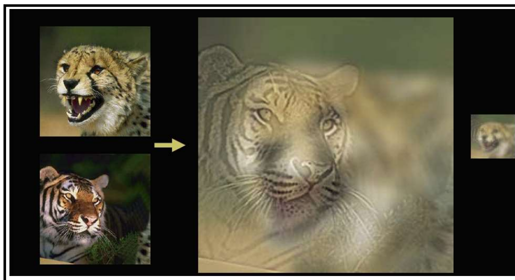
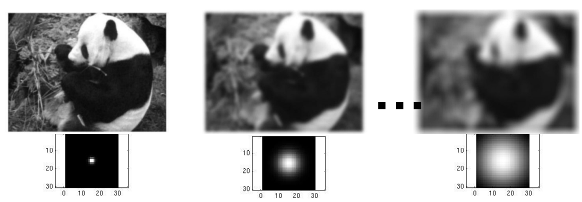


Linear Filters

Thurs Jan 19, 2017

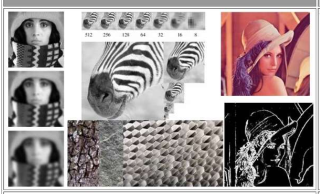


Announcements

- Piazza for assignment questions
- **A0** due Friday Jan 27. Submit on Canvas.

Course homepage

- **http://vision.cs.utexas.edu/378h-spring2017/**

	Tues Jan 17	**Course intro**	Textbook Sec 1.1-1.3 Course requirements UTCS account setup Basic Matlab tutorial Running Matlab at UT	slides	A0 out See optional Latex info
	Thurs Jan 19	**Features and filters**	Sec 3.1.1-2, 3.2	Linear filters	
	Tues Jan 24		Sec 3.2.3, 4.2	Gradients and	A0 due Friday Jan 27

Computer Vision: Algorit...

szeliski.org/Book/

Computer Vision: Algorithms and Applications

© 2010 Richard Szeliski, Microsoft Research

Welcome to the Web site (http://szeliski.org/Book) for my computer vision textbook, which you can now purchase at a variety of locations, including Springer, Amazon, and Barnes & Noble.

This book is largely based on the computer vision courses that I have co-taught at the University of Washington (2008, 2005, 2001) and Stanford (2003) with Steve Seitz and David Fleet.

You are welcome to download the PDF from this Web site for personal use, but **not** to repost it on any other Web site. Please post a link to this URL (http://szeliski.org/Book) instead. An elec manuscript will continue to be available even after the book is published. Note, however, that while the content of the electronic and hardcopy versions are the same, the page layout (pagination) electronic version is optimized for online reading.

The PDFs should be enabled for commenting directly in your viewer. Also, hyper-links to sections, equations, and references are enabled. To get back to where you were, use Alt-Left-Arrow i

Plan for today

- Image noise
- Linear filters
  – Examples: smoothing filters
- Convolution / correlation

