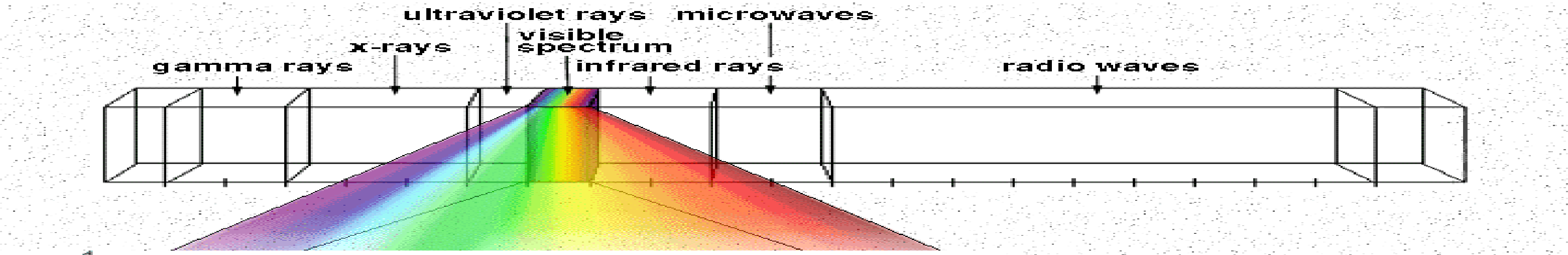
What is color?

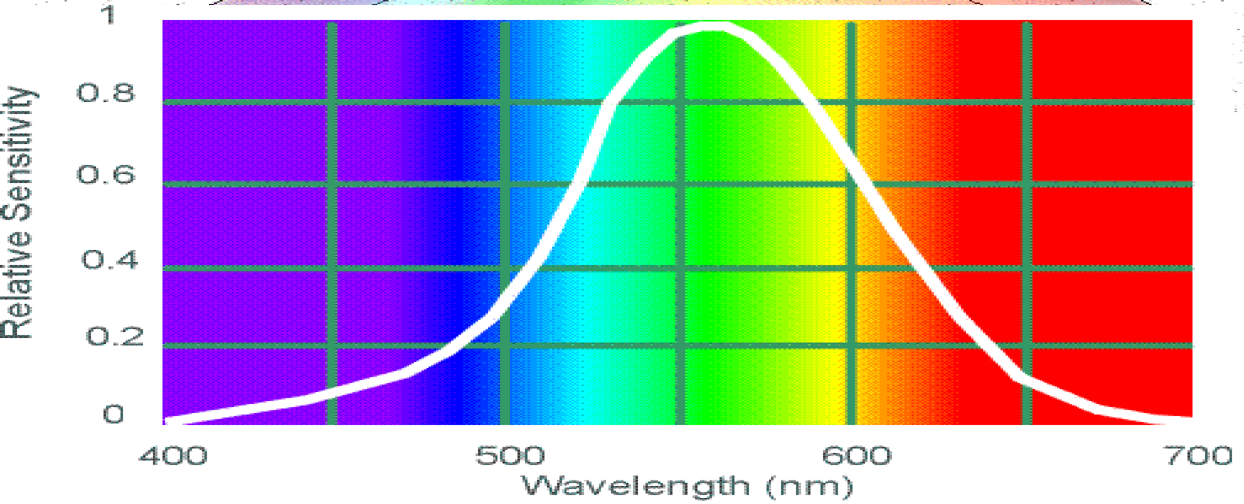
- Color is the result of interaction between physical light in the environment and our visual system
- Color is a psychological property of our visual experiences when we look at objects and lights, *not* a physical property of those objects or lights (S. Palmer, *Vision Science: Photons to Phenomenology*)



Color

Light is composed of a spectrum of wavelengths



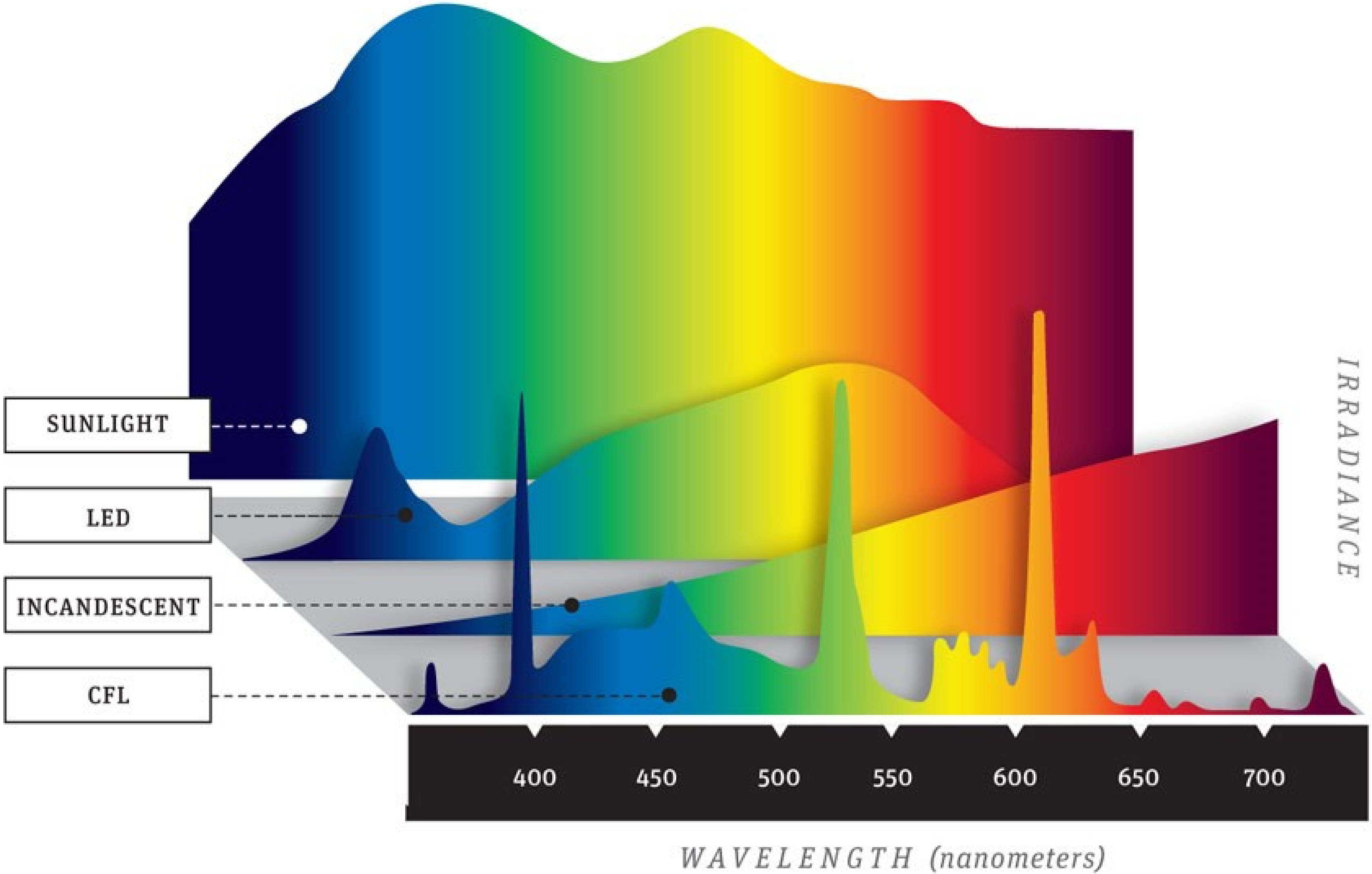


Human Luminance Sensitivity Function

Slide Credit: Efros

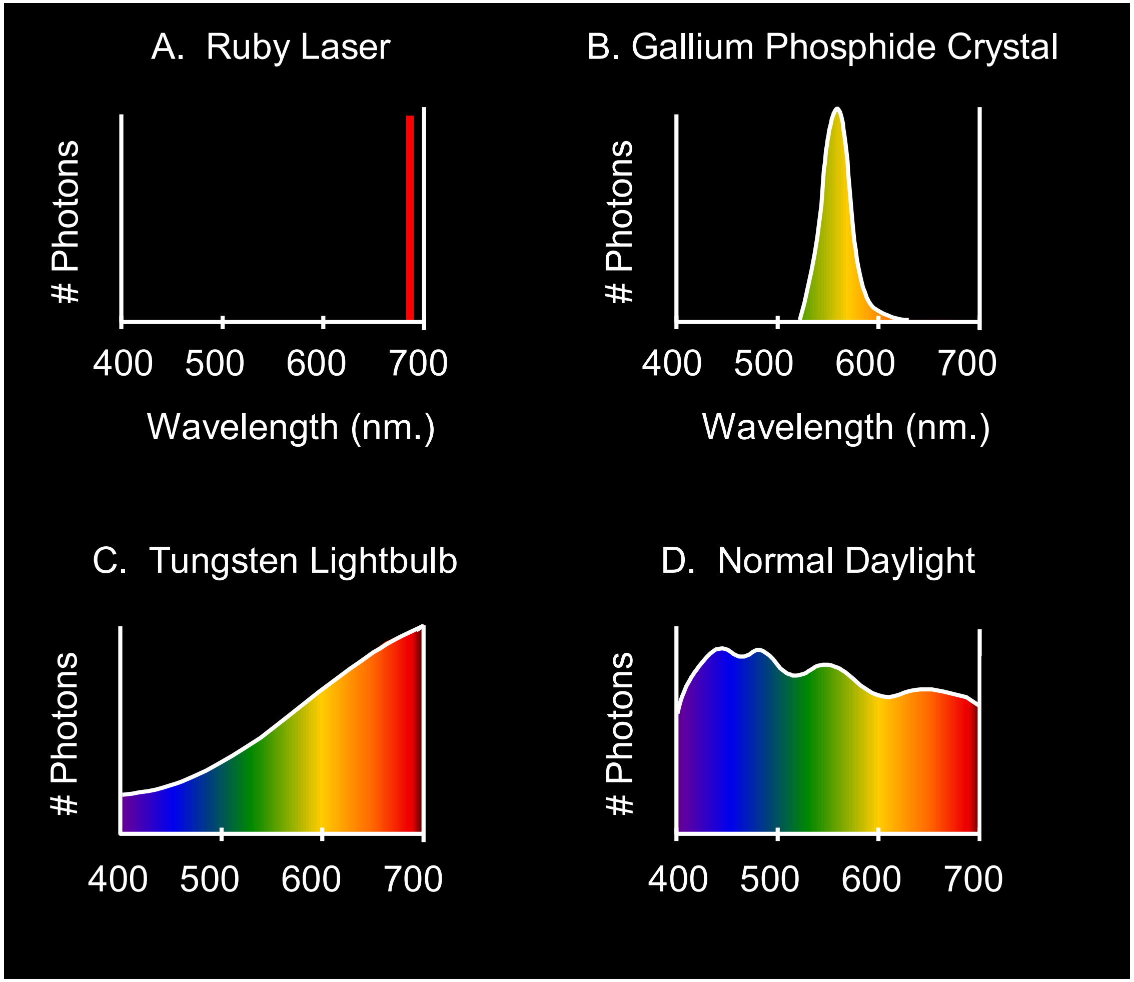
http://www.yorku.ca/eye/photopik.htm

Spectra of light sources



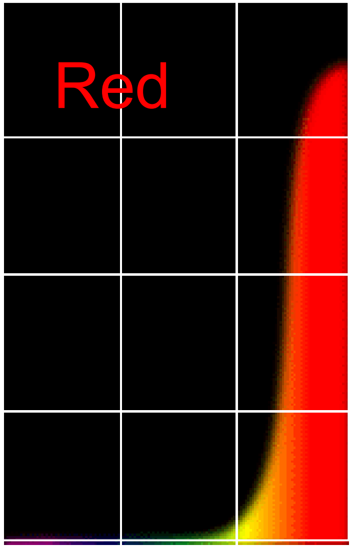
Source: Popular Mechanics

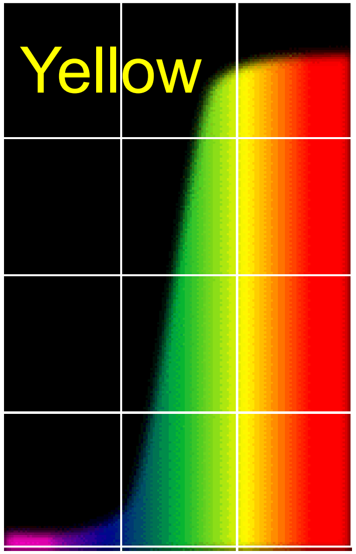
Some examples of the spectra of light sources

