Dr. George Karraz, Ph. D.

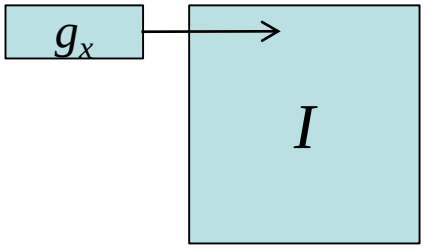
Digital Image Processing
Image Processing& Analysis

Image Filters

Dr. George Karraz, Ph.D.

Topics of This Lecture

- Linear filters
  - What are they? How are they applied?
  - Application: smoothing
  - Gaussian filter
  - What does it *mean* to filter an image?
- Nonlinear Filters
  - Median filter
- Multi-Scale representations
  - How to properly rescale an image?
- Image derivatives
  - How to compute gradients robustly?



Common Types of Noise

- Salt & 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.
- *Basic Assumption*
  - *Noise is i.i.d. (independent & identically distributed)*

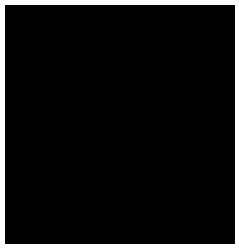
Original

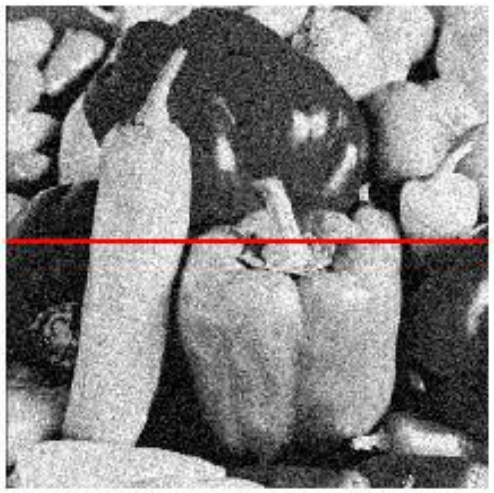
Salt and pepper noise

Impulse noise

Gaussian noise

Gaussian Noise





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

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

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

>> output = im + noise;
```

First Attempt at a Solution

- Assumptions:
  - Expect pixels to be like their neighbors
  - Expect noise processes to be independent from pixel to pixel ("i.i.d. = independent, identically distributed")

- Let's try to replace each pixel with an average of all the values in its neighborhood…

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							

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						

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					

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				

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	

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:

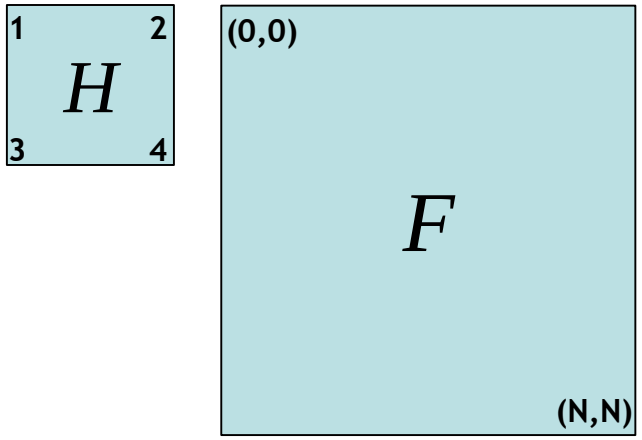
$$G[i,j] = \sum_{u=-k}^{k}\sum_{v=-k}^{k} \underbrace{H[u,v]}_{\text{Non-uniform weights}} F[i+u, j+v]$$

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 by a weighted combination of its neighbors.
  - The filter "kernel" or "mask" is the prescription for the weights in the linear combination.



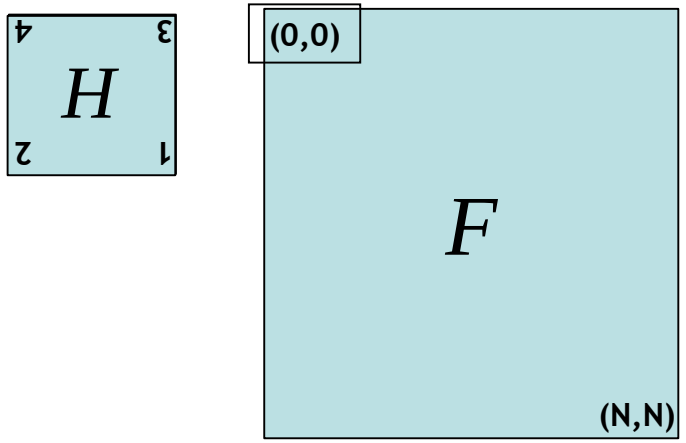
Convolution

- 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



Convolution vs. Correlation

- Correlation

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

$$G = H \otimes F$$

- Convolution

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