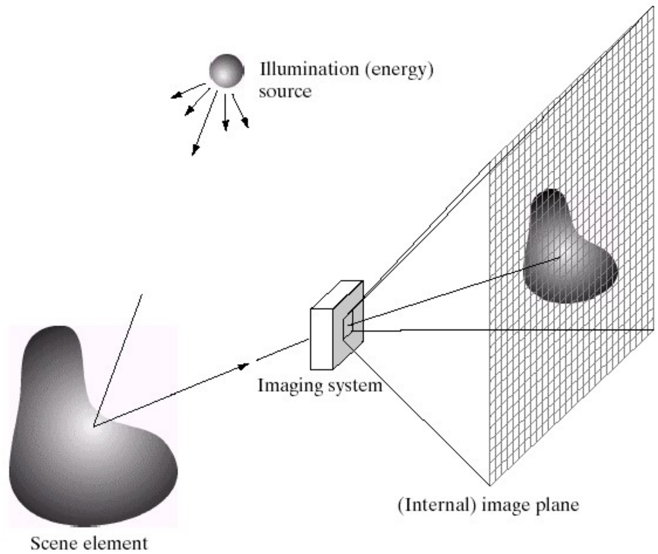
Images as matrices

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

Image Formation



Slide credit: Derek Hoiem

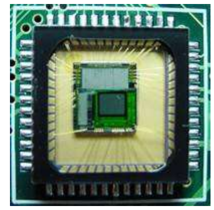
Digital camera

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 Steve Seitz

Digital images



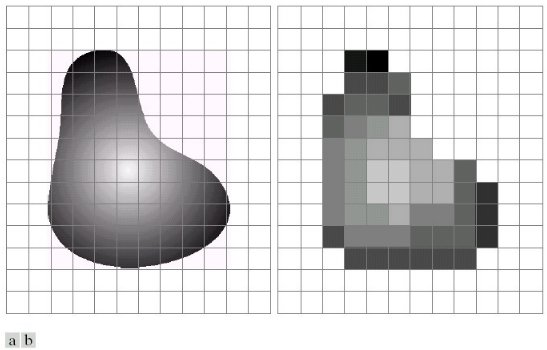
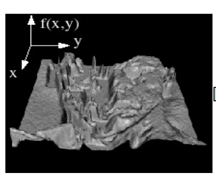


FIGURE 2.17 (a) Continuos image projected onto a sensor array. (b) Result of image sampling and quantization.

Slide credit: Derek Hoiem

Digital images

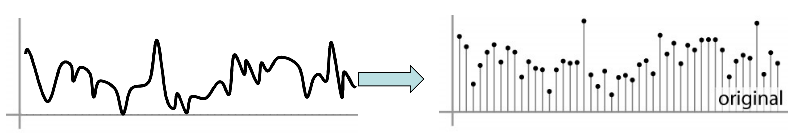
- **Sample** the 2D space on a regular grid
- **Quantize** each sample (round to nearest integer)
- Image thus represented as a matrix of integer values.