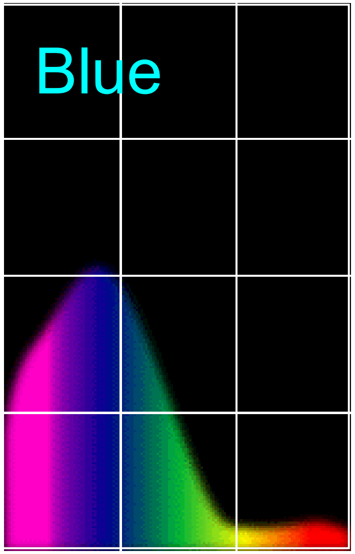


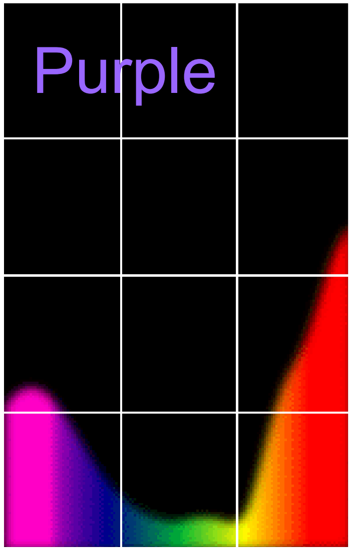
Some examples of the reflectance spectra of surfaces



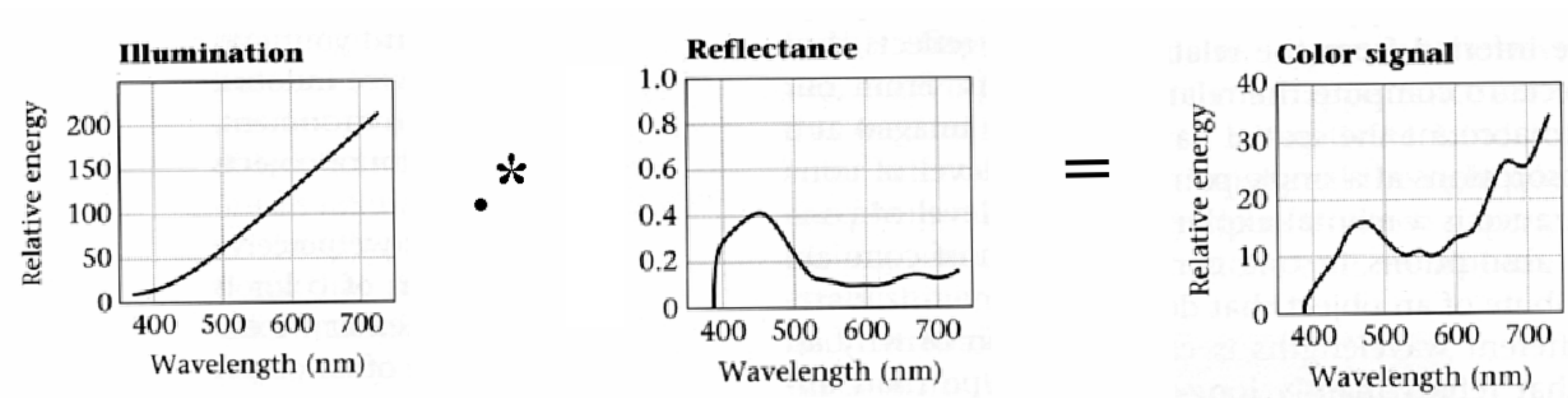






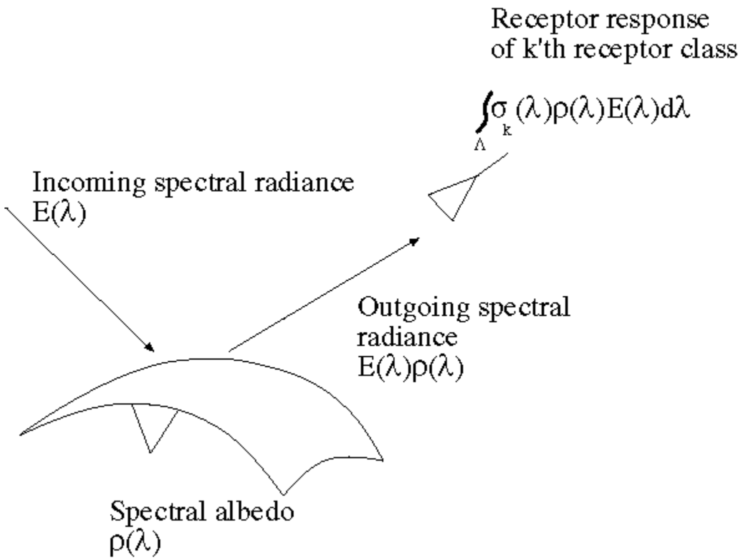
Interaction of light and surfaces

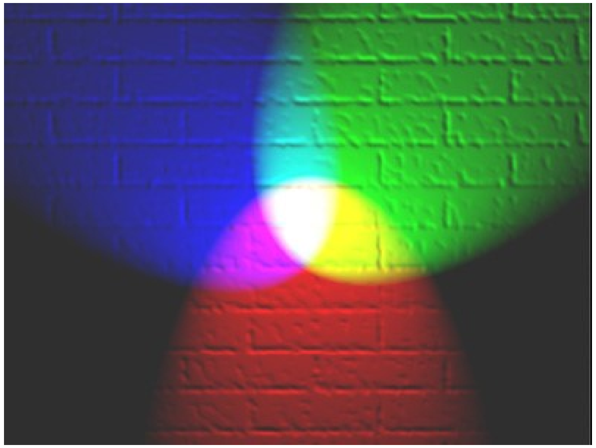
- Reflected color is the result of interaction of light source spectrum with surface reflectance



The color of objects

- Colored light arriving at the camera involves two effects
  - The color of the light source (illumination + inter-reflections)
  - The color of the surface



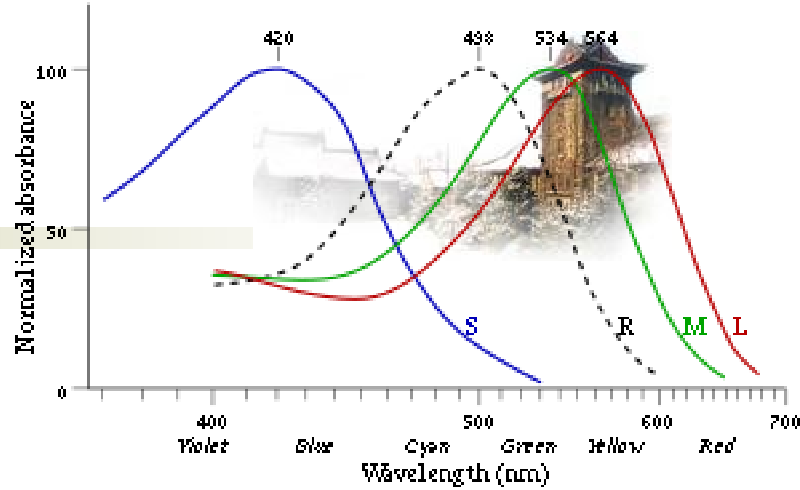


Slide: Forsyth

Why RGB?

If light is a spectrum, why are images RGB?

Human color receptors

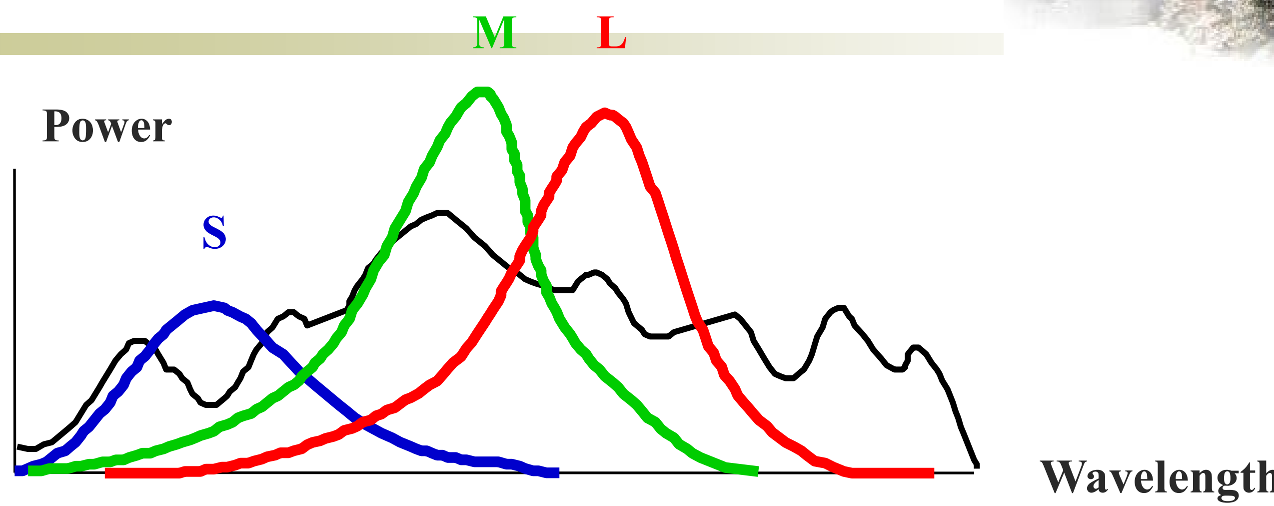


Long (red), Medium (green), and Short (blue) cones, plus intensity rods

- Fun facts
  - "M" and "L" on the X-chromosome
    - That's why men are more likely to be color blind (see what it's like: http://www.vischeck.com/vischeck/vischeckImage.php)
  - "L" has high variation, so some women are tetrachromatic
  - Some animals have 1 (night animals), 2 (e.g., dogs), 4 (fish, birds), 5 (pigeons, some reptiles/amphibians), or even 12 (mantis shrimp) types of cones

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

Color perception



Rods and cones act as *filters* on the spectrum

- To get the output of a filter, multiply its response curve by the spectrum, integrate over all wavelengths
  - Each cone yields one number

How can we represent an entire spectrum with three numbers?

We can't! Most of the information is lost

As a result, two different spectra may appear indistinguishable

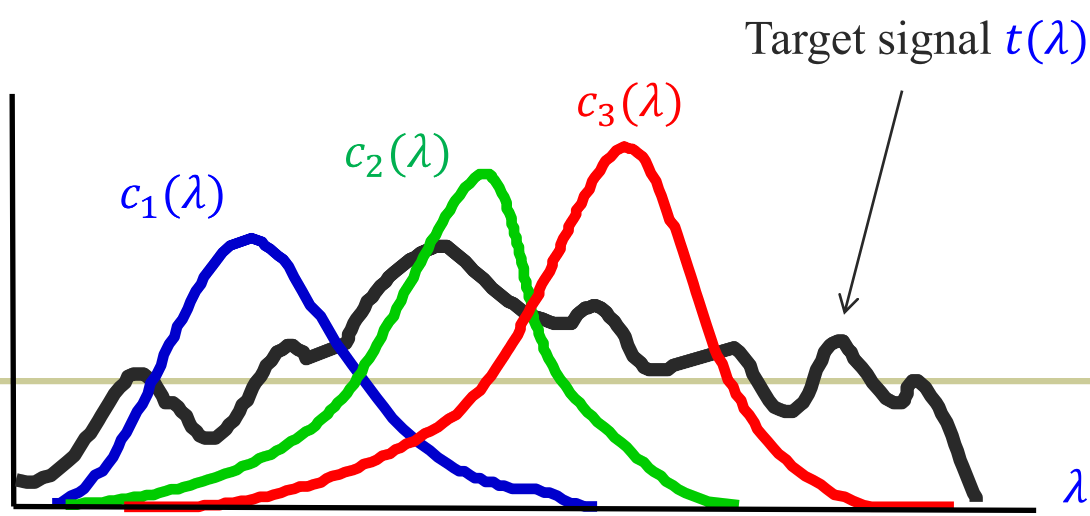
such spectra are known as metamers

Slide by Steve Seitz

Matching functions

- Let $t(\lambda)$ be the spectrum of the target signal
- Let $c_1(\lambda)$, $c_2(\lambda)$, and $c_3(\lambda)$ be the *matching functions*, or the amounts of each primary needed to match monochromatic sources with wavelengths λ
- Then the coordinates of t in the corresponding linear space are given by

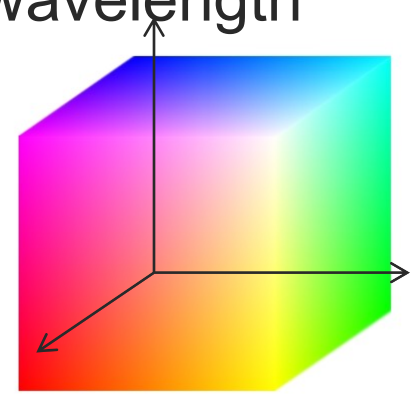
$$w_p = \int t(\lambda) c_p(\lambda) d\lambda$$



Matching functions act as filters on the target spectrum, like response curves of color receptors!