$$G = H \star F$$

Note the difference!

- Note
  - If $H[-u,-v] = H[u,v]$, then correlation ≡ convolution.

Shift Invariant Linear System

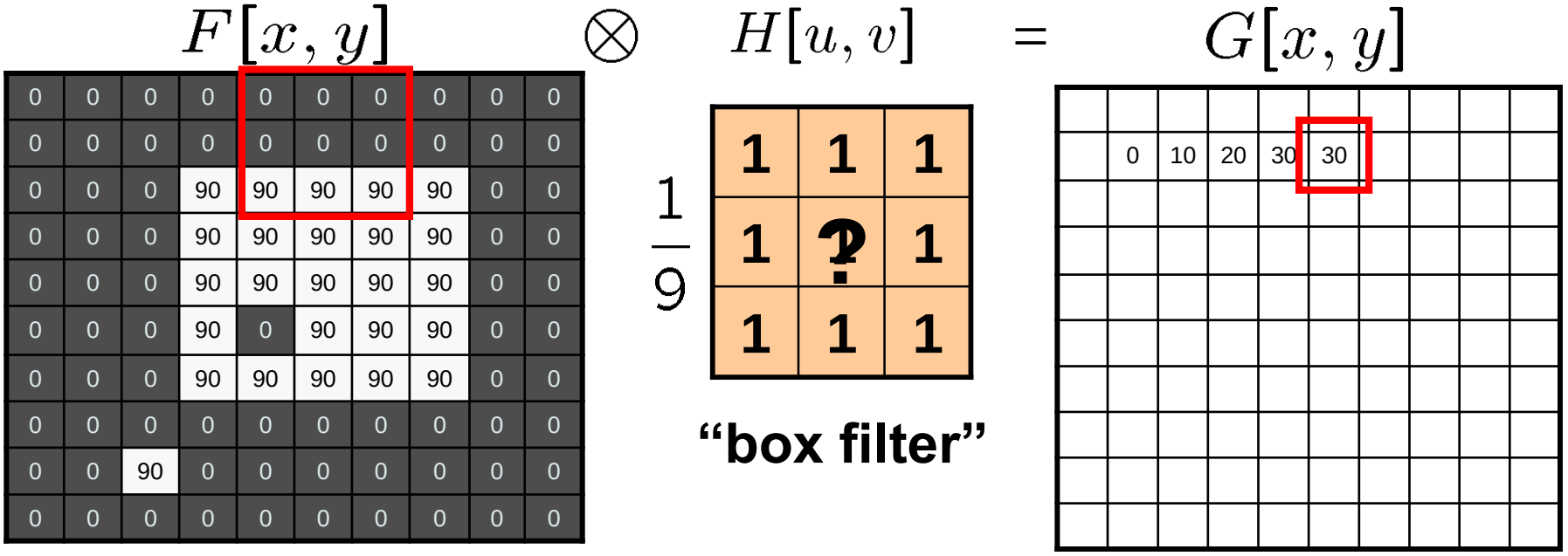
- Shift invariant:
  - Operator behaves the same everywhere, *i.e.* the value of the output depends on the pattern in the image neighborhood, not the position of the neighborhood.
- Linear:
  - Superposition: $h * (f1 + f2) = (h * f1) + (h * f2)$
  - Scaling: $h * (k f) = k (h * f)$

Properties of Convolution

- Linear & shift invariant
- Commutative: $f \sum g = g \sum f$
- Associative: $(f \sum g) \sum h = f \sum (g \sum h)$
  - Often apply several filters in sequence: $(((a \sum b_1) \sum b_2) \sum b_3)$
  - This is equivalent to applying one filter: $a \sum (b_1 \sum b_2 \sum b_3)$
- Identity: $f \sum e = f$
  - for unit impulse $e = [\ldots, 0, 0, 1, 0, 0, \ldots]$.
- Differentiation: $\frac{\partial}{\partial x}(f * g) = \frac{\partial f}{\partial x} * g$

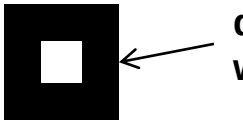
Averaging Filter

- What values belong in the kernel $H[u,v]$ for the moving average example?

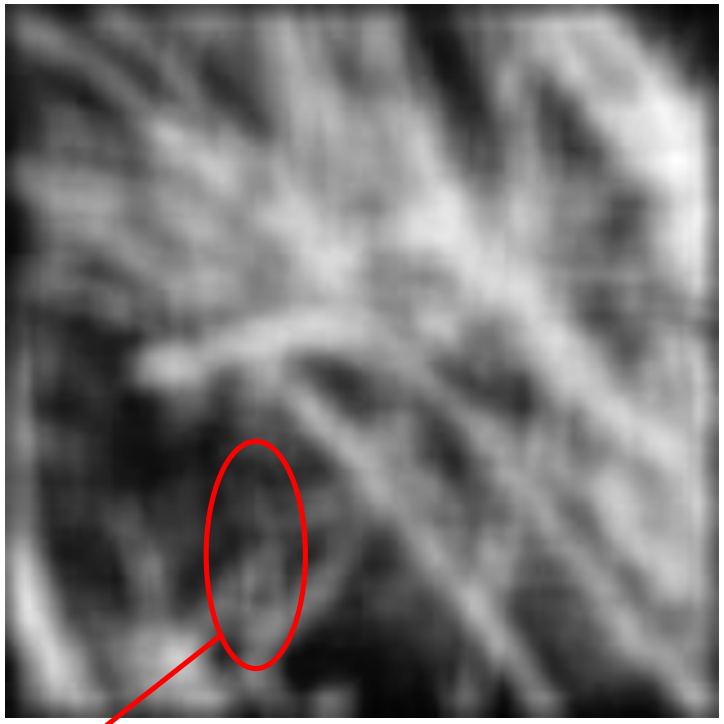


$$G = H \otimes F$$

Smoothing by Averaging



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

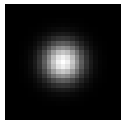


Original

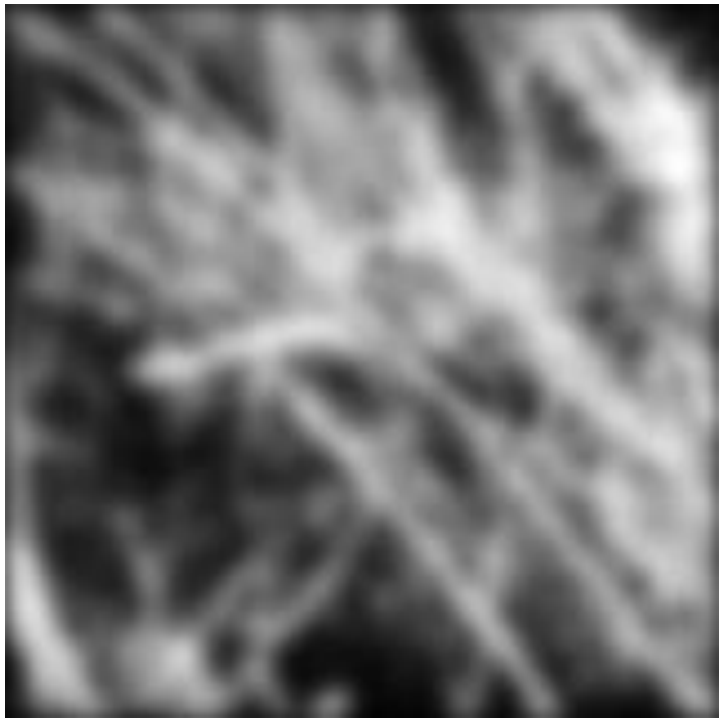
Filtered

"Ringing" artifacts!

Smoothing with a Gaussian



Original

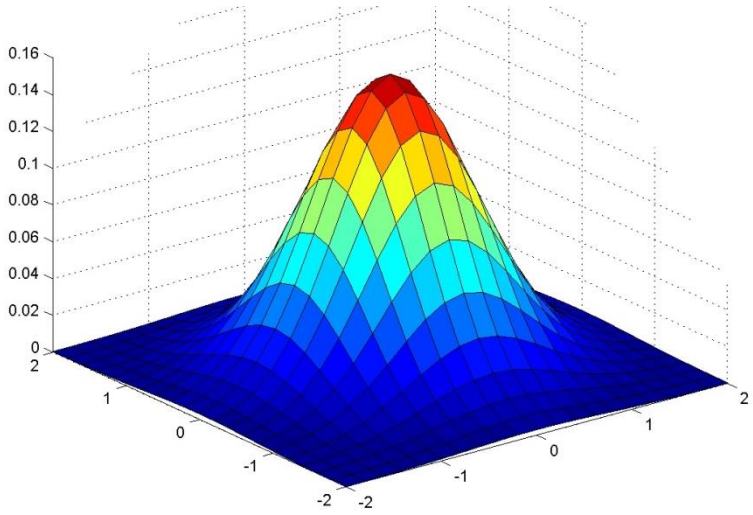


Filtered

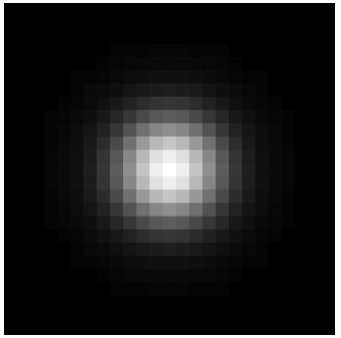
Gaussian Smoothing

- Gaussian kernel

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

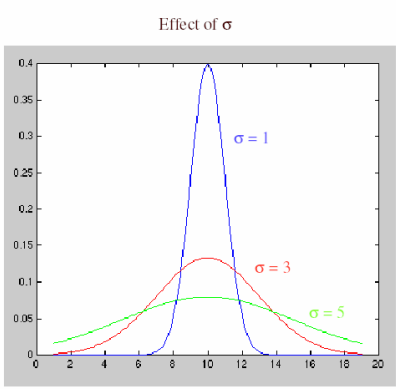


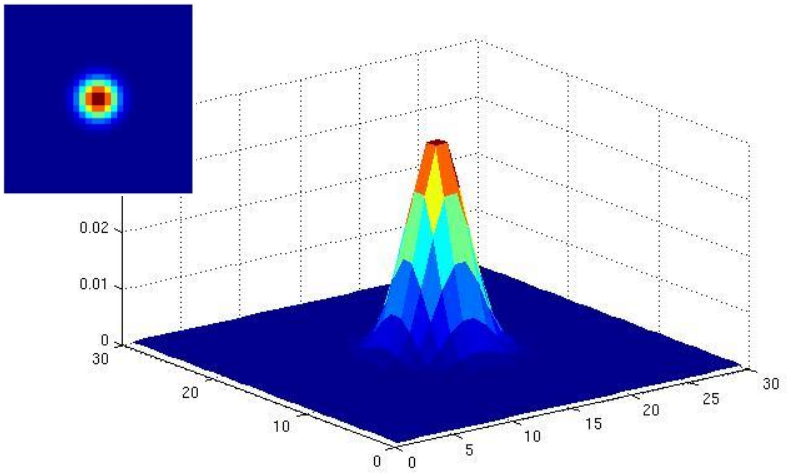
- Rotationally symmetric
- Weights nearby pixels more than distant ones
  - This makes sense as 'probabilistic' inference about the signal
- A Gaussian gives a good model of a fuzzy blob



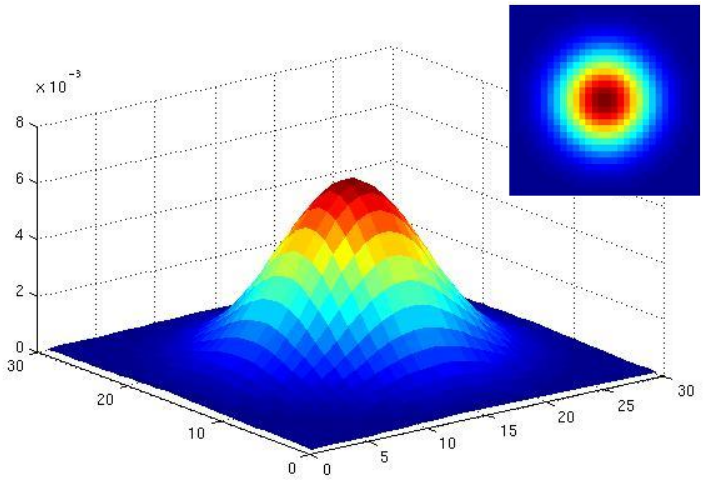
Gaussian Smoothing

- What parameters matter here?
- *Variance* σ of Gaussian
  - Determines extent of smoothing





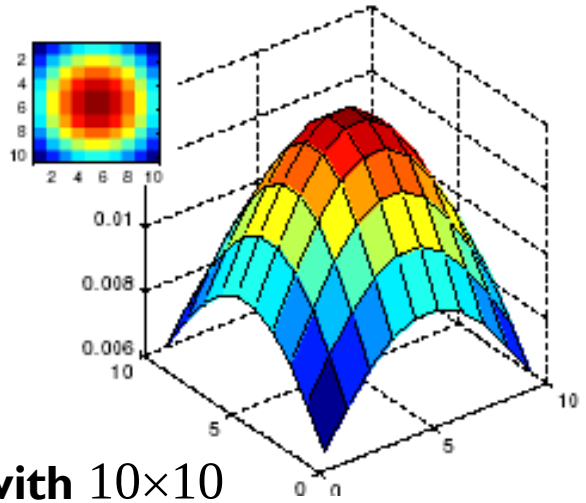
$\sigma = 2$ **with** 30×30 **kernel**



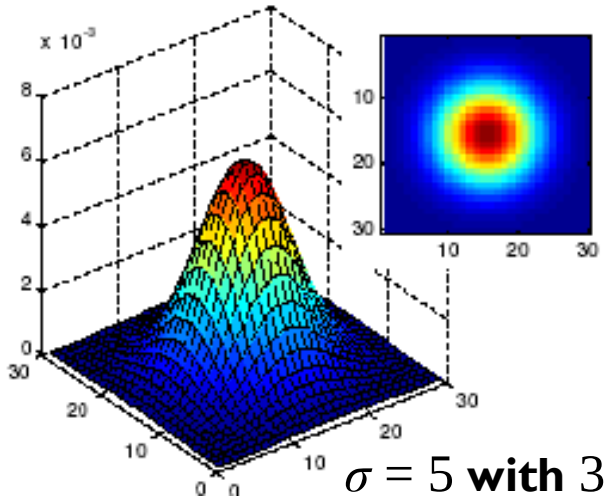
$\sigma = 5$ **with** 30×30 **kernel**

Gaussian Smoothing

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