j →, i ↓

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

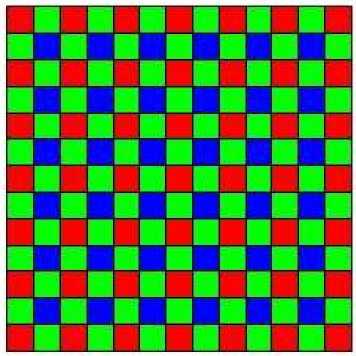
2D



1D

Adapted from S. Seitz

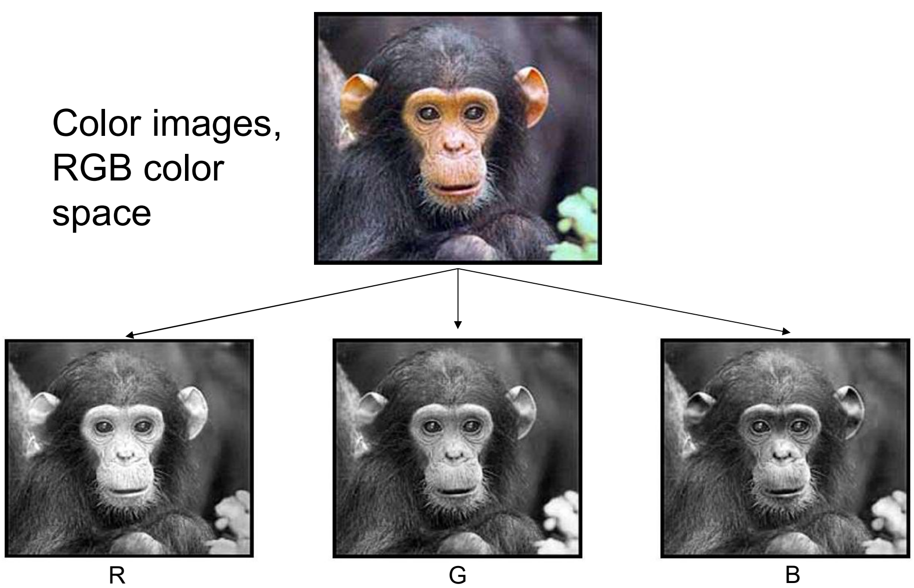
Digital color images



Bayer filter

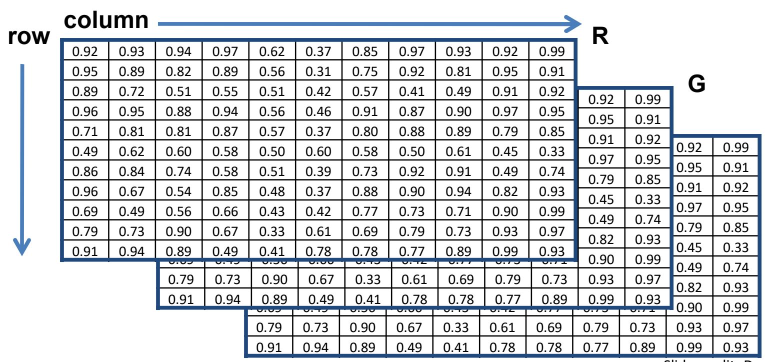
Digital color images

Color images, RGB color space



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



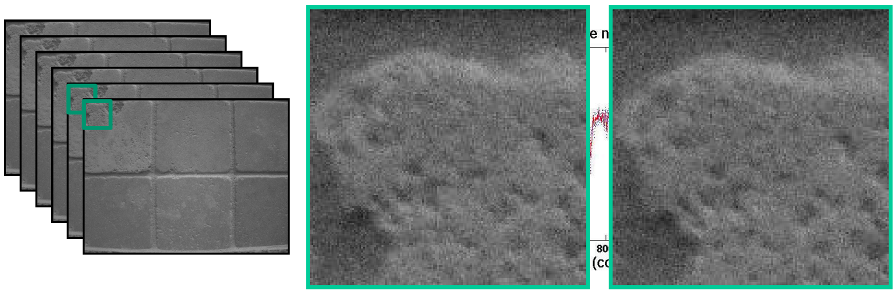
Slide credit: Derek Hoiem

Main idea: 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

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

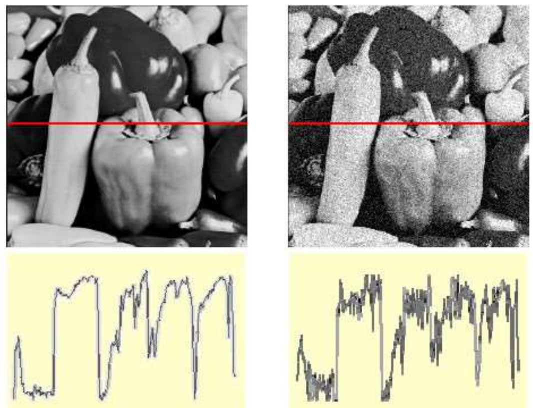
Salt and pepper noise

Impulse noise

Gaussian noise

Source: S. Seitz

Gaussian noise



$$f(x,y) = \overbrace{\widetilde{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$

Gaussian i.i.d. ("white") noise: $\eta(x,y) \sim \mathcal{N}(\mu,\sigma)$

```
>> noise = randn(size(im)).*sigma;
>> output = im + noise;
```

What is impact of the sigma?

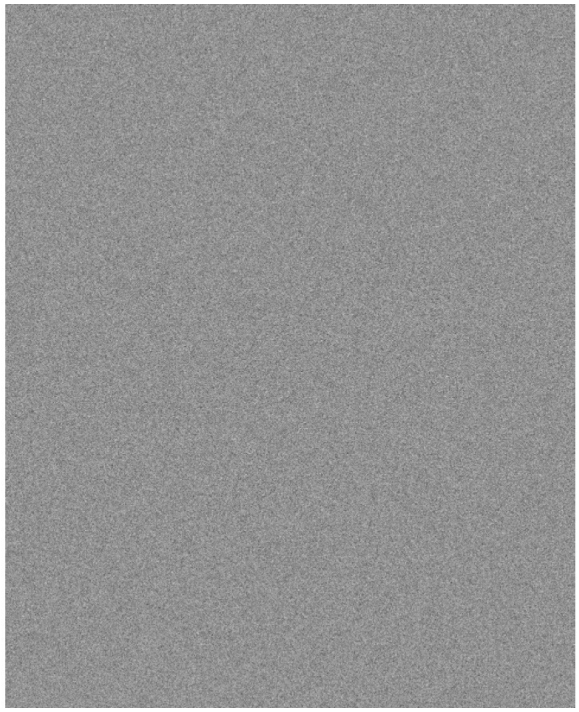
Fig: M. Hebert

sigma=1

Effect of sigma on Gaussian noise:

This shows the noise values added to the raw intensities of an image.

sigma=
16



Effect of sigma on Gaussian noise:

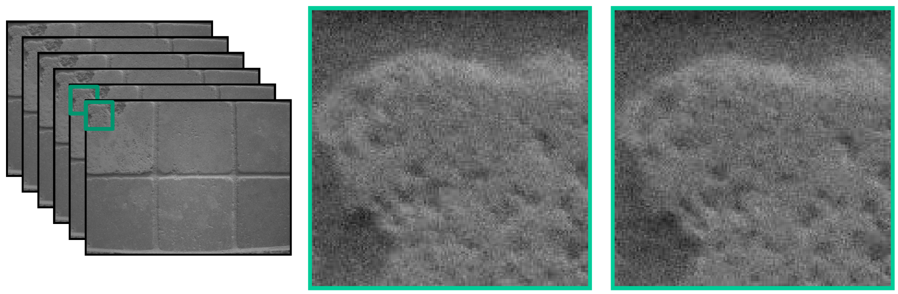
Image shows the noise values themselves.

sigma=16

Effect of sigma on Gaussian noise

This shows the noise values added to the raw intensities of an image.

