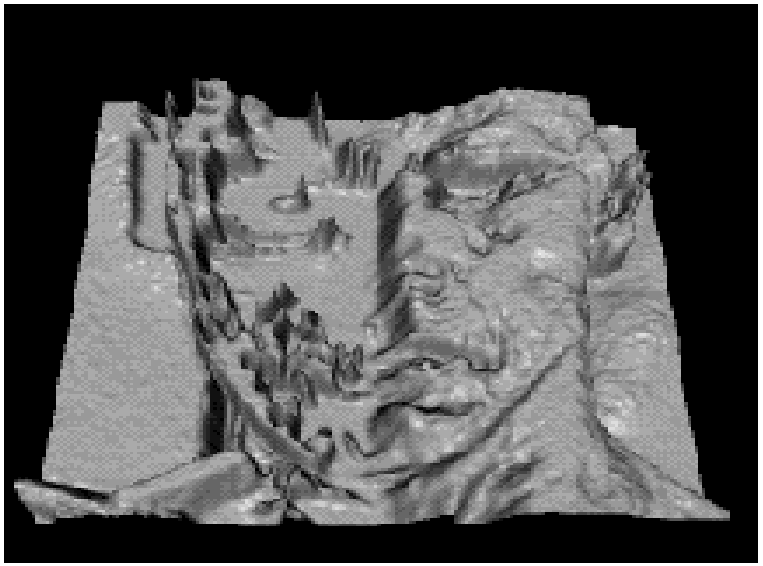
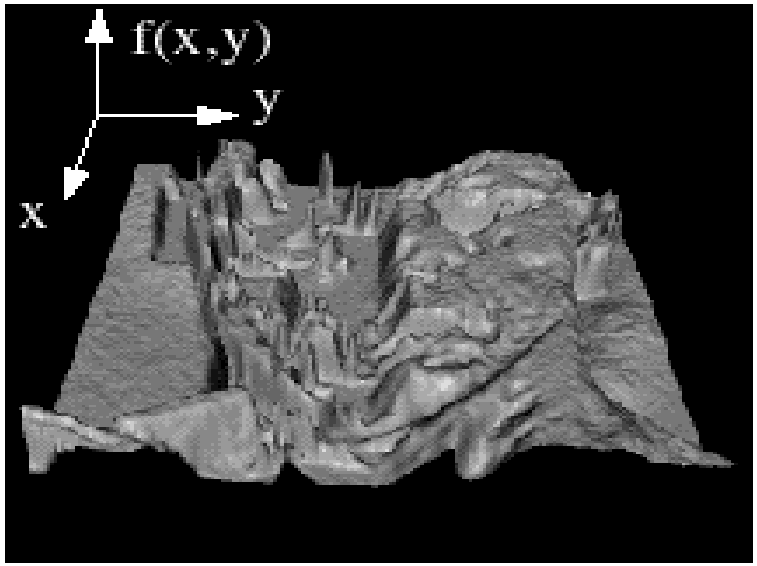
Image Processing and Filters

The content of most slides are from Dimitris Samaras and Minh Haoi (Stony Brook University)

Images as functions





What is a digital image?

- Digital images:
  - Sample the 2D space on a regular grid
  - Quantize each sample
- For samples being Δ apart:

  $f[i, j] = \text{Quantize}\{ f(i\,\Delta, j\,\Delta) \}$
- Image: matrix of integer values

$j \longrightarrow$

$i \downarrow$

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

- Color images: one 3-dimensional vector for each i,j

Image Noise

- Images are corrupted by “noise” mostly during acquisition
- Signal Processing techniques can be used to model and remove this noise
- Image Restoration: removal of noise
  - Additive Noise
  - Salt & Pepper Noise

Additive Noise

- In the additive noise model the signal is corrupted with random fluctuations $\hat{I}(i,j) = I(i,j) + n(i,j)$

Image source: http://www.iac.cnr.it/~vitulano/suvehpweb/SUVEHP.htm

Salt and Pepper Noise

- Salt and pepper noise can model errors introduced by the acquisition process.

$$\hat{I}(i,j) = \begin{cases} I(i,j) & \text{with probability } r \\ 0 \text{ or } 255 & \text{with probability } 1-r \end{cases}$$

Linear Filters

- General process:
  - Form new image whose pixels are a weighted sum of original pixel values, using the same set of weights at each point.
- Examples:
  - Average of pixels in a neighborhood
  - Gaussian smoothing: weighted averaging
  - Derivative

Properties of Linear Filters

- Linear:
  - Output is a linear function of the input
- Shift-invariant:
  - Output is a shift-invariant function of the input (i.e. shift the input image two pixels to the left, the output is shifted two pixels to the left)

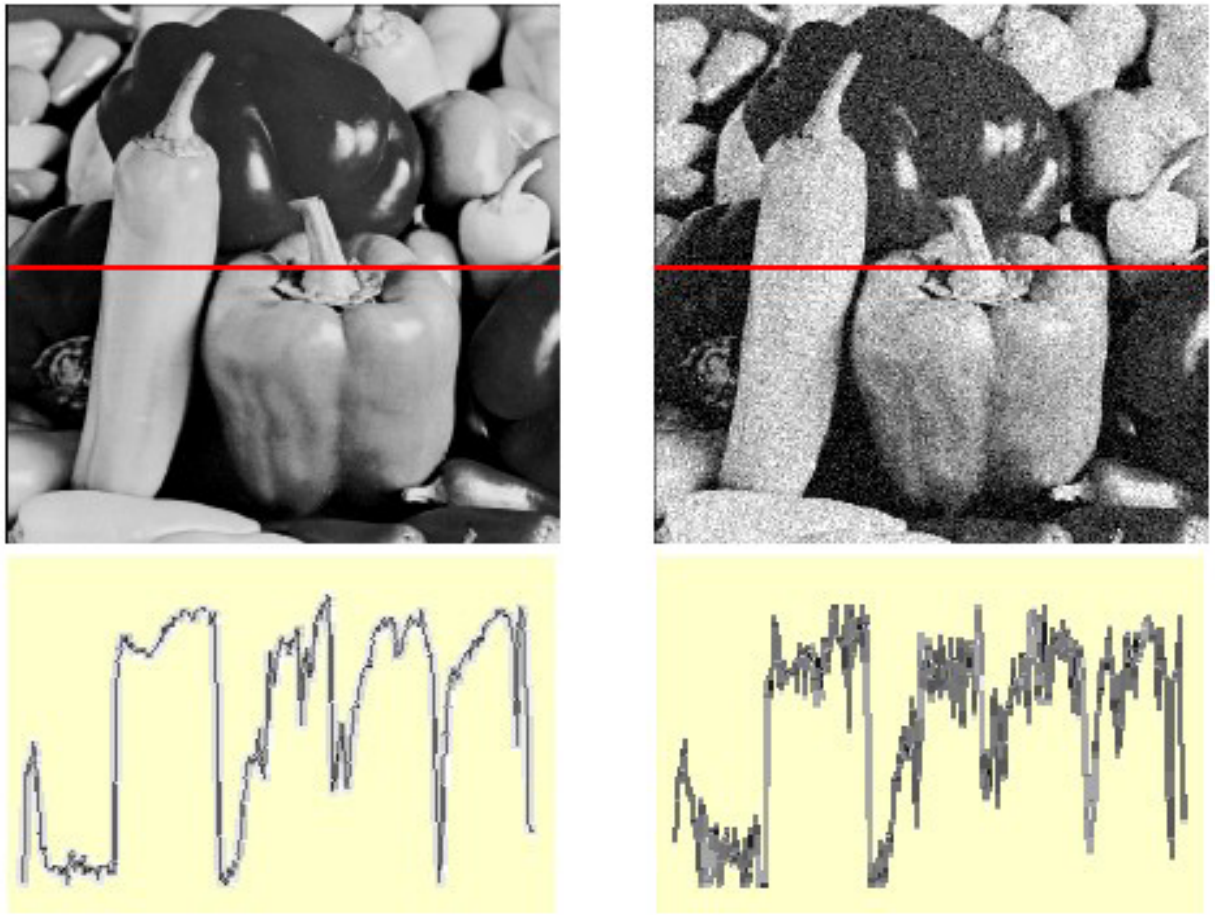
Image Filtering

- Noise reduction

Image Filtering

- Structure Extraction

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:
$\eta(x,y) \sim \mathcal{N}(\mu, \sigma)$