$\sigma = 5$ **with** 10×10 **kernel**



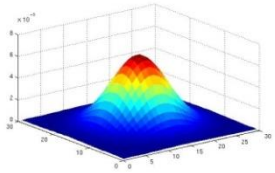
$\sigma = 5$ **with** 30×30 **kernel**

  - Rule : set filter half-width to about 3σ

Gaussian Smoothing in Matlab

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

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



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

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

Topics of This Lecture

- Linear filters
  - What are they? How are they applied?
  - Application: smoothing
  - Gaussian filter
  - What does it mean to filter an image?
- Nonlinear Filters
  - Median filter
- Multi-Scale representations
  - How to properly rescale an image?
- Image derivatives
  - How to compute gradients robustly?

Why Does This Work?

- A small excursion into the Fourier transform to talk about spatial frequencies…

=

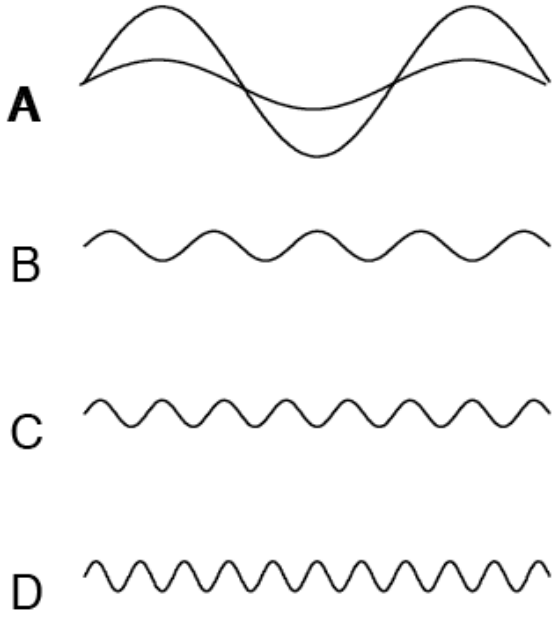
3 cos(*x*)

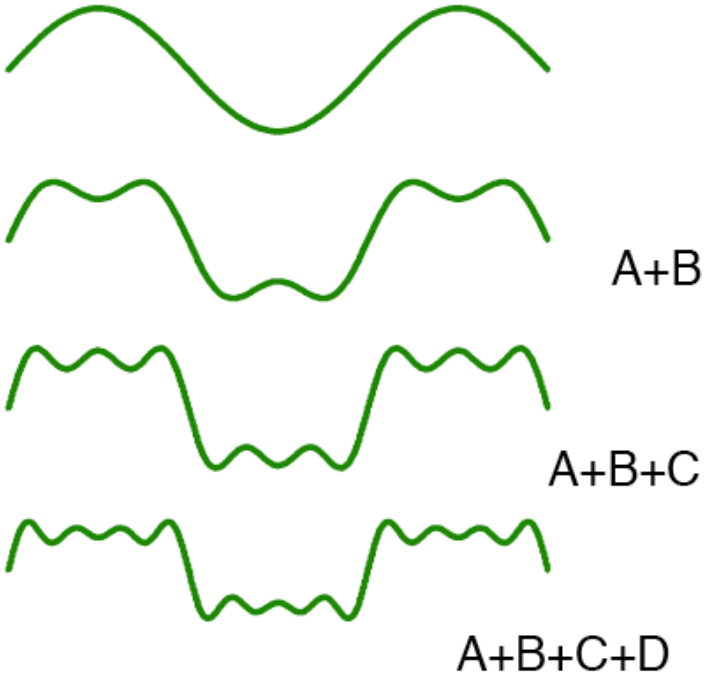
+ 1 cos(3*x*)

+ 0.8 cos(5*x*)

+ 0.4 cos(7*x*)

+ …





The Fourier Transform in Pictures

- A small excursion into the Fourier transform to talk about spatial frequencies…

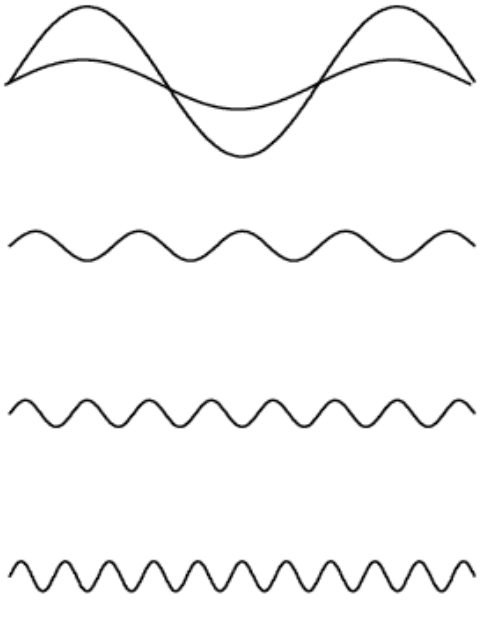
$3\cos(x)$

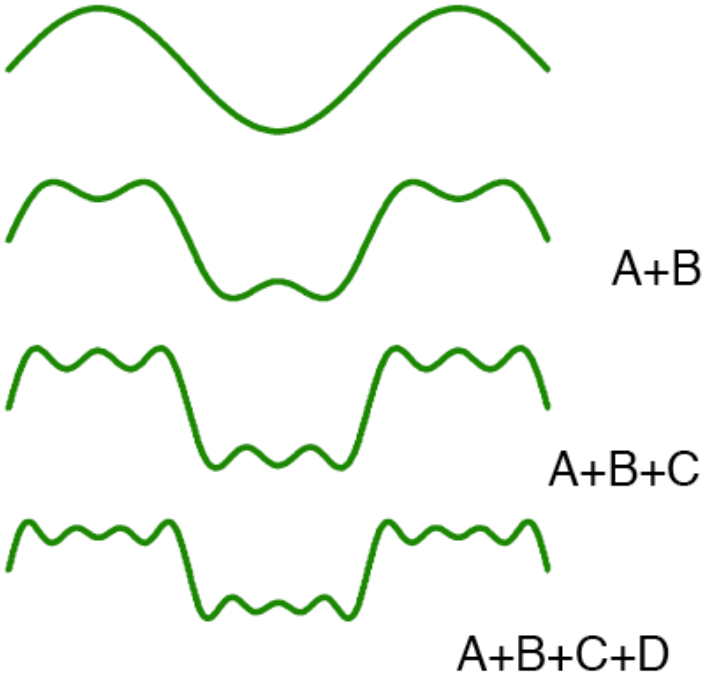
$+ 1\cos(3x)$

$+ 0.8\cos(5x)$

$+ 0.4\cos(7x)$

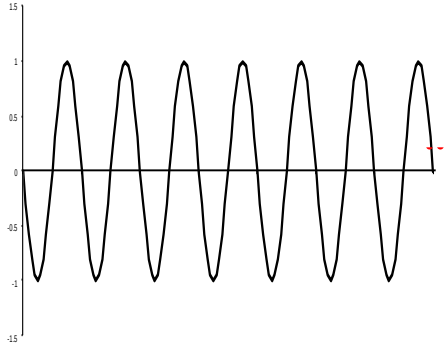
$+ \ldots$



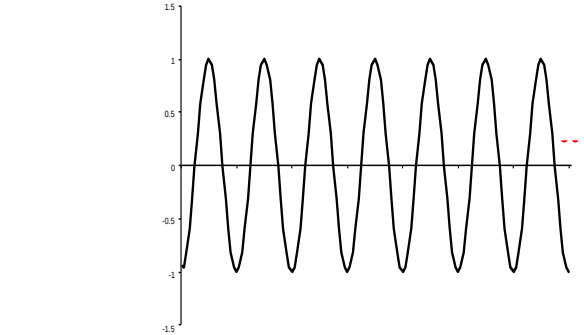


Fourier Transforms of Important Functions

- Sine and cosine transform to…



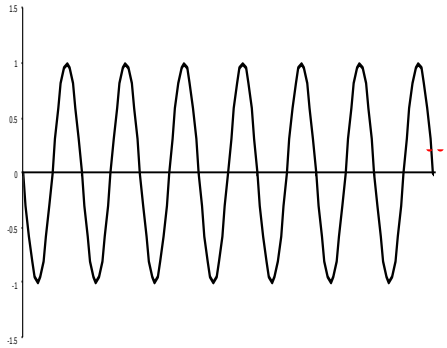
?

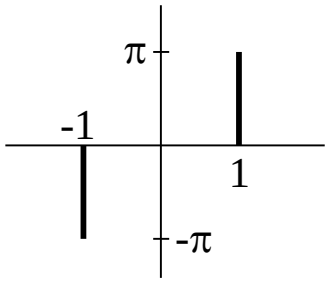


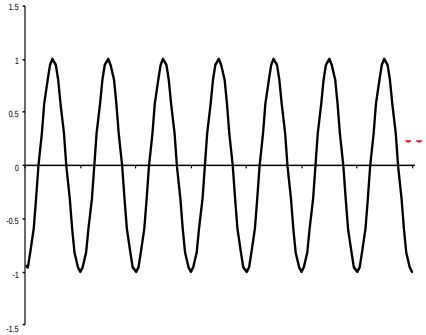
?

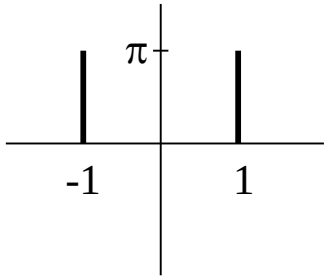
Fourier Transforms of Important Functions

- Sine and cosine transform to "frequency spikes"

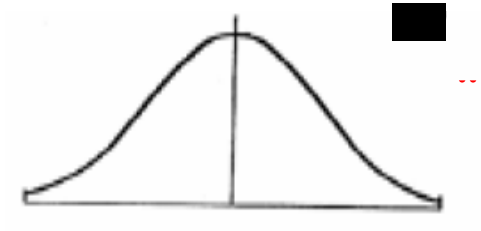








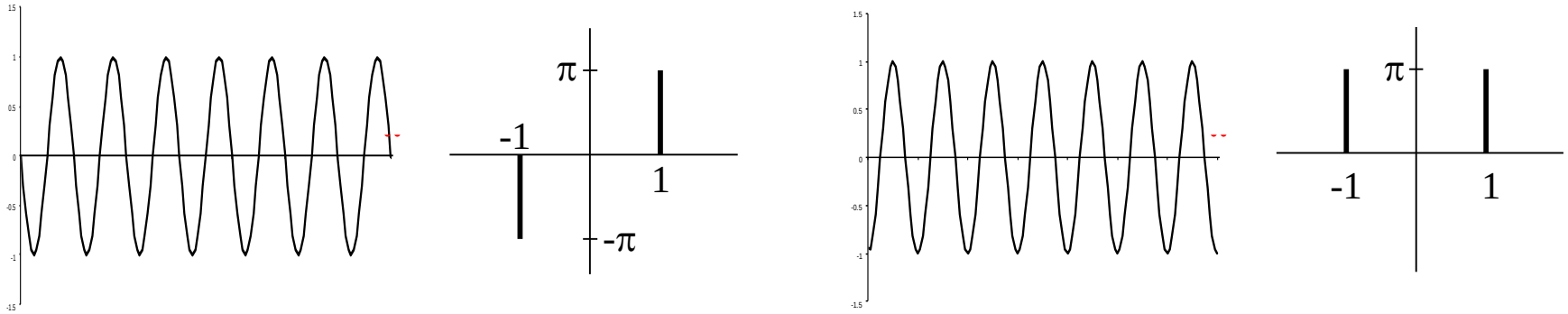
- A Gaussian transforms to…



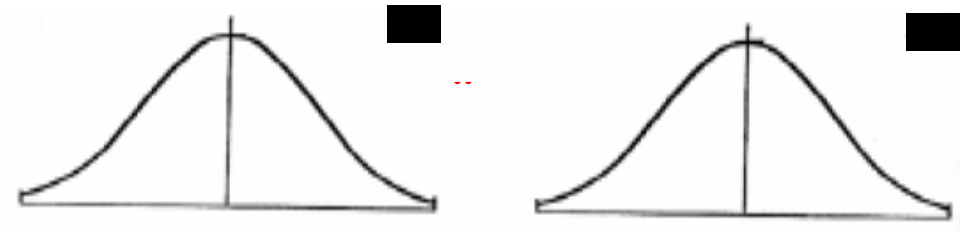
?

Fourier Transforms of Important Functions

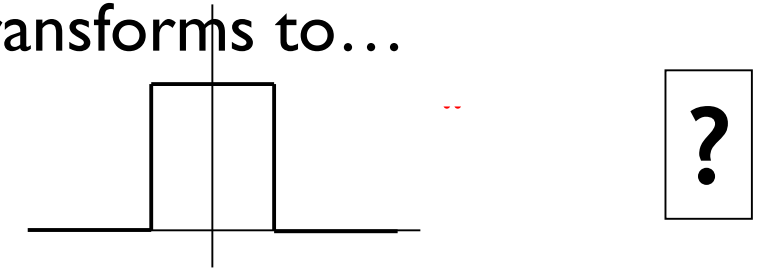
- Sine and cosine transform to "frequency spikes"



- A Gaussian transforms to a Gaussian

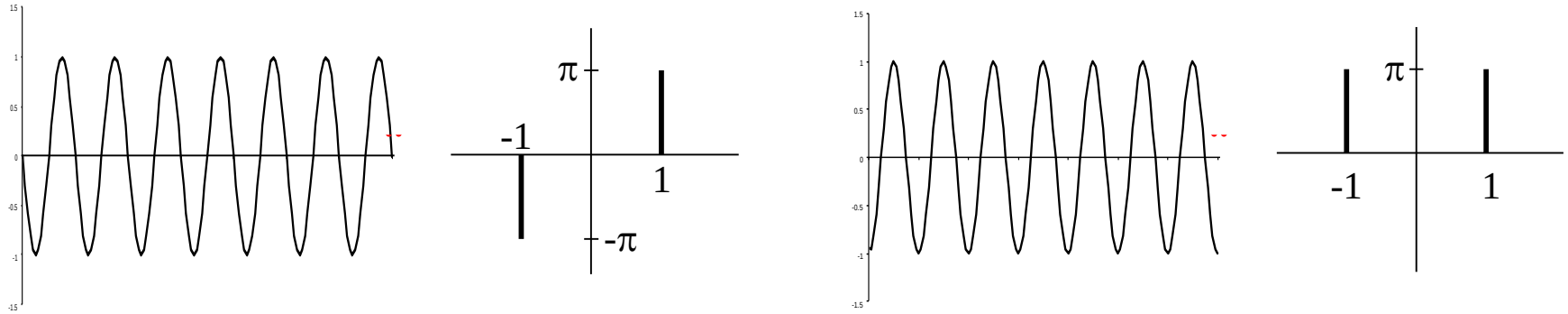


- A box filter transforms to…

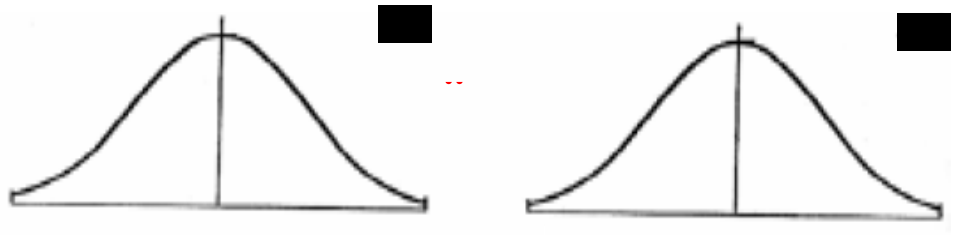


Fourier Transforms of Important Functions

- Sine and cosine transform to "frequency spikes"

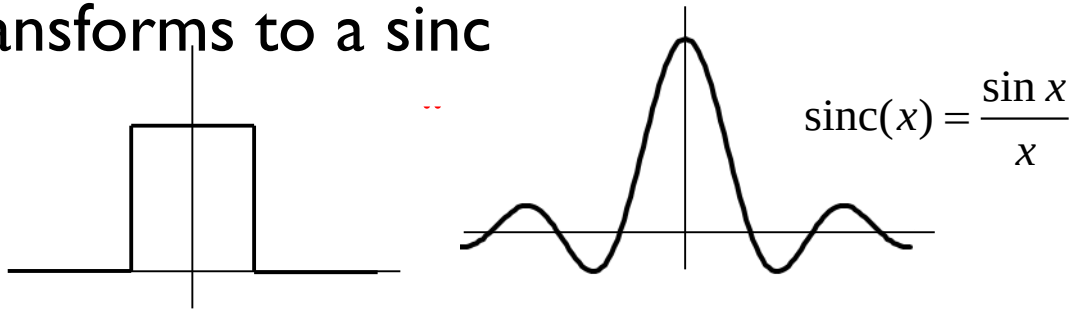