Motivation: noise reduction



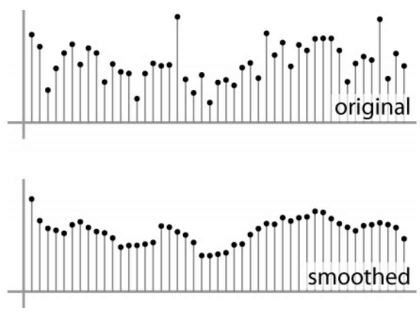
- Even multiple images of the same static scene will not be identical.
- How could we reduce the noise, i.e., give an estimate of the true intensities?
- **What if there's only one image?**

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:

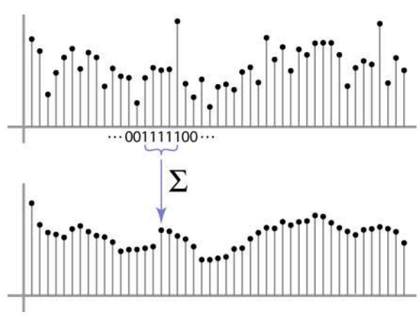


Source: S. Marschner

Weighted Moving Average

Can add weights to our moving average

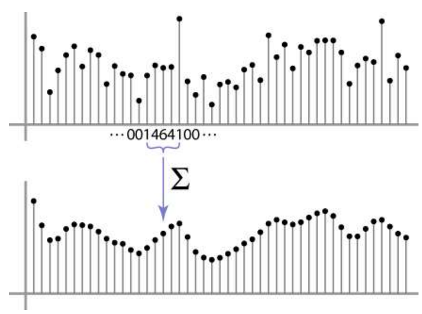
Weights [1, 1, 1, 1, 1] / 5



Source: S. Marschner

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	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:

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

Attribute uniform weight to each pixel

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:

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

Non-uniform weights

Correlation filtering

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

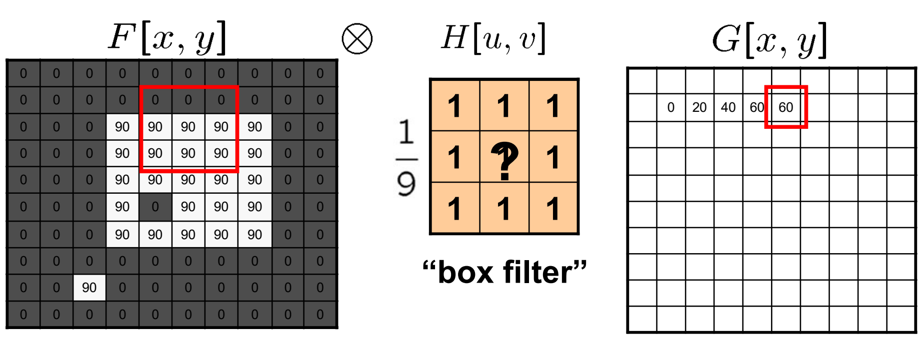
This is called **cross-correlation**, denoted $G = H \otimes F$

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.

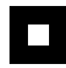
Averaging filter

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



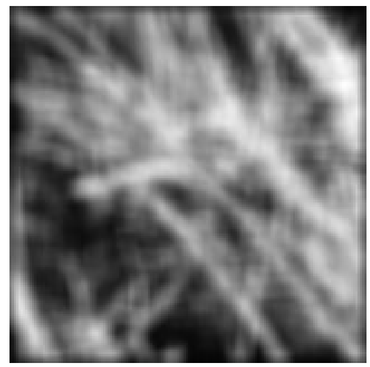
$$G = H \otimes F$$

Smoothing by averaging



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

original



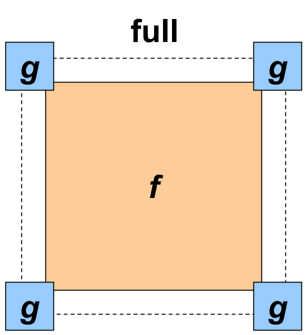
filtered

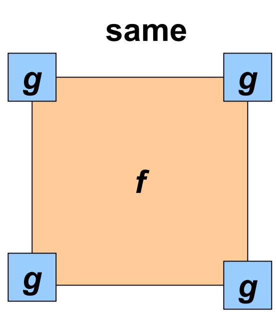
What if the filter size was 5 x 5 instead of 3 x 3?

