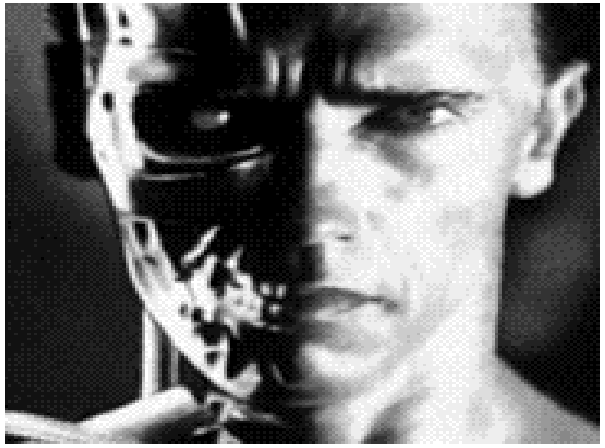
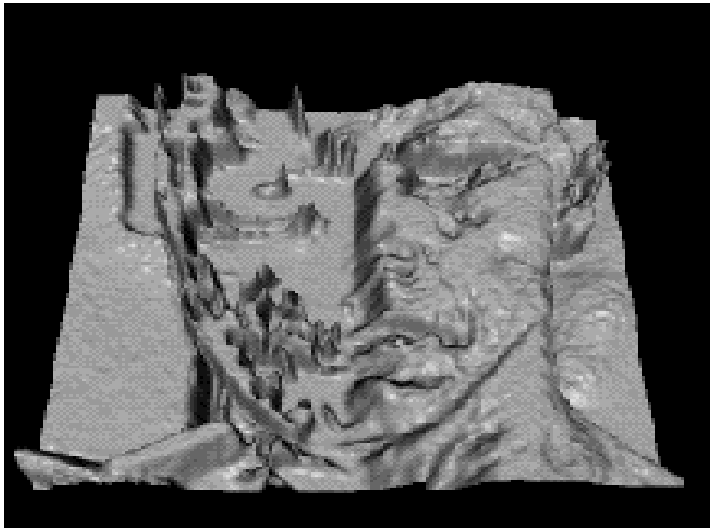
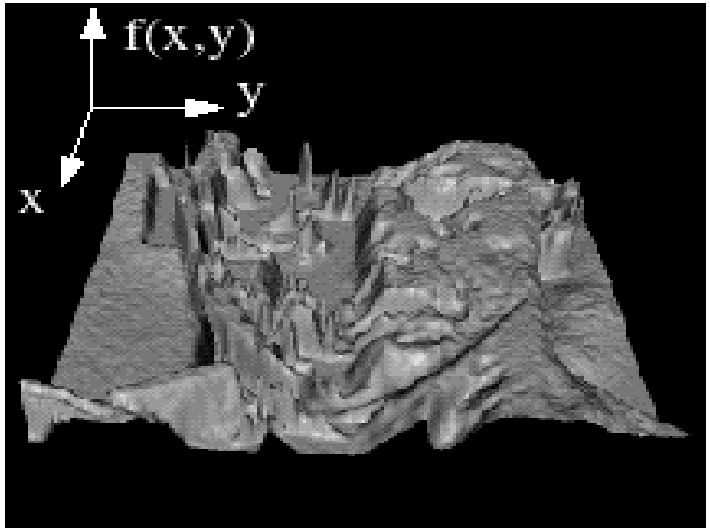
Images as functions







What is a digital image?

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

$$f[i, j] = \text{Quantize}\{ f(i D, j D) \}$$

- Image: matrix of integer values

i ↓ , j →

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

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

$$\hat{I}(i,j) = I(i,j) + n(i,j)$$

Salt and Pepper Noise

- Salt and pepper noise can model errors introduced by the acquisition process. (x and y are random variables in the range (0,1) and l is a constant)

$$\hat{I}(i,j) = \begin{cases} I(i,j) & x < l \\ i_{\min} + y(i_{\max} - i_{\min}) & x \geq l \end{cases}$$

Linear Filters

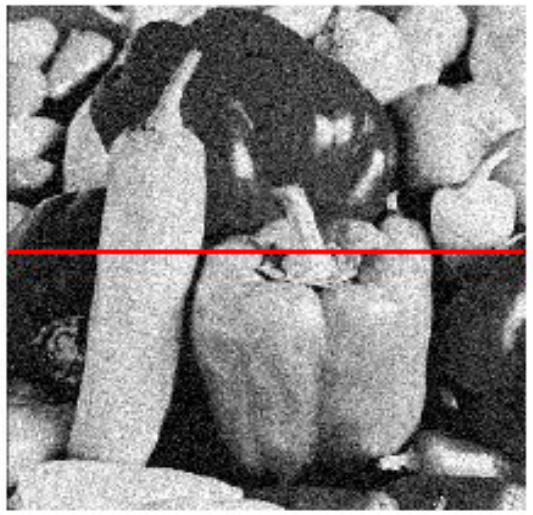
- General process:
  - Form new image whose pixels are a weighted sum of original pixel values, using the same set of weights at each point.
- Properties
  - Output is a linear function of the input
  - Output is a shift-invariant function of the input (i.e. shift the input image two pixels to the left, the output is shifted two pixels to the left)
- Example: smoothing by averaging
  - form the average of pixels in a neighbourhood
- Example: smoothing with a Gaussian
  - form a weighted average of pixels in a neighbourhood
- Example: finding a derivative
  - form a weighted average of pixels in a neighbourhood

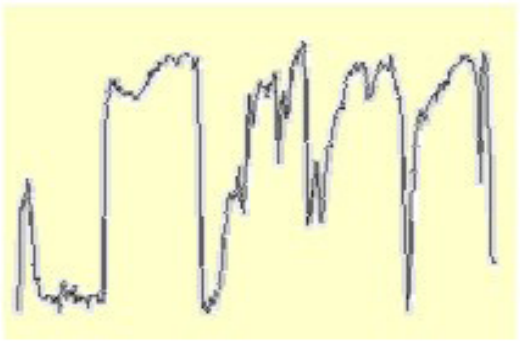
Image Filtering

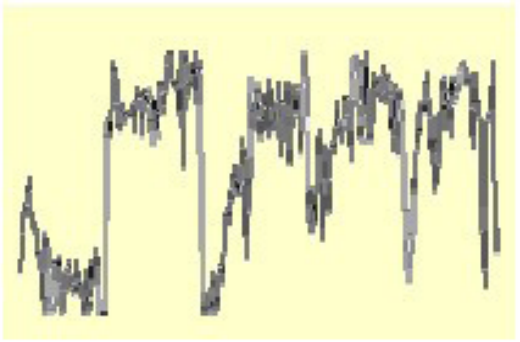
- Noise reduction/image restoration

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

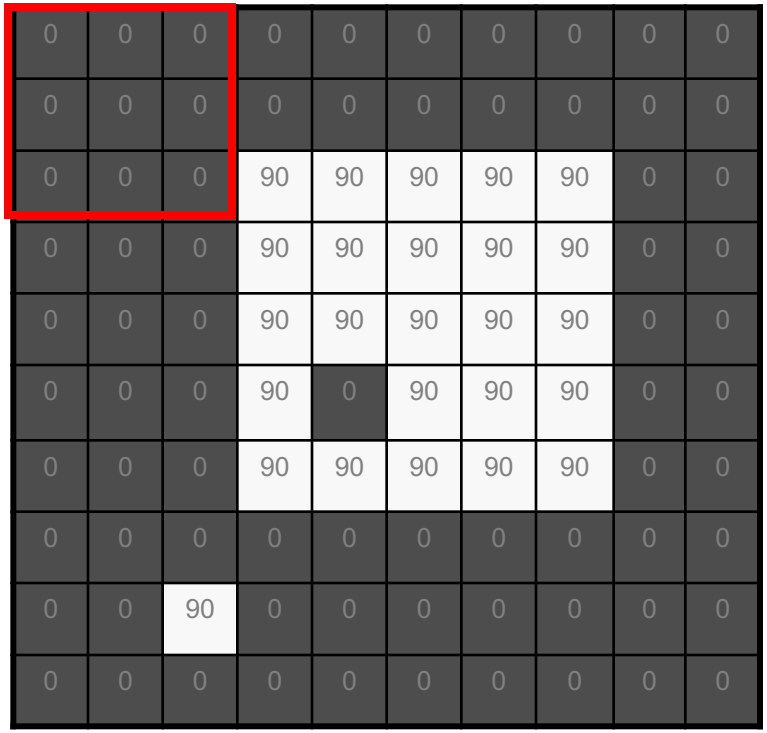
Removing noise

- Basic assumption
  - Noise process is independent, identically distributed
  - Image has a more regular underlying structure
    - Consider measuring the speed of wind or the value of a stock

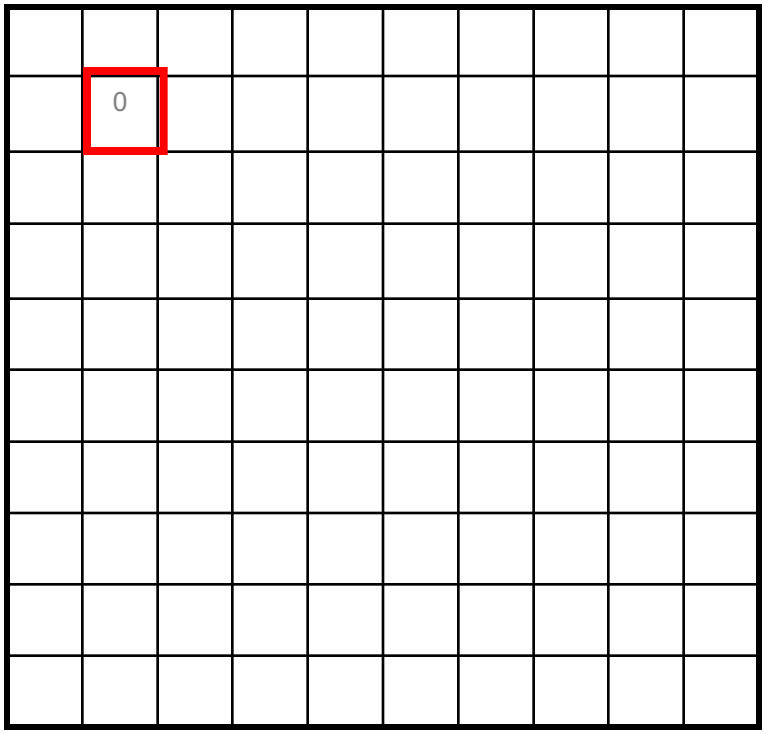
- By considering larger neighborhoods we can separate the signal from the noise