- A Gaussian transforms to a Gaussian



All of this is symmetric!

- A box filter transforms to a sinc



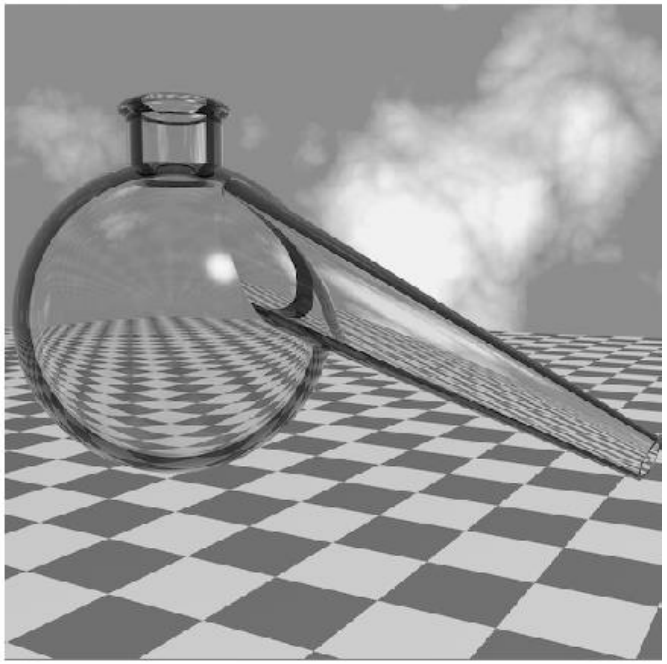
Effect of Convolution

- Convolving two functions in the image domain corresponds to taking the product of their transformed versions in the frequency domain.

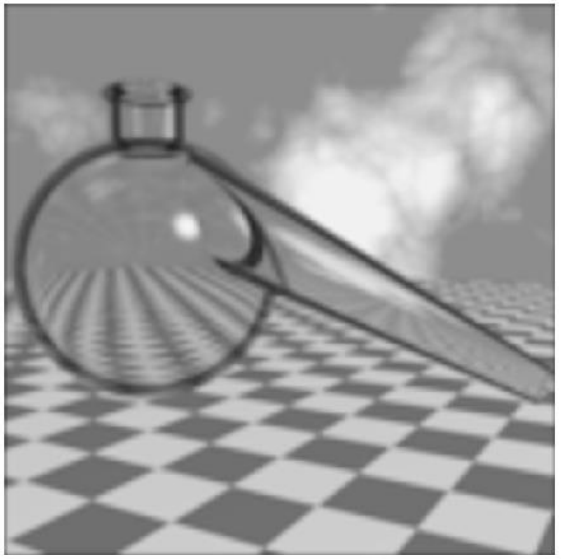
$$f * g \quad \ddot{} \quad \mathrm{F} \cdot \mathrm{G}$$

- This gives us a tool to manipulate image spectra.
  - A filter attenuates or enhances certain frequencies through this effect.

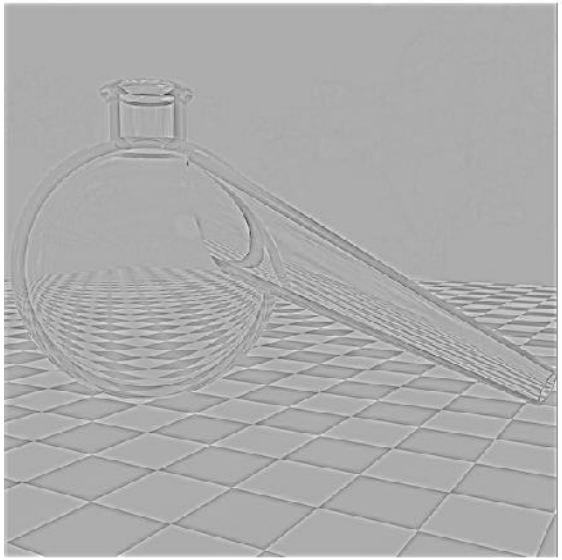
Low-Pass vs. High-Pass



Original image



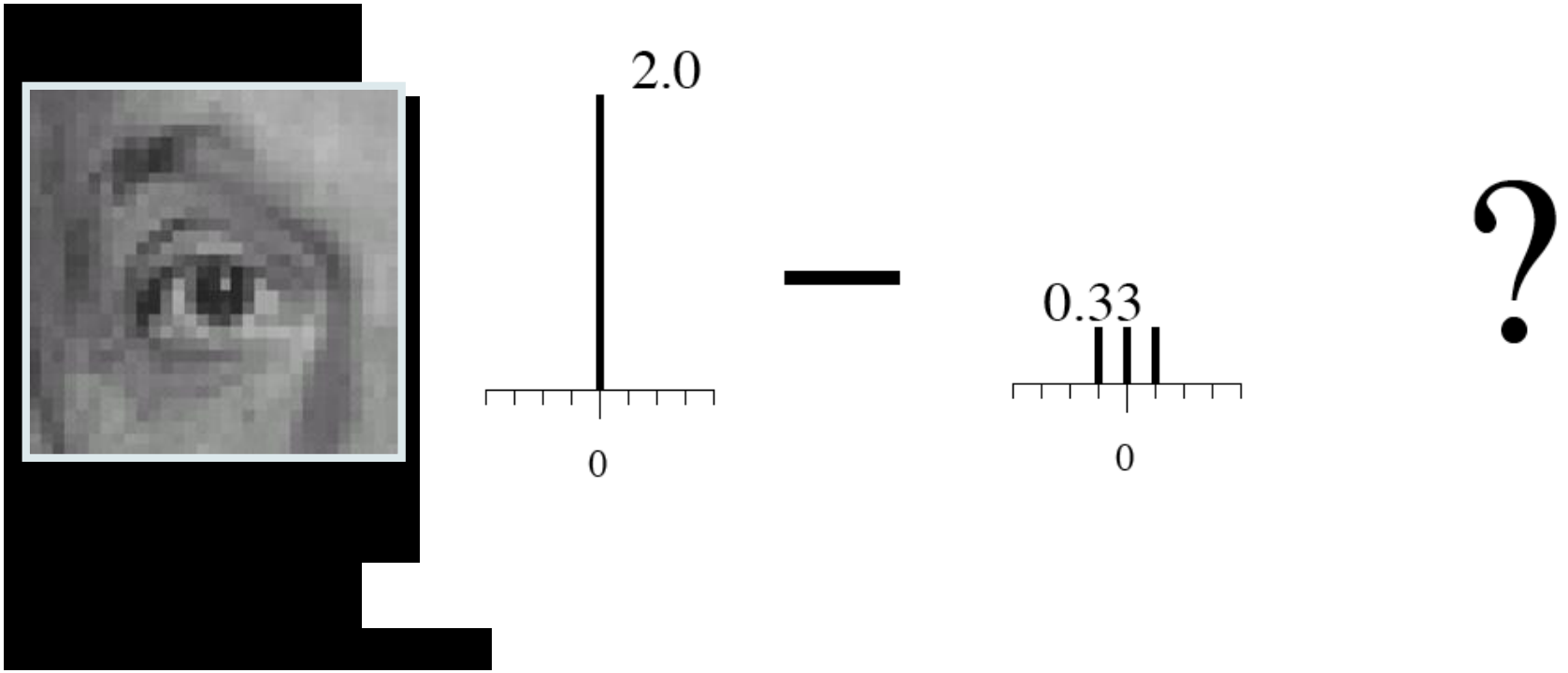
Low-pass filtered



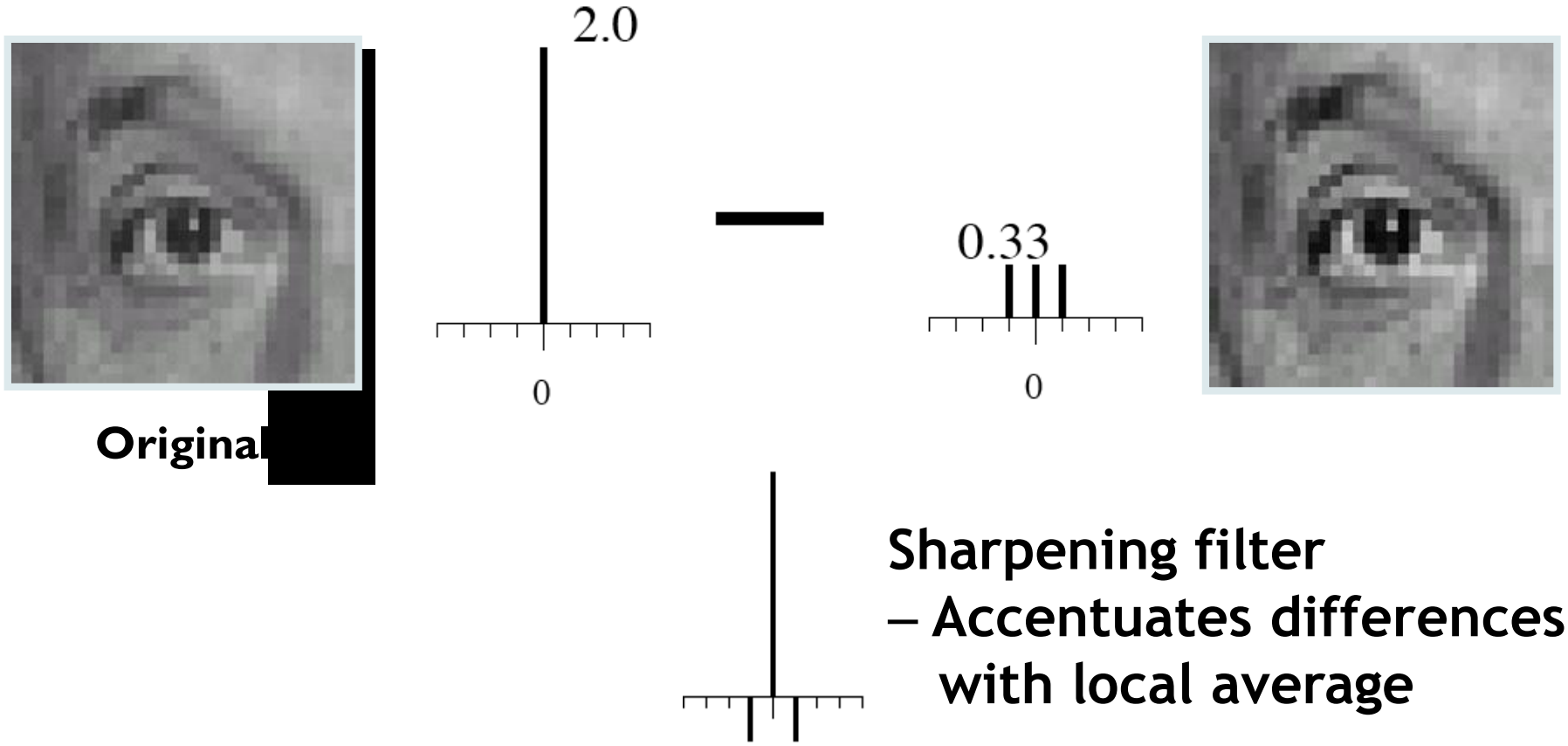
High-pass filtered

Image Source: S. Chenney

Quiz: What Effect Does This Filter Have?

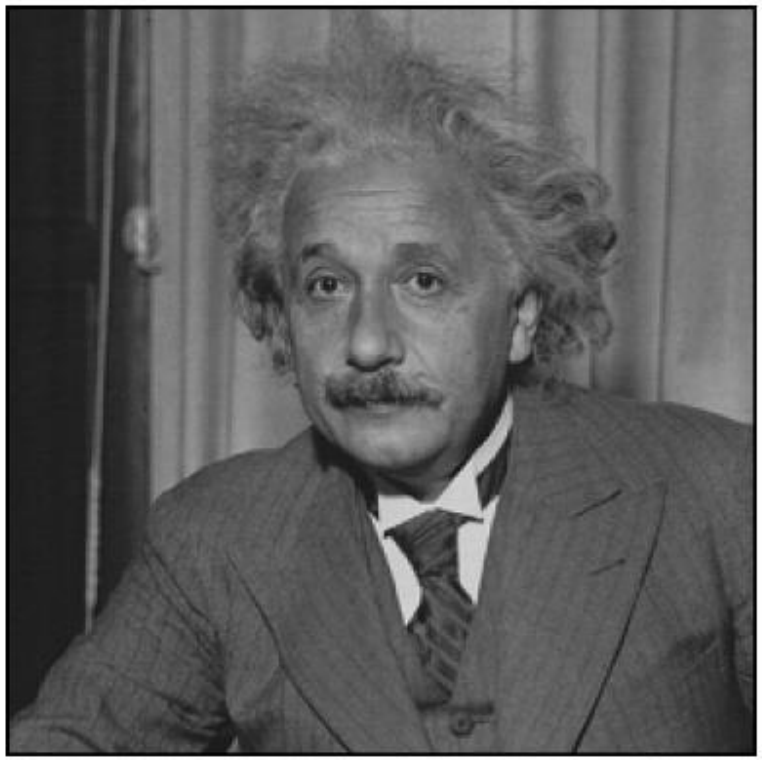


Sharpening Filter

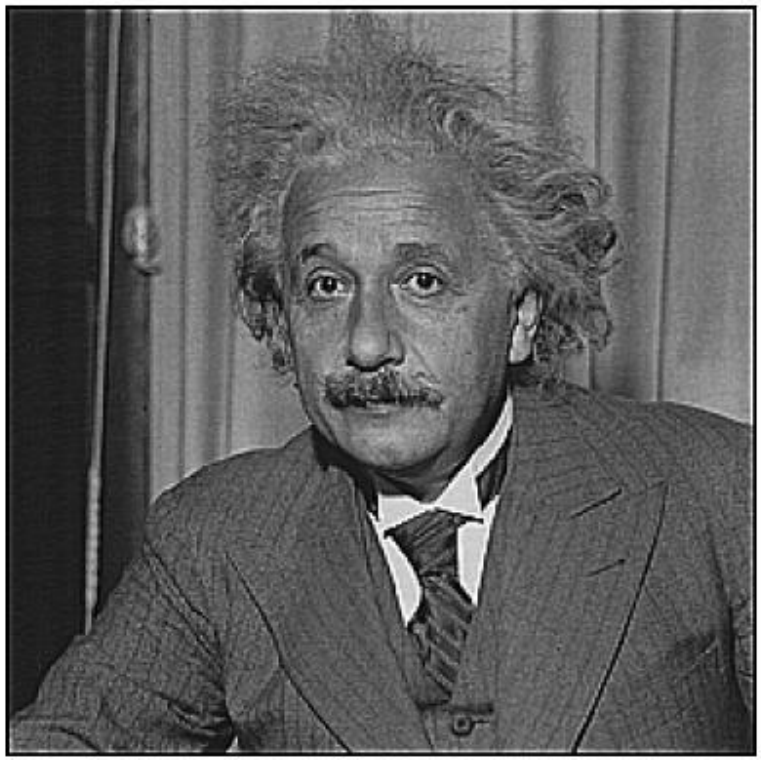


Sharpening filter
– Accentuates differences with local average

Sharpening Filter



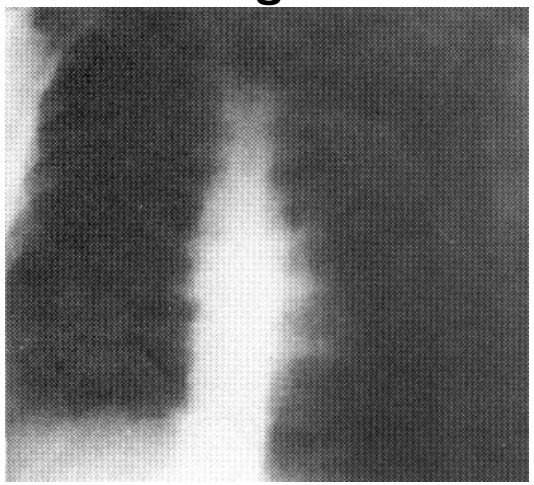
before



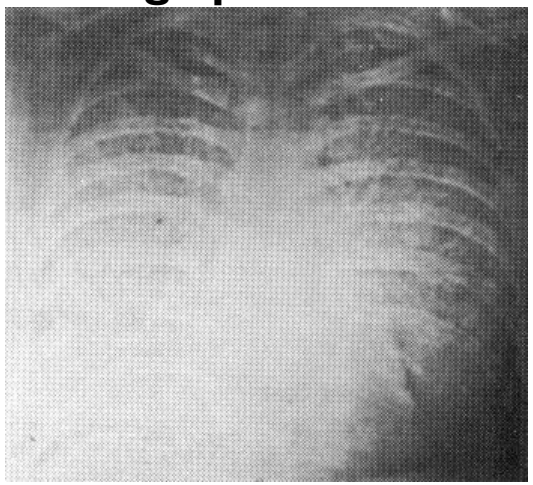
after

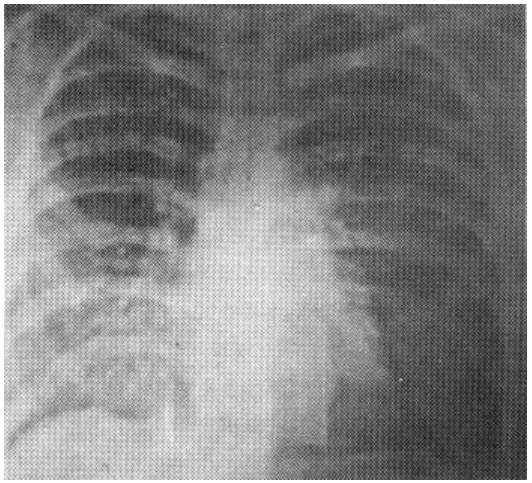
Application: High Frequency Emphasis

Original

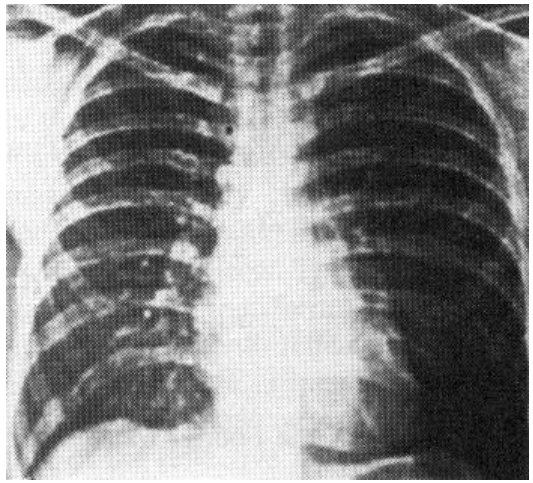


High pass Filter





High Frequency Emphasis



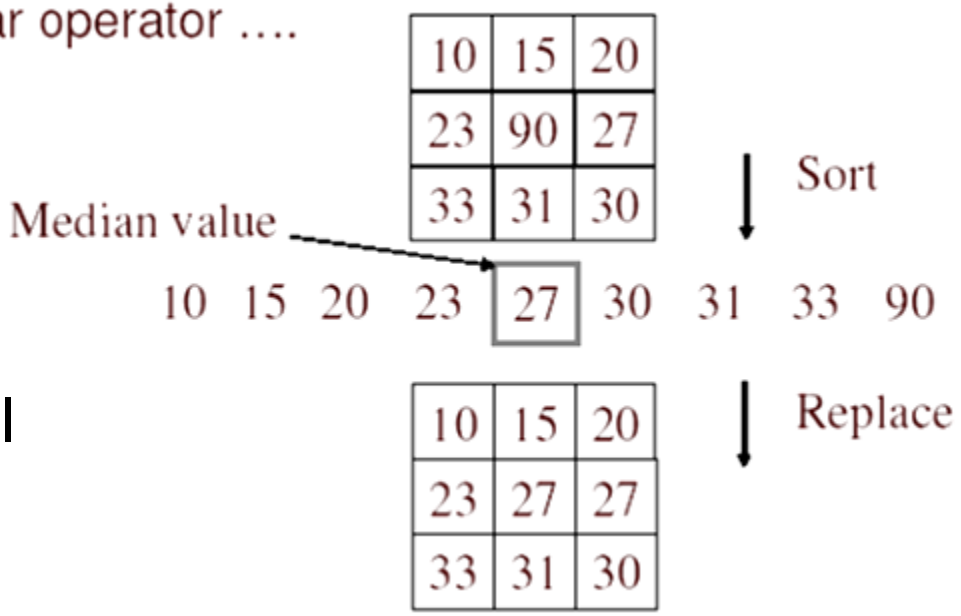
High Frequency Emphasis + Histogram Equalization

Topics of This Lecture

- Linear filters
  - What are they? How are they applied?
  - Application: smoothing
  - Gaussian filter
  - What does it *mean* to filter an image?
- Nonlinear Filters
  - Median filter
- Multi-Scale representations
  - How to properly rescale an image?
- Image derivatives
  - How to compute gradients robustly?

Non-Linear Filters: Median Filter

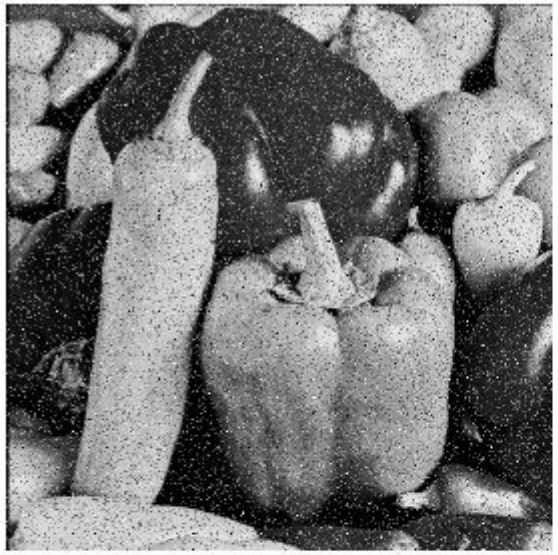
- Basic idea
  - Replace each pixel by the ar operator …. median of its neighbors.



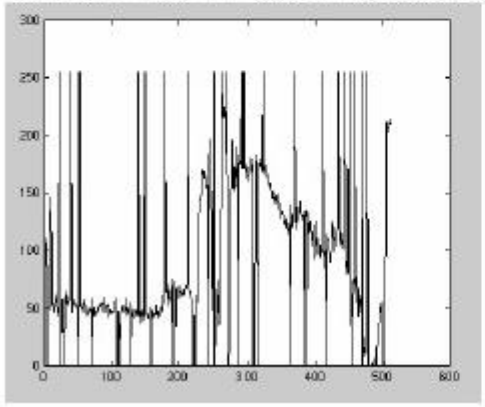
- Properties
  - Doesn't introduce new pixel values
  - Removes spikes: good for impulse, salt & pepper noise
  - Linear?

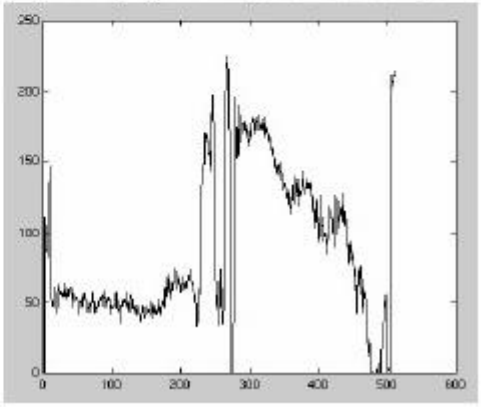
Median Filter

Salt and pepper noise



Median filtered

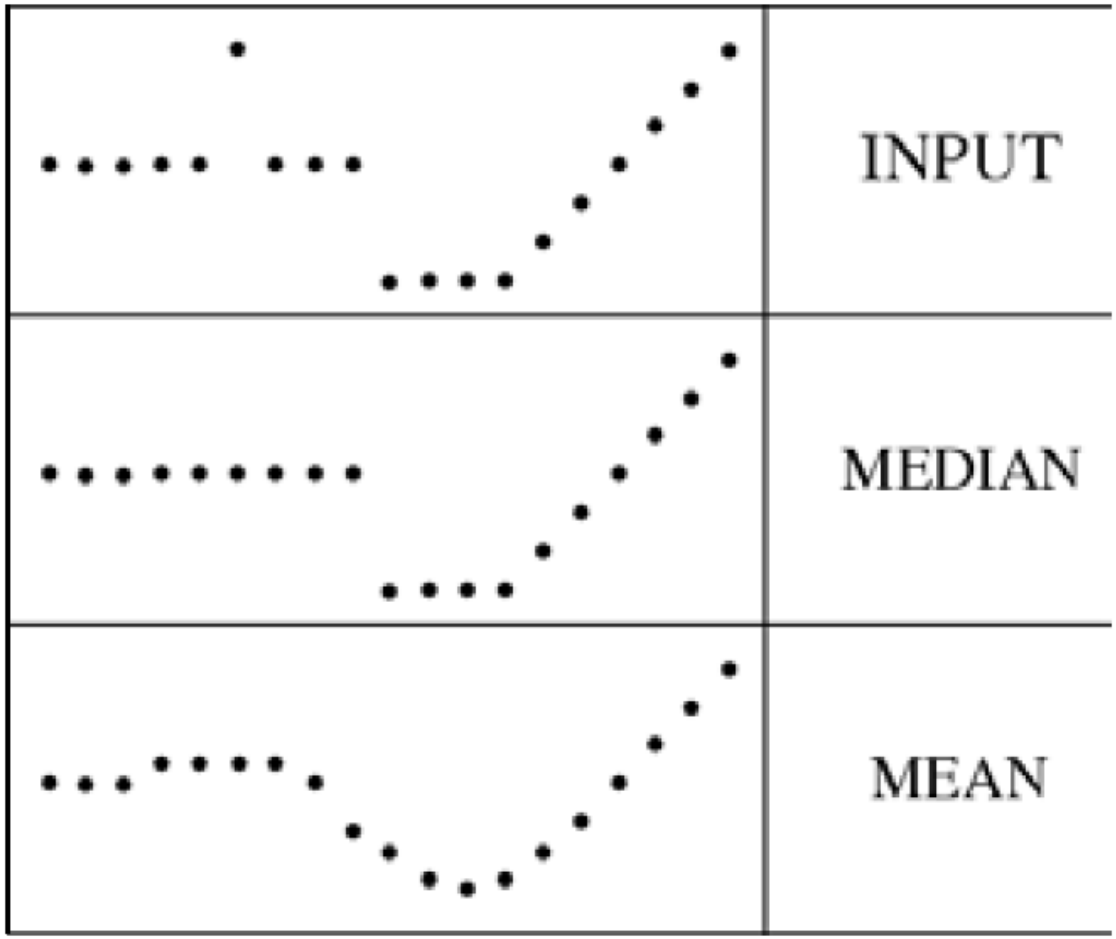




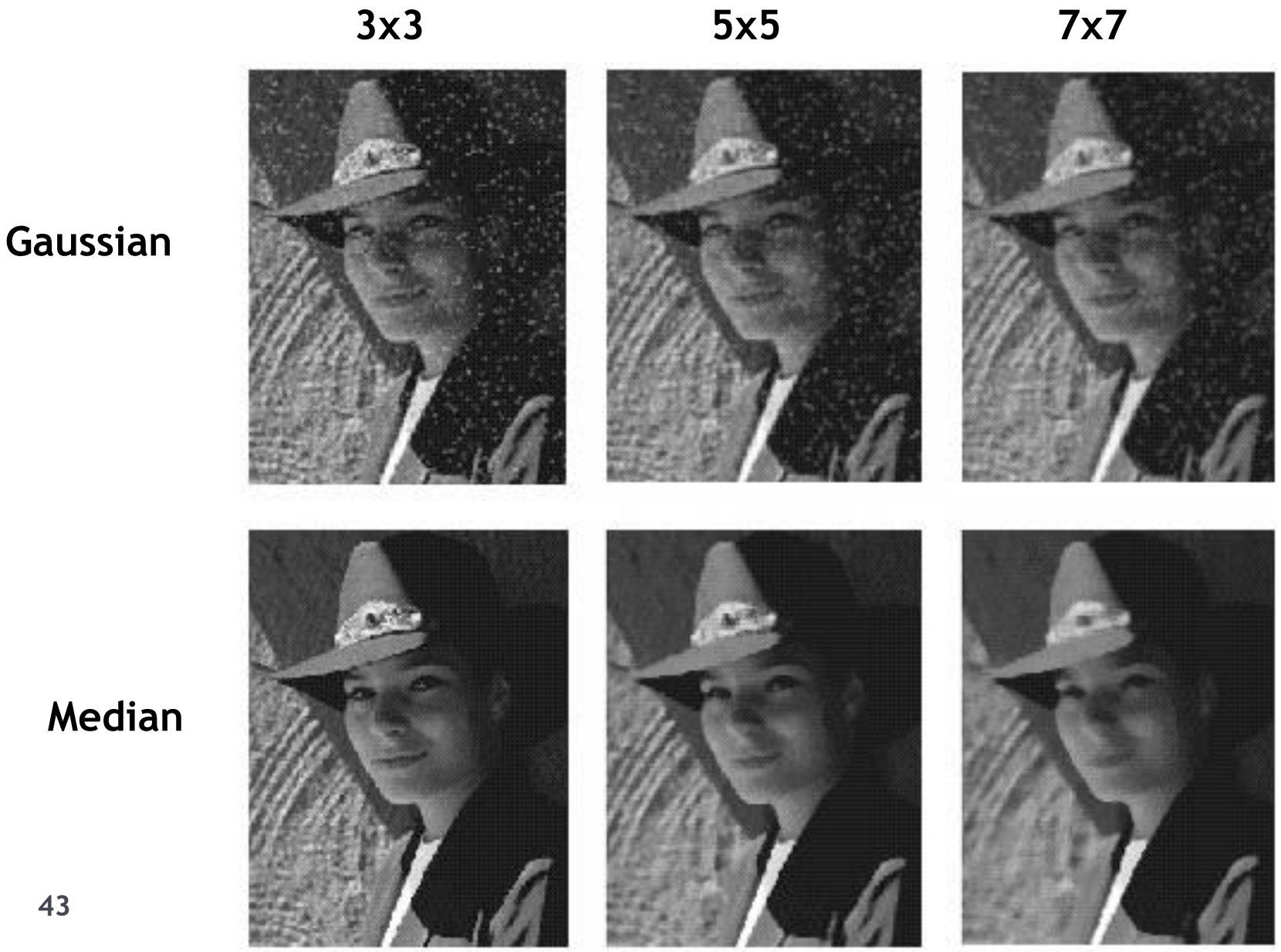
Plots of a row of the image

Median Filter

- The Median filter is edge preserving.