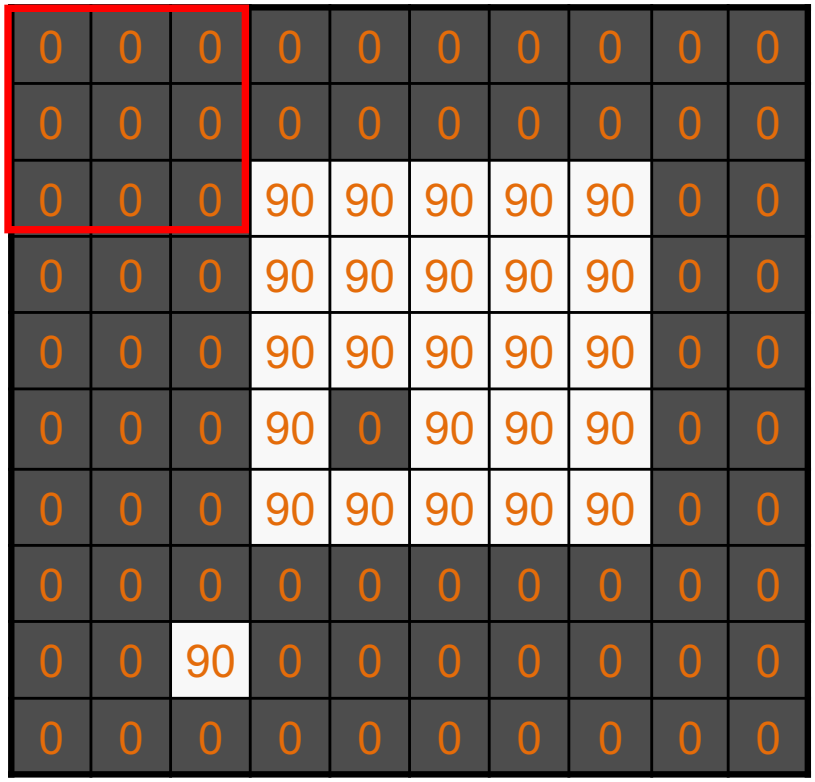
Removing noise

- Basic assumption
  - Noise process is independent, identically distributed
  - Image has a more regular underlying structure

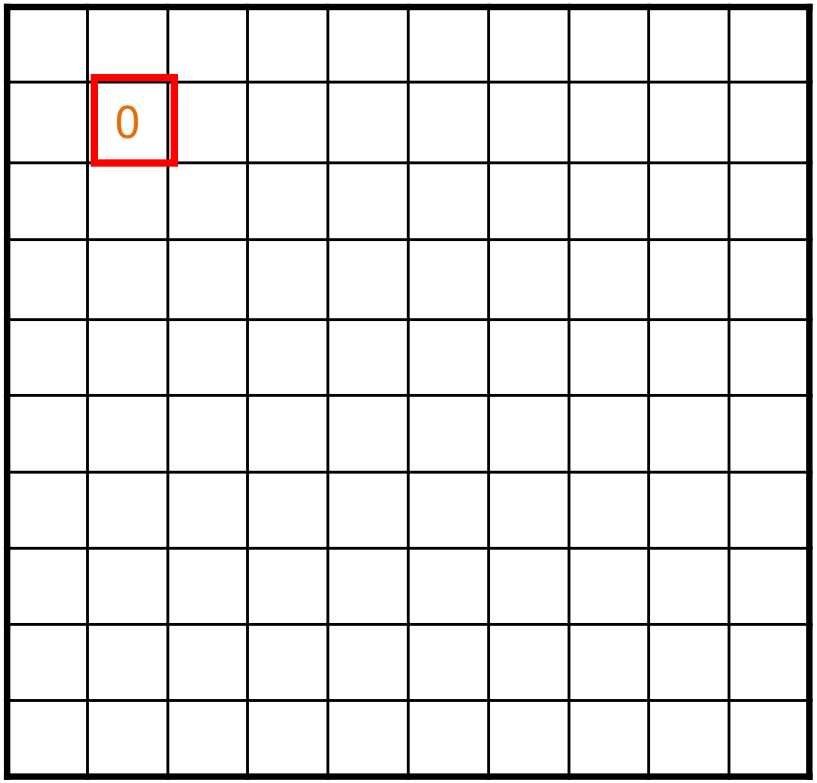
- By considering larger neighborhoods we can separate the signal from the noise

Moving Average in 2D

$F[x, y]$



$G[x, y]$



Moving Average in 2D

$F[x, y]$

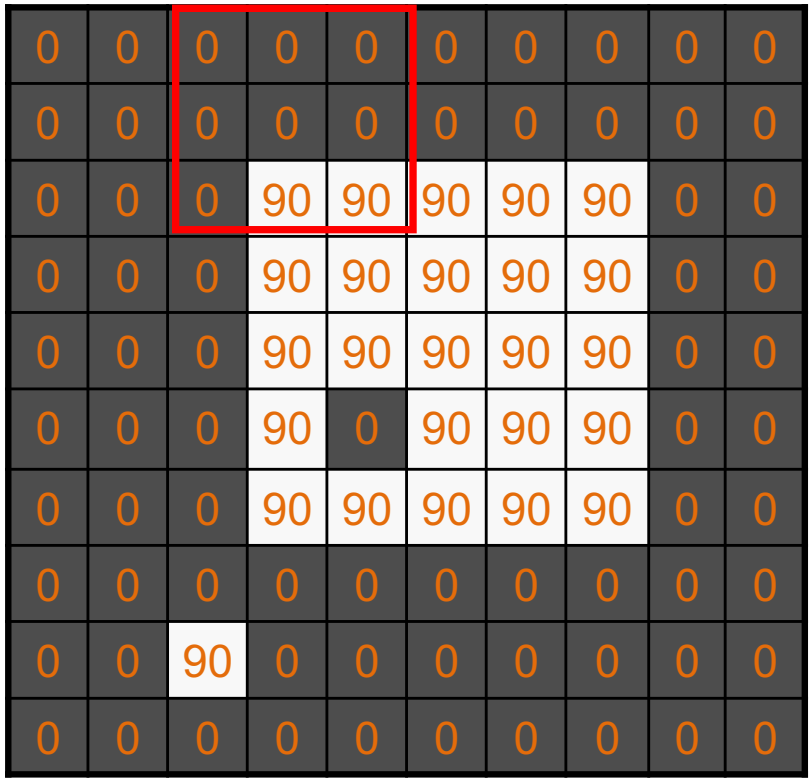
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

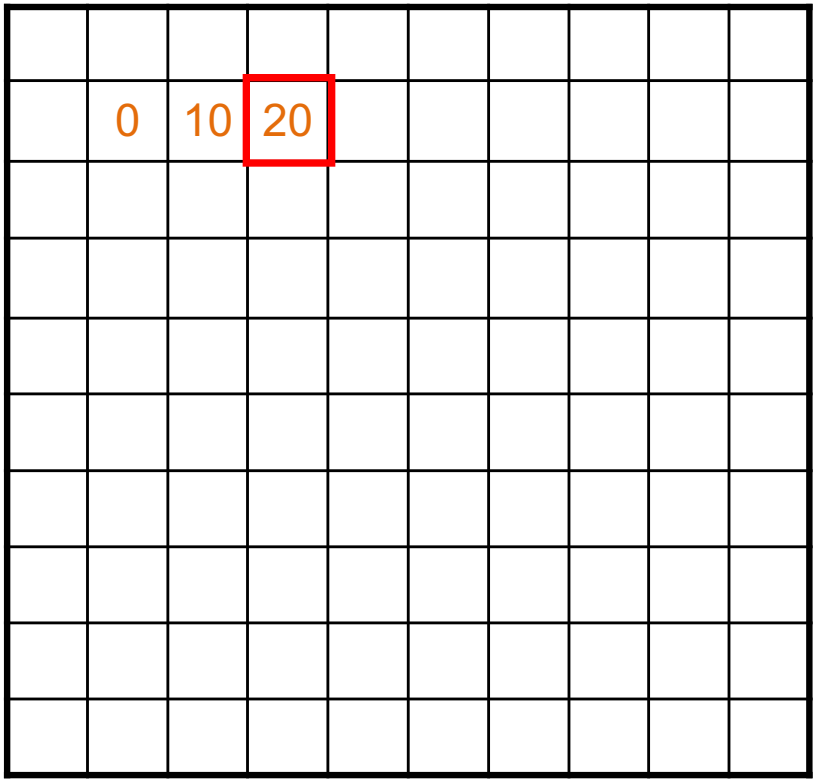
	0	10							

Moving Average in 2D

$F[x, y]$



$G[x, y]$



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:

$$G[i,j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]$$

- Different weights depending on neighboring pixel's relative position:

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

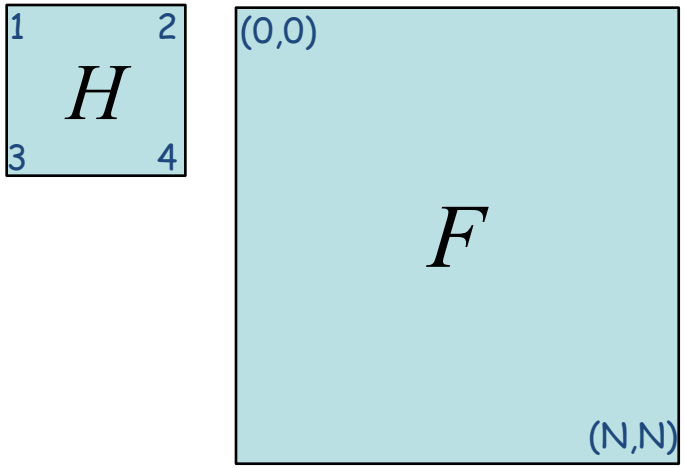
- Correlation filtering:

$$G = H \otimes F$$

Correlation Filtering

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

- 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

- Represent these weights as an image H
- H is usually called the **kernel**
- Operation is called **convolution**
  - it's associative

- Result is:

$$G_{ij} = \sum_{u,v} H_{uv} F_{i-u,i-v}$$

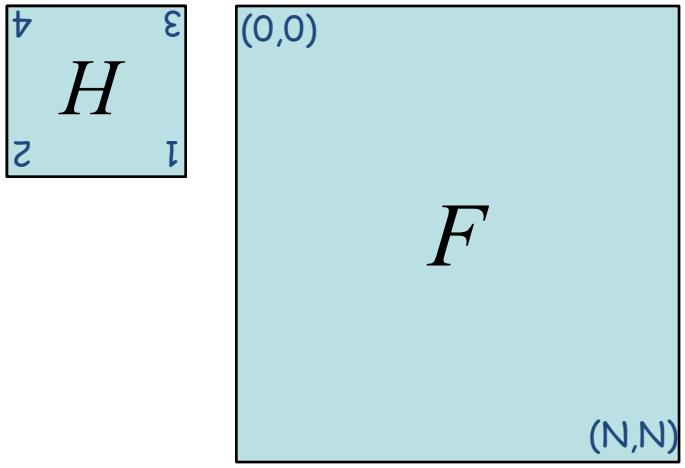
- Notice the order of indices
  - it's a result of the derivation expressing any shift-invariant linear operator as a convolution.