Moving Average in 2D

$F[x, y]$

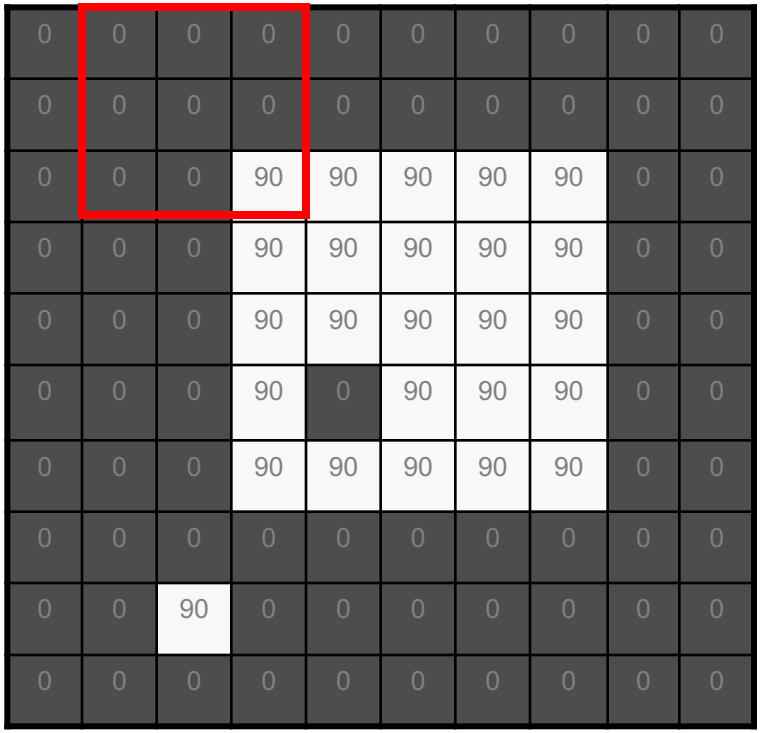


$G[x, y]$

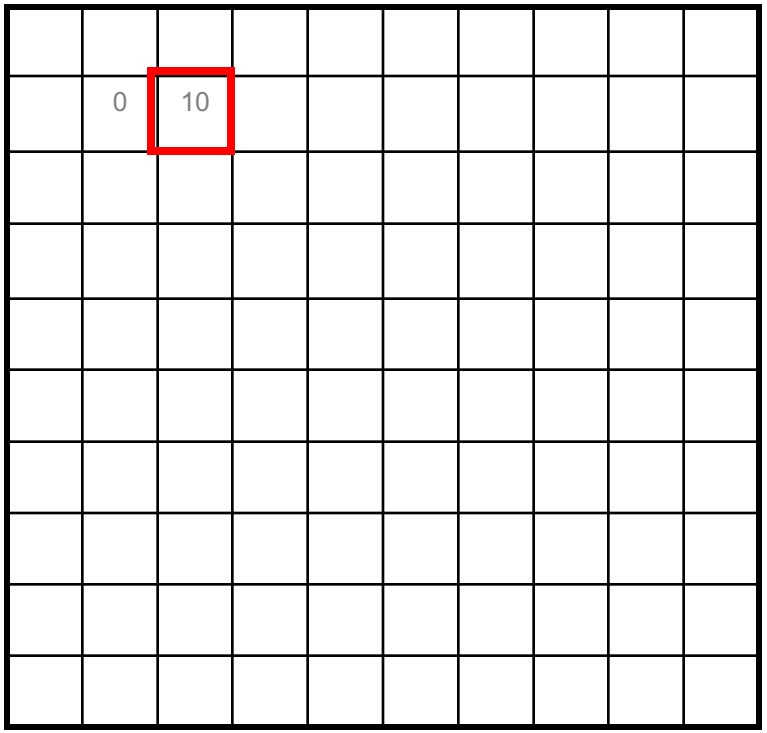


Moving Average in 2D

$F[x, y]$

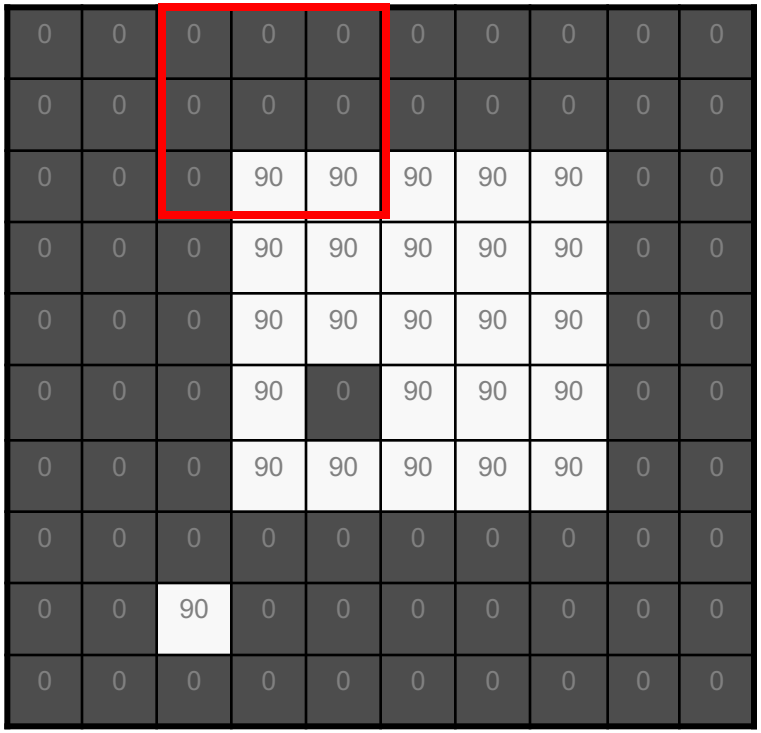


$G[x, y]$

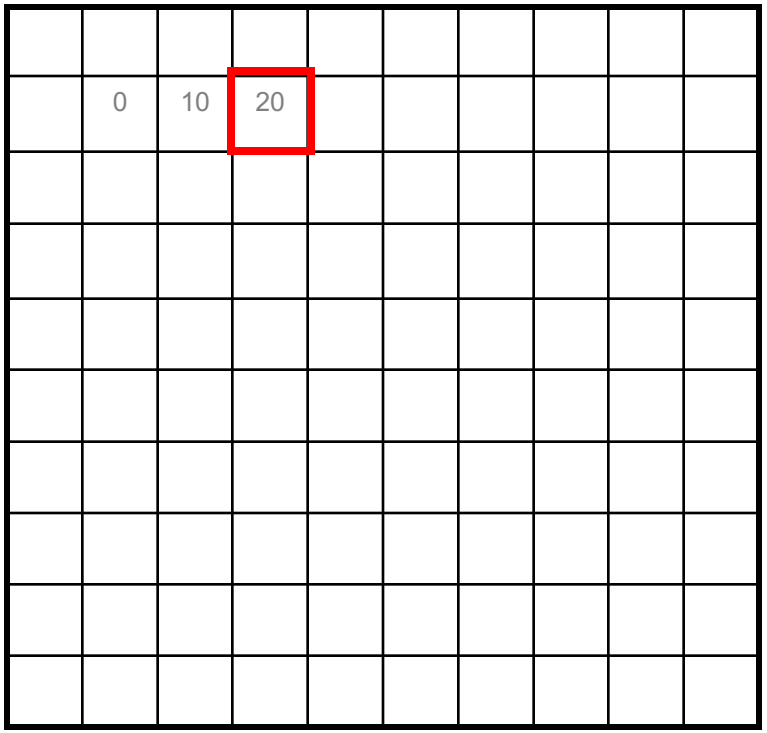


Moving Average in 2D

$F[x, y]$

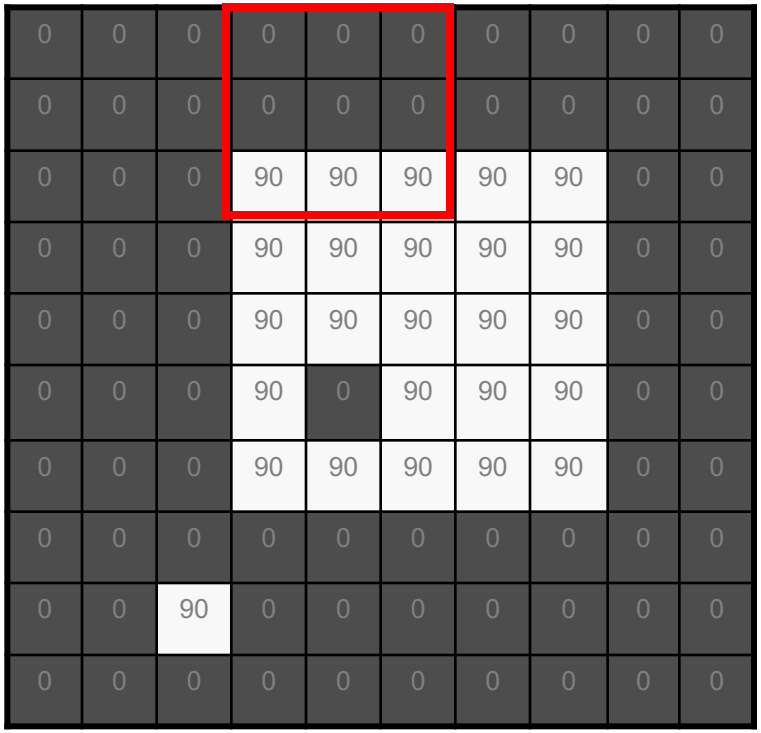


$G[x, y]$

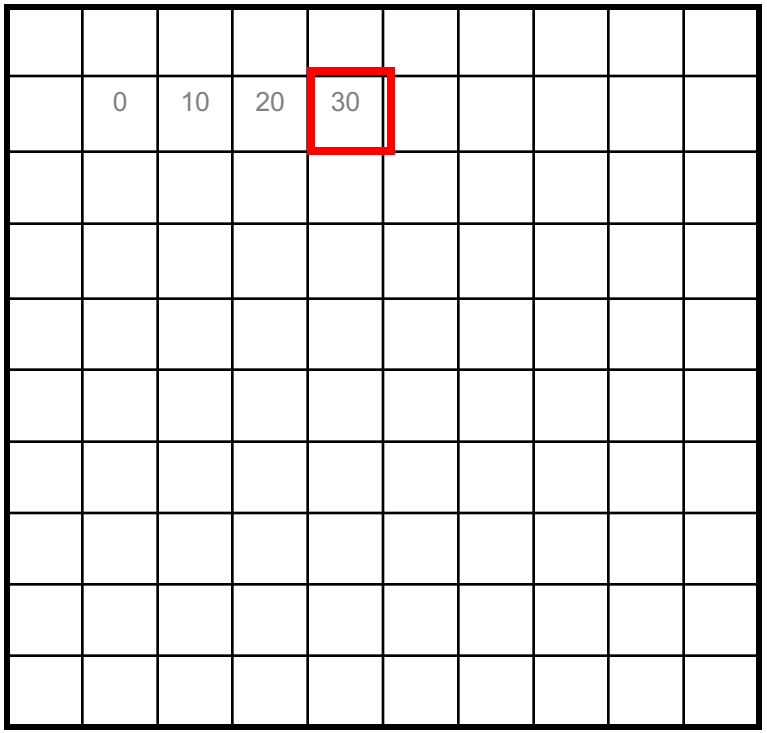


Moving Average in 2D

$F[x, y]$

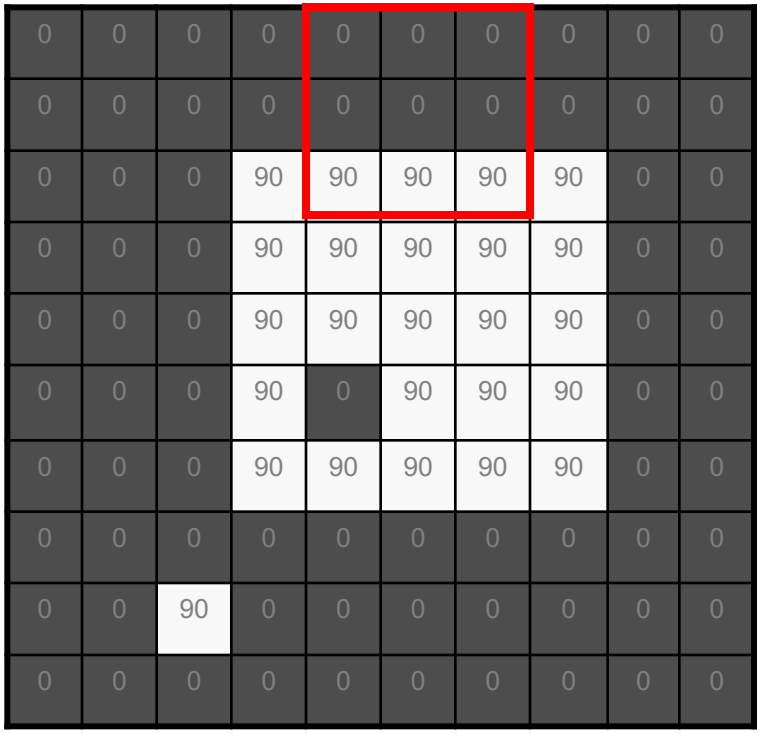


$G[x, y]$

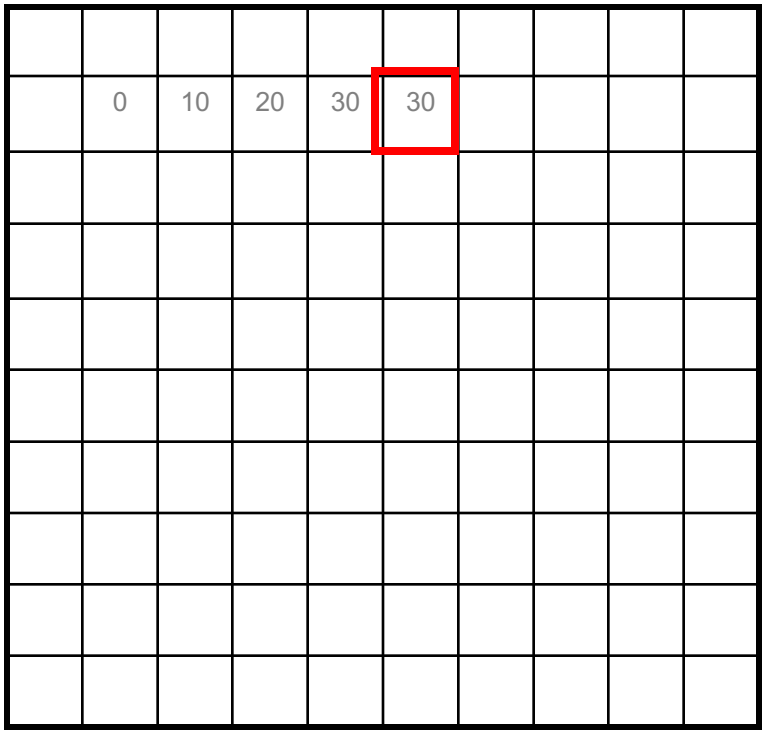


Moving Average in 2D

$F[x, y]$

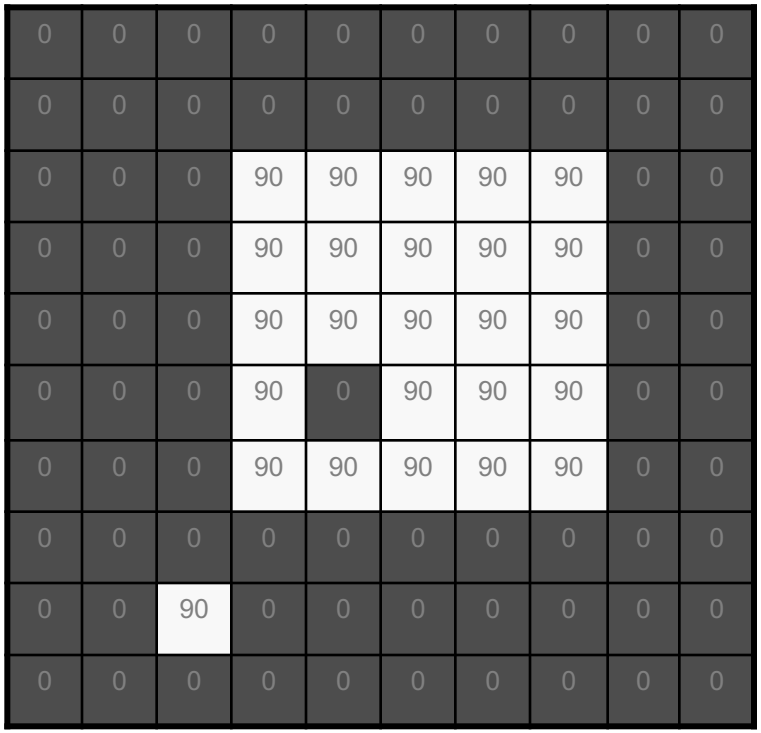


$G[x, y]$

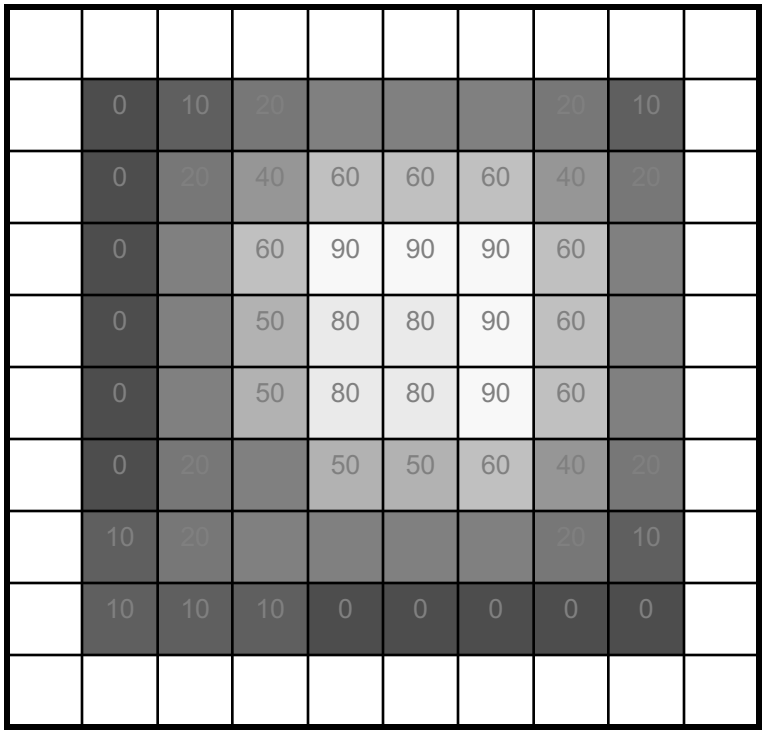


Moving Average in 2D

$F[x, y]$



$G[x, y]$



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:

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

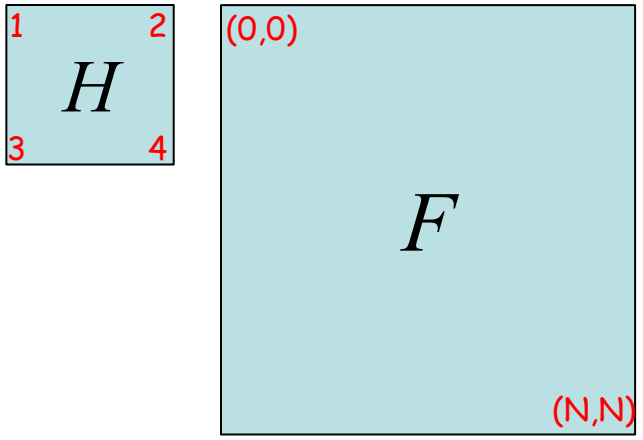