Linear color spaces

- Defined by a choice of three **primaries**
- The **coordinates** of a color are given by the weights of the primaries used to match it
- In addition to primaries, need to specify **matching functions:** the amount of each primary needed to match a monochromatic light source at each wavelength

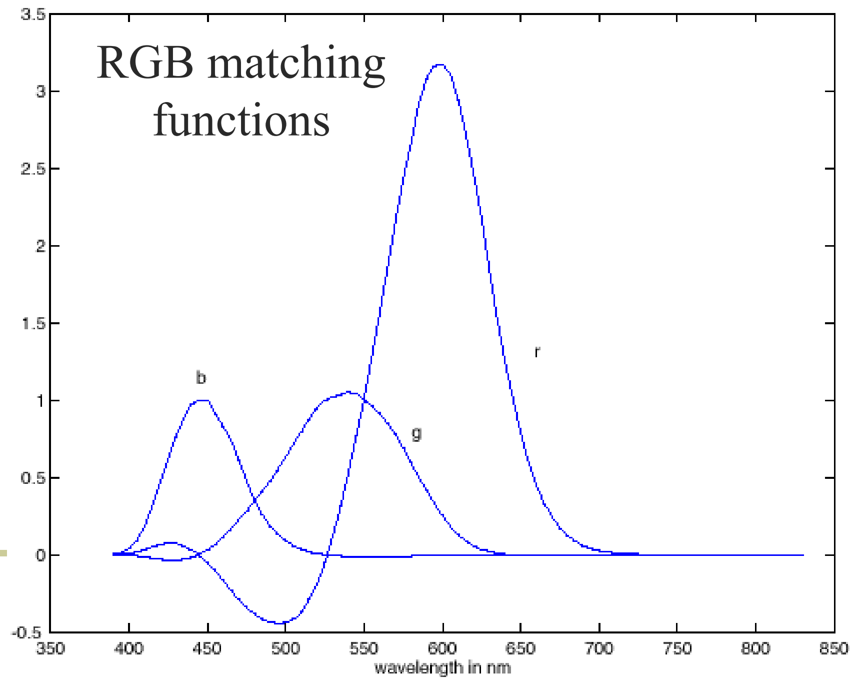


RGB primaries

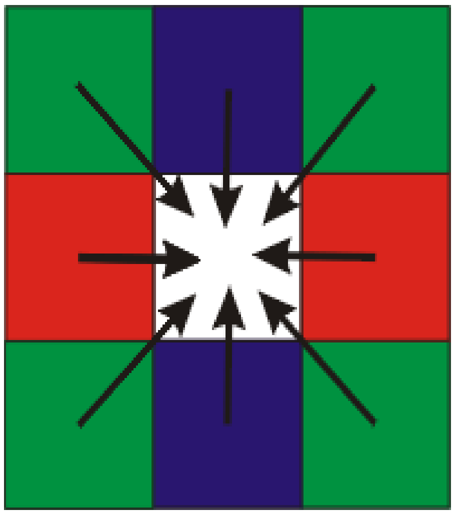
$p_1 = 645.2$ nm

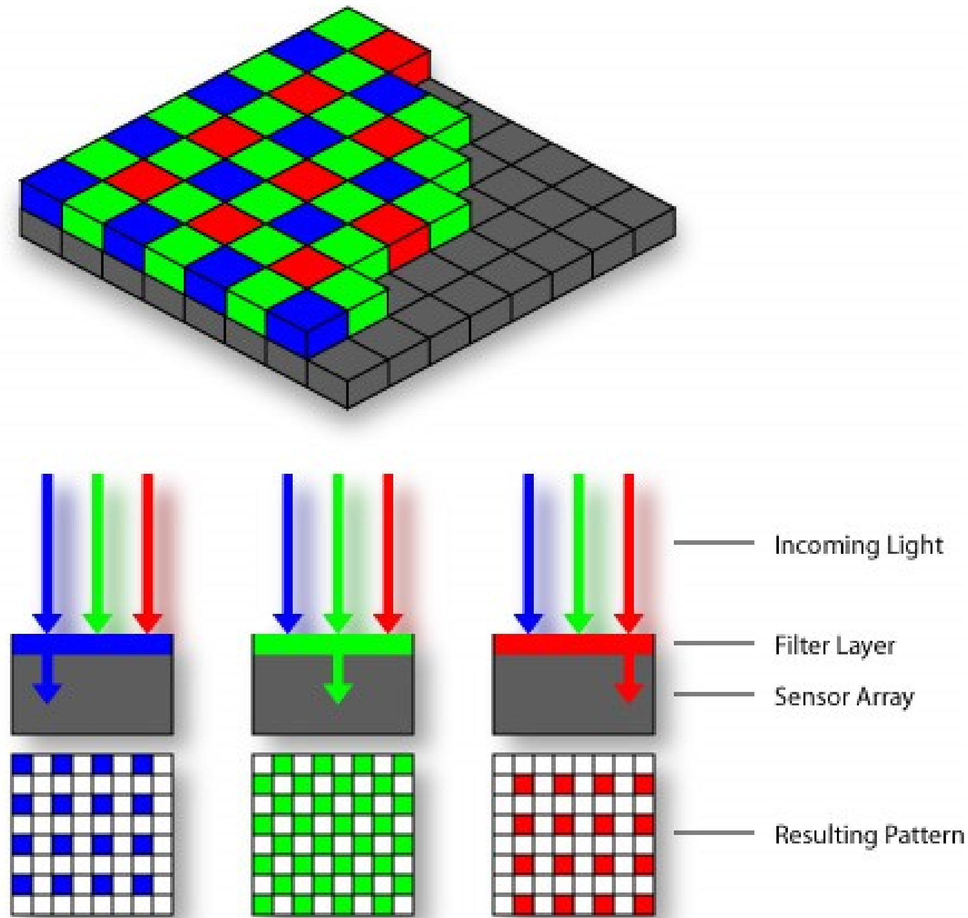
$p_2 = 525.3$ nm

$p_3 = 444.4$ nm



Color Sensing: Bayer Grid



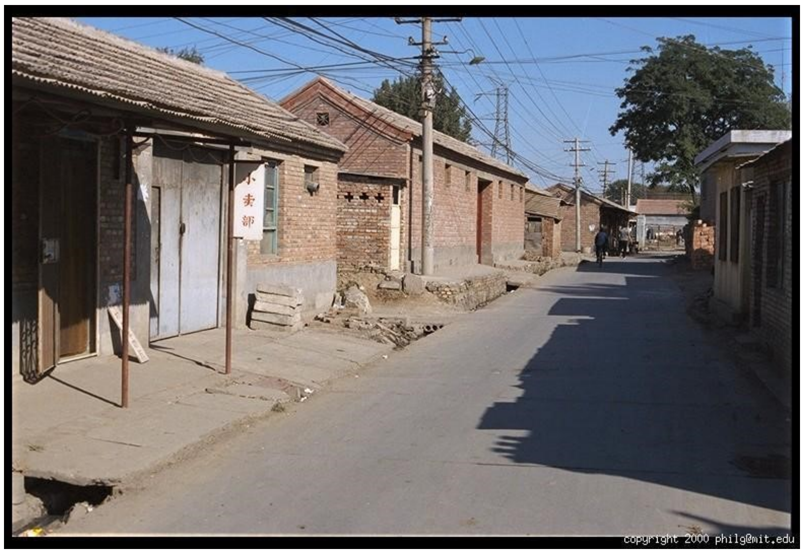


- Estimate RGB at each cell from neighboring values

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

Slide by Steve Seitz

Color Image


copyright 2000 philg@mit.edu


R
G
B
copyright 2000 philg@mit.edu

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
- imread(filename) returns a uint8 image (values 0 to 255)
  - Convert to double format (values 0 to 1) with im2double

column

row

R

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

G

									0.92	0.99
									0.95	0.91
									0.91	0.92
									0.97	0.95
									0.79	0.85
									0.45	0.33
									0.49	0.74
									0.82	0.93
									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

B

									0.92	0.99
									0.95	0.91
									0.91	0.92
									0.97	0.95
									0.79	0.85
									0.45	0.33
									0.49	0.74
									0.82	0.93
									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

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

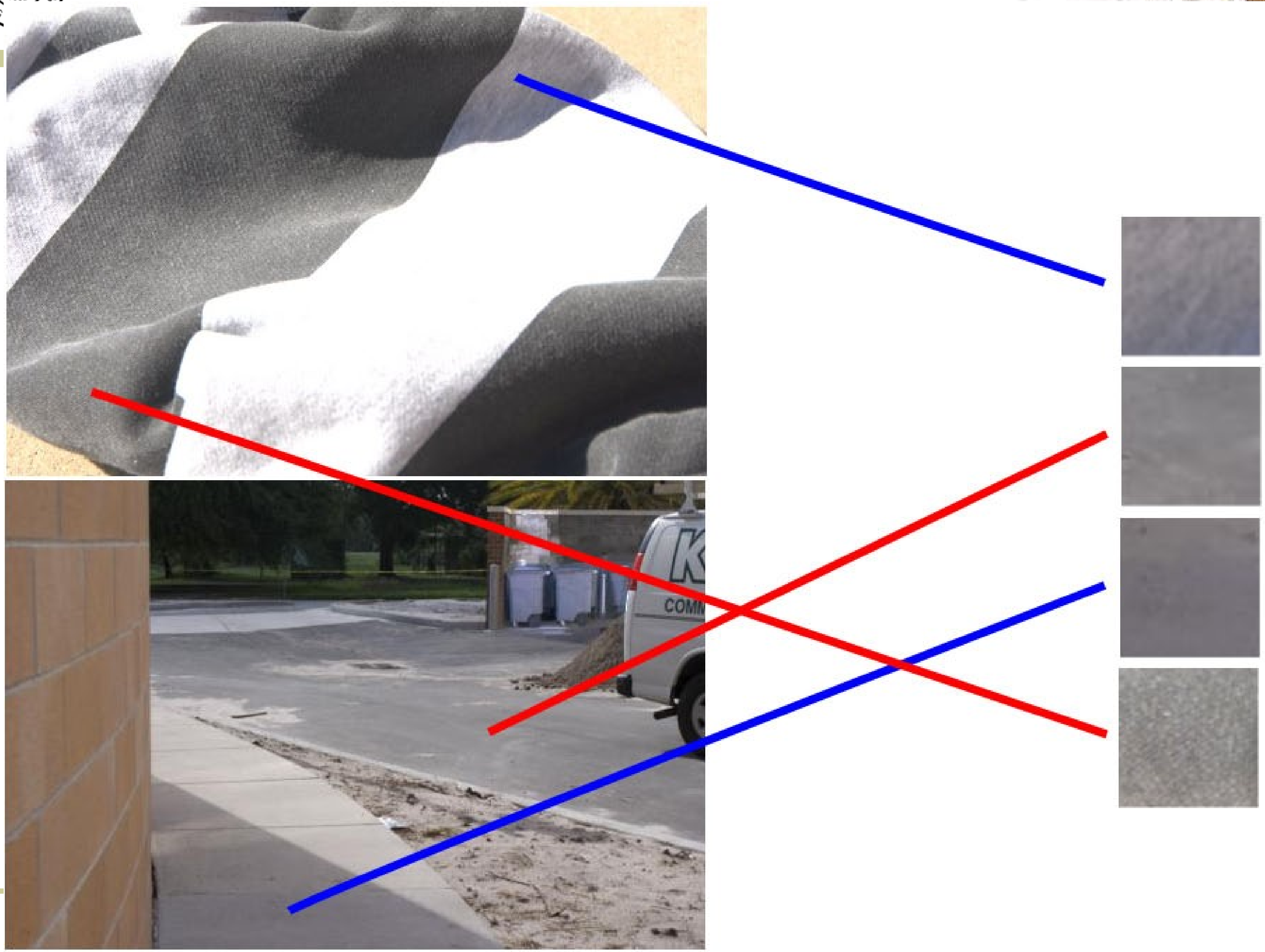
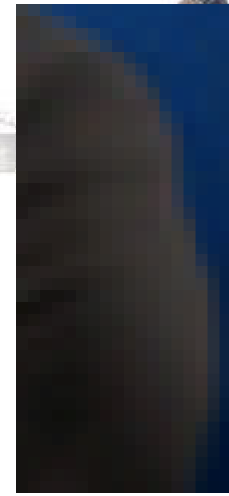