Convolution

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

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

$$G = H * F$$



- Convolution of continuous signals

$$g(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(u,v) f(x-u, y-v) du dv$$

Some facts about Convolution

- A linear operation
  - $X*(aY + bZ) = a(X*Y) + b(X*Z)$
- An associative operation
  - $X*(Y*Z) = (X*Y)*Z$
- Convolution in the spatial or time domain corresponds to multiplication in the frequency domain

Computation

- Convolution is the most important method to analyze digital signals
- Convolution is a fairly expensive operation requiring a large number of computations on typical images.
- Many computer architectures provide specialized instructions for these kinds of operations.

Common Kernels

- Two convolution kernels that are commonly used for noise reduction are
  - The mean kernel
  - The Gaussian Kernel

Mean Filtering

- Smoothing by averaging

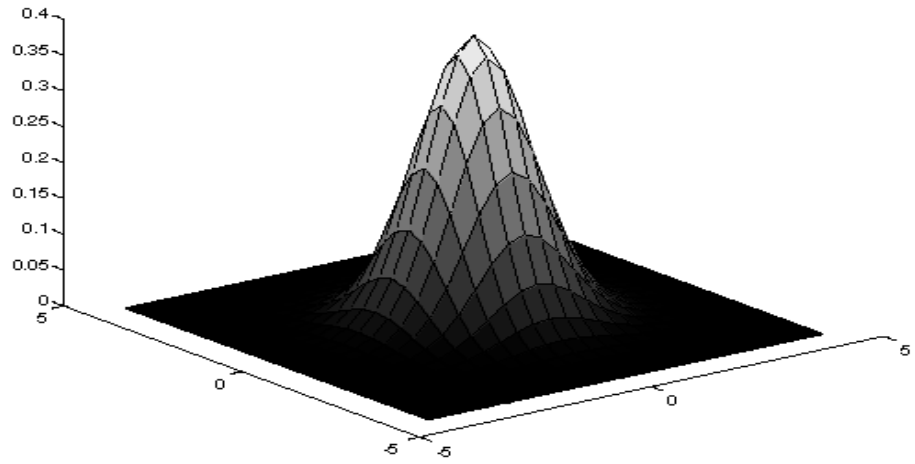
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

- Entries must add up to one. Why?

Example: Smoothing by Averaging

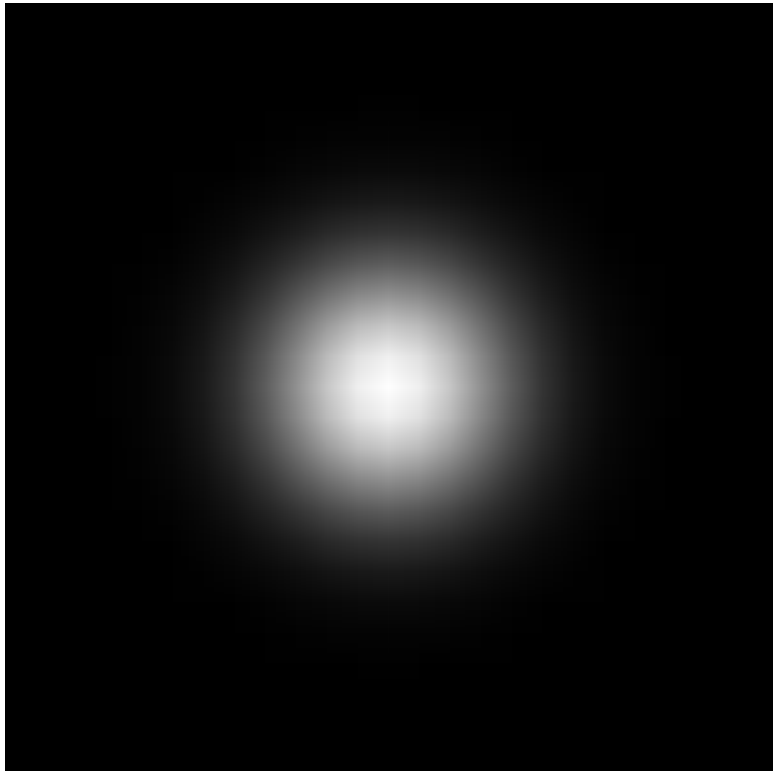
Smoothing with a Gaussian

- Smoothing with an average actually doesn't compare at all well with a defocussed lens
  - Most obvious difference is that a single point of light viewed in a defocussed lens looks like a fuzzy blob; but the averaging process would give a little square.



- A Gaussian gives a good model of a fuzzy blob

An Isotropic Gaussian

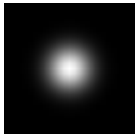


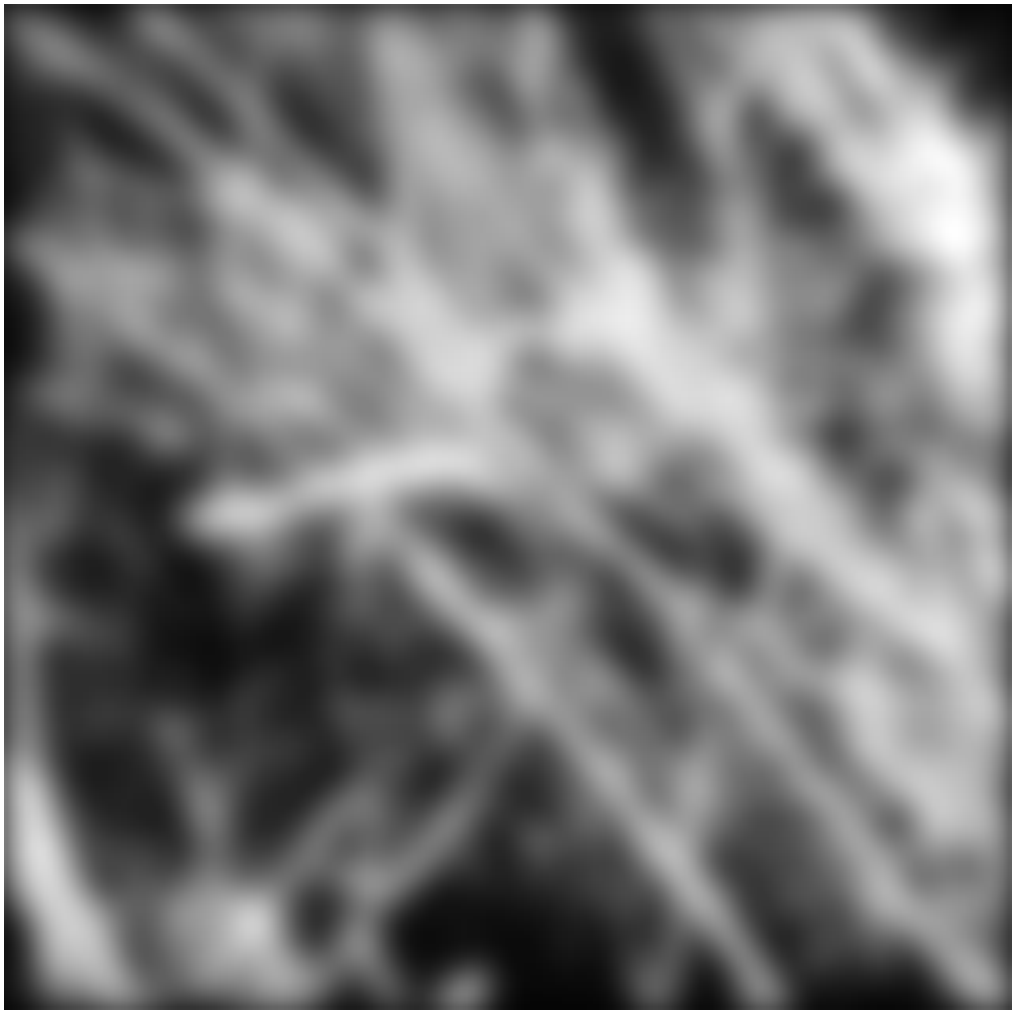
- The picture shows a smoothing kernel proportional to

$$\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

(which is a reasonable model of a circularly symmetric fuzzy blob)

Smoothing with a Gaussian





Differentiation and convolution

- Recall

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

- Now this is linear and shift invariant, so must be the result of a convolution.