- Correlation filtering:

$$G = H \otimes F$$

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:

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

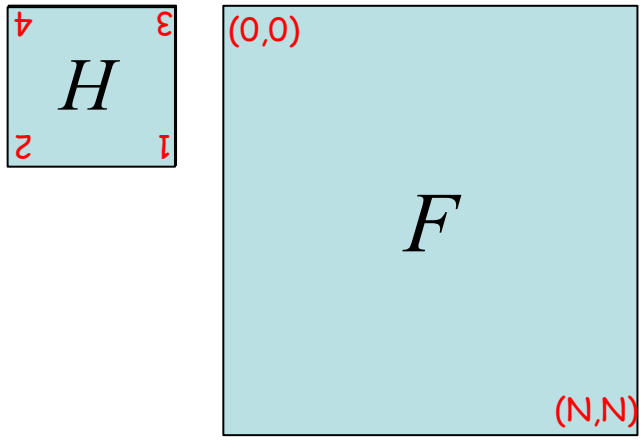
- Notice the order of indices
  - all examples can be put in this form
  - it's a result of the derivation expressing any shift-invariant linear operator as a convolution.

Convolution

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

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

$$G = H * F$$



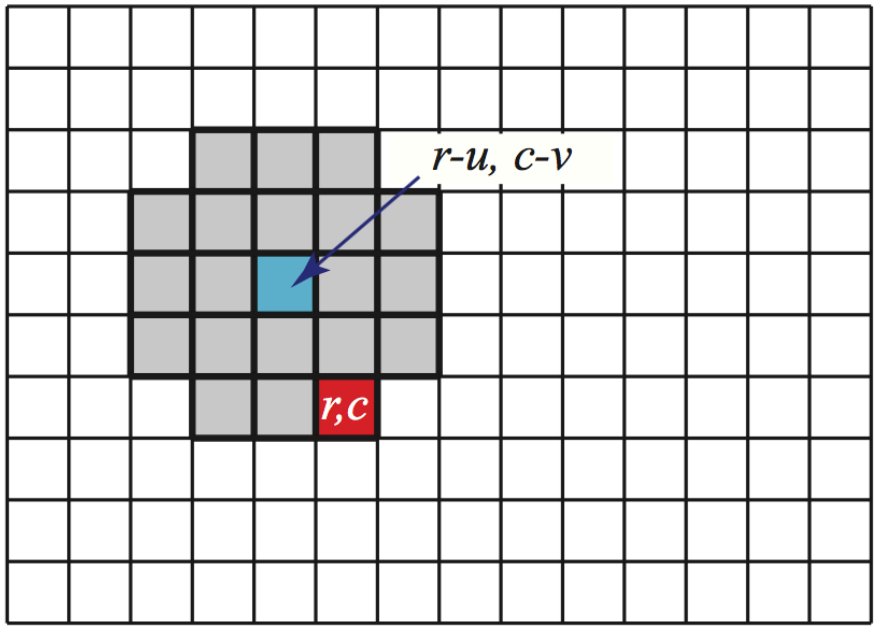
- Convolution of continuous signals

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

Convolution

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

- H is usually called the **kernel**
- H is the weight of how a source/input pixel value contribute to output pixels
- r = i, c = j