Photo by nickwheeleroz, Flickr

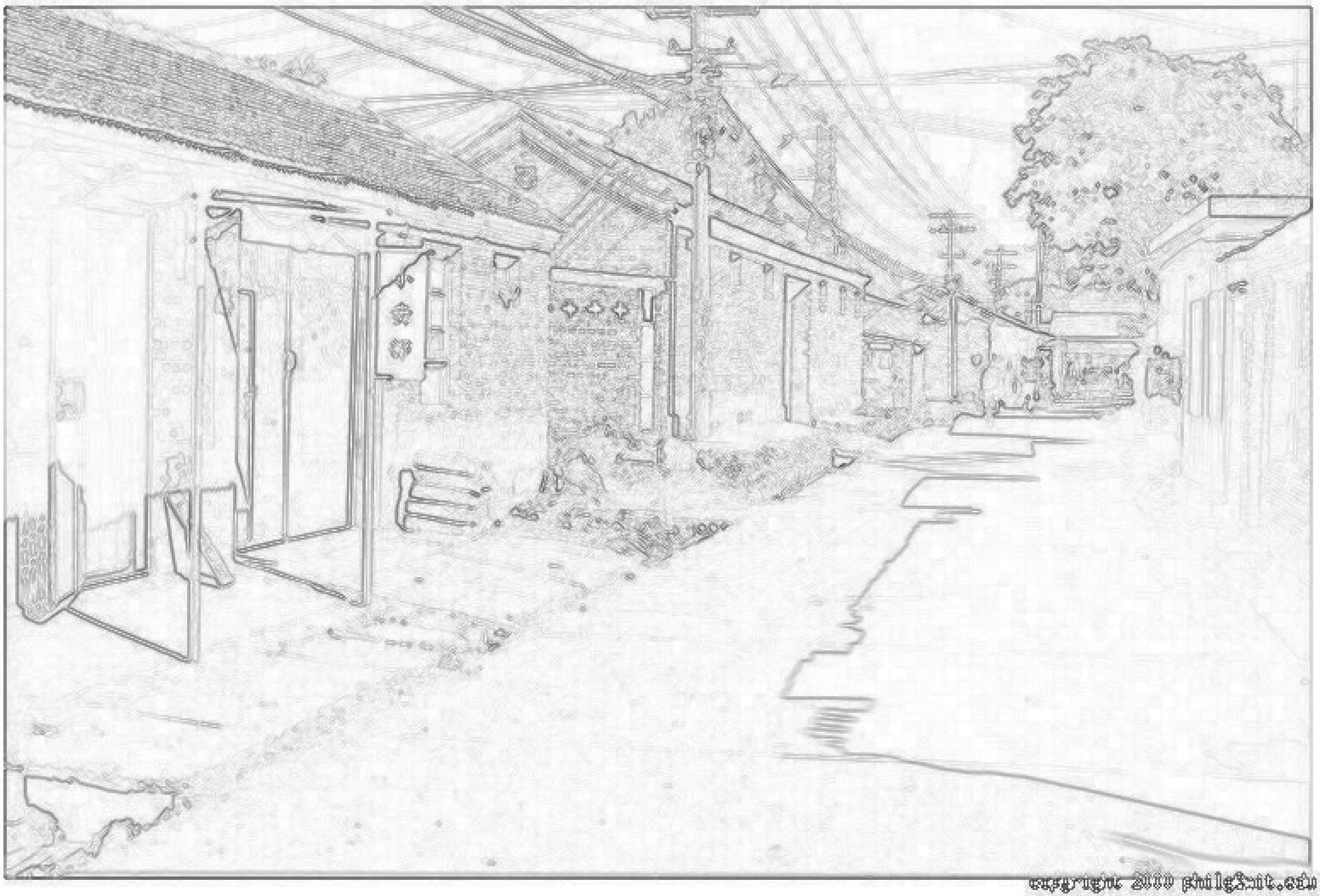


Slide: Forsyth

And yet we can interpret 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

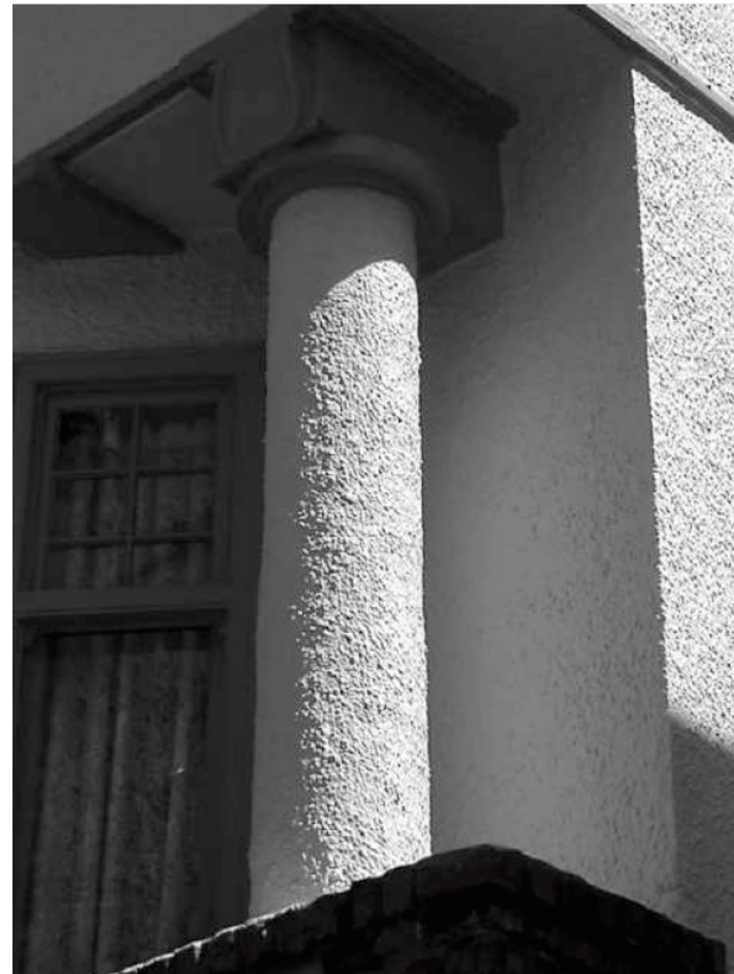


What is this?


HONDA

What differences in intensity tell us about shape

- Changes in surface normal
- Texture
- Proximity
- Indents and bumps
- Grooves and creases



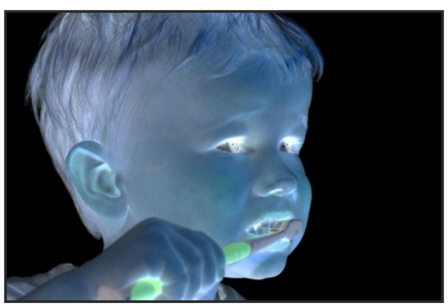
Photos Koenderink slides on image texture and the flow of light

Overview

- What is an image?
- Image formation: light and color
- **Image transformation**
- Image noise and image smoothing
- Convolution operation
- Media filter

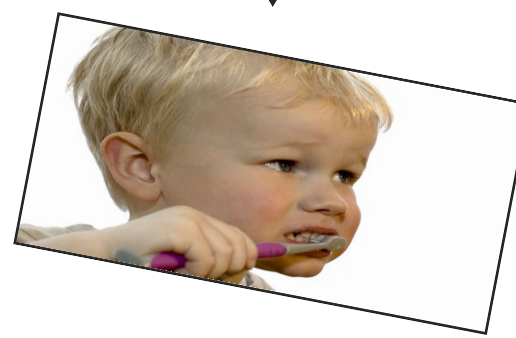
Image transformations

Filtering



changes pixel *values*

Warping



changes pixel *locations*

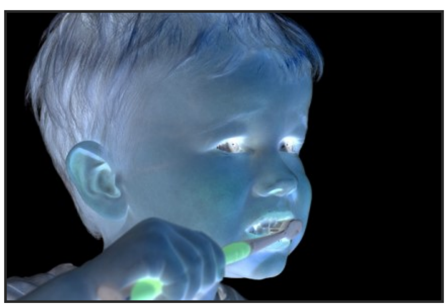
Image transformations

F

Filtering

$$G(\boldsymbol{x}) = h\{F(\boldsymbol{x})\}$$

G



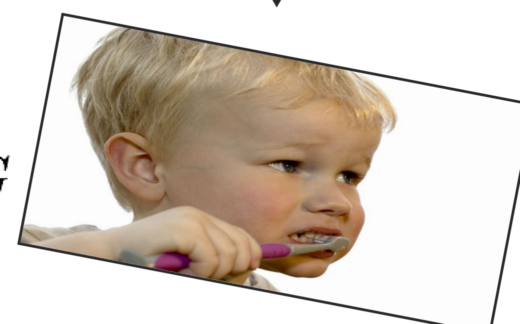
changes *range* of image function

F

Warping

$$G(\boldsymbol{x}) = F(h\{\boldsymbol{x}\})$$

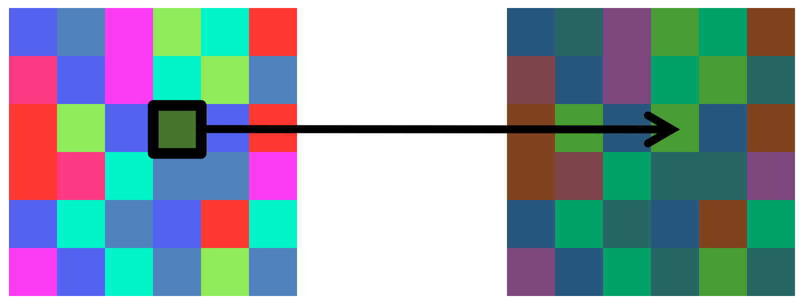
G



changes *domain* of image function

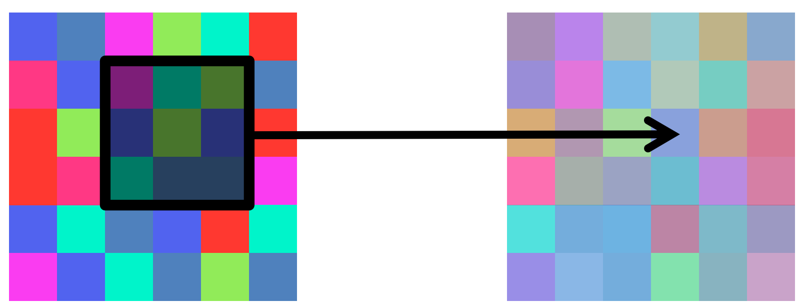
Image filtering

Point Operation



point processing

Neighborhood Operation



“filtering”

Examples of point processing

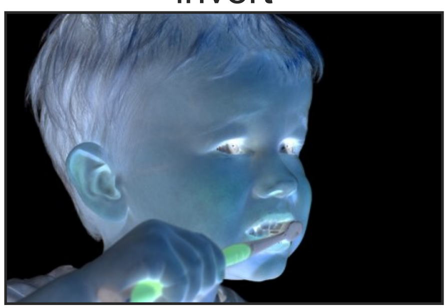
original

darken

lower contrast

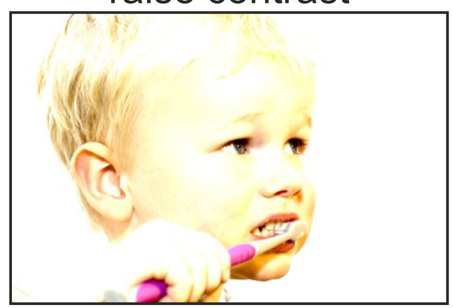
non-linear lower contrast

invert



lighten

raise contrast



non-linear raise contrast

Examples of point processing

How would you implement these?

original

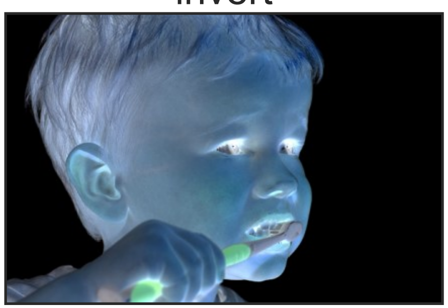
darken

lower contrast

non-linear lower contrast

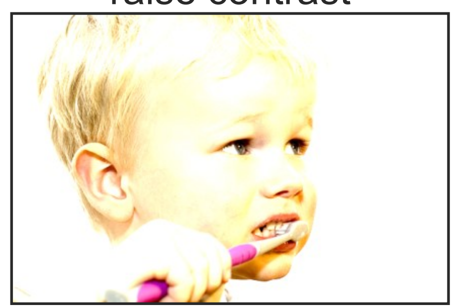
x

invert



lighten

raise contrast



non-linear raise contrast

Examples of point processing

How would you implement these?

original

x

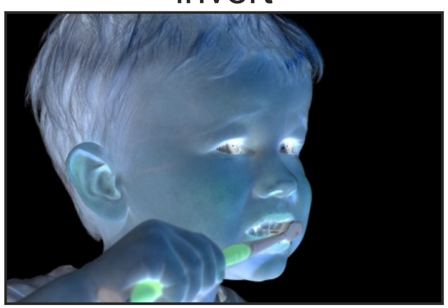
darken

$x - 128$

lower contrast

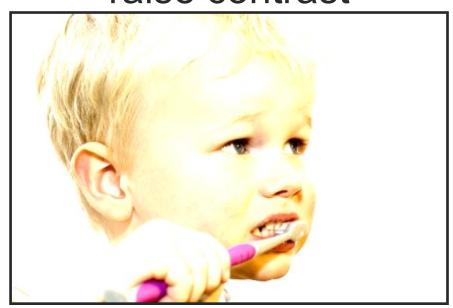
non-linear lower contrast

invert



lighten

raise contrast



non-linear raise contrast

Examples of point processing

How would you implement these?