- We could approximate this as

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

- It is obviously a convolution; it's not a very good way to do things, as we shall see

A Noise Model

- Simplest noise model
  - independent stationary additive Gaussian noise
  - the noise value at each pixel is given by an independent draw from the same normal probability distribution
- Issues
  - this model allows noise values that could be greater than maximum camera output or less than zero
  - for small standard deviations, this isn't too much of a problem - it's a fairly good model
  - independence may not be justified (e.g. damage to lens)
  - may not be stationary (e.g. thermal gradients in the CCD)

sigma=1

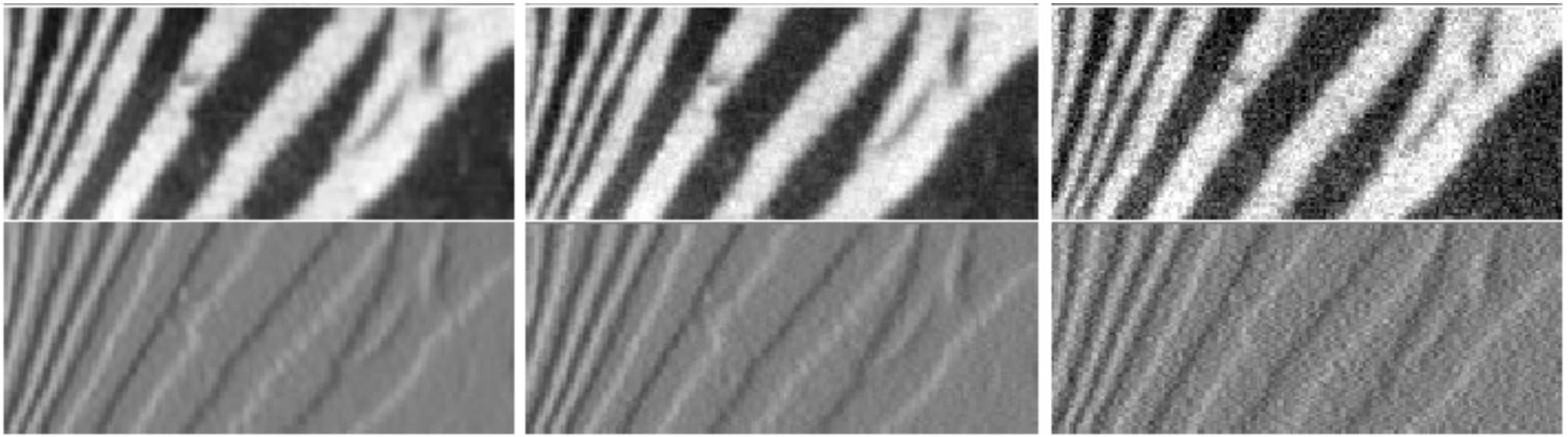
sigma=16

Finite differences and noise

- Finite difference filters respond strongly to noise
  - obvious reason: image noise results in pixels that look very different from their neighbors
- Generally, the larger the noise the stronger the response

- What is to be done?
  - intuitively, most pixels in images look quite a lot like their neighbors
  - this is true even at an edge; along the edge they're similar, across the edge they're not
  - suggests that smoothing the image should help, by forcing pixels different to their neighbors (=noise pixels?) to look more like neighbors

Finite differences responding to noise



Increasing noise ---------->

(This is zero mean additive Gaussian noise)

The response of a linear filter to noise

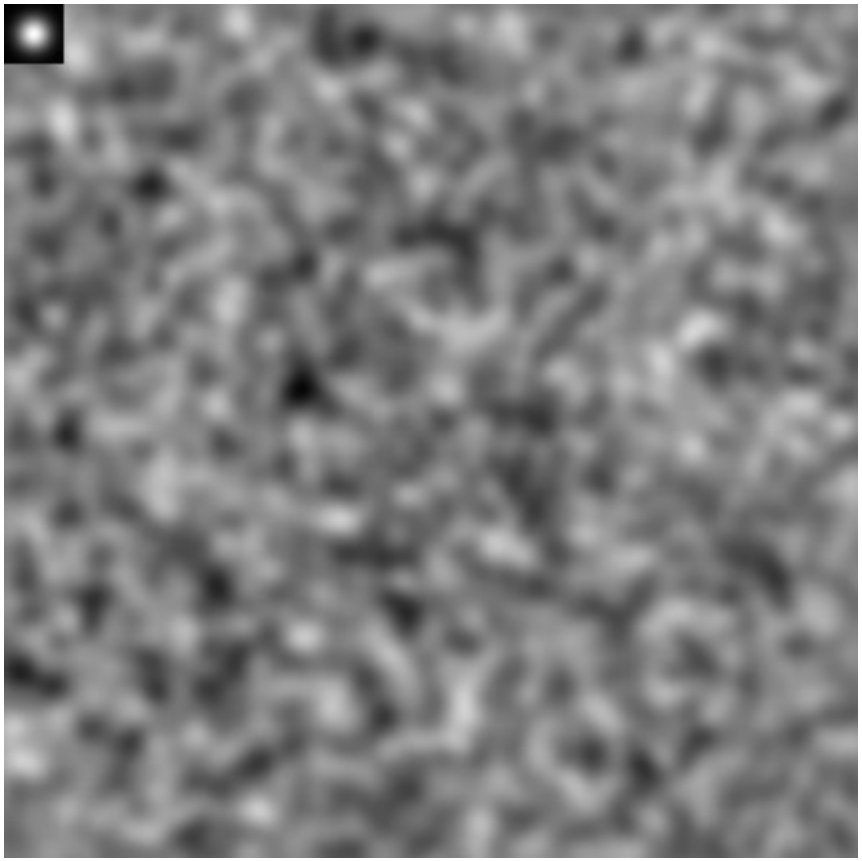
- Consider stationary independent additive Gaussian noise with zero mean (non-zero mean can be easily dealt with)
- What would be the mean and variance of the noise after filtering?
- **Mean**:
  - Output is a weighted sum of inputs
  - It's a weighted sum of zero mean normal random variables
  - Must be zero
- **Variance**:
  - variance of a sum of random variables is sum of their variances
  - variance of constant times random variable is constant^2 times variance
  - then if s is noise variance and kernel is K, variance of response is

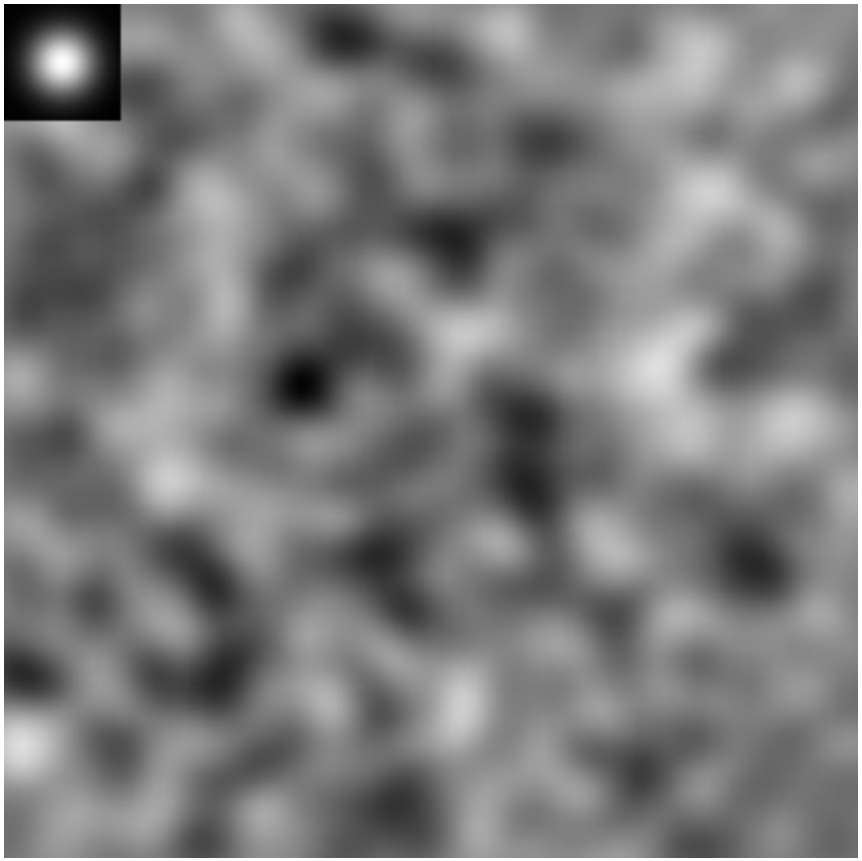
$$\underbrace{s^2 \sum_{u,v} K_{uv}^2}$$