Pic from Carlo Tomasi

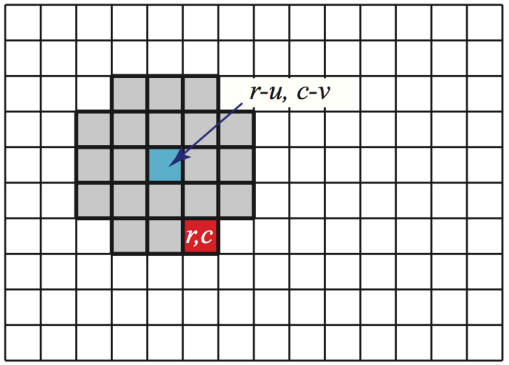
Convolution vs Correlation Filtering

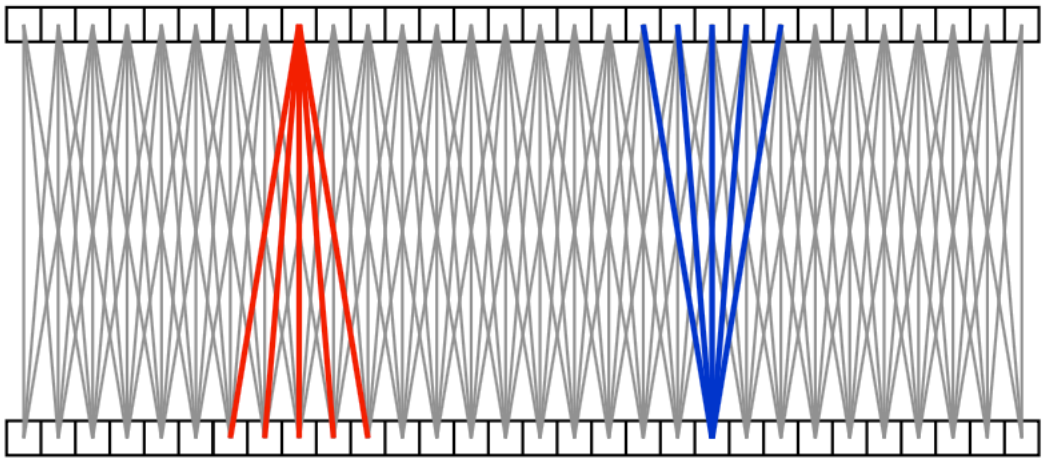
Correlation filtering

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

Convolution

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





Algebraic properties

Commutativity

$$f * g = g * f$$

Distributivity

$$f * (g + h) = (f * g) + (f * h)$$

Associativity

$$f * (g * h) = (f * g) * h$$

Associativity with scalar multiplication

$$a(f * g) = (af) * g$$

- Any linear operation
  - X*(aY + bZ) = a(X*Y) + b(X*Z)

Relationship with differentiation

$$(f * g)' = f' * g = f * g'$$

Computation

- 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

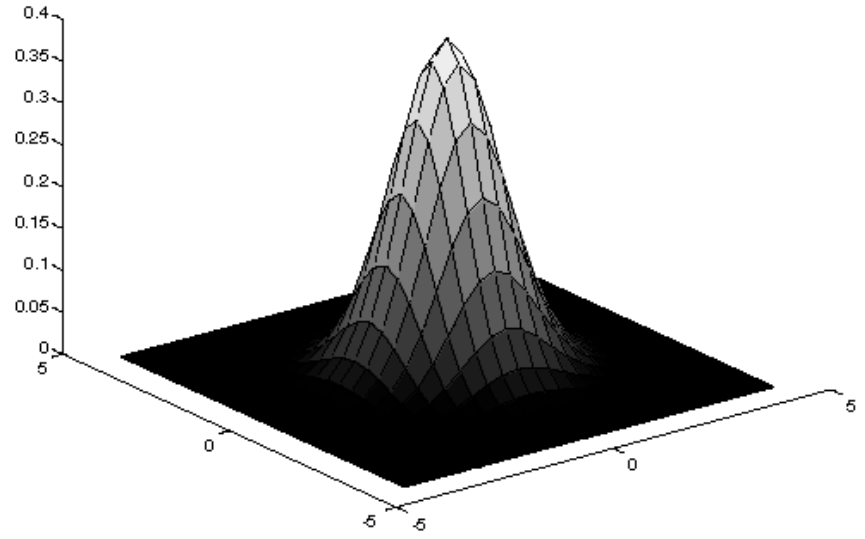
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

- Entries must add up to one. Why?

Example: Smoothing by Averaging

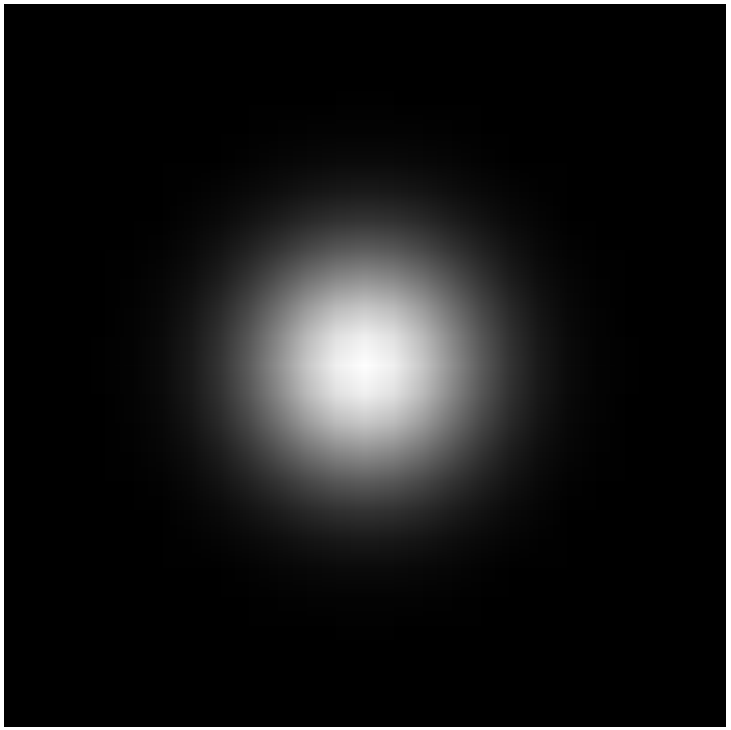
Smoothing with a Gaussian

- 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

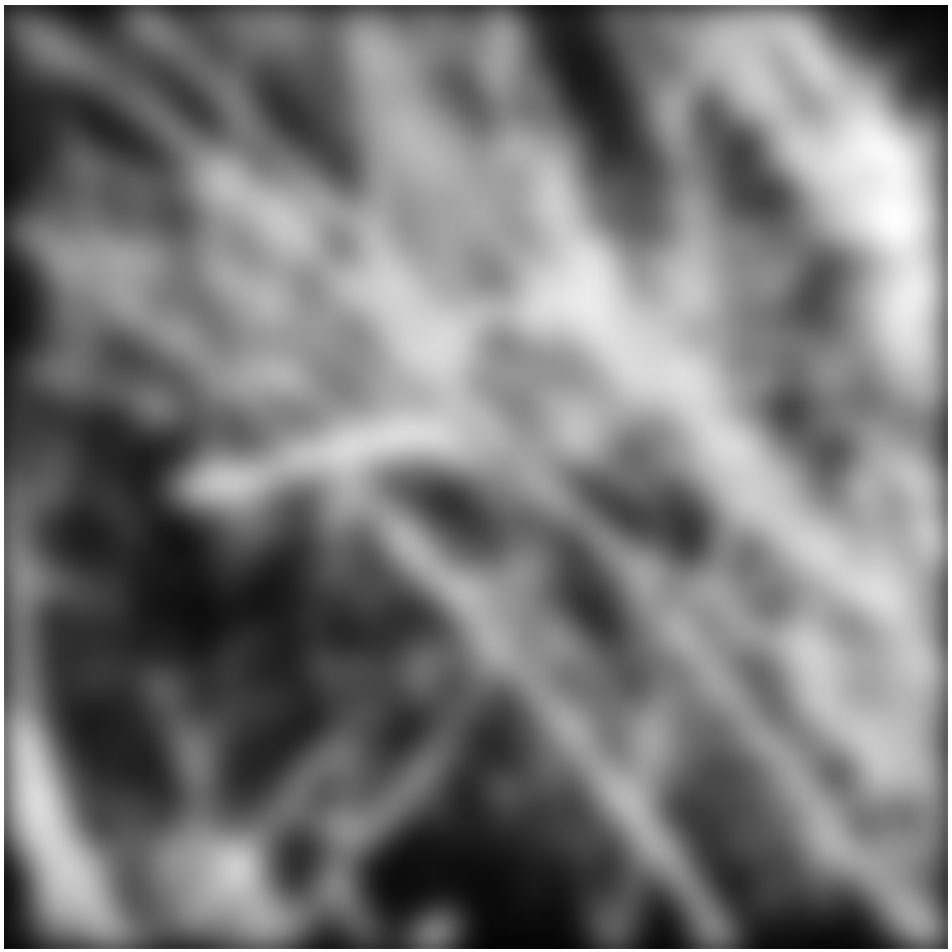


- The picture shows a smoothing kernel proportional to

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

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

Smoothing with a Gaussian



Differentiation and convolution

- Recall

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

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

- We could approximate this as

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

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

Frequency Domain: Fourier Series, Transform

Idea: Fourier series is a way to represent a function as the sum of simple sine waves.

$$s_N(x) = \frac{A_0}{2} + \sum_{n=1}^{N} A_n \cdot \sin\left(\frac{2\pi n x}{P} + \phi_n\right)$$

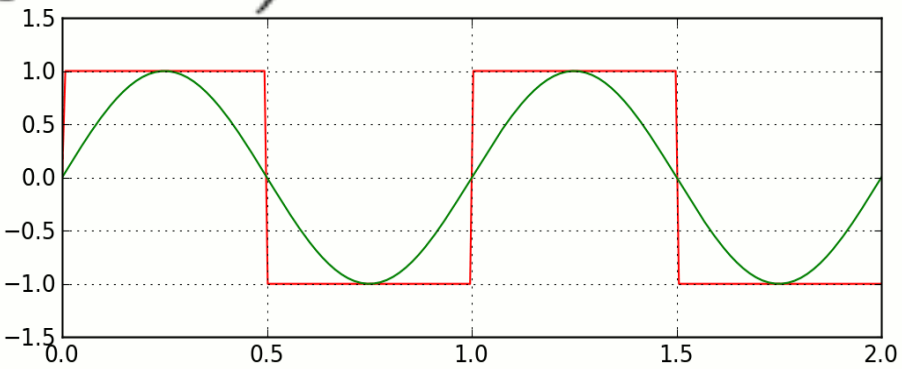
Euler's formula:

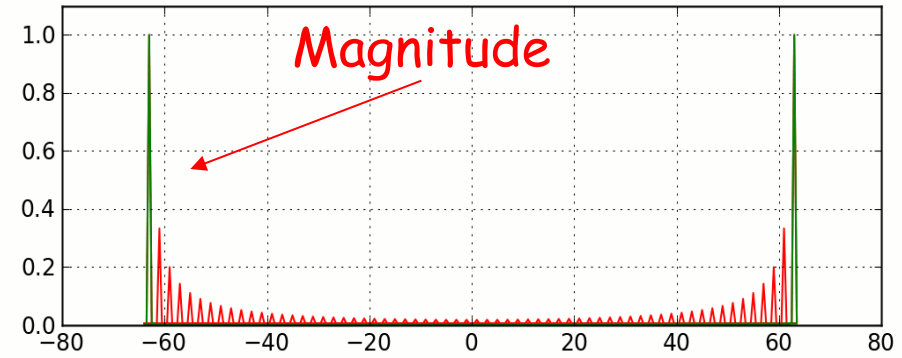
$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{i 2\pi k n / N}$$

Fourier Transform: Go from signal to series coefficients (frequencies magnitude).