original

$$x$$

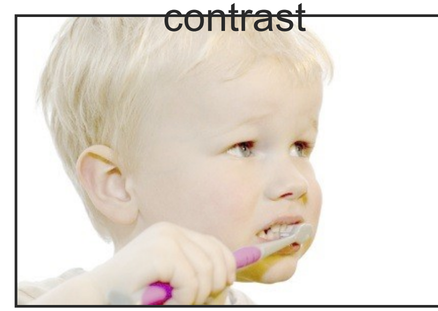
darken

$$x - 128$$

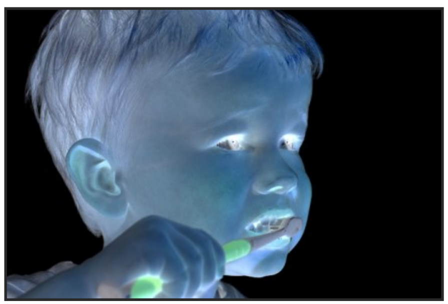
lower contrast

$$\frac{x}{2}$$

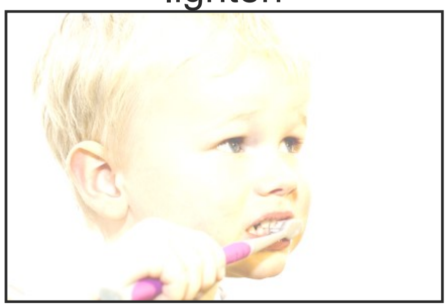
non-linear lower contrast



invert



lighten



raise contrast

non-linear raise contrast

Examples of point processing

How would you implement these?

original

$$x$$

darken

$$x - 128$$

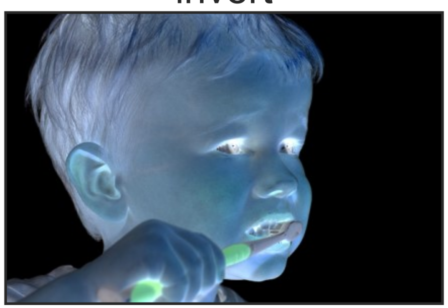
lower contrast

$$\frac{x}{2}$$

non-linear lower contrast

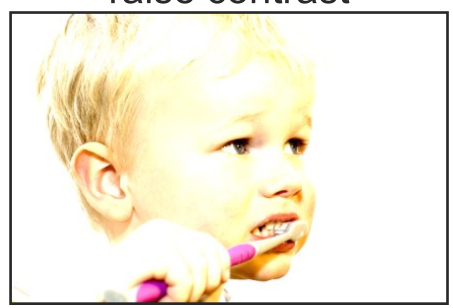
$$\left(\frac{x}{255}\right)^{1/3} \times 255$$

invert

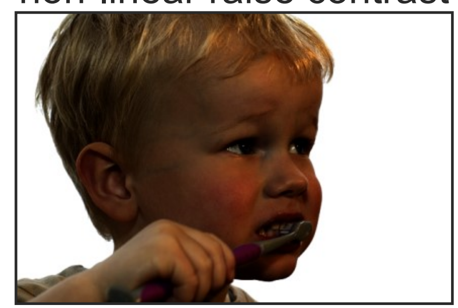


lighten

raise contrast



non-linear raise contrast



Examples of point processing

How would you implement these?

original

$$x$$

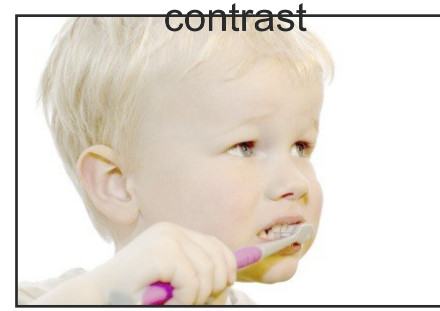
darken

$$x - 128$$

lower contrast

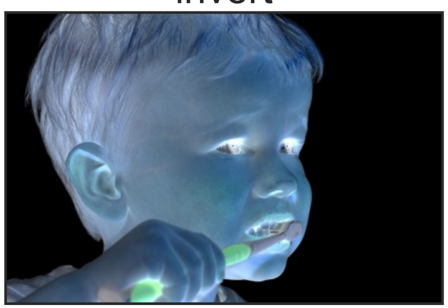
$$\frac{x}{2}$$

non-linear lower contrast



$$\left(\frac{x}{255}\right)^{1/3} \times 255$$

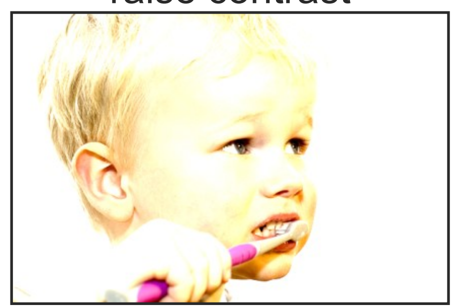
invert



$$255 - x$$

lighten

raise contrast



non-linear raise contrast

Examples of point processing

How would you implement these?

original

x

darken

$x - 128$

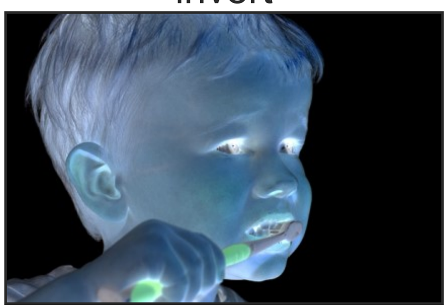
lower contrast

$\frac{x}{2}$

non-linear lower contrast

$\left(\frac{x}{255}\right)^{1/3} \times 255$

invert

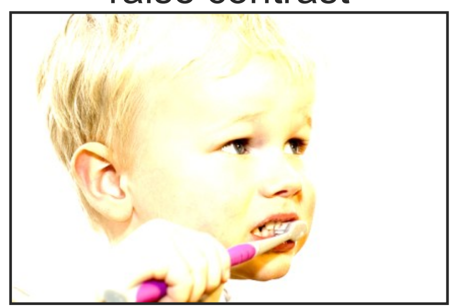


$255 - x$

lighten

$x + 128$

raise contrast



non-linear raise contrast

Examples of point processing

How would you implement these?

original

x

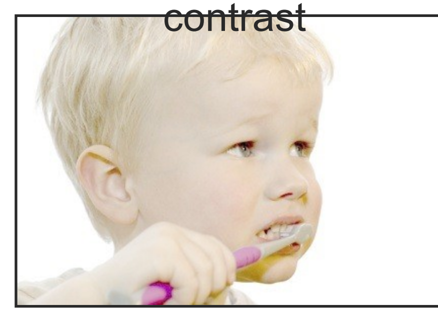
darken

$x - 128$

lower contrast

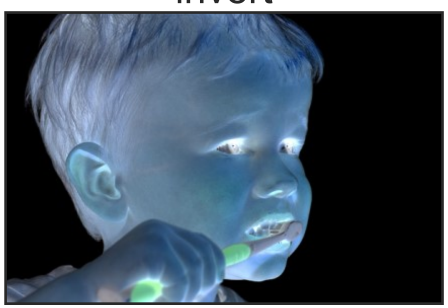
$\frac{x}{2}$

non-linear lower contrast



$$\left(\frac{x}{255}\right)^{1/3} \times 255$$

invert

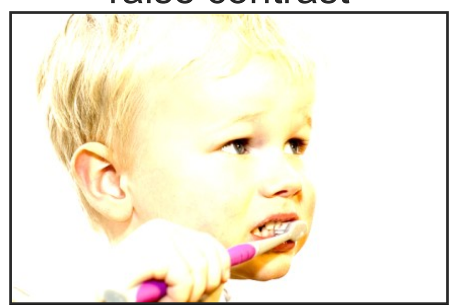


$255 - x$

lighten

$x + 128$

raise contrast



$x \times 2$

non-linear raise contrast

Examples of point processing

How would you implement these?

original

$$x$$

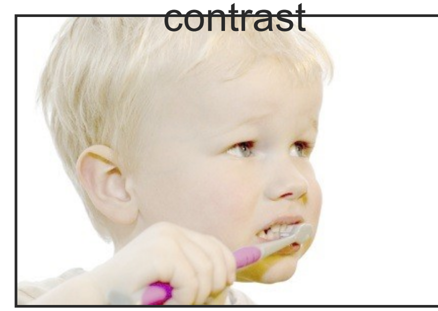
darken

$$x - 128$$

lower contrast

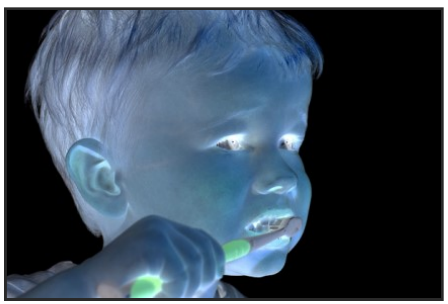
$$\frac{x}{2}$$

non-linear lower contrast



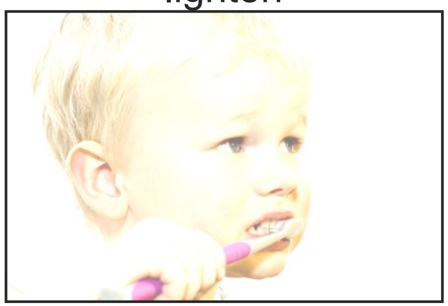
$$\left(\frac{x}{255}\right)^{1/3} \times 255$$

invert



$$255 - x$$

lighten



$$x + 128$$

raise contrast

$$x \times 2$$

non-linear raise contrast

$$\left(\frac{x}{255}\right)^{2} \times 255$$

Image filtering

- Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a “filter” or mask saying how to combine values from neighbors.

- Uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)

Adapted from Derek Hoiem

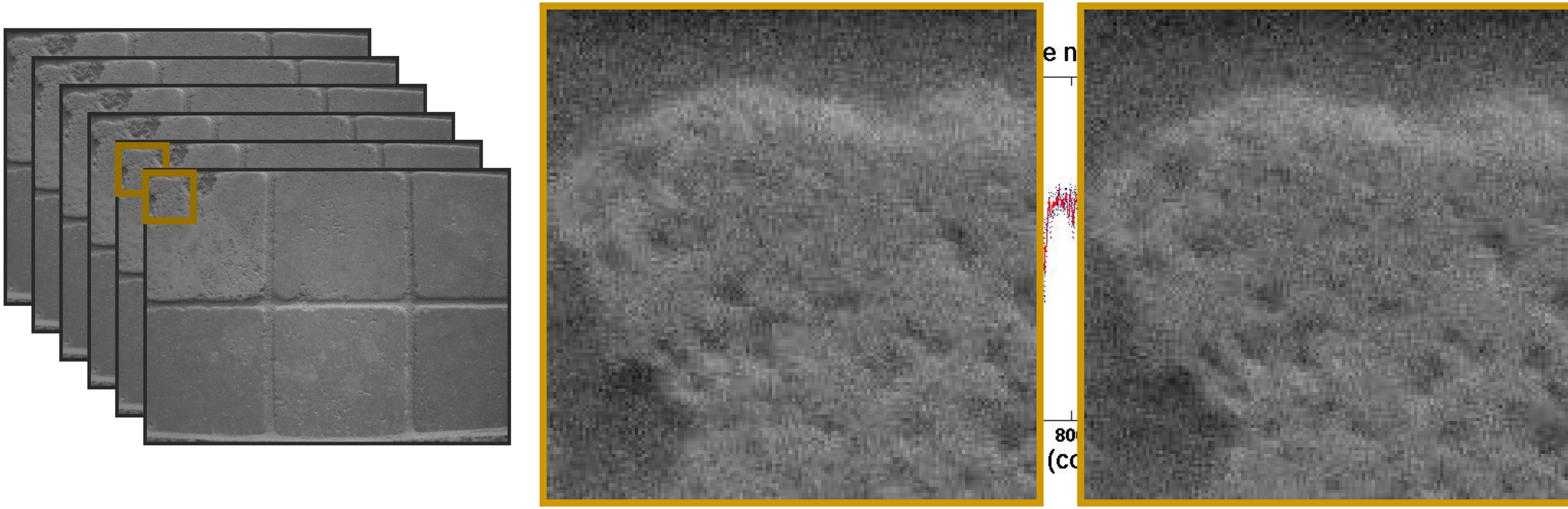
Three views of filtering

- Image filters in spatial domain
  - Filter is a mathematical operation on values of each patch
  - 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

Overview

- What is an image?
- Image formation: light and color
- Image transform
- **Image noise and image smoothing**
- Convolution operation
- Media filter

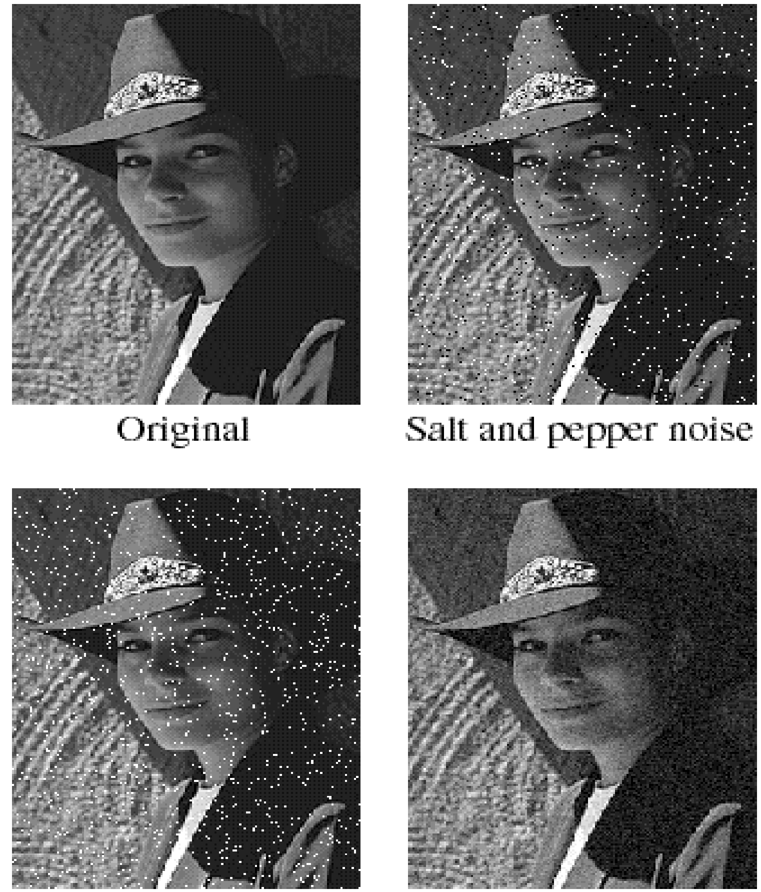
Motivation: noise reduction



- Even multiple images of the **same static scene** will not be identical.