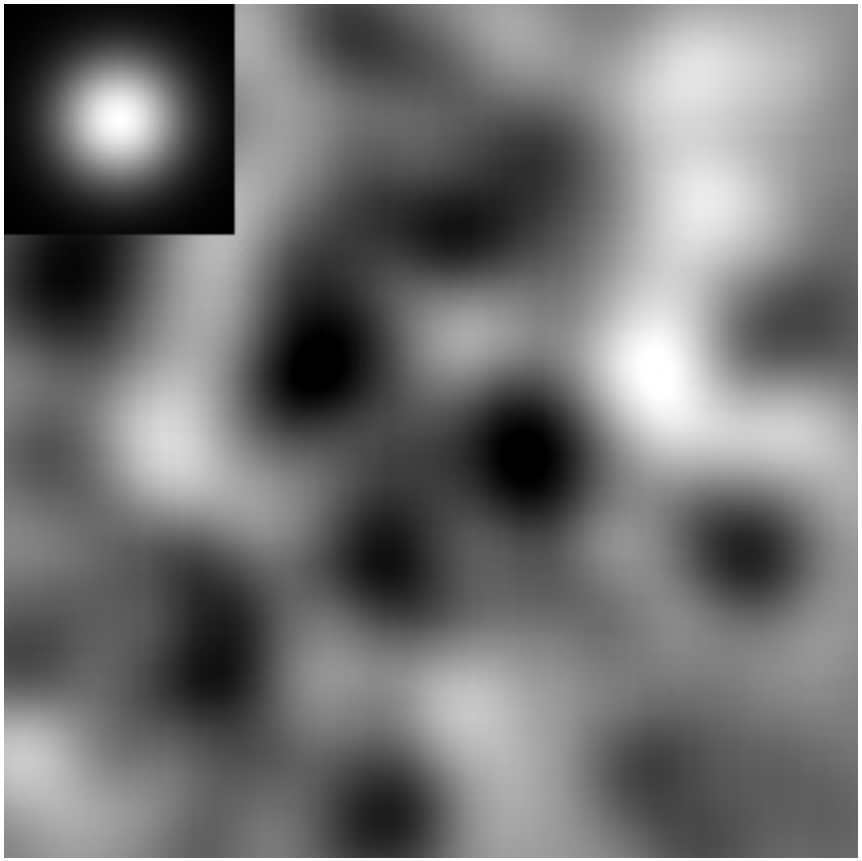
could be small

Filter responses are correlated

- Over scales similar to the scale of the filter
- Filtered noise is sometimes useful
  - looks like some natural textures, can be used to simulate fire, etc.



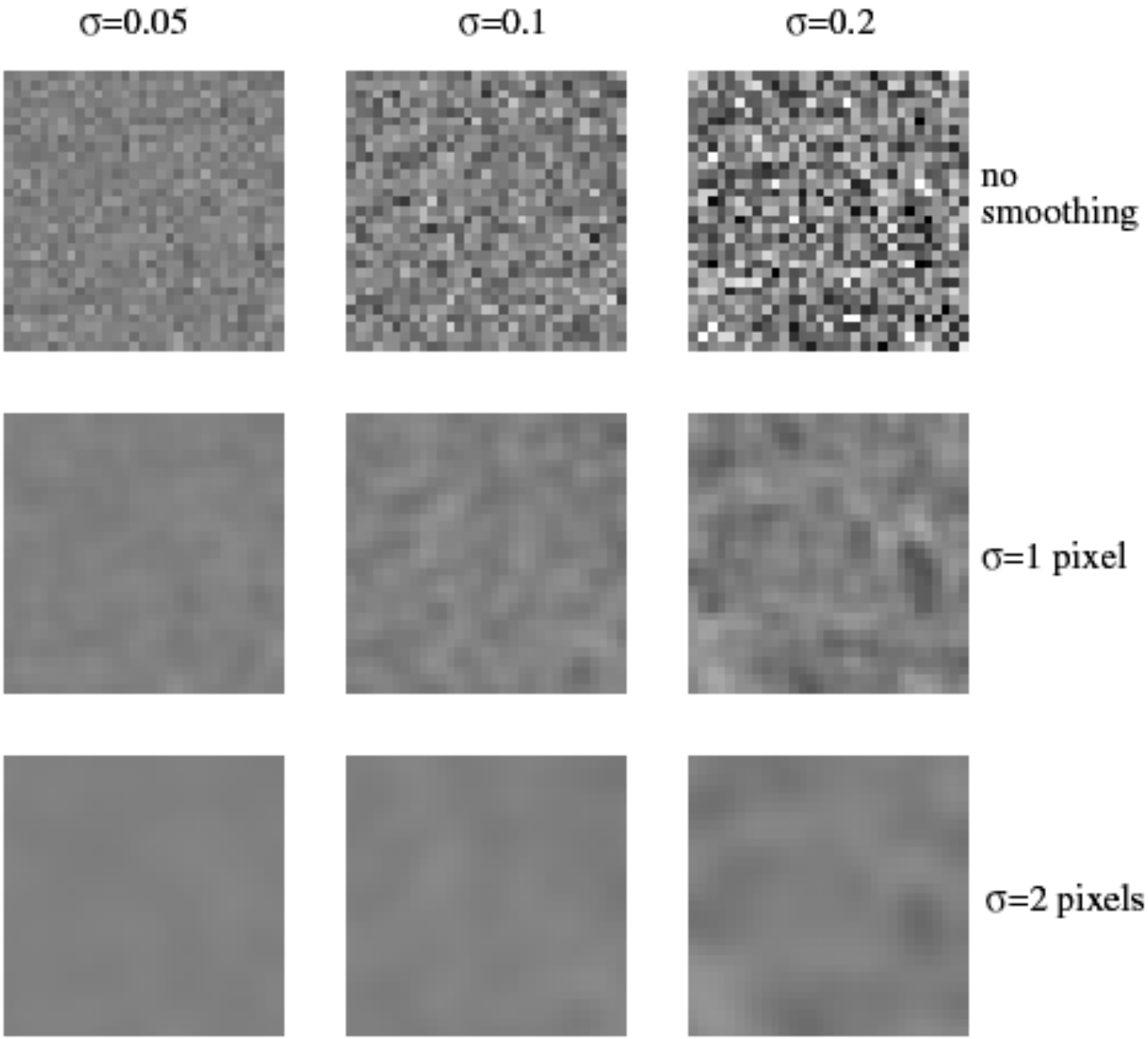




Smoothing reduces noise

- Generally expect pixels to "be like" their neighbors
  - surfaces turn slowly
  - relatively few reflectance changes
- Generally expect noise processes to be independent from pixel to pixel
- Implies that smoothing suppresses noise, for appropriate noise models
- Scale
  - the parameter in the symmetric Gaussian
  - as this parameter goes up, more pixels are involved in the average
  - and the image gets more blurred
  - and noise is more effectively suppressed (noise variance gets smaller)

Effects of Smoothing



- Each row shows smoothing with Gaussians of different width
- Each column shows different realizations of an image of Gaussian noise.

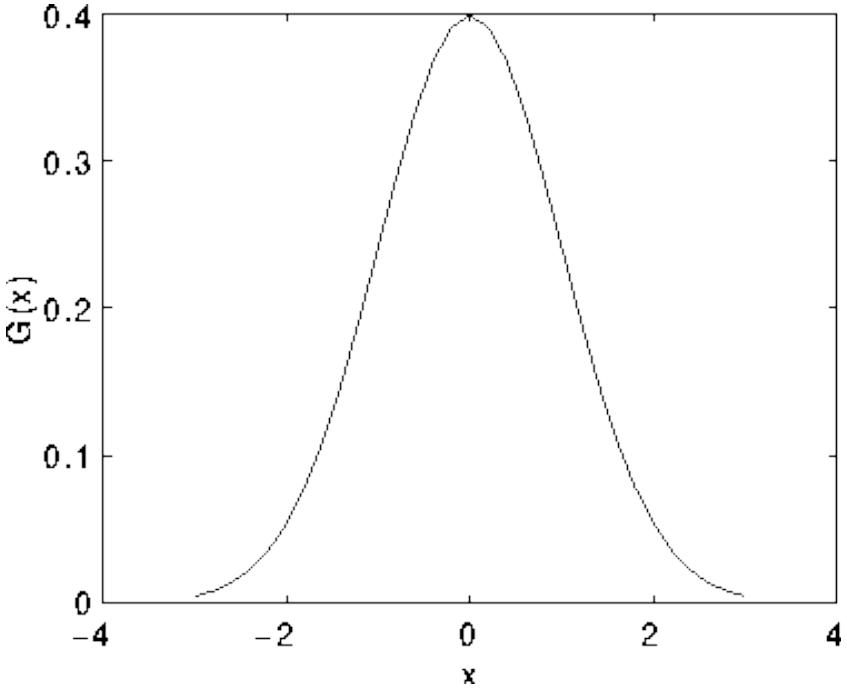
Separable filters

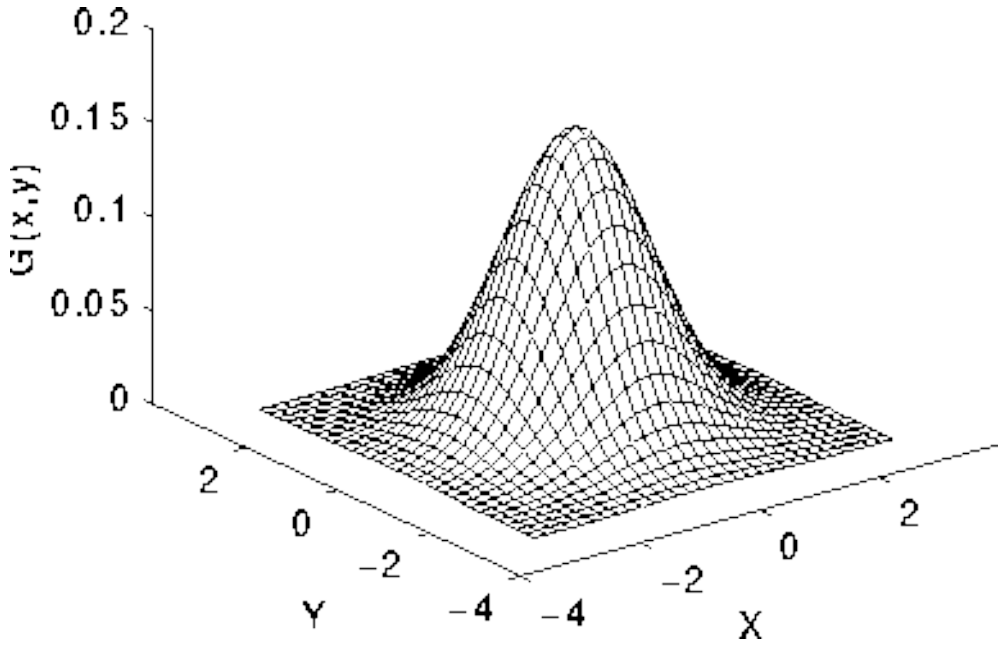
- Some 2D image filters can actually be implemented as two 1D filters - one in the horizontal direction followed by another in the vertical direction.
- This can significantly lower the computational complexity of the operation