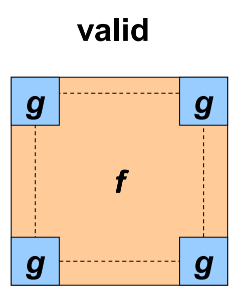
Boundary issues

What is the size of the output?

- 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 (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

$F[x, y]$

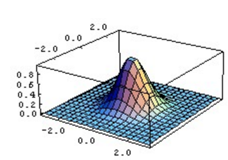
$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$H[u, v]$

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

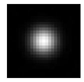
$$h(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$

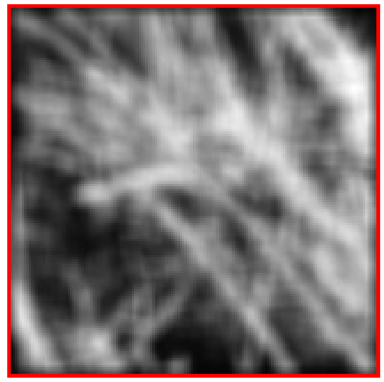


- Removes high-frequency components from the image (“low-pass filter”).

Source: S. Seitz

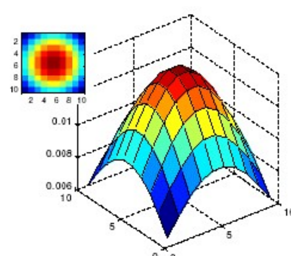
Smoothing with a Gaussian



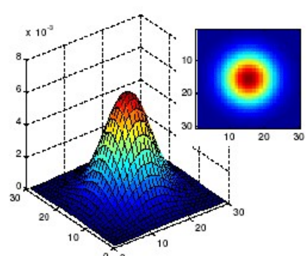


Gaussian filters

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



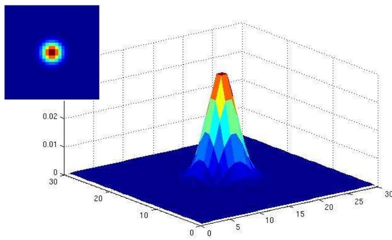
σ = 5 with 10 x 10 kernel



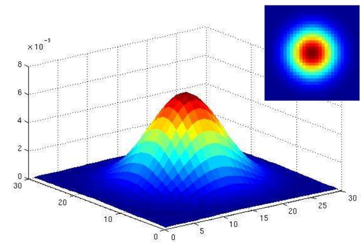
σ = 5 with 30 x 30 kernel

Gaussian filters

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



σ = 2 with 30 x 30 kernel

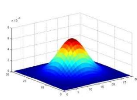


σ = 5 with 30 x 30 kernel

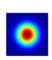
Matlab

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

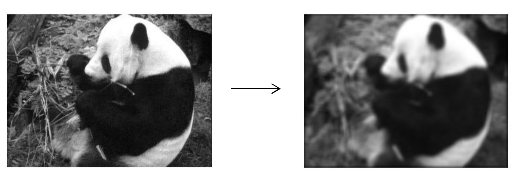
```
>> mesh(h);
```



```
>> imagesc(h);
```



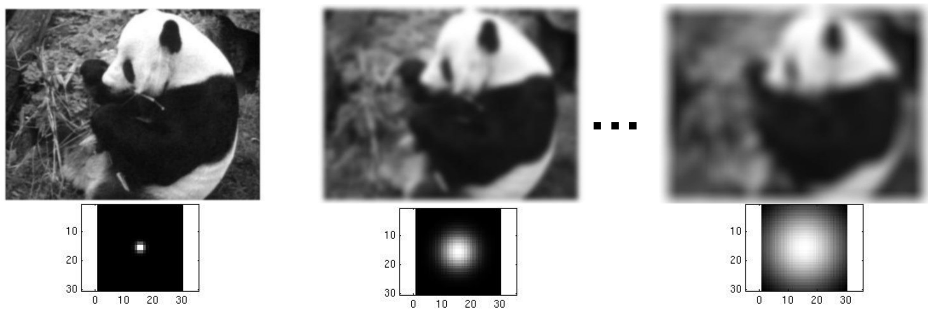
```
>> outim = imfilter(im, h); % correlation
>> imshow(outim);
```