Has an inverse.

Slide by R Shilkrot

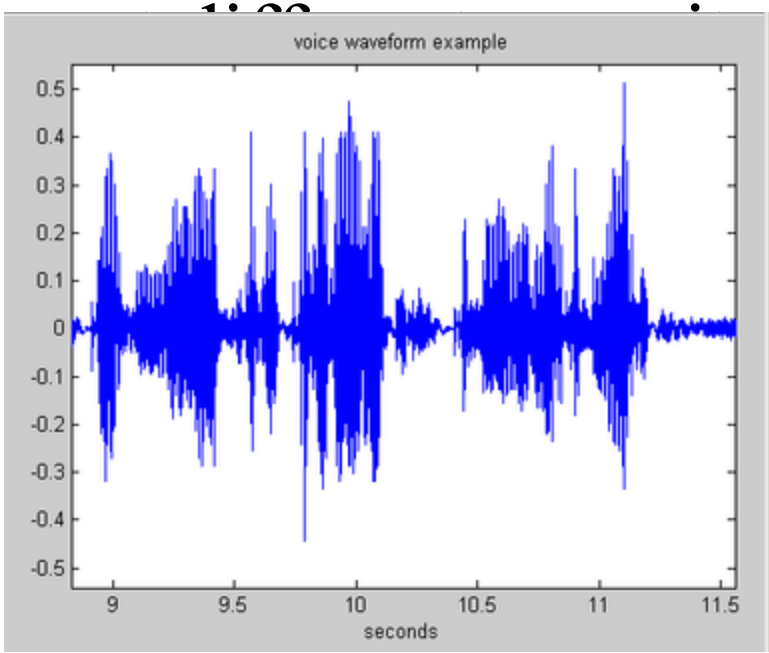


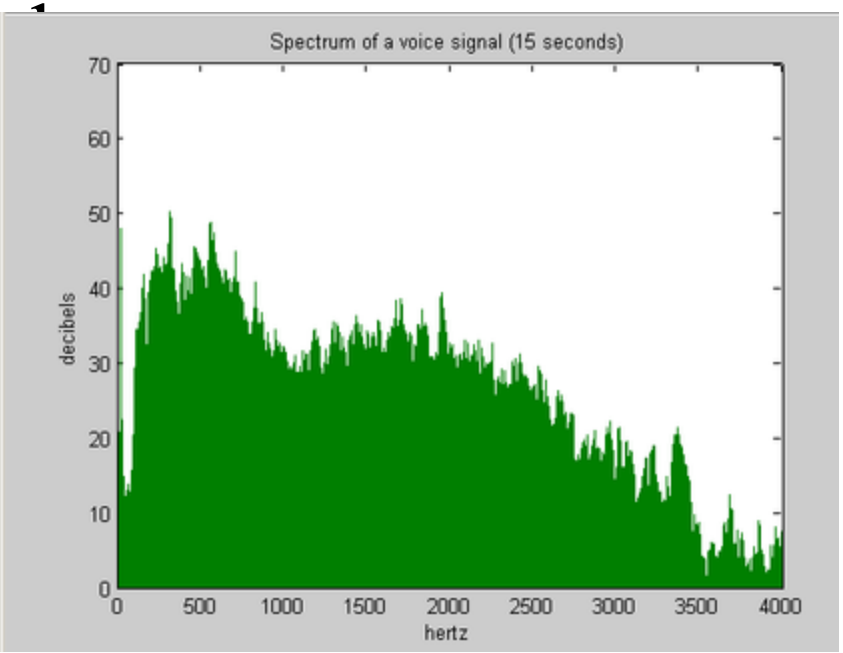


Frequency spectrum:

Example: Music

- We think of music in terms of frequencies

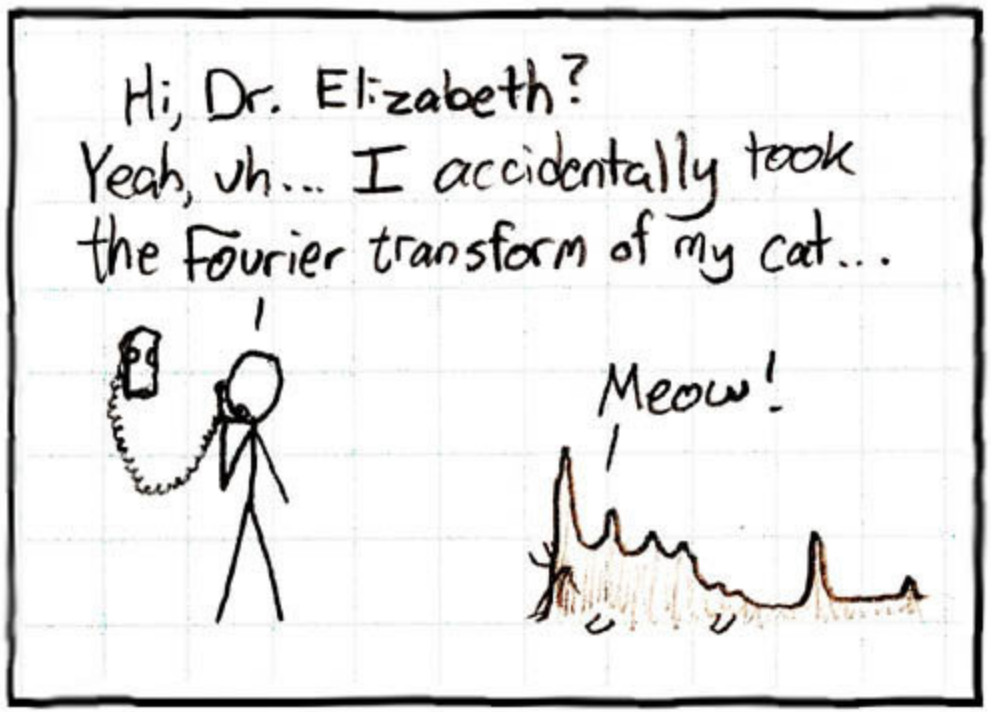




Slide: Hoiem

Other signals

- We can also think of all kinds of other signals the same way

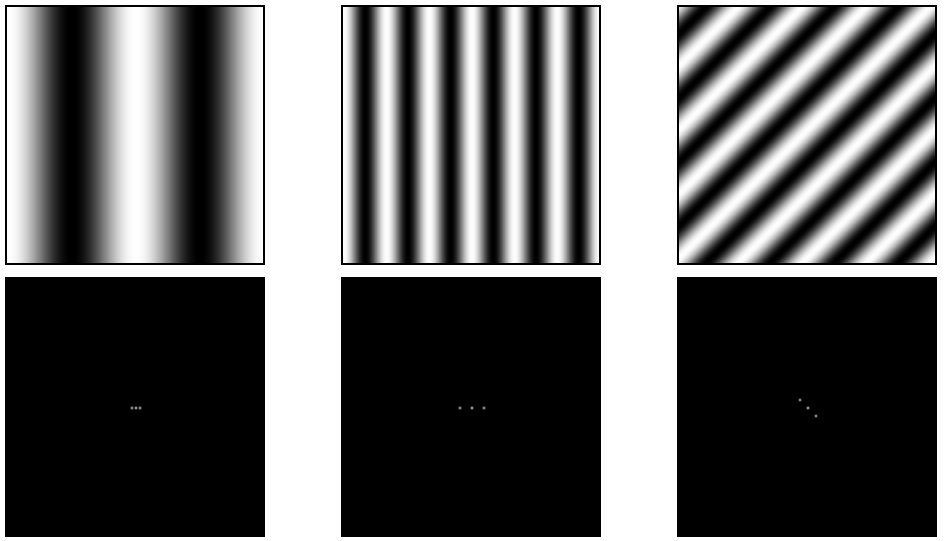


xkcd.com

Fourier analysis in images

Intensity Image

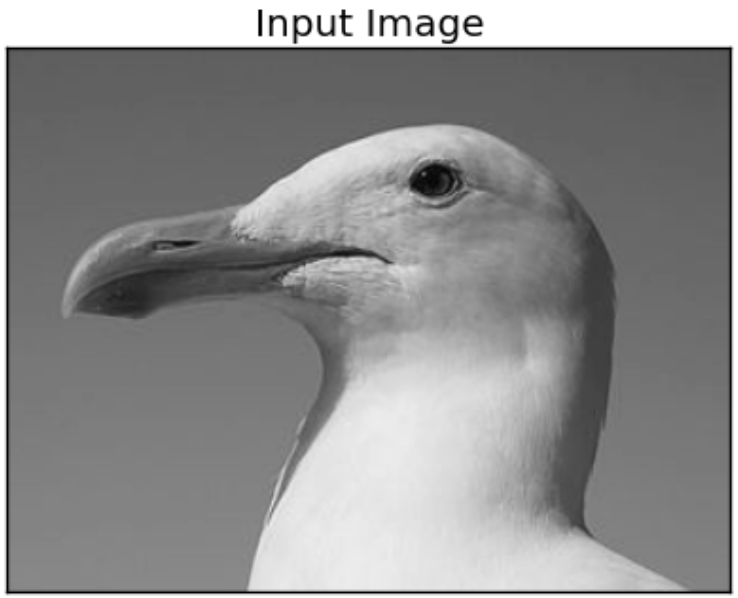
Fourier Image

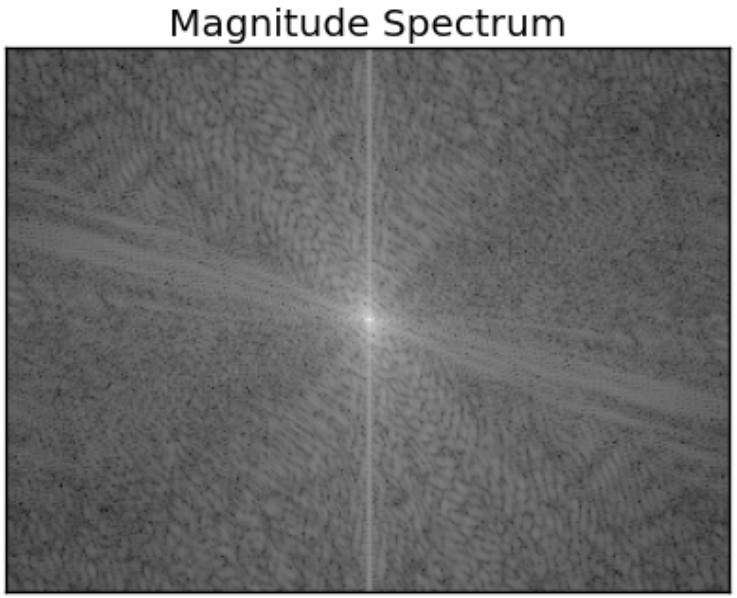


The FT encodes the change in image intensity based on magnitude, frequency and phase sinusoids.
So it goes from image domain to frequency domain.