Gaussian Filtering

- Separable filter

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \qquad G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$





Gaussian Filtering (II)

- $\sigma = 1$

$\frac{1}{10.7}$

1.3	3.2	3.8	3.2	1.3

$\frac{1}{115}$

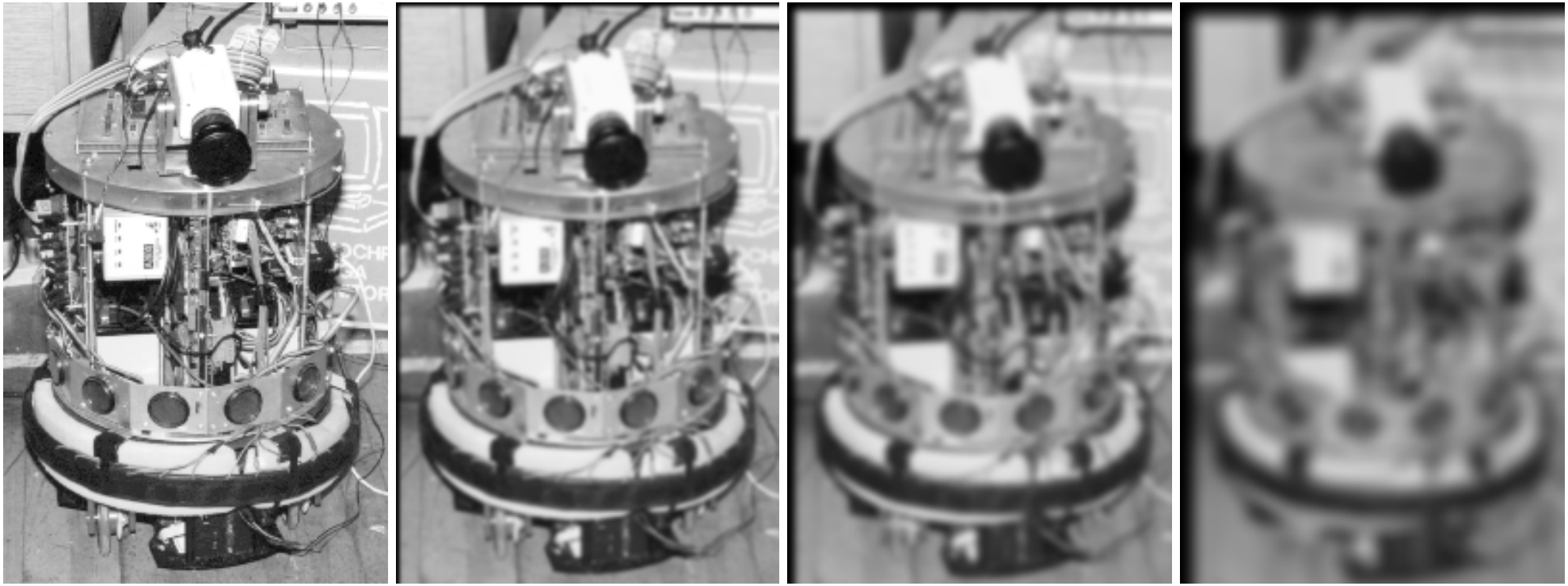
2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Gaussian Filtering (III)

- Sampling theorem
- To keep the energy in 98.76% area $\sigma > 0.8$
- Repeated averaging

Example

- $\sigma = 1.0$, $\sigma = 2.0$, $\sigma = 4.0$



Sharpening

Input = Coarse + Fine

Output = Coarse + Fine

Slide content: Andrew Adams (http://goo.gl/6GR56Q)

Sharpening

- Any Filter which removes fine details can be used to sharpen
  1. Coarse = Remove Fine Details from Input
  2. Fine = Input - Coarse
  3. Output = Input + Fine x 0.5

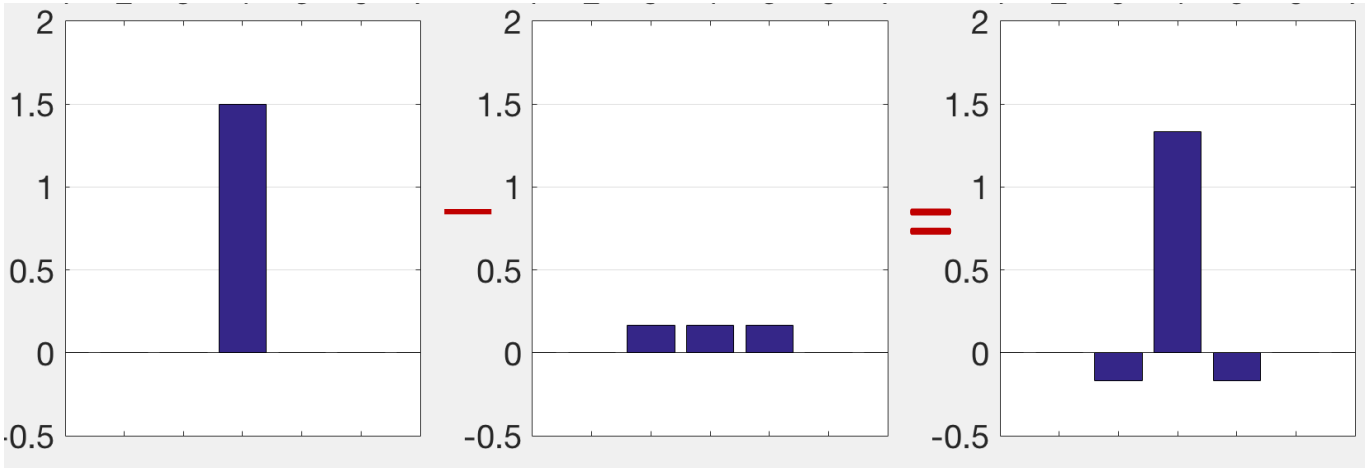
Slide content: Andrew Adams (http://goo.gl/6GR56Q)

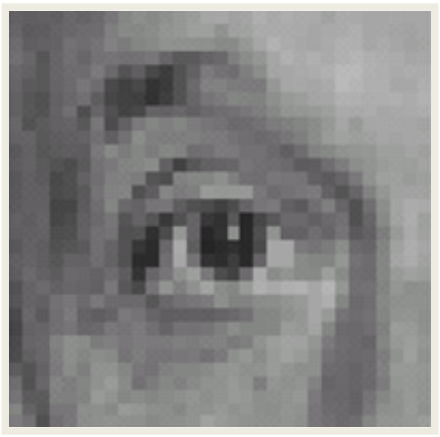
Linear Sharpening Filters

- Let G be a Gaussian kernel
  1. Coarse = G * Input
  2. Fine = Input - Coarse
  3. Output = Input + Fine x 0.5
- Recall convolution: G*(a+b) = G*a + G*b
- Output = (1.5I – 0.5G)*Input

Slide content: Andrew Adams (http://goo.gl/6GR56Q)