outim

Smoothing with a Gaussian

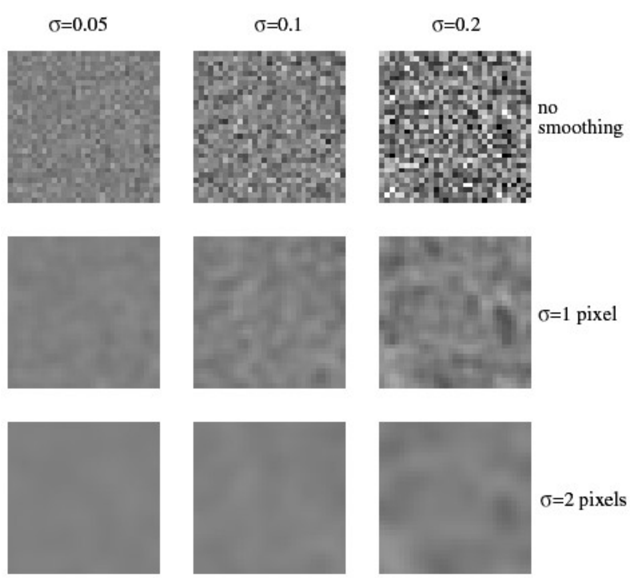
Parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.




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

Keeping the two Gaussians in play straight…

More noise →



Wider smoothing kernel →

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

Predict the outputs using correlation filtering

$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = ?$$

$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = ?$$

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

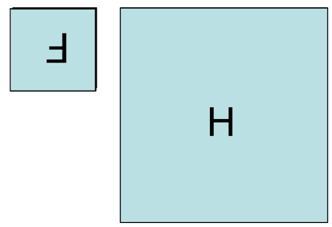
Convolution

- Convolution:
  - Flip the filter in both dimensions (bottom to top, right to left)
  - Then apply cross-correlation

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

$$G = H \star F$$

Notation for convolution operator



Properties of convolution

- **Shift invariant:**
  - Operator behaves the same everywhere, i.e. the value of the output depends on the pattern in the image neighborhood, not the position of the neighborhood.
- **Superposition:**
  - h * (f1 + f2) = (h * f1) + (h * f2)

Properties of convolution

- Commutative:
  f * g = g * f
- Associative
  (f * g) * h = f * (g * h)
- Distributes over addition
  f * (g + h) = (f * g) + (f * h)
- Scalars factor out
  kf * g = f * kg = k(f * g)
- Identity:
  unit impulse e = […, 0, 0, 1, 0, 0, …]. f * e = f

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

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

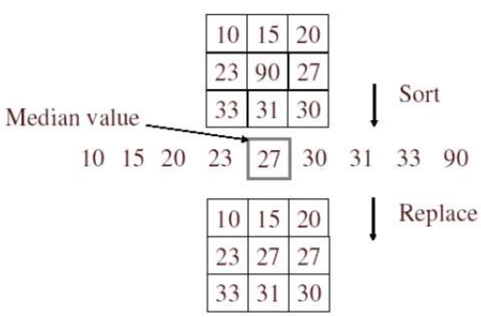
Effect of smoothing filters

5x5

Additive Gaussian noise

Salt and pepper noise

Median filter

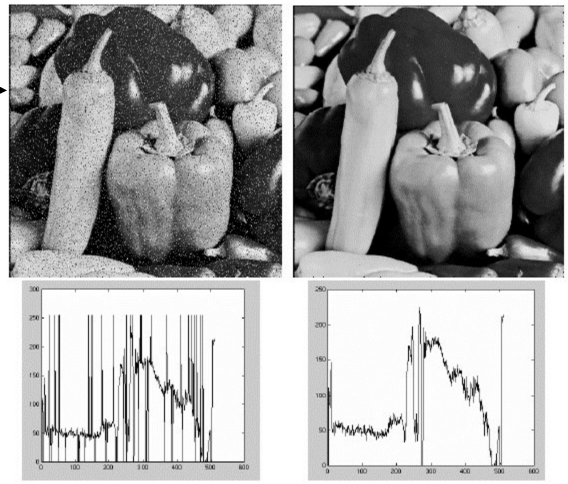


- No new pixel values introduced
- Removes spikes: good for impulse, salt & pepper noise
- Non-linear filter