Median vs. Gaussian Filtering



Topics of This Lecture

- Linear filters
  - What are they? How are they applied?
  - Application: smoothing
  - Gaussian filter
  - What does it *mean* to filter an image?
- Nonlinear Filters
  - Median filter

- Multi-Scale representations
  - How to properly rescale an image?
- Image derivatives
  - How to compute gradients robustly?

Motivation: Fast Search Across Scales

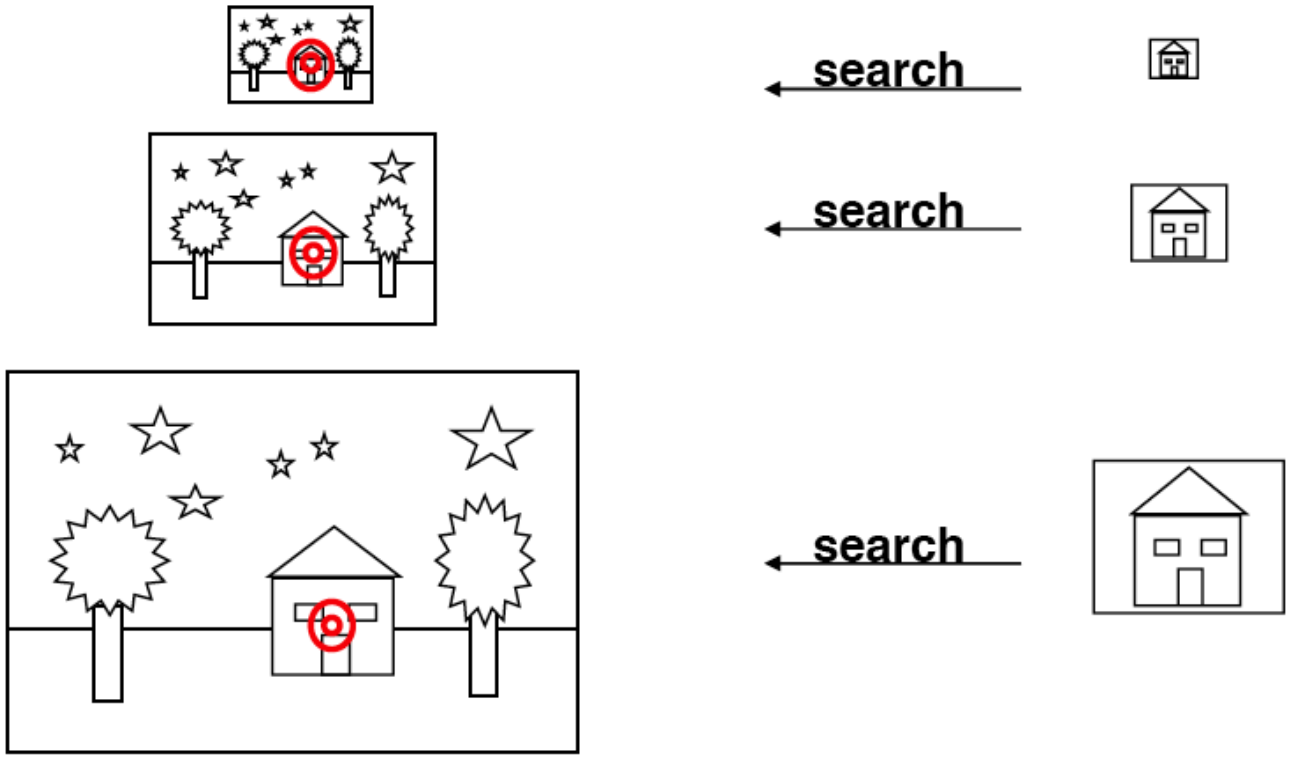
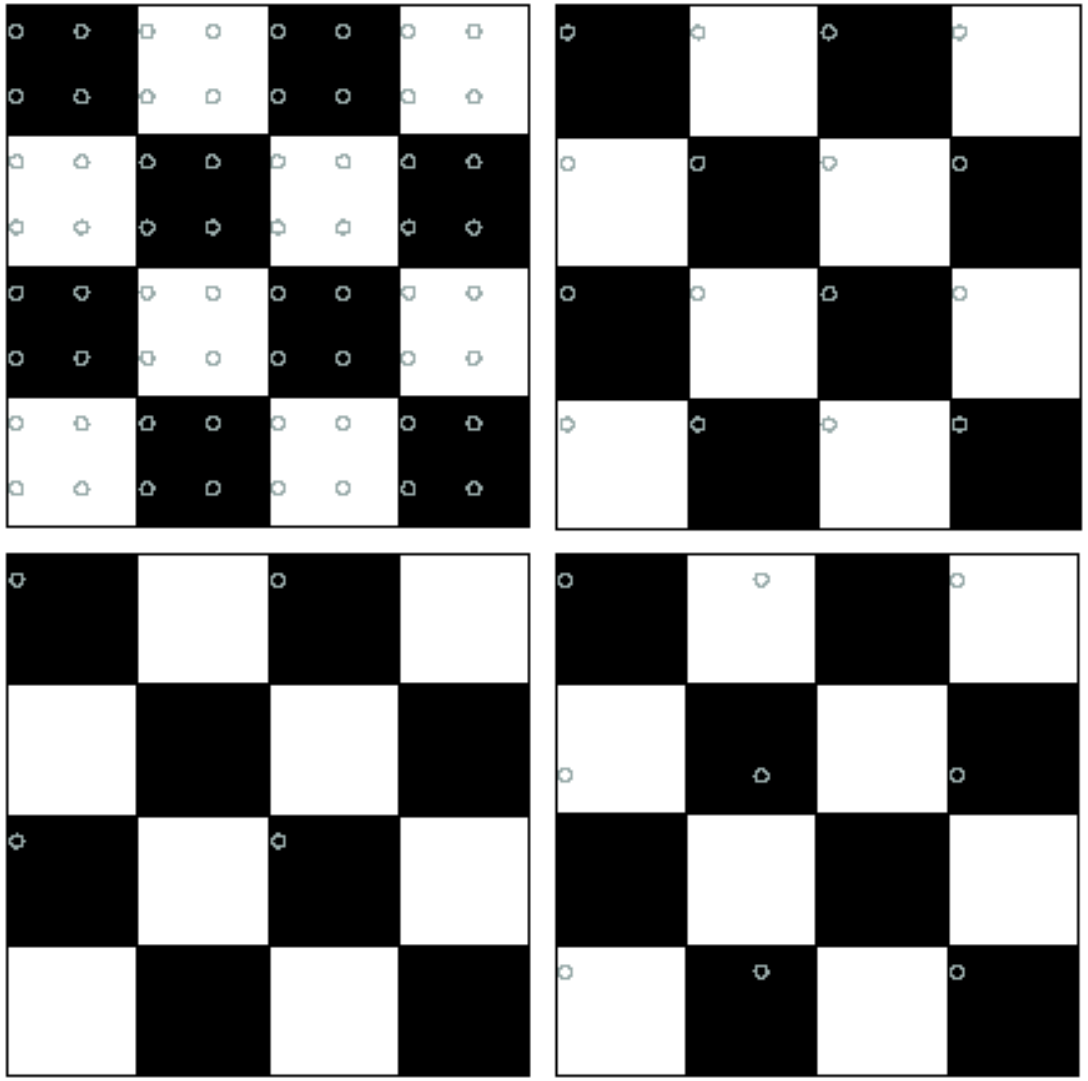


Image Pyramid

Low resolution

High resolution

How Should We Go About Resampling?



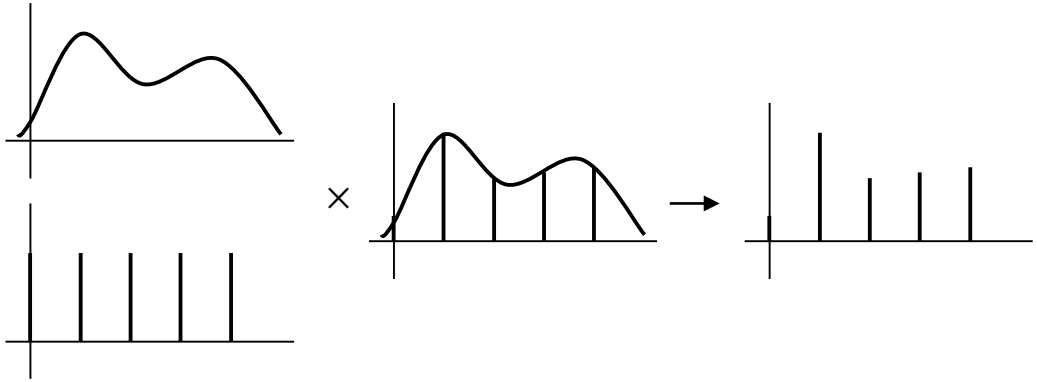
Let's resample the checkerboard by taking one sample at each circle.

In the top left board, the new representation is reasonable. Top right also yields a reasonable representation.

Bottom left is all black (dubious) and bottom right has checks that are too big.

Fourier Interpretation: Discrete Sampling

- Sampling in the spatial domain is like multiplying with a spike function.



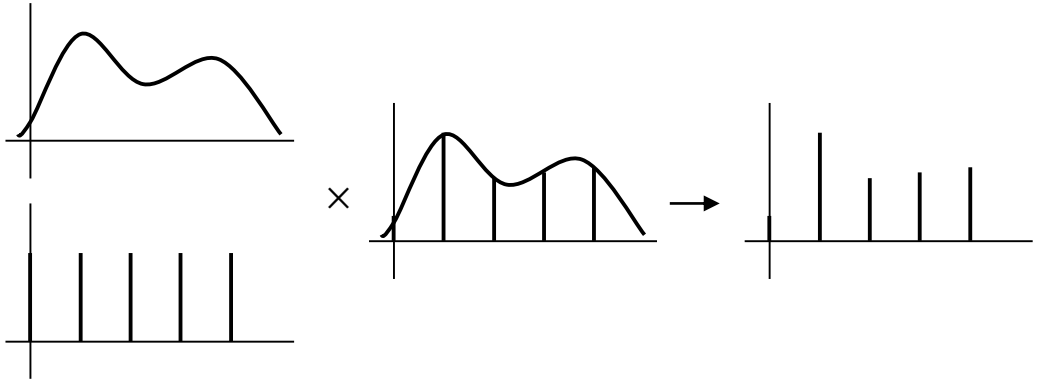
- Sampling in the frequency domain is like...

?

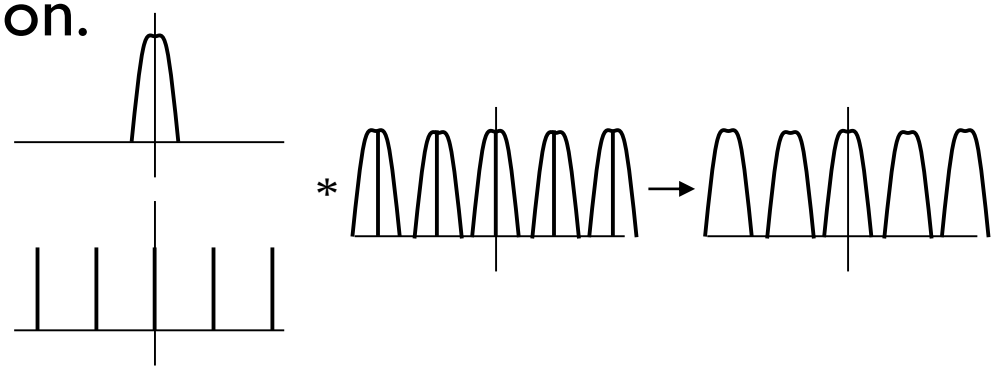
Source: S. Chenney

Fourier Interpretation: Discrete Sampling

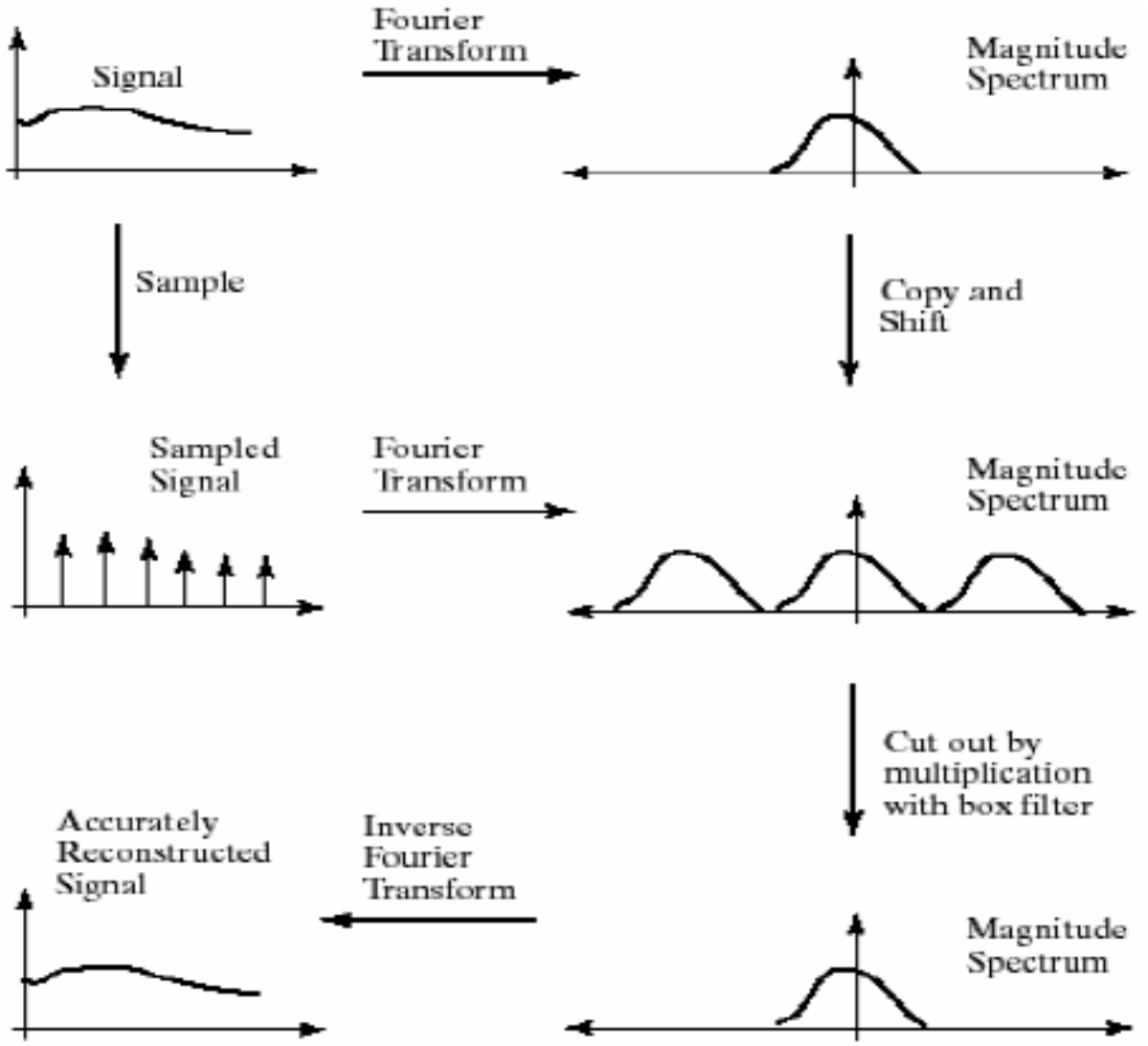
- Sampling in the spatial domain is like multiplying with a spike function.



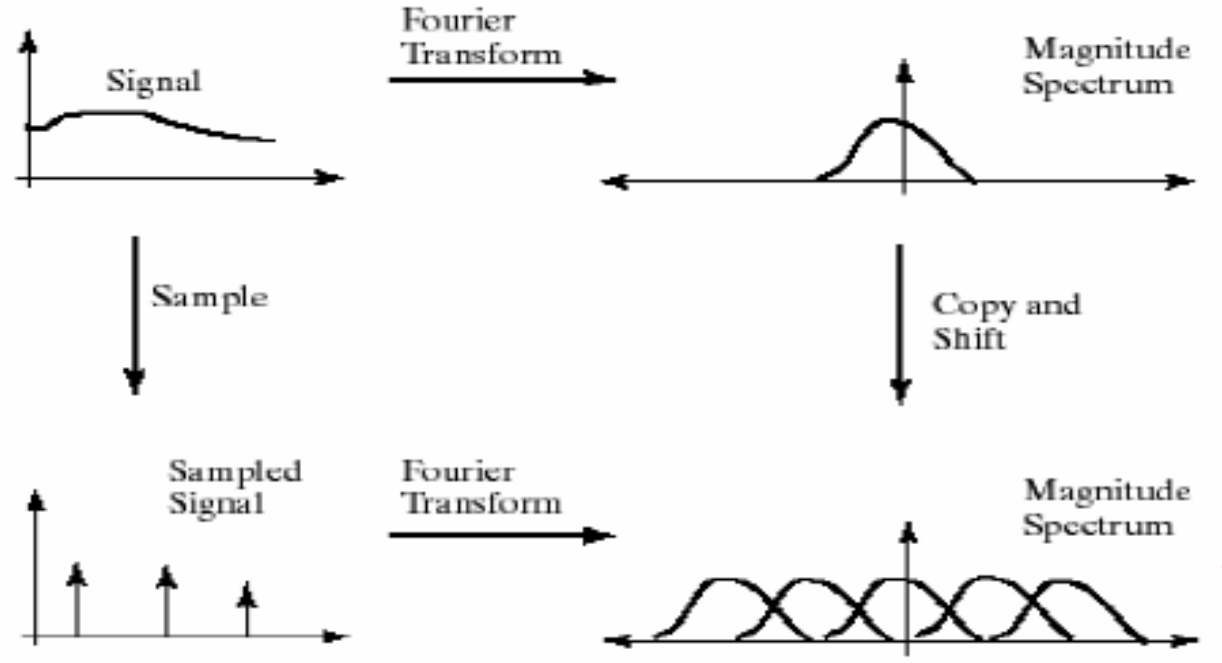
- Sampling in the frequency domain is like convolving with a spike function.



Sampling and Aliasing



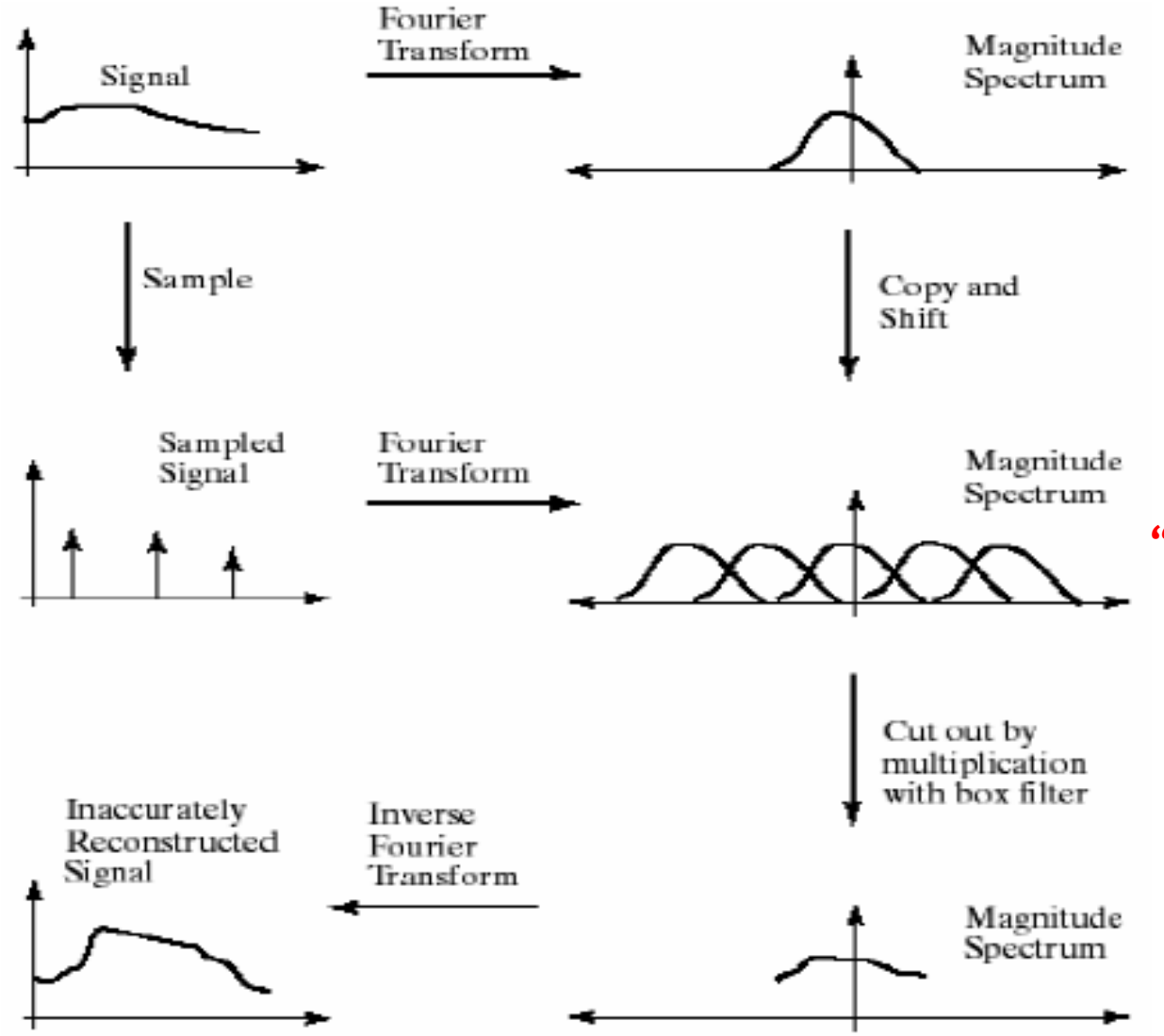
Sampling and Aliasing



"Nyquist limit"

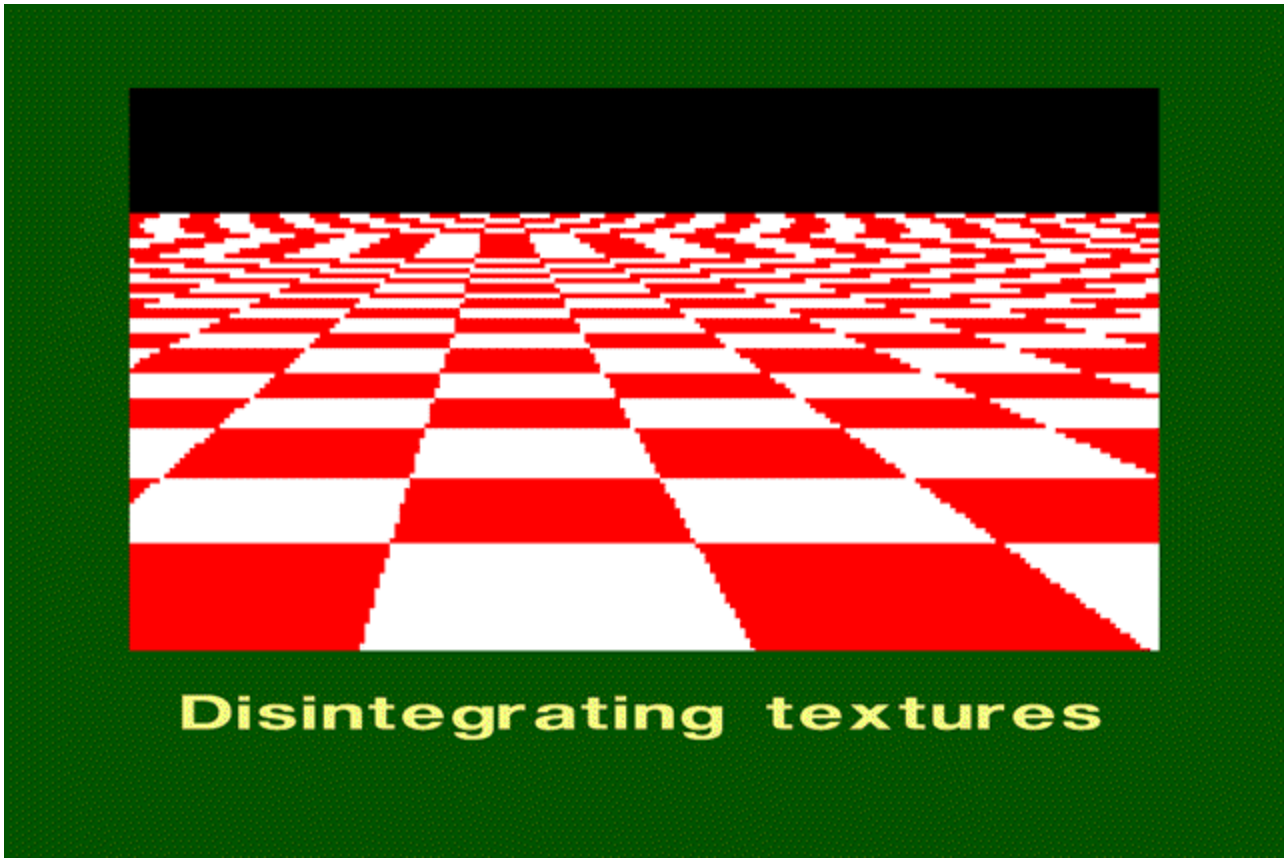
- Nyquist theorem:
  - In order to recover a certain frequency f, we need to sample with at least $2f$.
  - This corresponds to the point at which the transformed frequency spectra start to overlap.

Sampling and Aliasing

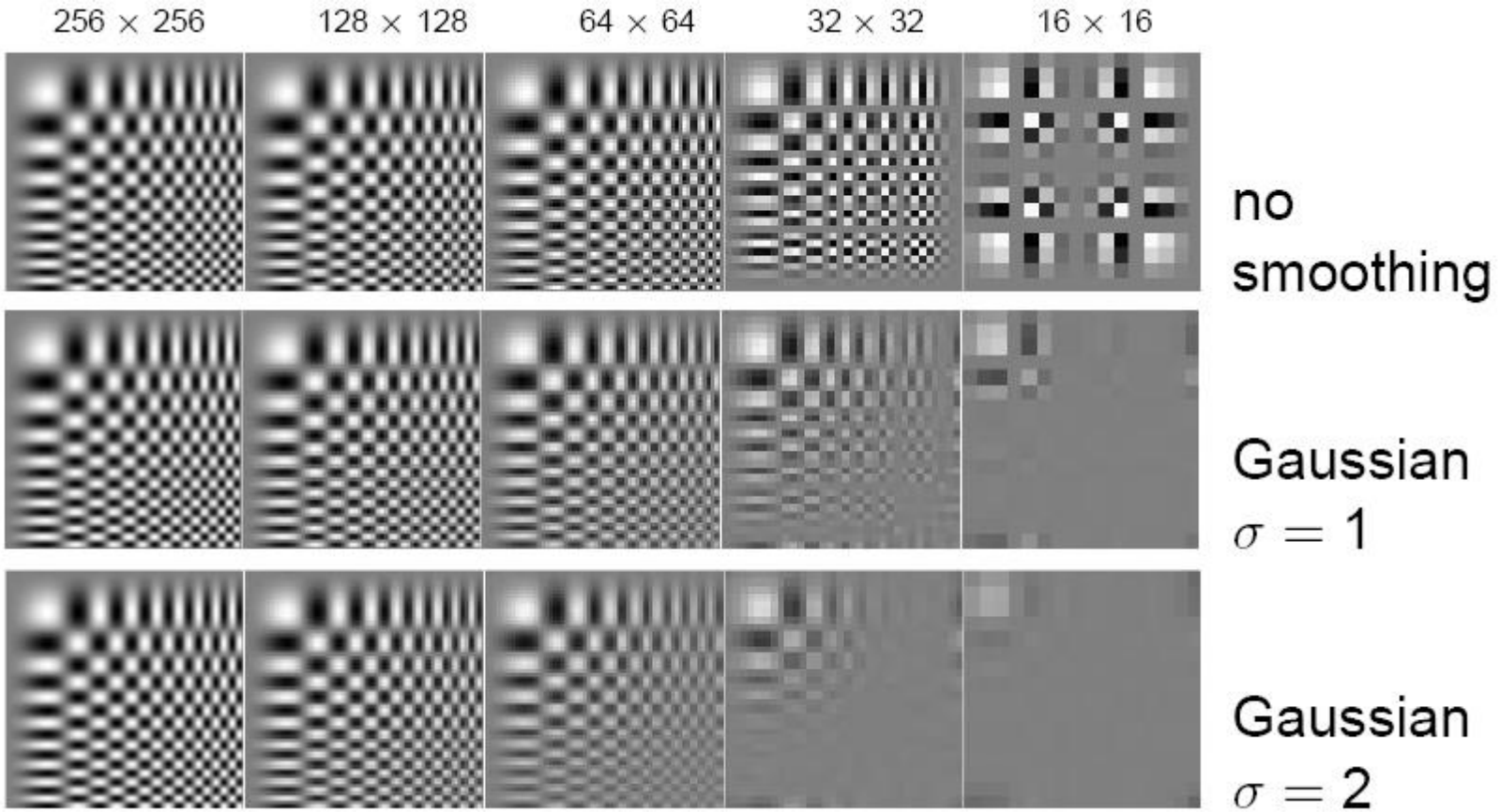


"Nyquist limit"

Aliasing in Graphics

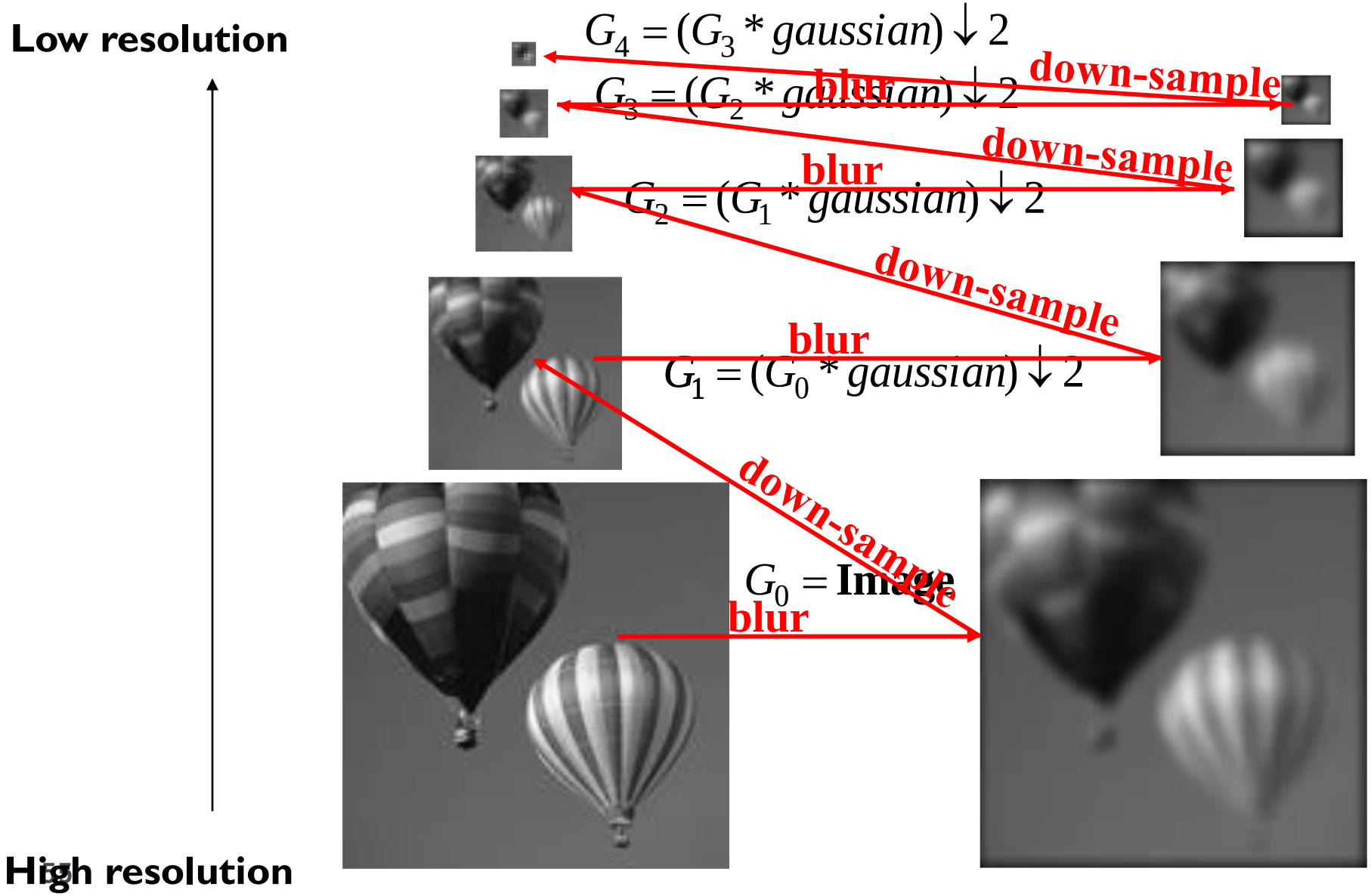


Resampling with Prior Smoothing

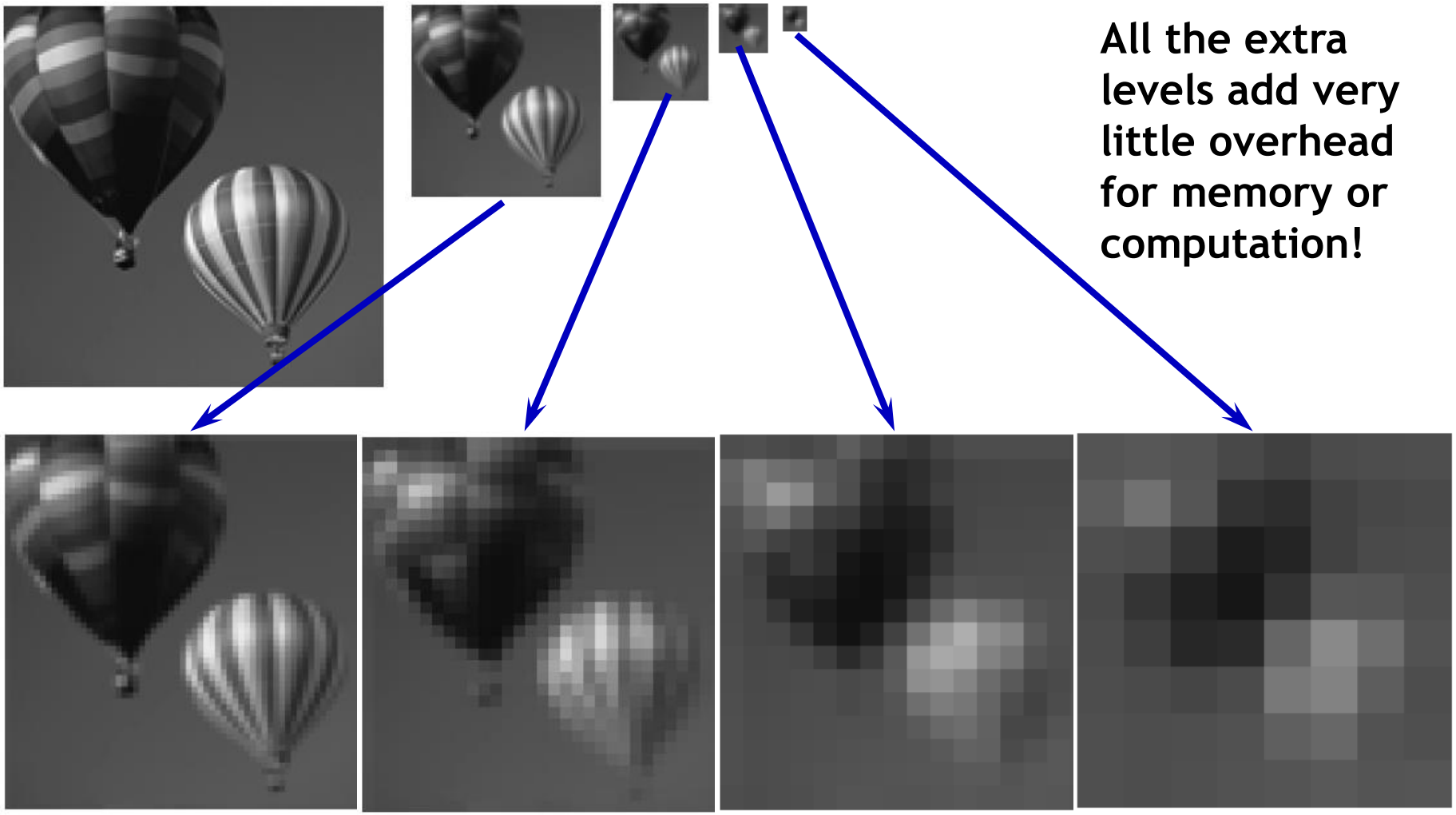


- Note: We cannot recover the high frequencies, but we can avoid artifacts by smoothing before resampling.

The Gaussian Pyramid



Gaussian Pyramid – Stored Information



All the extra levels add very little overhead for memory or computation!

Summary: Gaussian Pyramid