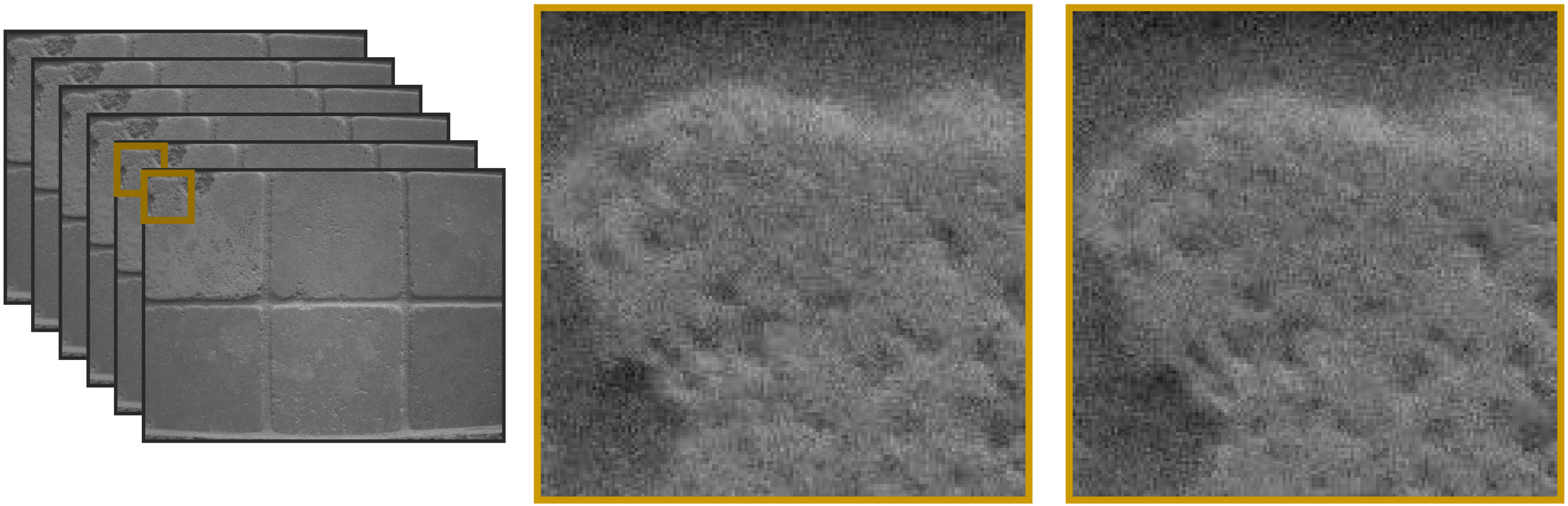
Common types of noise

- **Salt and pepper noise**: random occurrences of black and white pixels
- **Impulse noise:** random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution



Original | Salt and pepper noise | Impulse noise | Gaussian noise

Motivation: noise reduction



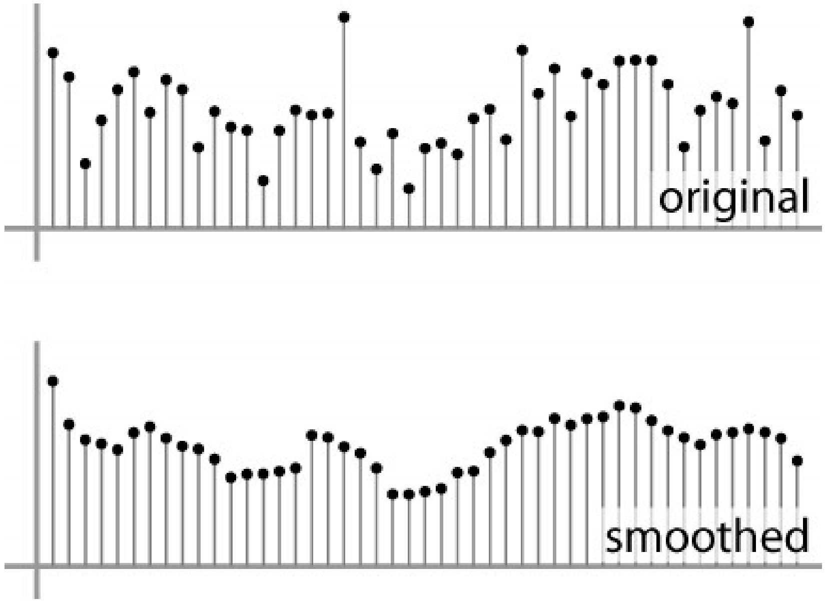
- Even multiple images of the same static scene will not be identical.
- How could we reduce the noise, i.e., give an estimate of the true intensities?
- **What if there's only one image?**

First attempt at a solution

- Let’s replace each pixel with an average of all the values in its neighborhood
- Assumptions:
  - Expect pixels to be like their neighbors
  - Expect noise processes to be independent from pixel to pixel

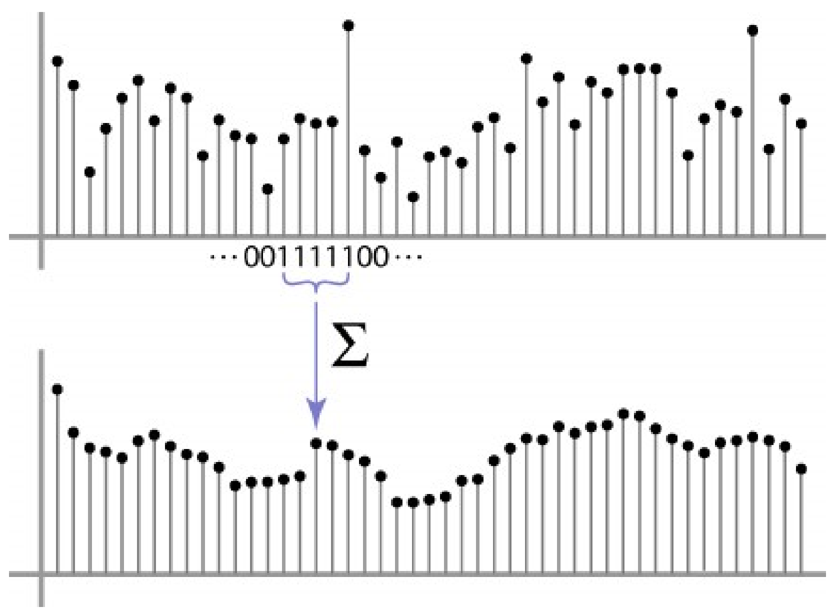
First attempt at a solution

- Let's replace each pixel with an average of all the values in its neighborhood
- Moving average in 1D:



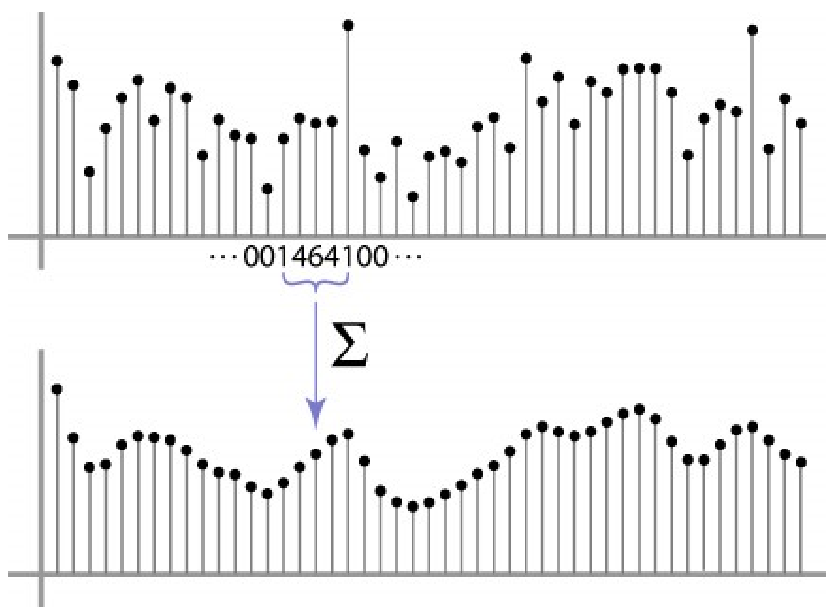
Weighted Moving Average

- Can add weights to our moving average
- *Weights* [1, 1, 1, 1, 1] / 5



Weighted Moving Average

- Non-uniform weights [1, 4, 6, 4, 1] / 16



Source: S. Marschner

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0								

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0	10							

Source: S. Seitz

Moving Average In 2D

$$F[x, y]$$

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

$$G[x, y]$$

	0	10	20						

Source: S. Seitz

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20	30					

Source: S. Seitz

Moving Average In 2D

$$F[x, y]$$

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

$$G[x, y]$$

	0	10	20	30	30				

Source: S. Seitz

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	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	

Source: S. Seitz

Correlation filtering

Say the averaging window size is 2k+1 x 2k+1:

$$G[i,j] = \underbrace{\frac{1}{(2k+1)^2}}_{\text{Attribute uniform weight to each pixel}} \underbrace{\sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]}_{\text{Loop over all pixels in neighborhood around image pixel } F[i,j]}$$

Now generalize to allow different weights depending on neighboring pixel's relative position:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \underbrace{H[u,v]}_{\text{Non-uniform weights}} F[i+u, j+v]$$

Correlation filtering

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

This is called cross-correlation, denoted $G = H \otimes F$

Filtering an image: replace each pixel with a linear combination of its neighbors.

The filter "kernel" or "mask" $H[u,v]$ is the prescription for the weights in the linear combination.

Averaging filter

- What values belong in the kernel *H* for the moving average example?

$F[x, y]$

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

$\otimes$

$H[u, v]$

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

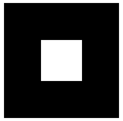
"box filter"