Median filter

Salt and pepper noise

Median filtered



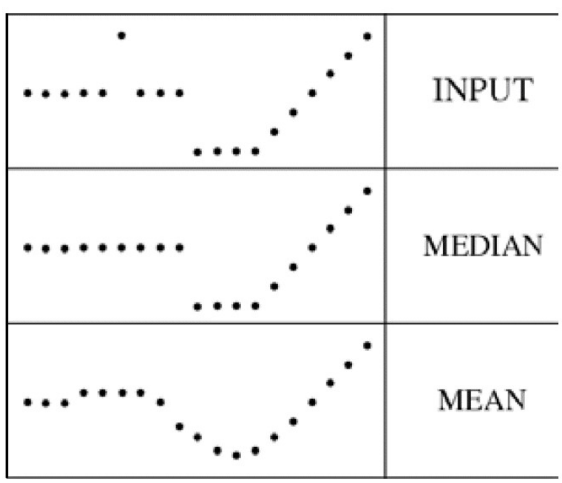
Plots of a row of the image

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

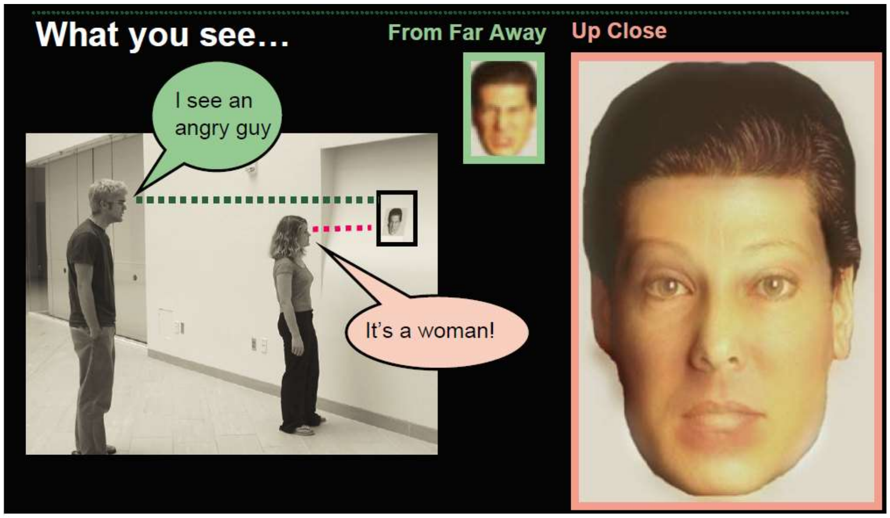
Source: M. Hebert

Median filter

- Median filter is edge preserving



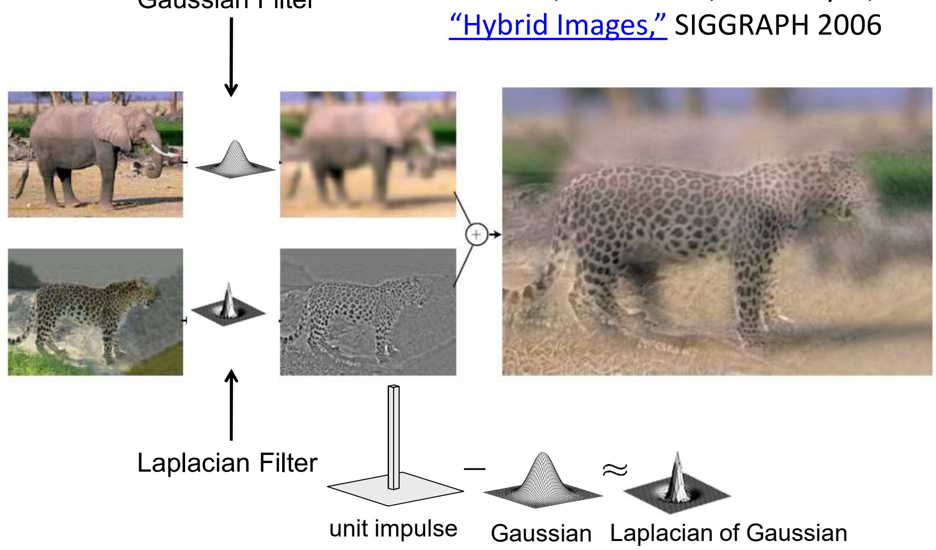
Filtering application: Hybrid Images

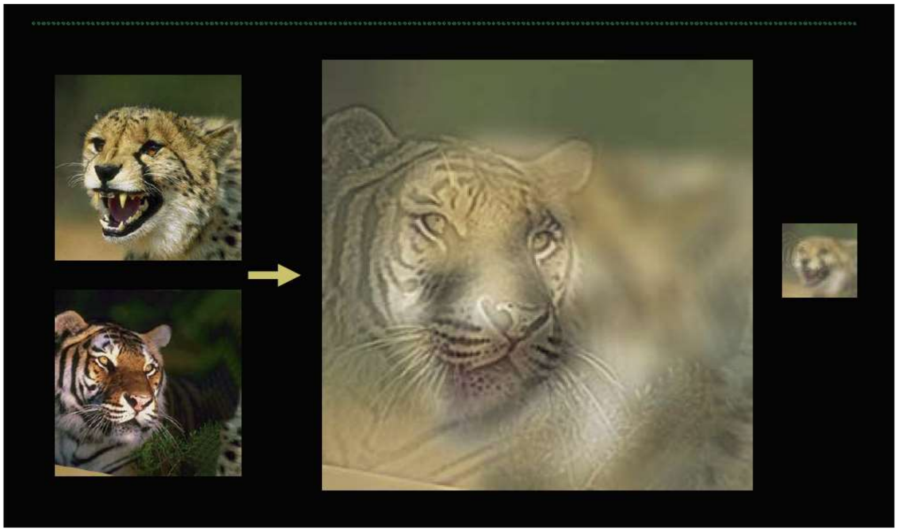


Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006

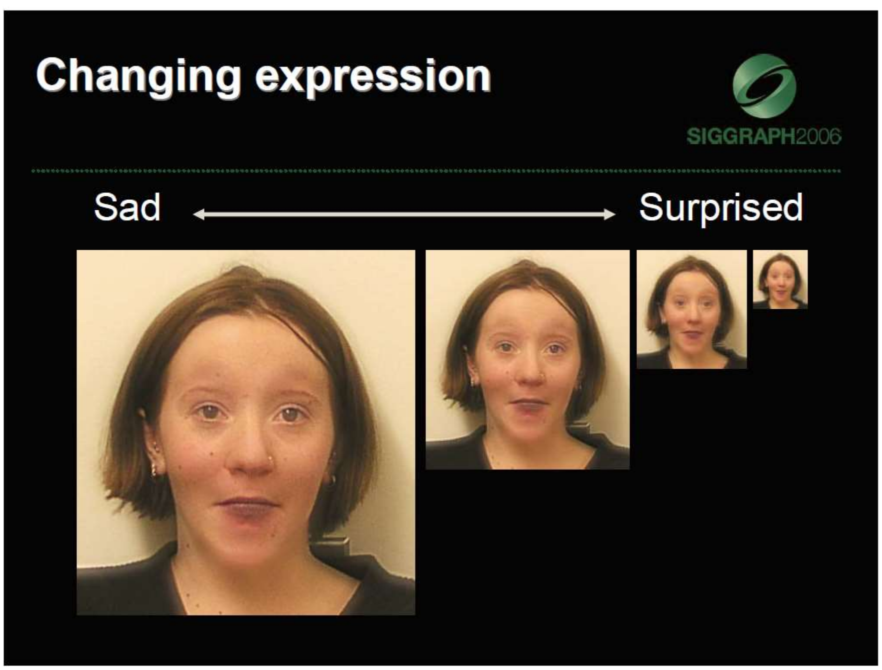
Application: Hybrid Images

A. Oliva, A. Torralba, P.G. Schyns, "Hybrid Images," SIGGRAPH 2006





Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006



Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006

Summary

- Image "noise"
- Linear filters and convolution useful for
  - Enhancing images (smoothing, removing noise)
    - Box filter
    - Gaussian filter
    - Impact of scale / width of smoothing filter
  - Detecting features (next time)
- Separable filters more efficient
- Median filter: a non-linear filter, edge-preserving

Coming up

- **Tuesday:**
  - Filtering part 2: filtering for features (edges, gradients, seam carving application)
  - See reading assignment on webpage
- **Next Friday:**
  - Assignment 0 is due on Canvas 11:59 PM