- Construction: create each level from previous one
  - Smooth and sample
- Smooth with Gaussians, in part because
  - Gaussian*Gaussian = another Gaussian
  - $G(\sigma_1) * G(\sigma_2) = G(\mathrm{sqrt}(\sigma_1^{\,2} + \sigma_2^{\,2}))$
- Gaussians are low-pass filters, so the representation is redundant once smoothing has been performed.

  $\Rightarrow$ There is no need to store smoothed images at the full original resolution.

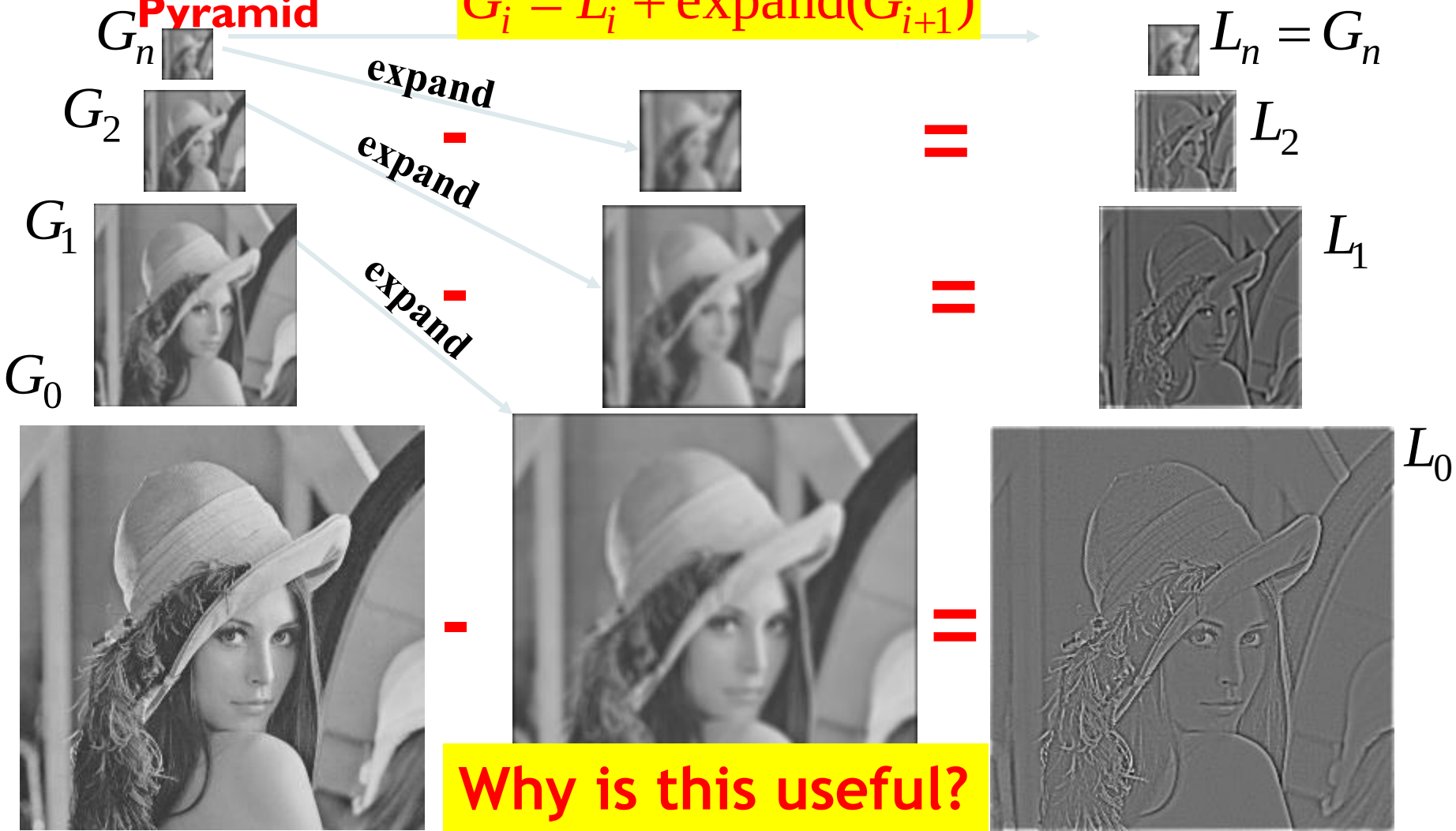
The Laplacian Pyramid

$$L_i = G_i - \text{expand}(G_{i+1})$$

$$G_i = L_i + \text{expand}(G_{i+1})$$

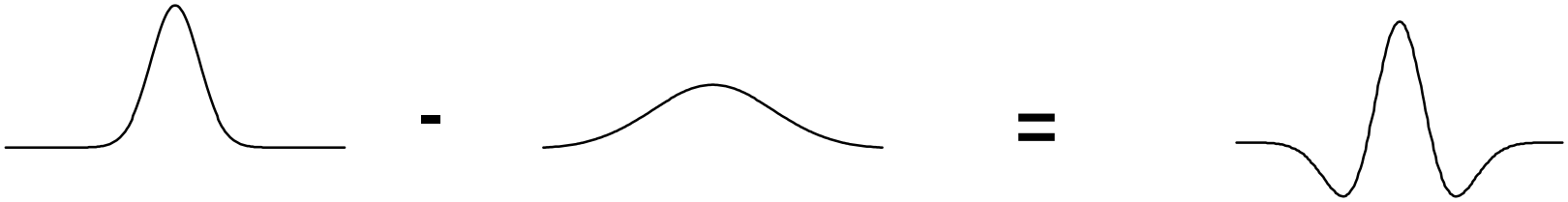
Gaussian Pyramid

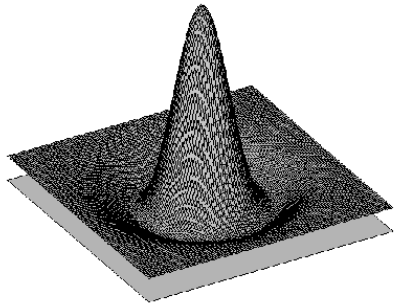
Laplacian Pyramid



Why is this useful?

Laplacian ~ Difference of Gaussian



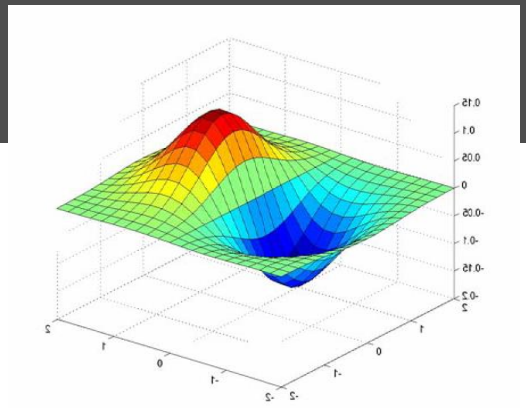


DoG = Difference of Gaussians

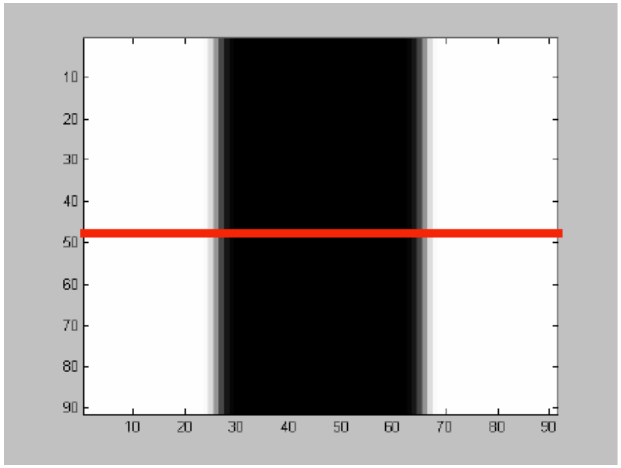
Cheap approximation - no derivatives needed.

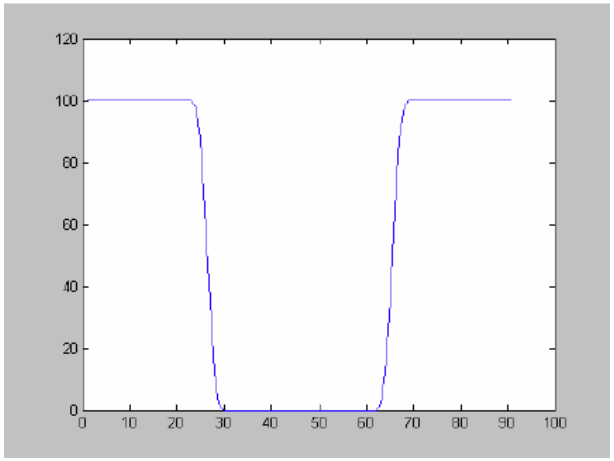
Topics of This Lecture

- Linear filters
  - What are they? How are they applied?
  - Application: smoothing
  - Gaussian filter
  - What does it *mean* to filter an image?
- Nonlinear Filters
  - Median filter
- Multi-Scale representations
  - How to properly rescale an image?
- How to compute gradients robustly?

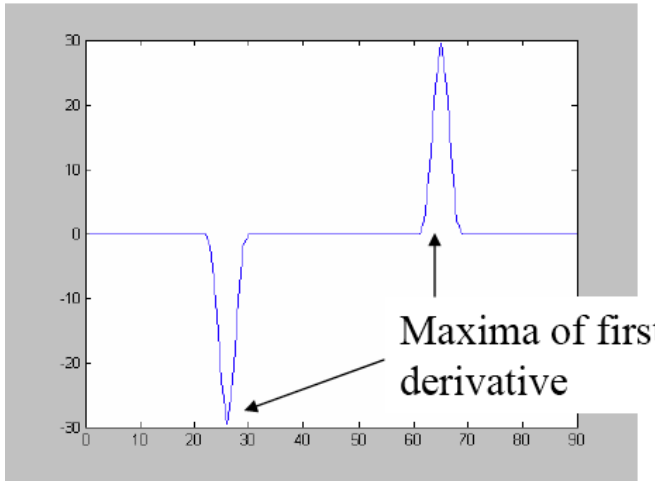


Edges and Derivatives…

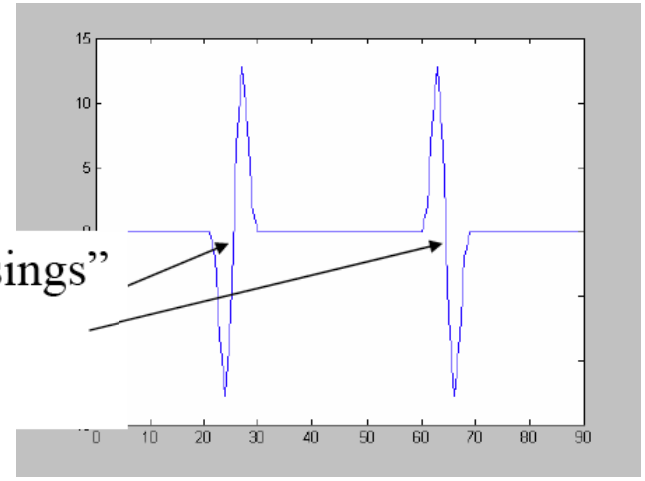




1st derivative



2nd derivative



Differentiation and Convolution

- For the 2D function $f(x,y)$, the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

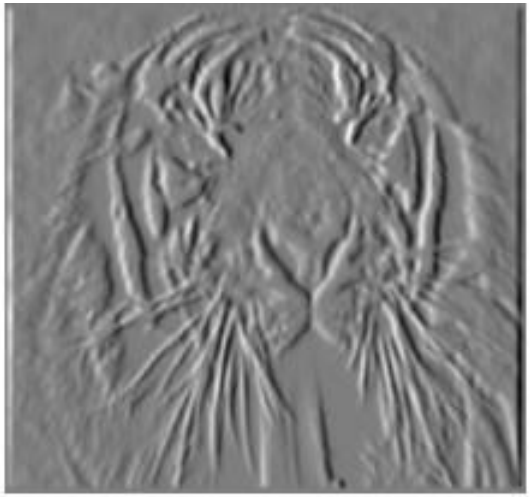
- For discrete data, we can approximate this using finite differences:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

- To implement the above as convolution, what would be the associated filter?

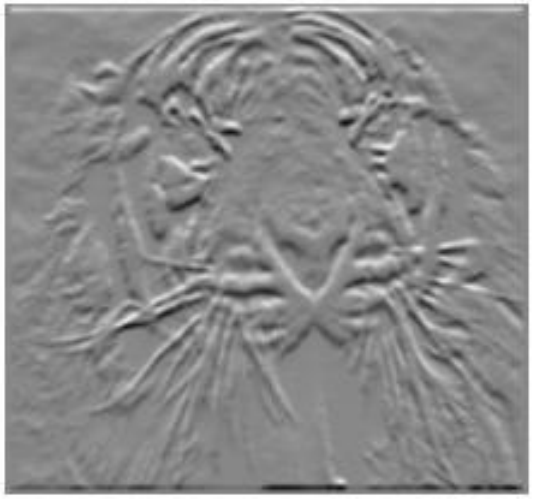
Partial Derivatives of an Image

$$\frac{\partial f(x,y)}{\partial x}$$



-1 1

$$\frac{\partial f(x,y)}{\partial y}$$



-1
1

?

or

1
-1

Which shows changes with respect to x?

Assorted Finite Difference Filters

Prewitt: $M_x =$

-1	0	1
-1	0	1
-1	0	1

; $M_y =$

1	1	1
0	0	0
-1	-1	-1

Sobel: $M_x =$

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

; $M_y =$

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

Roberts: $M_x =$

0	1
-1	0

; $M_y =$

1	0
0	-1