$G[x, y]$

	0	20	40	60	60				

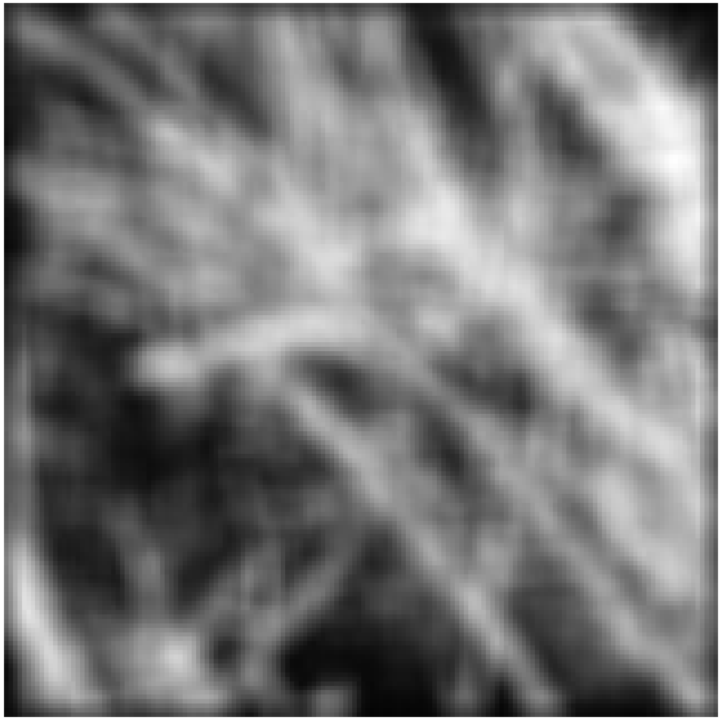
$$G = H \otimes F$$

Smoothing by averaging



depicts box filter:
white = high value, black = low value

original



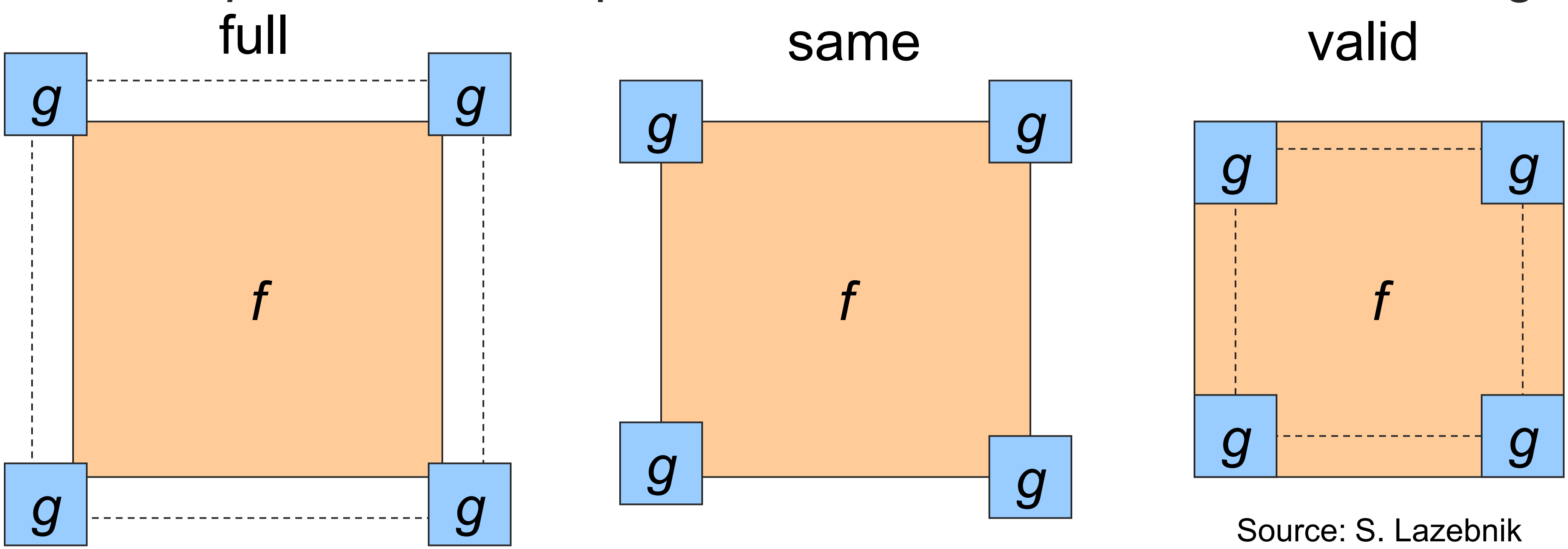
filtered

What if the filter size was 5 x 5 instead of 3 x 3?

Boundary issues

MATLAB: output size / “shape” options

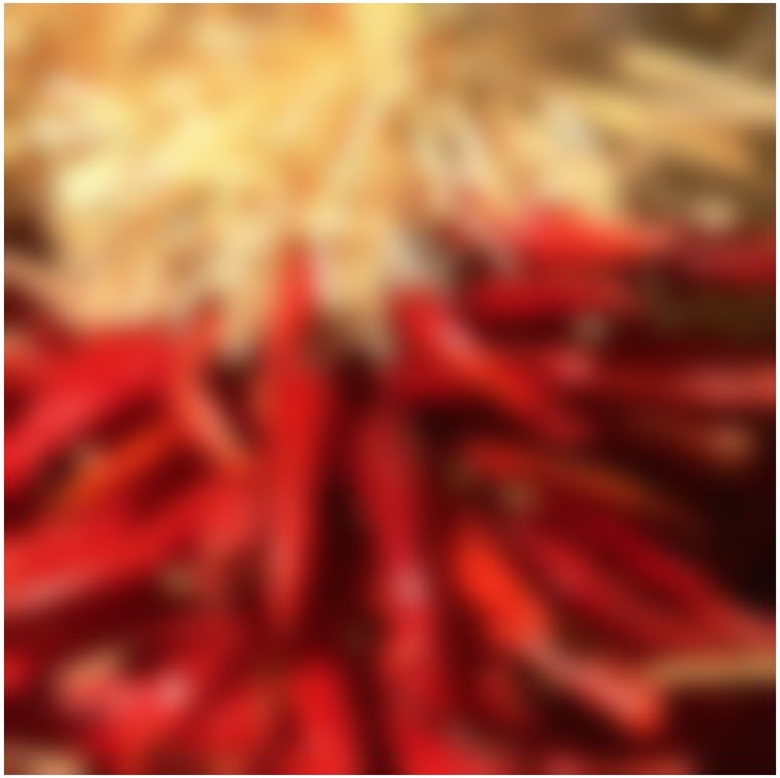
- *shape* = ‘full’: output size is sum of sizes of f and g
- *shape* = ‘same’: output size is same as f
- *shape* = ‘valid’: output size is difference of sizes of f and g



Source: S. Lazebnik

Boundary issues

- 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

Boundary issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - 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

Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?

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

$F[x, y]$

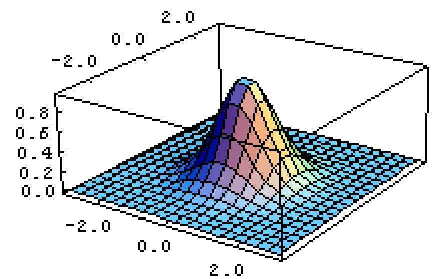
$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$H[u, v]$

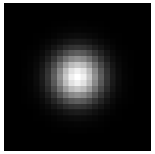
This kernel is an approximation of a 2d Gaussian function:

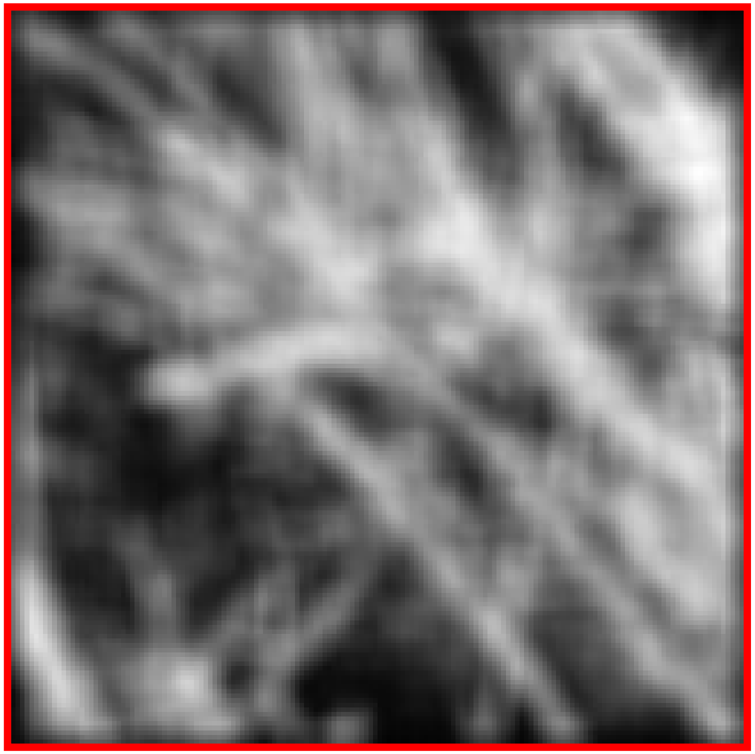
$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



- Removes high-frequency components from the image (“low-pass filter”).

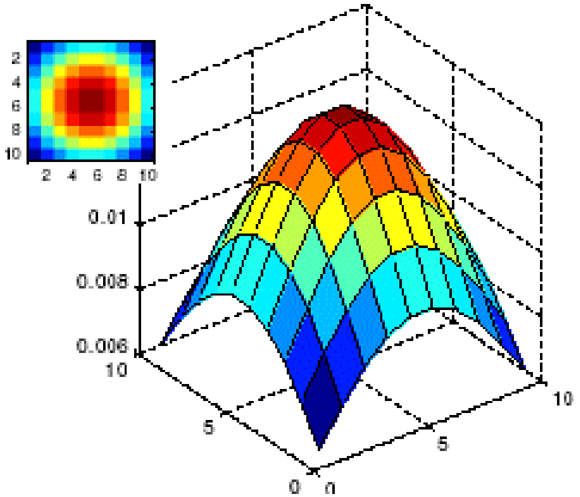
Smoothing with a Gaussian



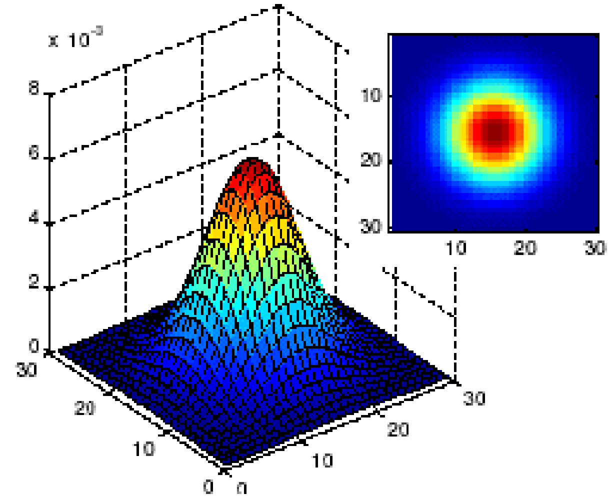


Gaussian filters

- What parameters matter here?
- **Size** of kernel or mask
  - Note, Gaussian function has infinite support, but discrete filters use finite kernels



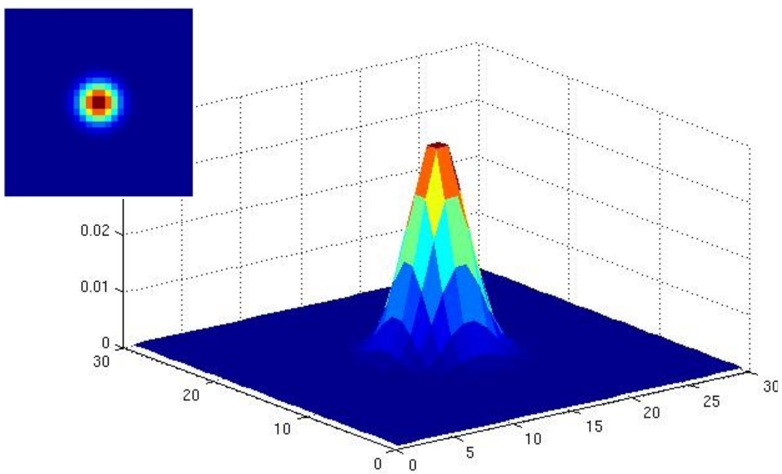
σ = 5 with 10 x 10 kernel



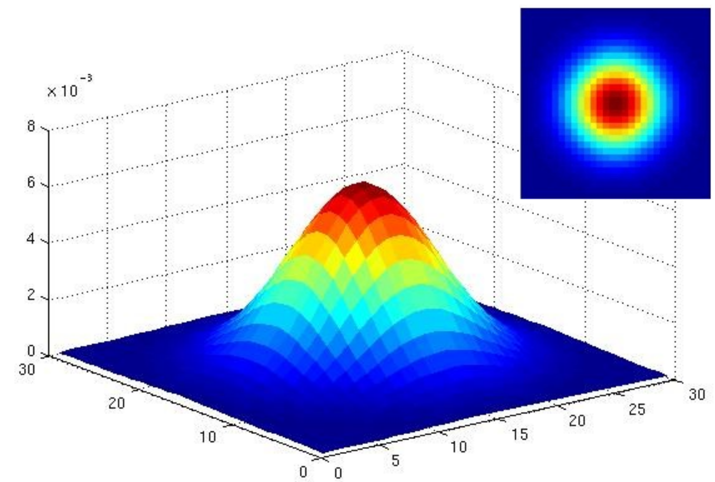
σ = 5 with 30 x 30 kernel

Gaussian filters

- What parameters matter here?
- **Variance** of Gaussian: determines extent of smoothing



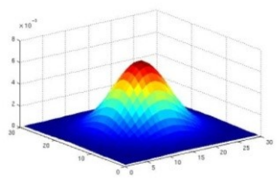
σ = 2 with 30 x 30 kernel



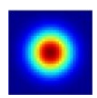
σ = 5 with 30 x 30 kernel

Matlab

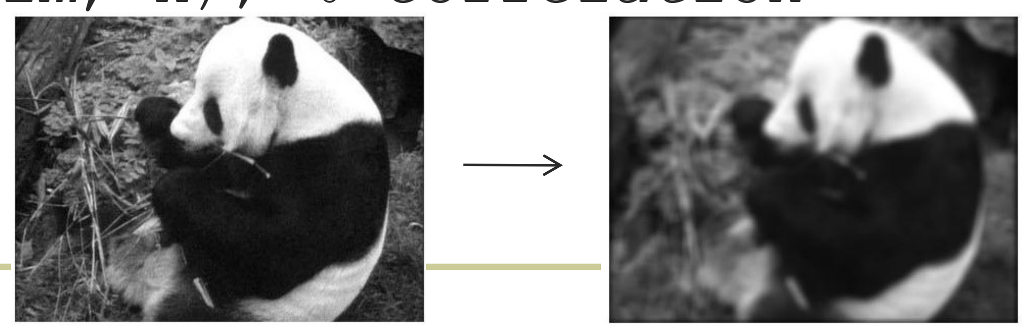
```
>> hsize = 10;
>> sigma = 5;
>> h = fspecial(‘gaussian’ hsize, sigma);
```



```
>> mesh(h);
```