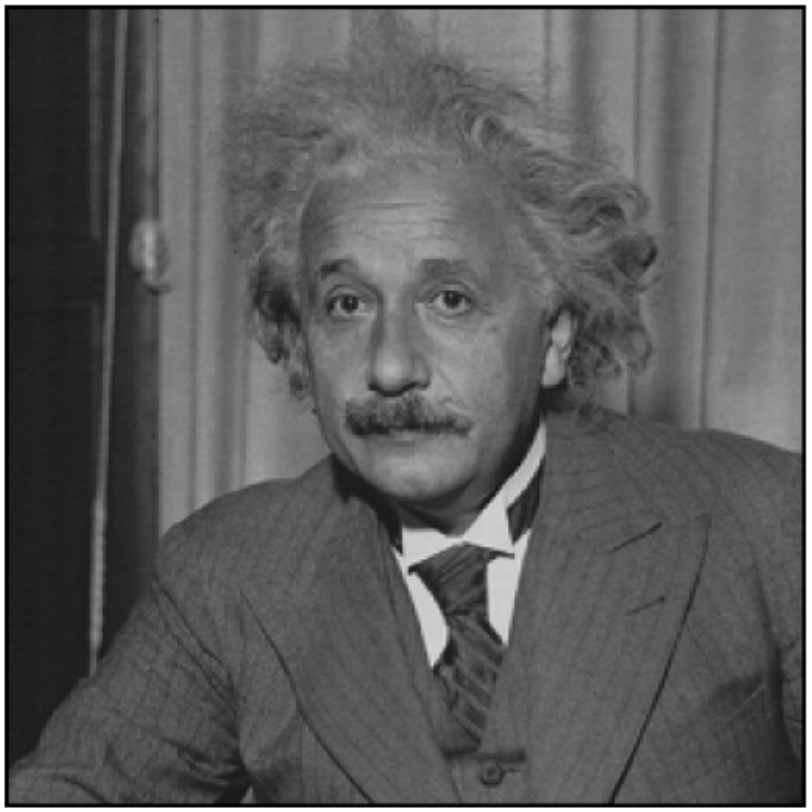
Sharpening Filter



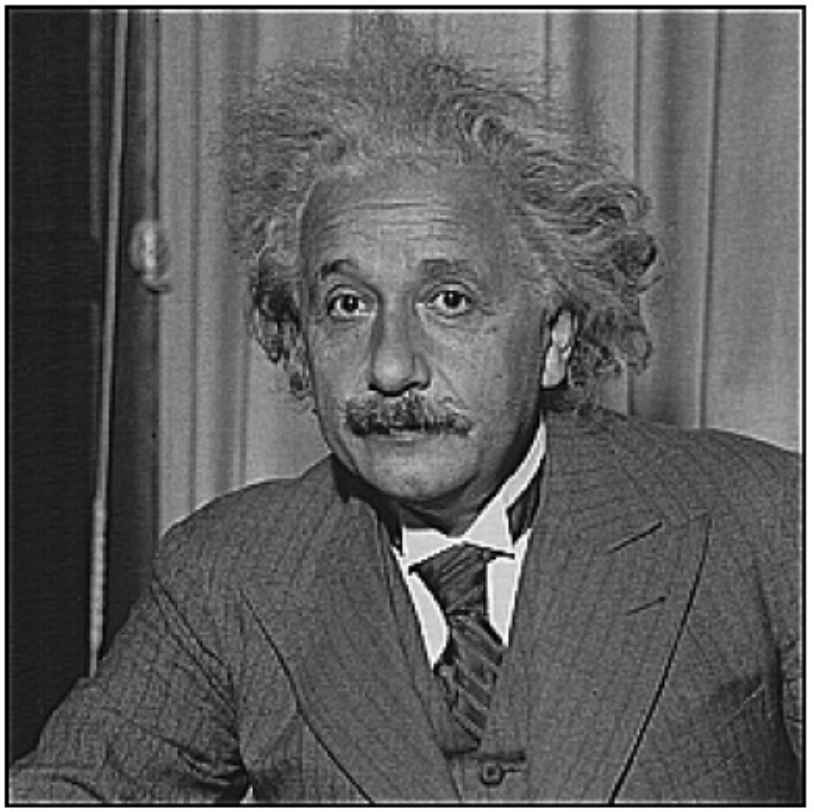


Original

Sharpening Filter

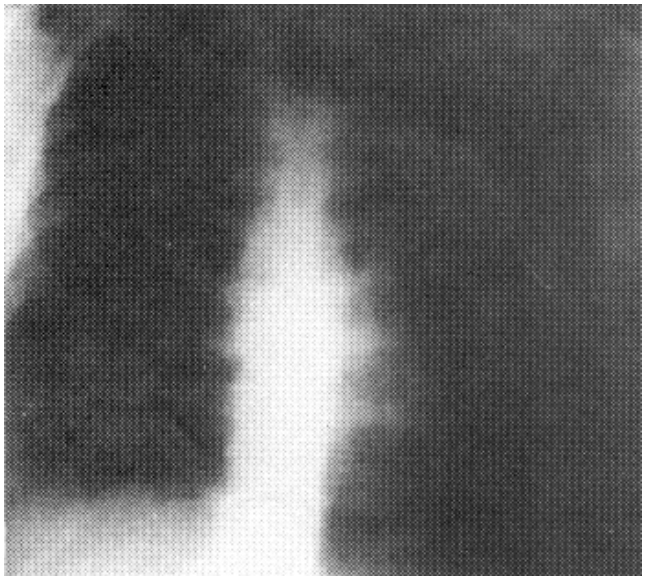


before

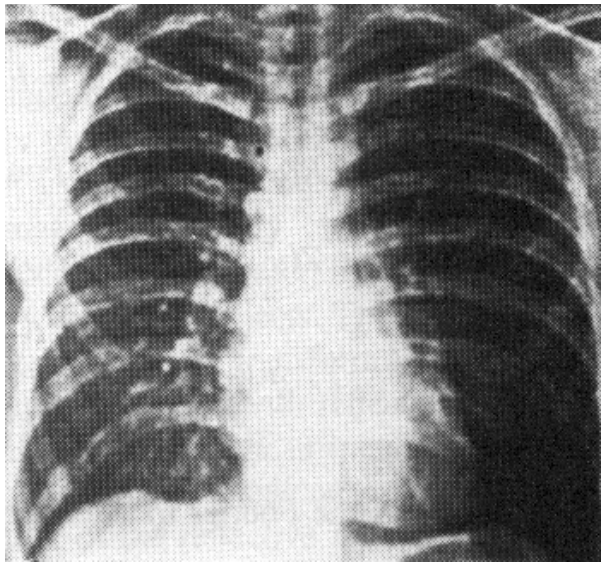


after

Application: High Frequency Emphasis



Original

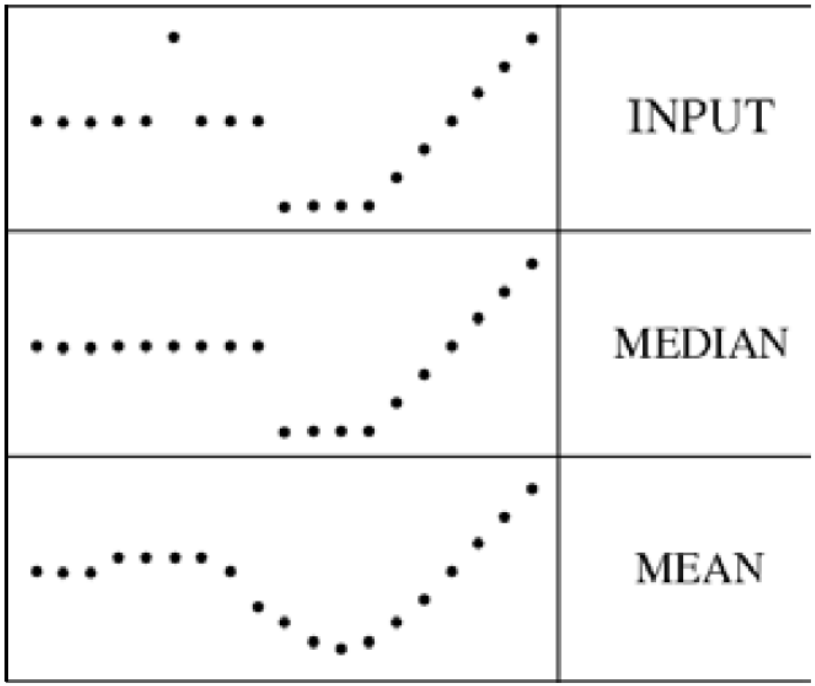


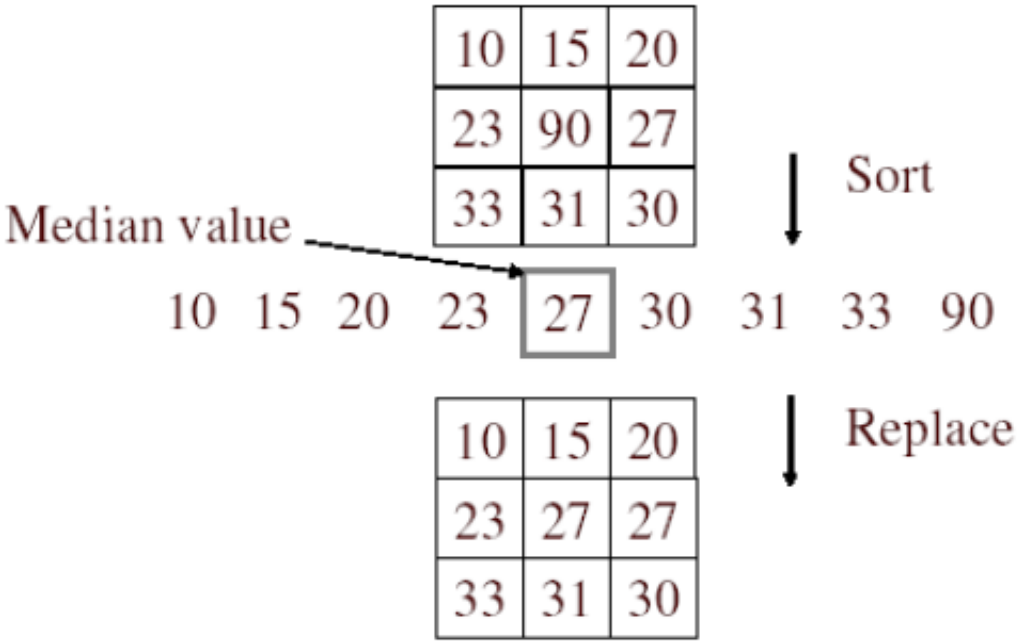
High Frequency Emphasis
+
Histogram Equalization

Non-Linear Filters: Median Filter

– Replace each pixel by the median of its neighbors.