http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering

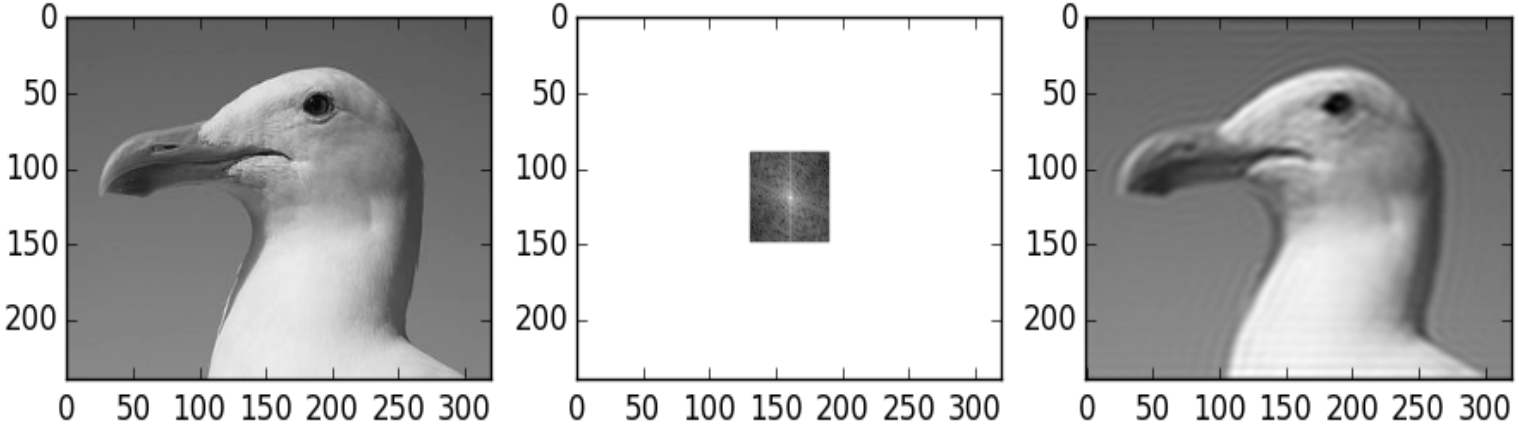
Example



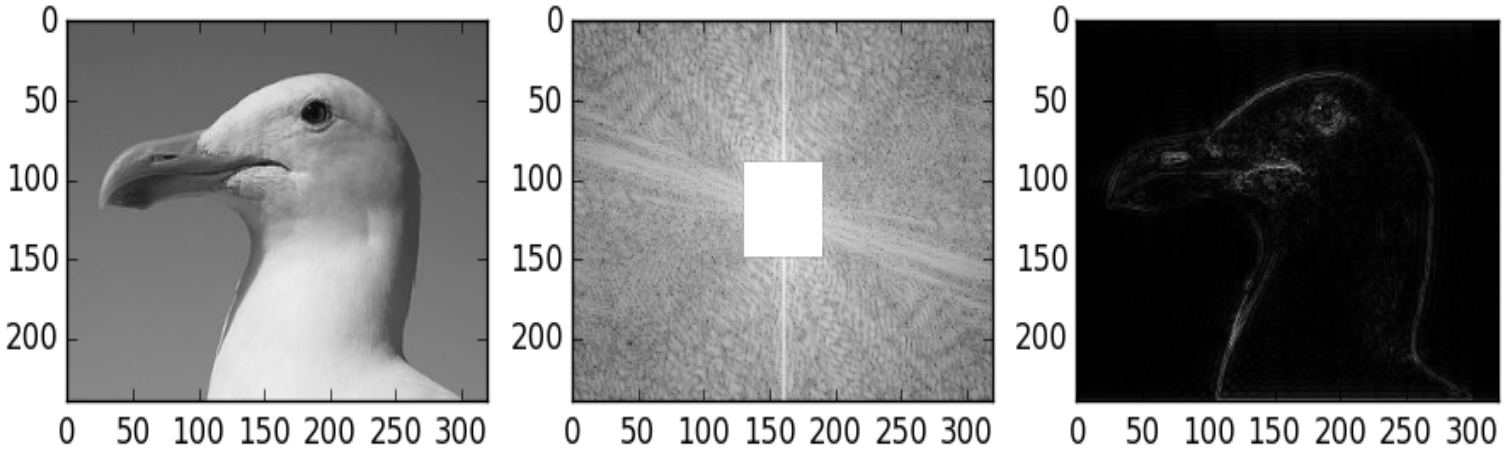


Slide by R Shilkrot

Low and High Pass Filters



LPF: Mask high frequencies of the freq. domain.

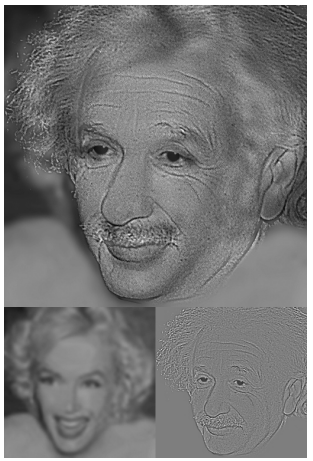


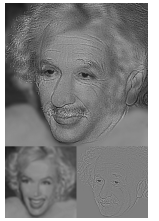
HPF: Mask low frequencies of the freq. domain.

Slide by R Shilkrot

Hybrid Images

Low-pass
A
("shape")
+
High-pass
B
("details")





Who can you see?

Slide by R Shilkrot

Convolution Theorem

The FT of a convolution is the pointwise product of FTs.

Let $\mathcal{F}$ denote the Fourier transform operator, so $\mathcal{F}\{f\}$ and $\mathcal{F}\{g\}$ are the Fourier transforms of f and g, respectively. Then

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\}$$

where $\cdot$ denotes point-wise multiplication. It also works the other way around:

$$\mathcal{F}\{f \cdot g\} = \mathcal{F}\{f\} * \mathcal{F}\{g\}$$

By applying the inverse Fourier transform $\mathcal{F}^{-1}$, we can write:

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\}$$

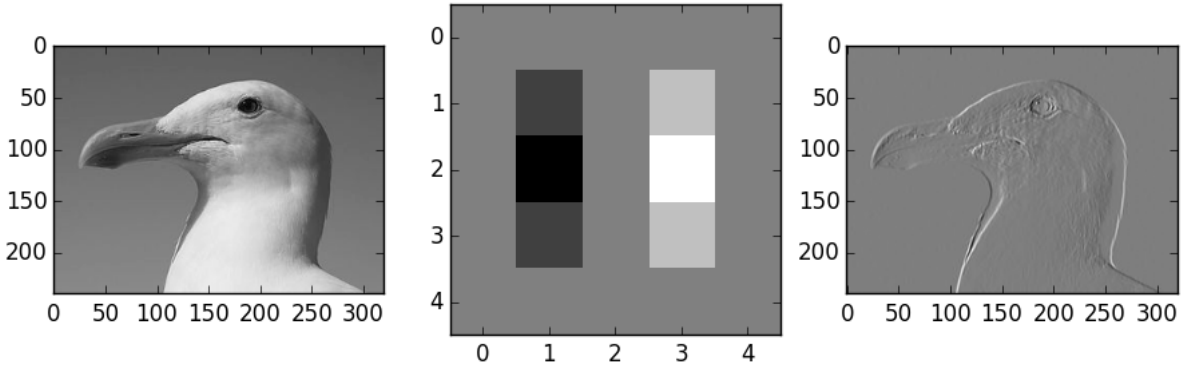
and:

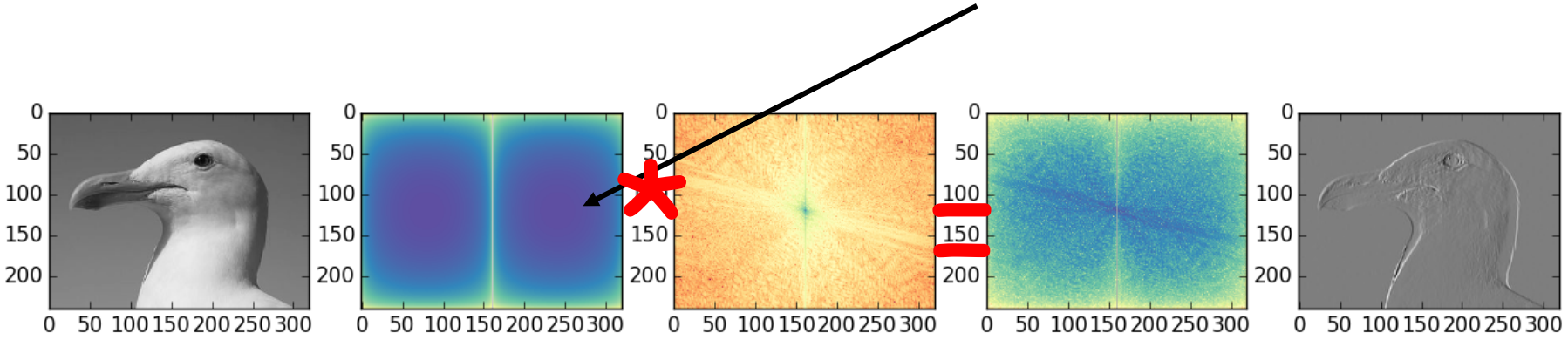
$$f \cdot g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} * \mathcal{F}\{g\}\}$$

Slide by R Shilkrot

Example

Simple Sobel filter in image ("spatial") domain:





The FT way:

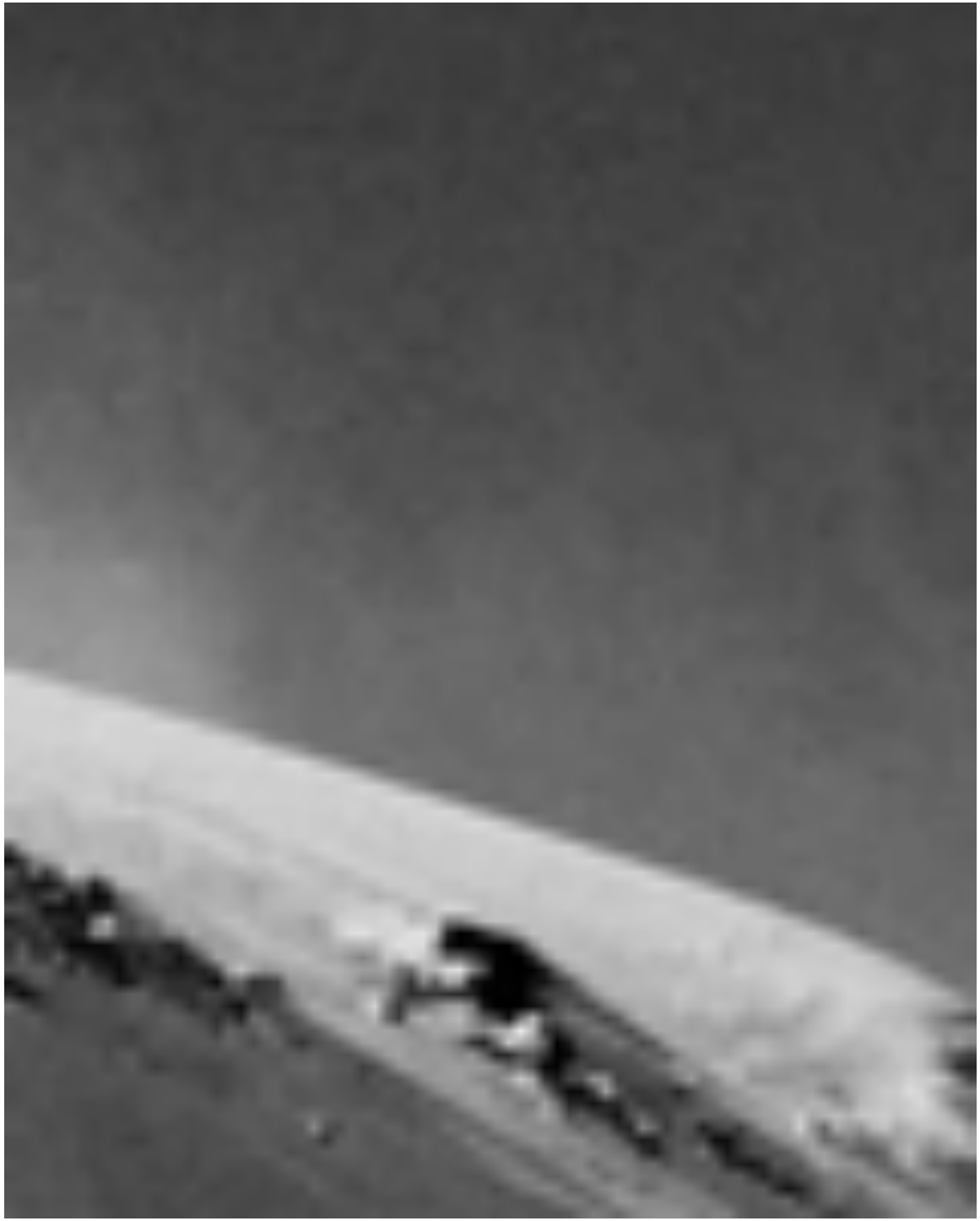
F^{-1}

Slide by R Shilkrot

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

Differentiation and convolution

- Recall

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

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

- We could approximate this as

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

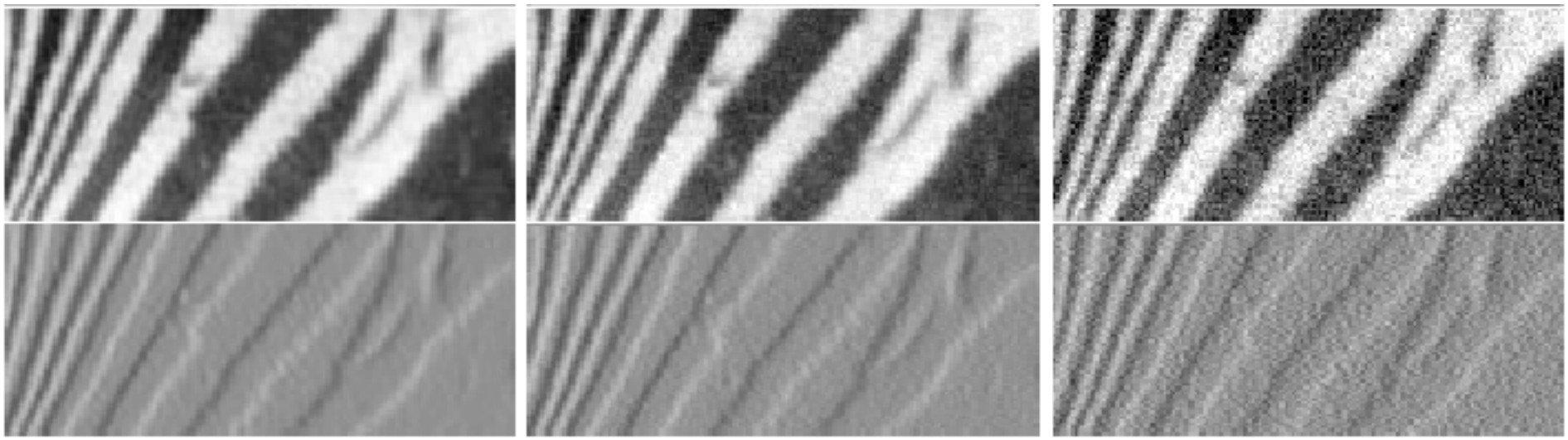
(which is obviously a convolution; it's not a very good way to do things, as we shall see)

Finite differences and noise

- Finite difference filters respond strongly to noise
  - obvious reason: image noise results in pixels that look very different from their neighbours
- 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 neighbours
  - 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 neighbours (=noise pixels?) to look more like neighbours

Finite differences responding to noise



Increasing noise ->
(this is zero mean additive gaussian noise)

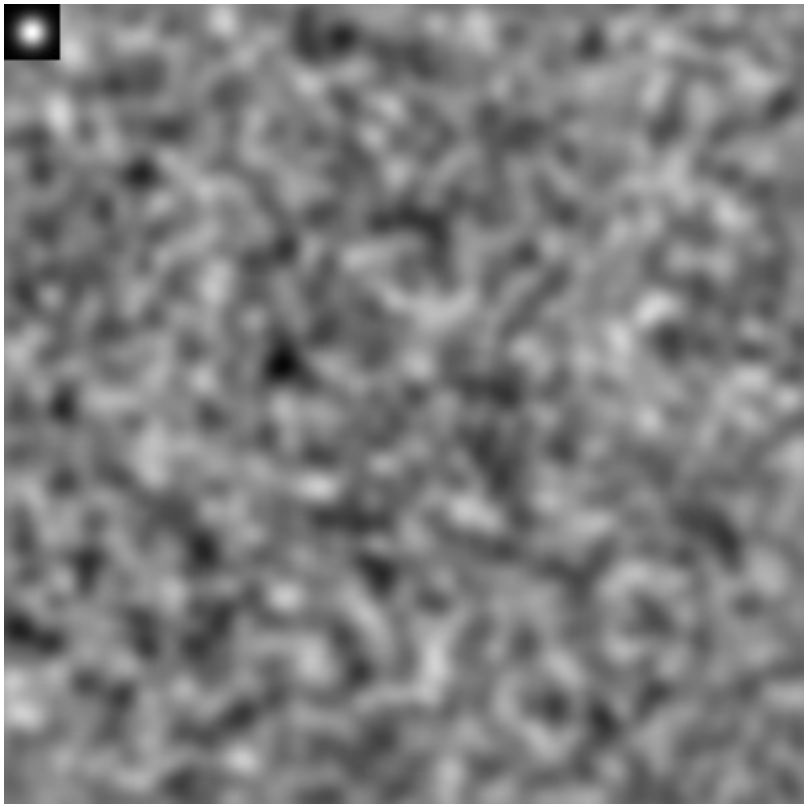
The response of a linear filter to noise

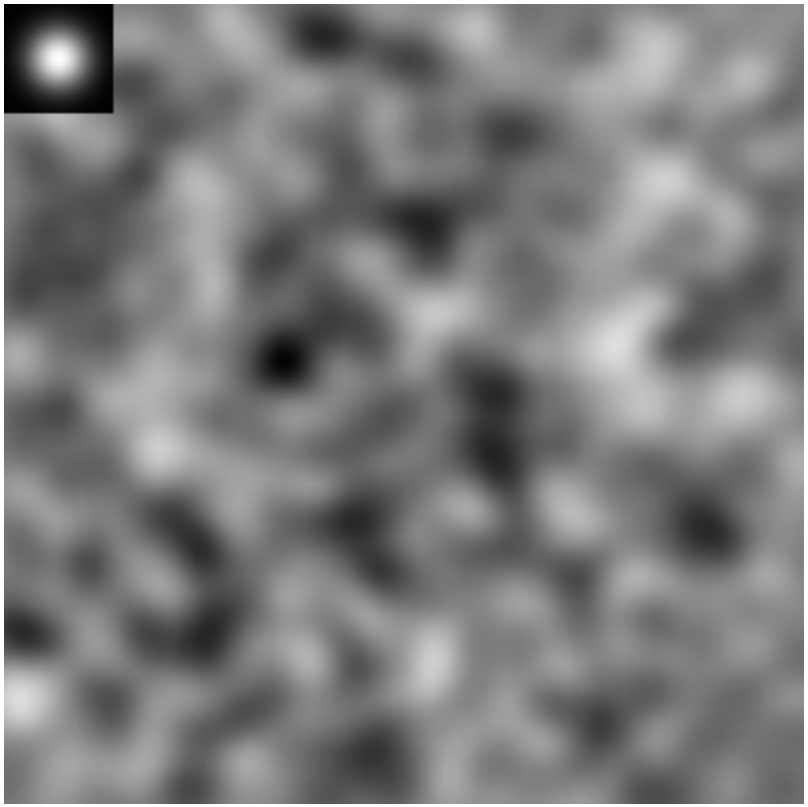
- Do only stationary independent additive Gaussian noise with zero mean (non-zero mean is easily dealt with)
- **Mean**:
  - output is a weighted sum of inputs
  - so we want mean of a weighted sum of zero mean normal random variables
  - must be zero
- **Variance**:
  - recall
    - variance of a sum 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

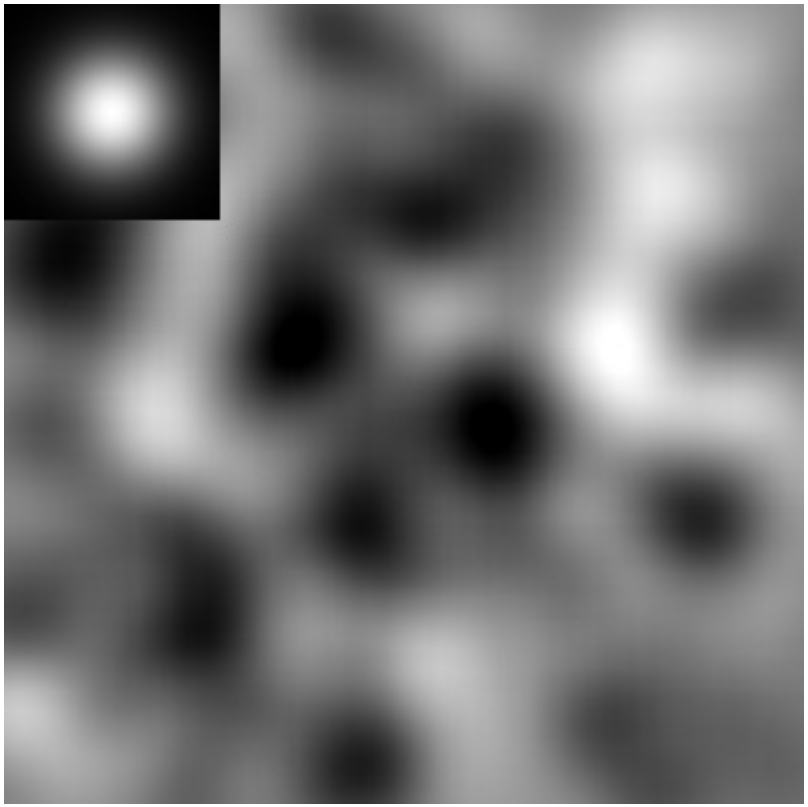
$$\sigma^2 \sum_{u,v} K_{u,v}^2$$

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.