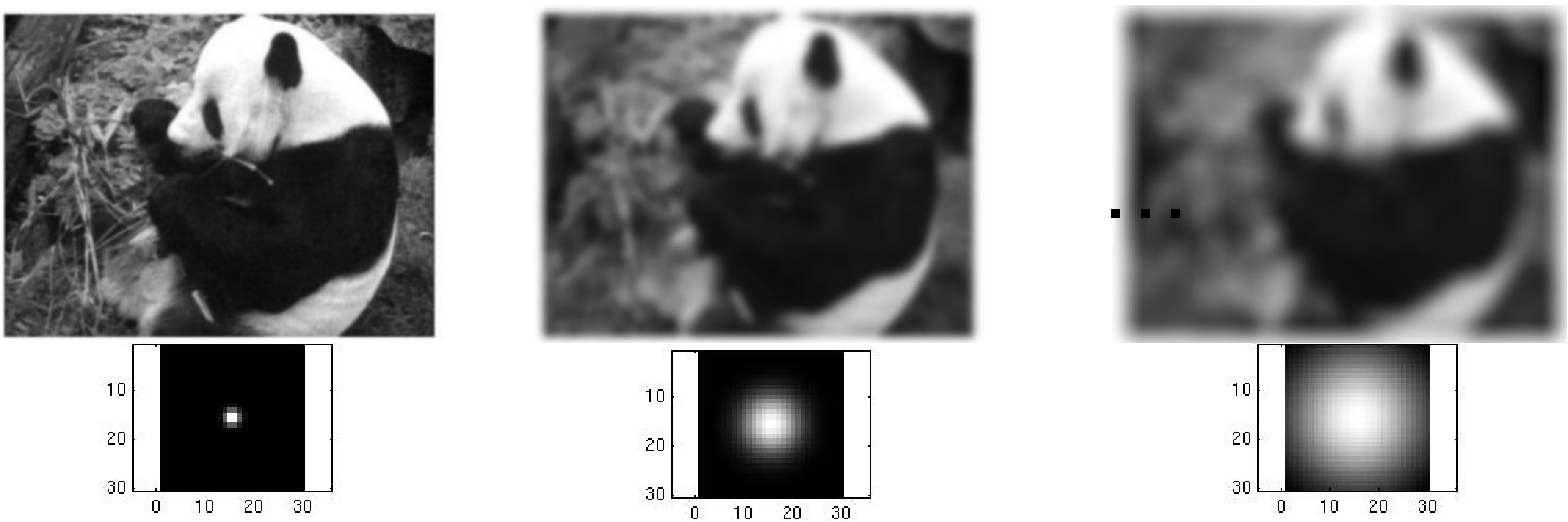
```
>> imagesc(h);

>> outim = imfilter(im, h); % correlation
>> imshow(outim);
```



outim

Smoothing with a Gaussian

Parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.




```
for sigma=1:3:10
    h = fspecial('gaussian‘, fsize,
    sigma);
    out = imfilter(im, h);
    imshow(out);
    pause;
end
```

Properties of smoothing filters

- Smoothing
  - Values positive
  - Sum to 1 → constant regions same as input
  - Amount of smoothing proportional to mask size
  - Remove “high-frequency” components; “low-pass” filter

Overview

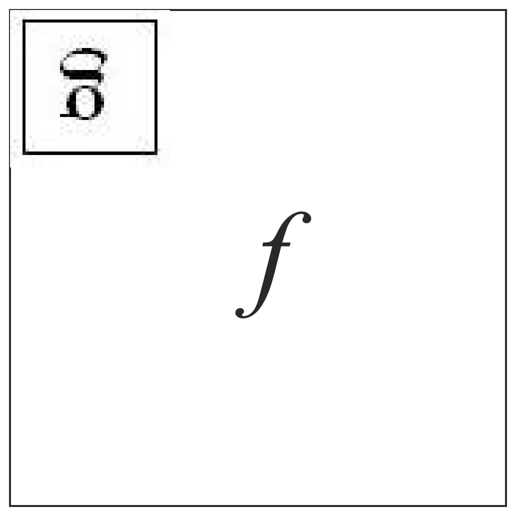
- What is an image?
- Image formation: light and color
- Image transform
- Image noise and image smoothing
- **Convolution operation**
- Media filter

Defining convolution

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

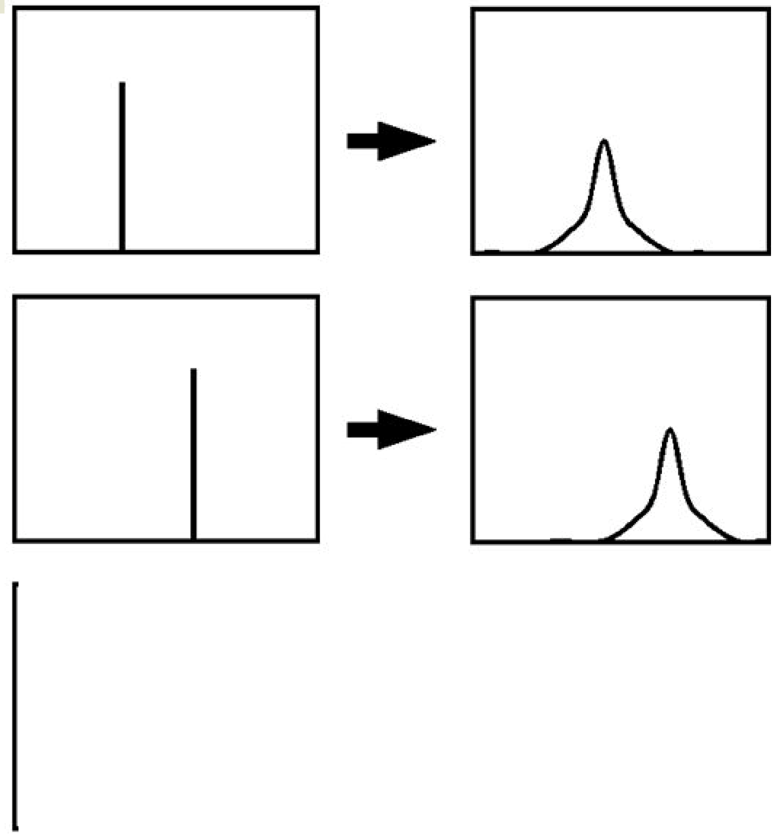
$$(f * g)[m,n] = \sum_{k,l} f[m-k, n-l]g[k,l]$$

Convention:
kernel is "flipped"



Key properties

- **Shift invariance:** same behavior regardless of pixel location: filter(shift(f)) = shift(filter(f))
- **Linearity:** filter(f_1 + f_2) = filter(f_1) + filter(f_2)



- Theoretical result: any linear shift-invariant operator can be represented as a convolution

Properties in more detail

- 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 $e = [\ldots, 0, 0, 1, 0, 0, \ldots]$, $a * e = a$

Convolution vs correlation

Definition of discrete 2D convolution:

notice the flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j) I(x - i, y - j)$$

Definition of discrete 2D correlation:

notice the lack of a flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j) I(x + i, y + j)$$

- Most of the time won't matter, because our kernels will be symmetric.

Separability

- In some cases, filter is separable, and we can factor into two steps:
  - Convolve all rows with a 1D filter
  - Convolve all columns with a 1D filter

$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \end{bmatrix} \circ \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 \end{bmatrix}$$

Overview

- What is an image?
- Image formation: light and color
- Image transform
- Image noise and image smoothing
- Convolution operation
- **Media filter**

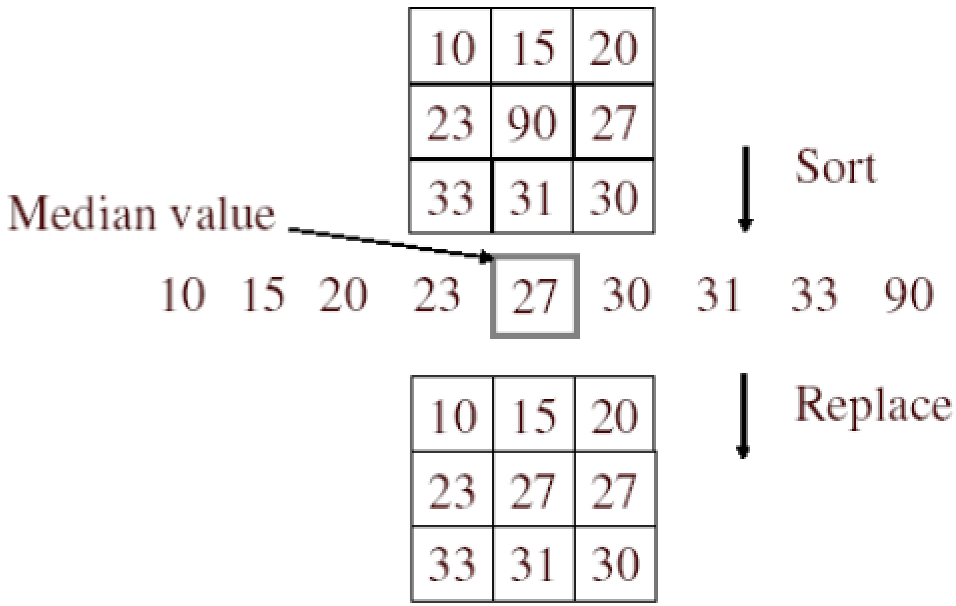
Effect of smoothing filters

5x 5

Additive Gaussian noise

Salt and pepper noise

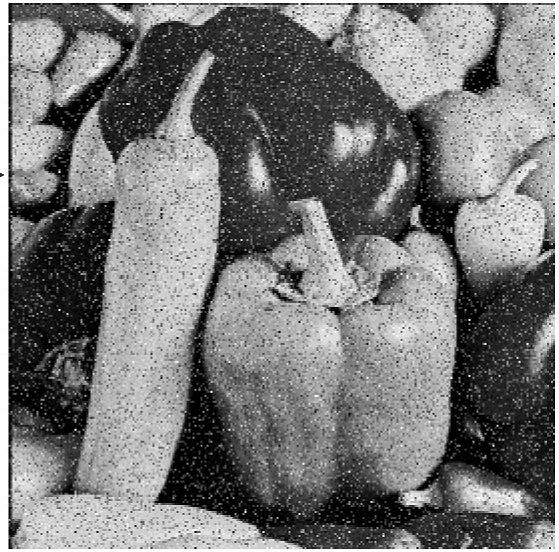
Median filter

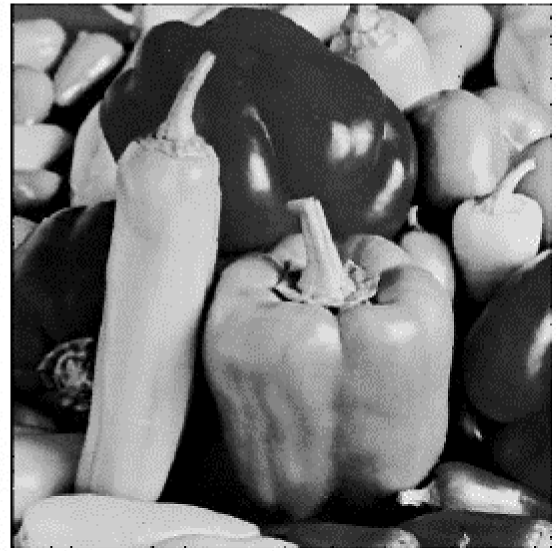


- No new pixel values introduced
- Removes spikes: good for impulse, salt & pepper noise
- Non-linear filter

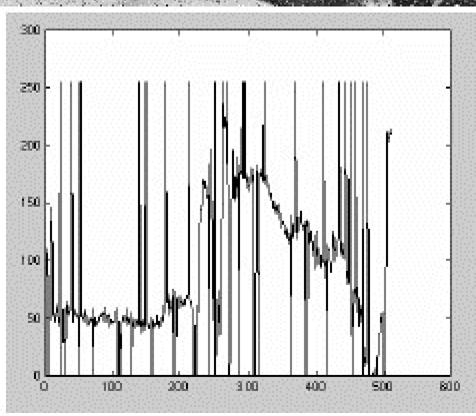
Median filter

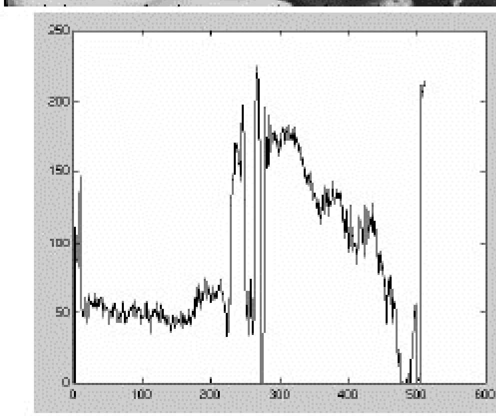
Salt and pepper noise





Median filtered





Plots of a row of the image

Matlab: output im = medfilt2(im, [h w]);

Median filter

- Median filter is edge preserving