```
>> My = fspecial('sobel');
>> outim = imfilter(double(im), My);
>> imagesc(outim);
>> colormap gray;
```

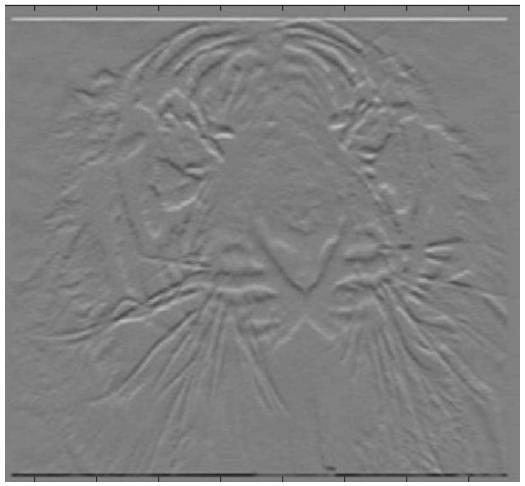
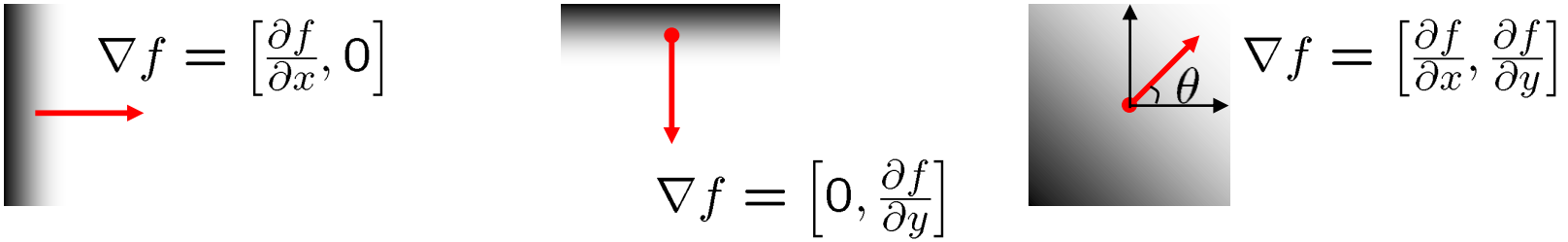


Image Gradient

- The gradient of an image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- The gradient points in the direction of most rapid intensity change



- The gradient direction (orientation of edge normal) is given by:

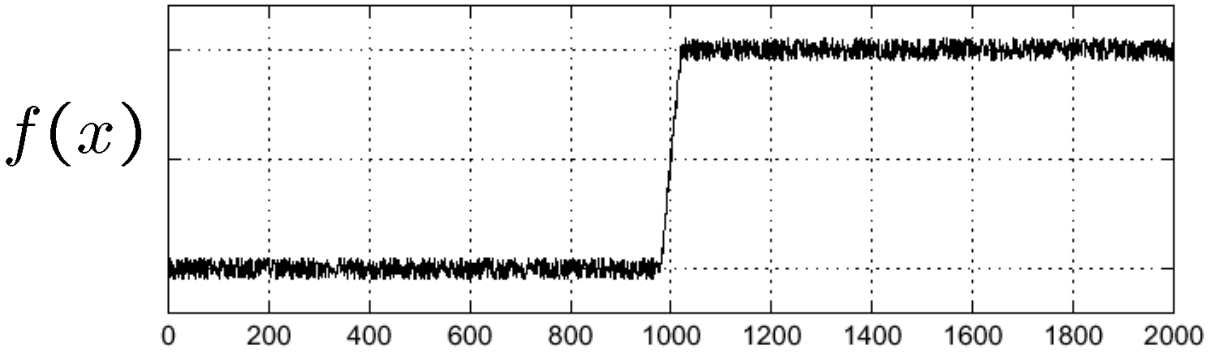
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

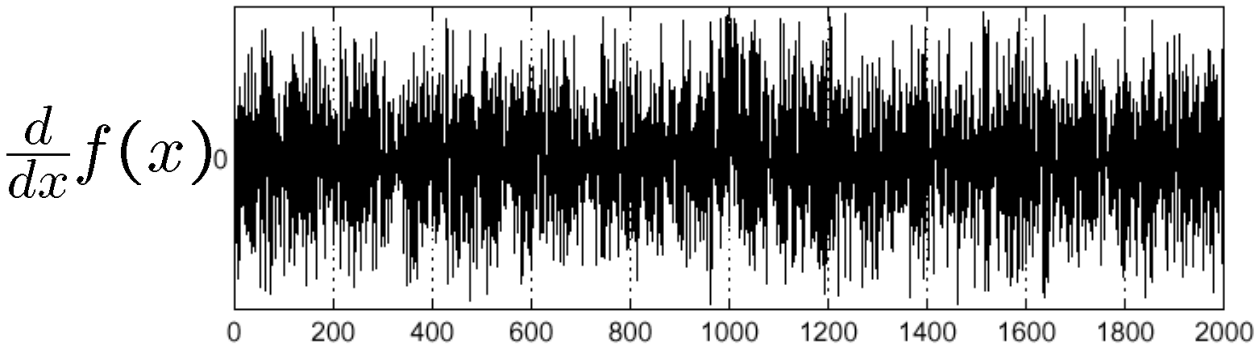
- The edge strength is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Effect of Noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal





Where is the edge?

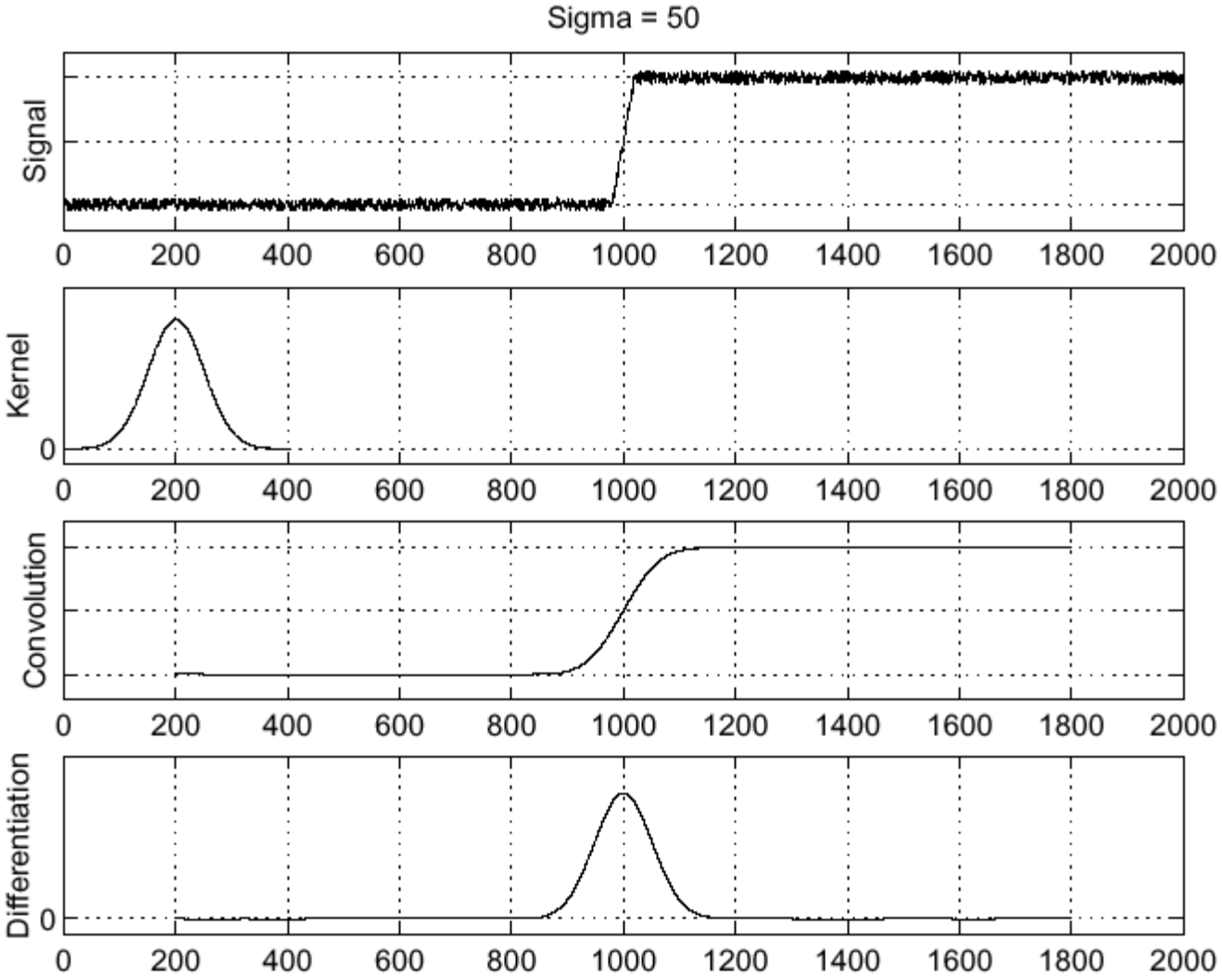
Solution: Smooth First

f

h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$



Where is the edge? **Look for peaks in** $\frac{\partial}{\partial x}(h \star f)$

Derivative Theorem of Convolution

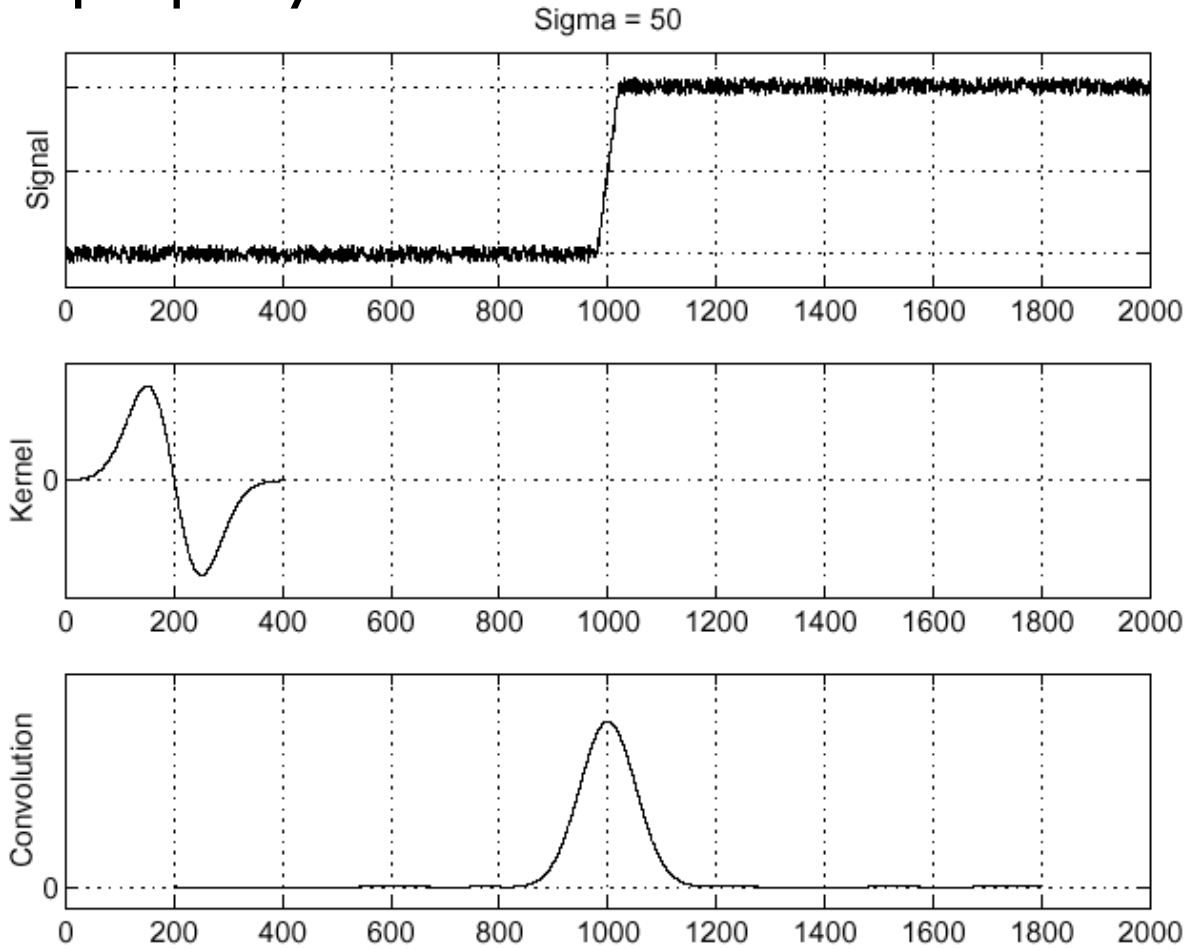
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

- Differentiation property of convolution.

f

$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$

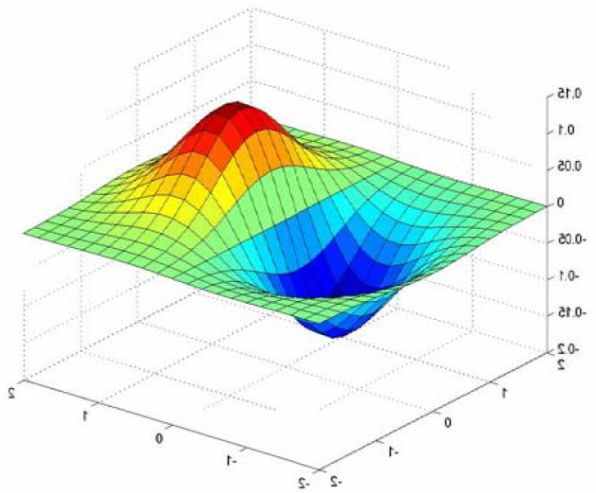


Derivative of Gaussian Filter

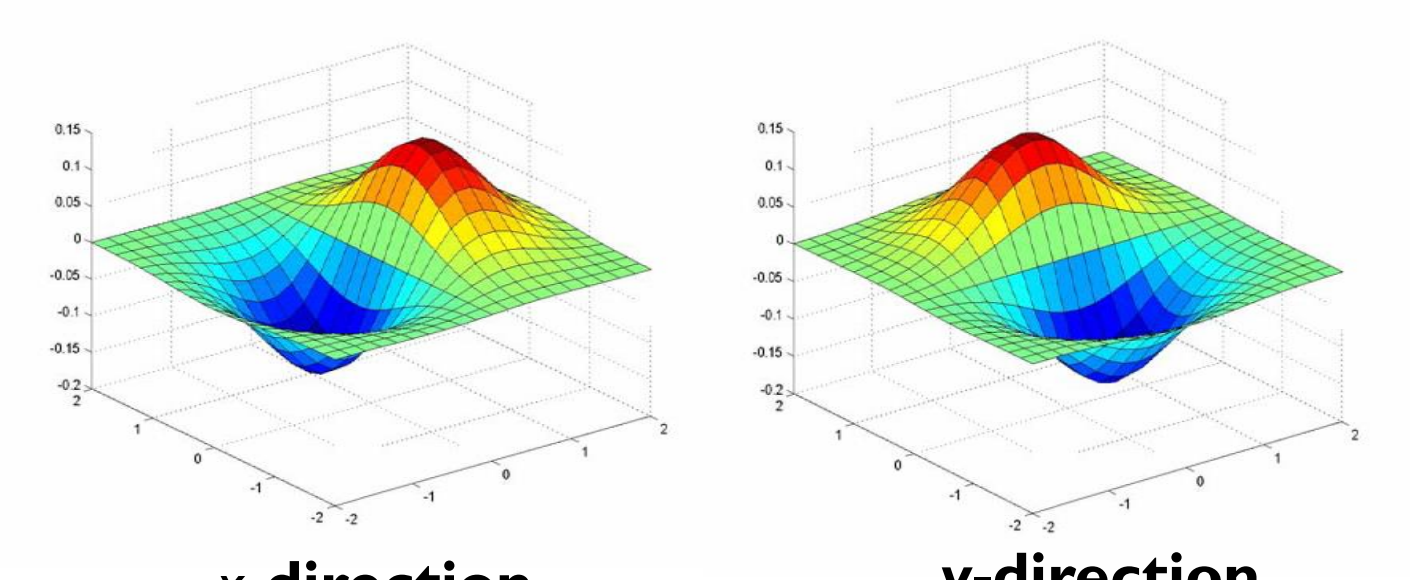
$$(I * g) * h = I * (g * h)$$

$$\begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix} * \begin{bmatrix} 1 & -1 \end{bmatrix}$$

Why is this preferable?

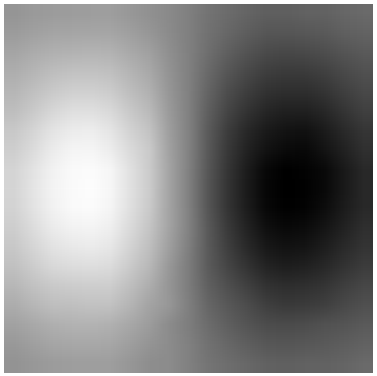


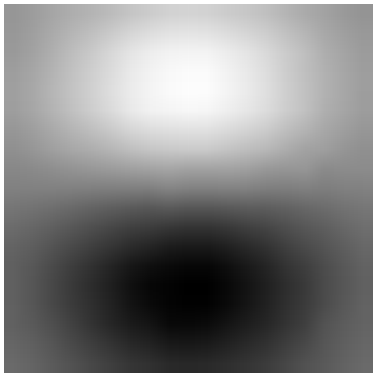
Derivative of Gaussian Filters



x-direction

y-direction



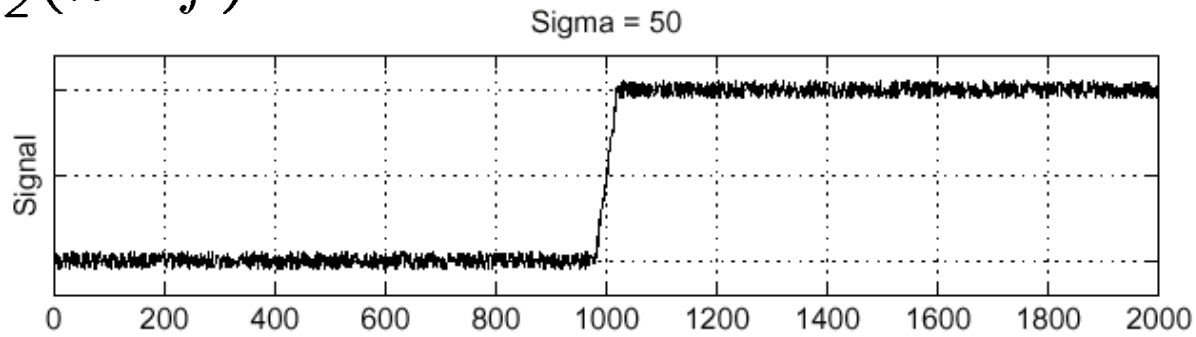


Source: Svetlana Lazebnik

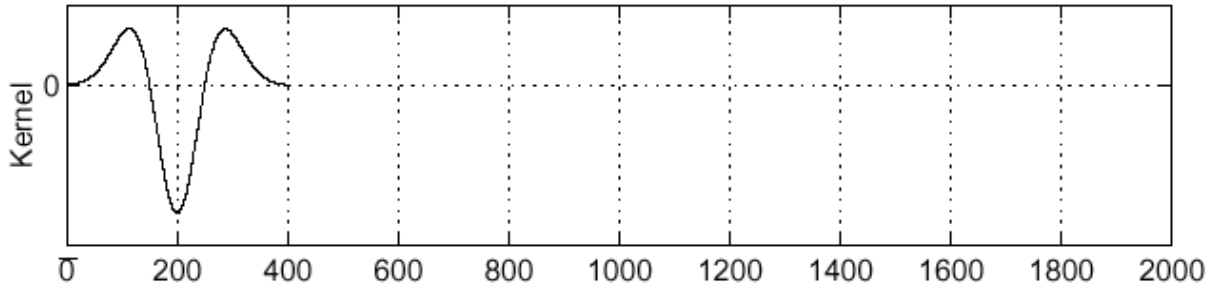
Laplacian of Gaussian (LoG)

- Consider $\frac{\partial^2}{\partial x^2}(h \star f)$

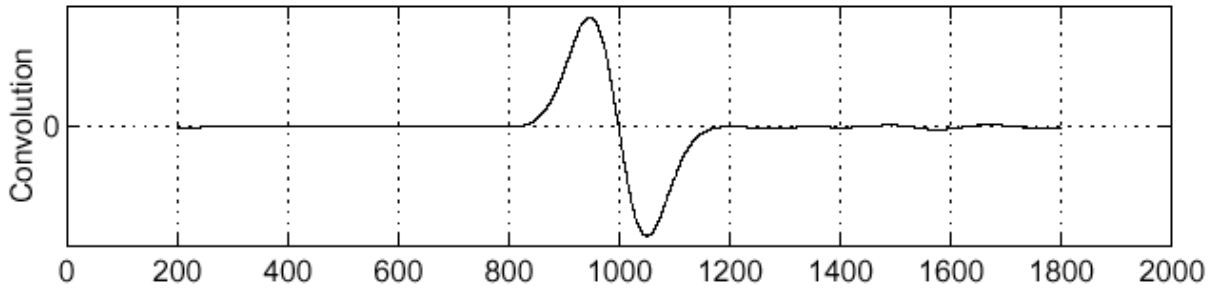
f



$\frac{\partial^2}{\partial x^2}h$

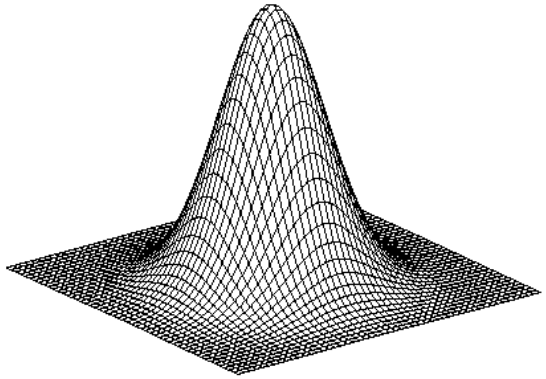


$(\frac{\partial^2}{\partial x^2}h) \star f$



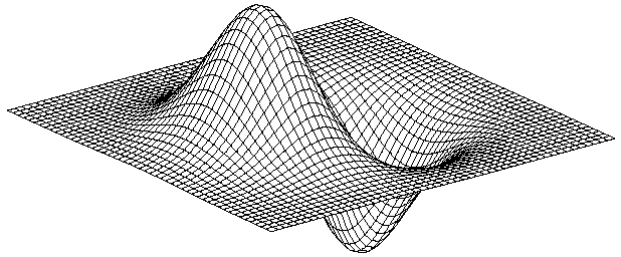
Where is the edge? **Zero-crossings of bottom graph**

Summary: 2D Edge Detection Filters



Gaussian

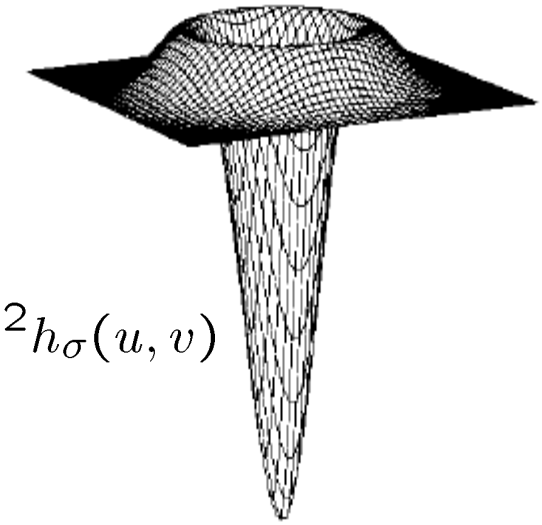
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



Derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian



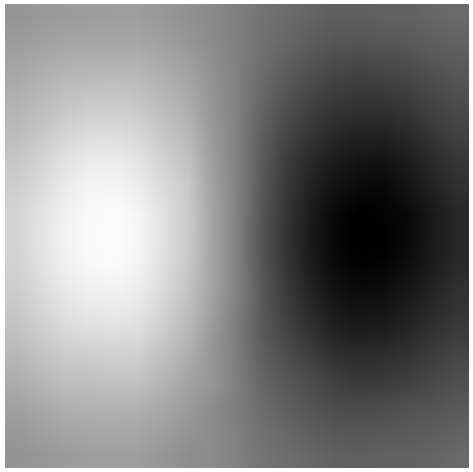
$$\nabla^2 h_\sigma(u,v)$$

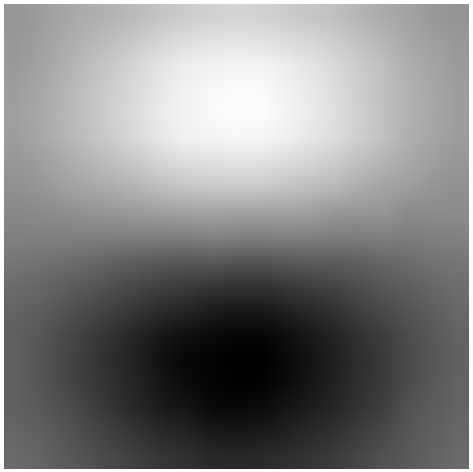
- ∇^2 is the Laplacian operator: $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

Note: Filters are Templates

- Applying a filter at some point can be seen as taking a dot-product between the image and some vector.
- Filtering the image is a set of dot products.

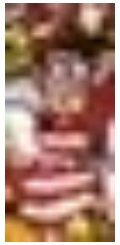
- Insight
  - Filters look like the effects they are intended to find.
  - Filters find effects they look like.





Where's Waldo?

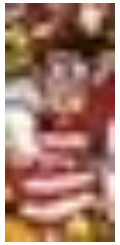
Scene



Template

Where's Waldo?

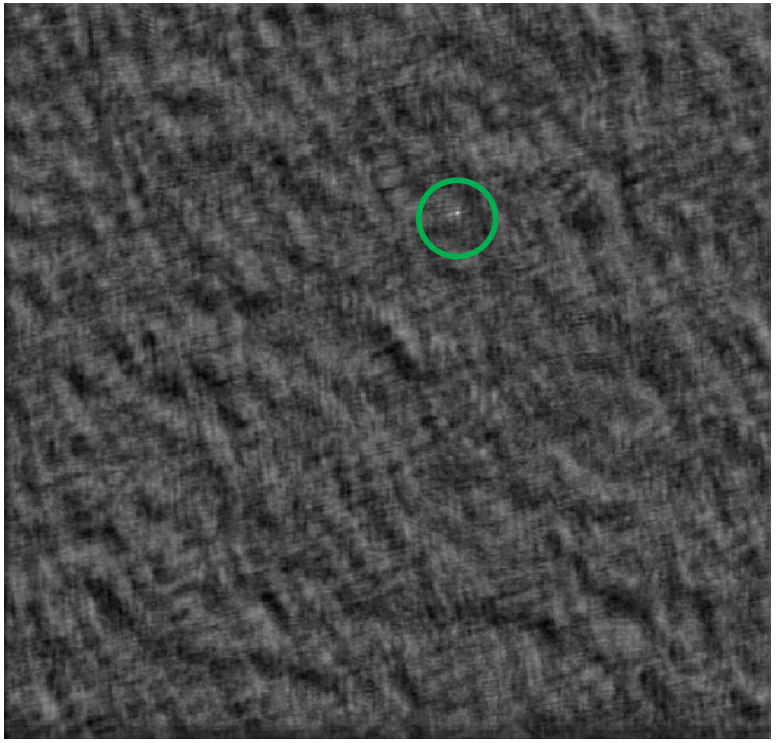
Detected template



Template

Where's Waldo?

Detected template



Correlation map

Correlation as Template Matching

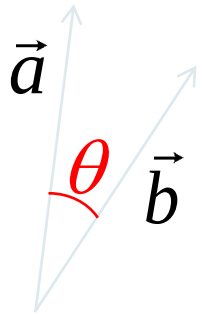
- Think of filters as a dot product of the filter vector with the image region
  - Now measure the angle between the vectors

$$a \cdot b = |a||b|\cos\theta \qquad \cos\theta = \frac{a \cdot b}{|a||b|}$$

  - Angle (similarity) between vectors can be measured by normalizing length of each vector to 1.

Template

Image region



Vector interpretation

Summary: Mask Properties

- Smoothing
  - Values positive
  - Sum to 1 $\Rightarrow$ constant regions same as input
  - Amount of smoothing proportional to mask size
  - Remove "high-frequency" components; "low-pass" filter

- Derivatives
  - Opposite signs used to get high response in regions of high contrast
  - Sum to 0 $\Rightarrow$ no response in constant regions
  - High absolute value at points of high contrast

- Filters act as templates
  - Highest response for regions that "look the most like the filter"
  - Dot product as correlation

Summary Linear Filters

Linear filtering:

- **Form a new image whose pixels are a weighted sum of original pixel values**

Properties

- **Output is a shift-invariant function of the input (same at each image location)**

Examples:

- **Smoothing with a box filter**
- **Smoothing with a Gaussian**
- **Finding a derivative**
- **Searching for a template**

Pyramid representations

- **Important for describing and searching an image at all scales**