– Comparison with mean:



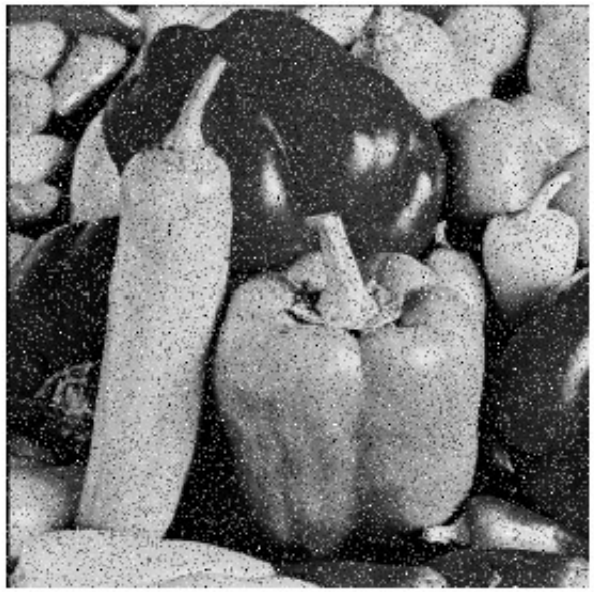


Median Filtering

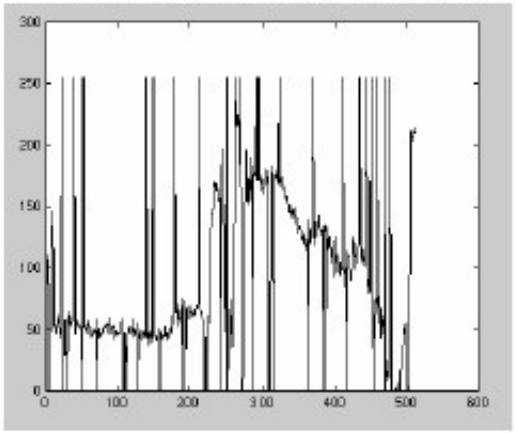
- Median filtering is a non-linear operation that is particularly effective at removing salt and pepper noise.

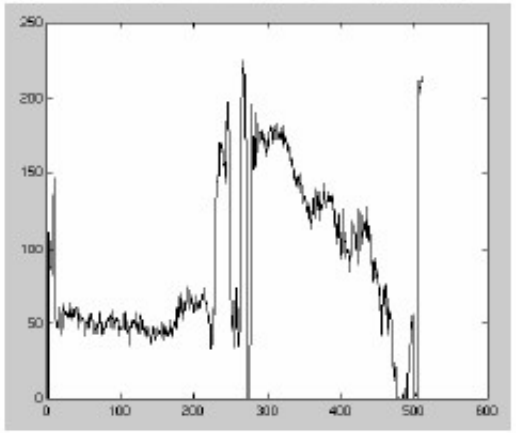
Median Filter

Salt and pepper noise



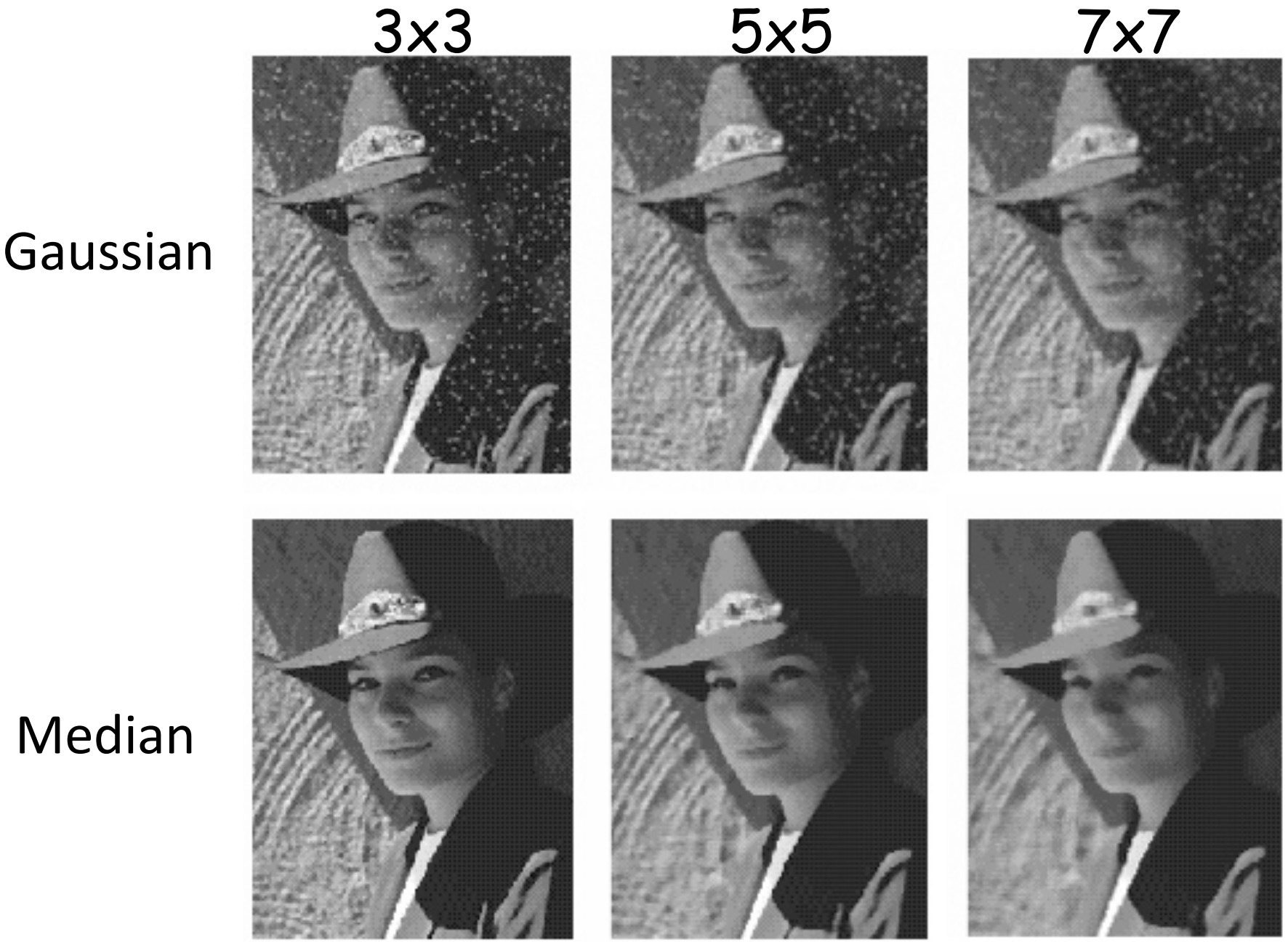
Median filtered



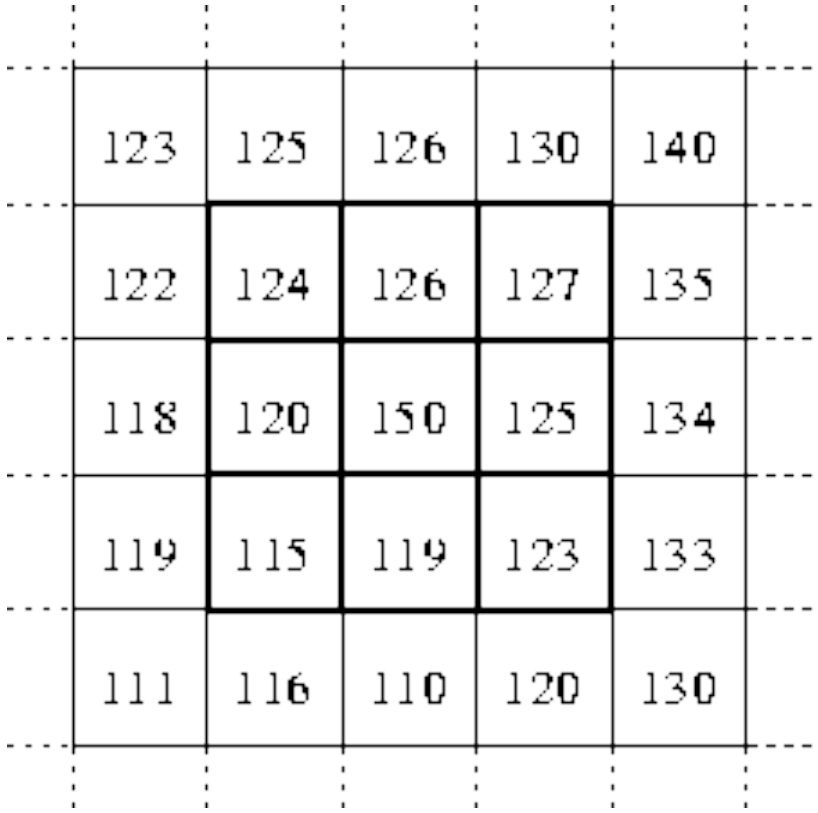


Plots of a row of the image

Median vs. Gaussian Filtering



3x3 Median Filter



Neighbourhood values:

115, 119, 120, 123, 124,
125, 126, 127, 150

Median value: 124

Example

What you need to know

- Noise model
- Linear filter
  - Correlation
  - Convolution
- Gaussian filter
  - Separability
- Smoothing
- Sharpening
- Non-linear filter
  - Median filter

Readings and References

- Section 3.1-3.5 of Szeliski's book
- Denoising: http://web.stanford.edu/class/cs448f/lectures/2.1/Denoising.pdf
- Sharpening: http://web.stanford.edu/class/cs448f/lectures/2.1/Sharpening.pdf