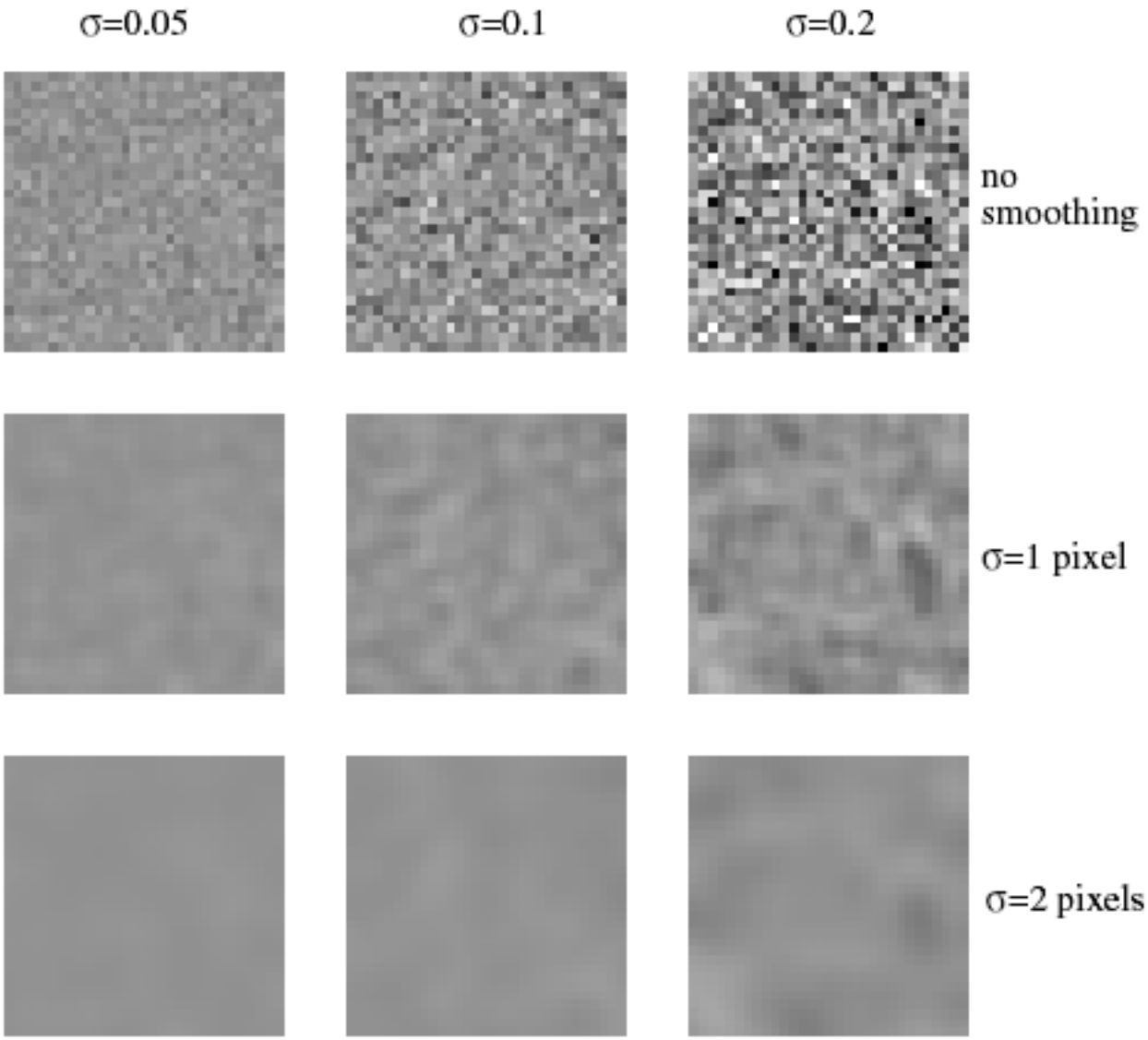






Smoothing reduces noise

- Generally expect pixels to "be like" their neighbours
  - 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



The effects of smoothing
Each row shows smoothing with gaussians of different width; each column shows different realisations of an image of gaussian noise.

Separable filters

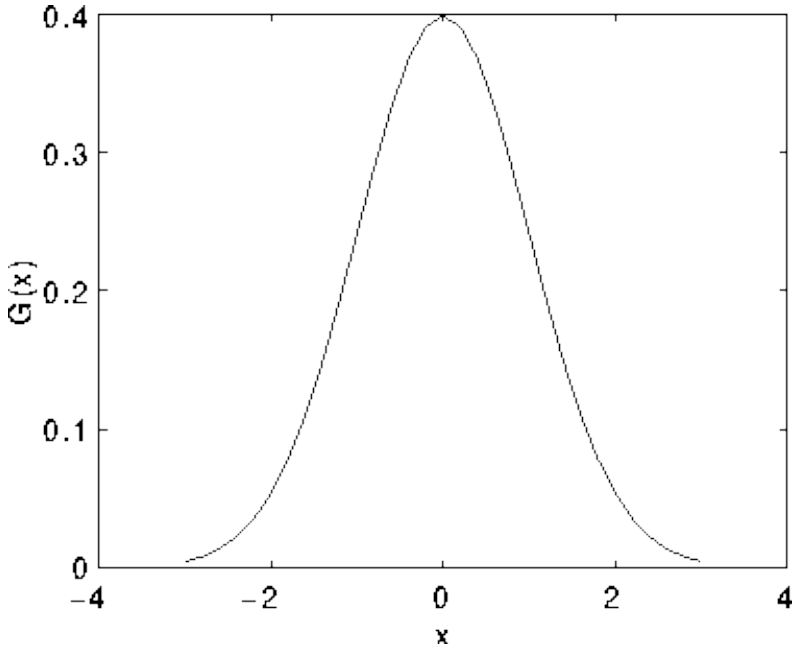
- Some 2D image filters can actually be implemented as 2 1D filters - one in the horizontal direction followed by another in the vertical direction.
- This can significantly lower the computational complexity of the operation

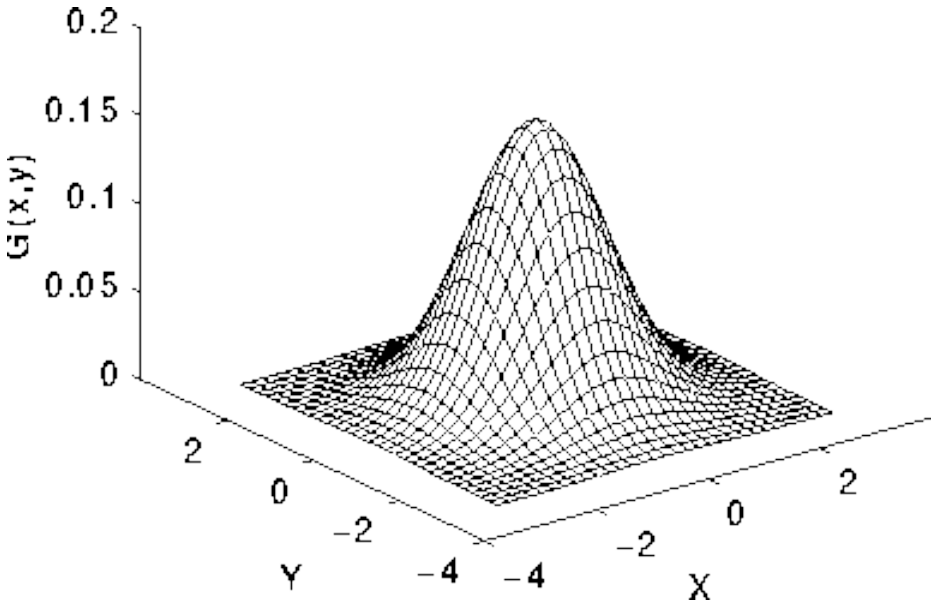
Gaussian Filtering

- Separable filter

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

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





Gaussian Filtering (II)

- $\sigma = 1$

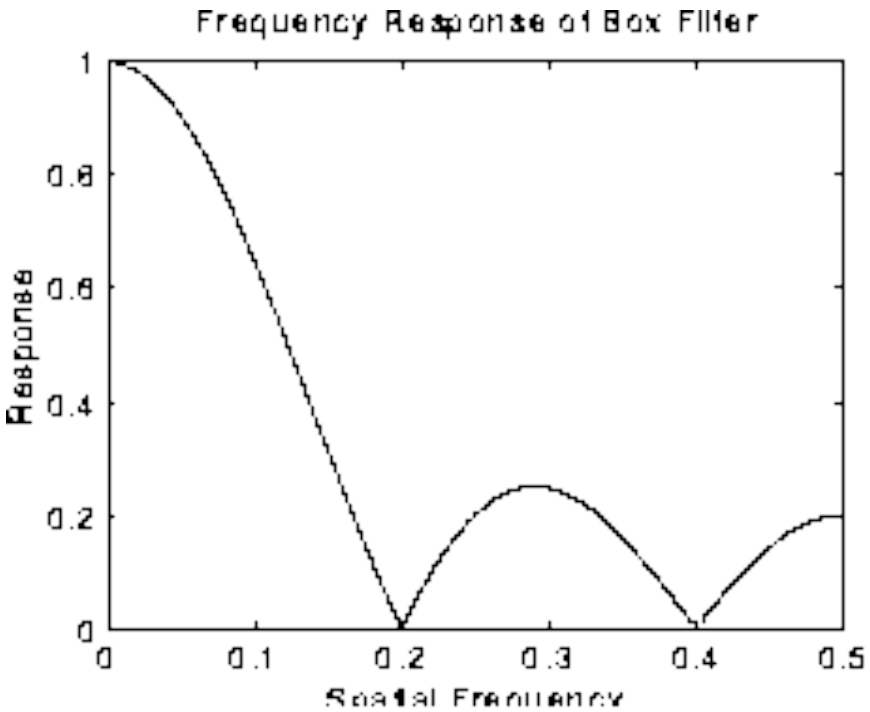
$\frac{1}{10.7}$

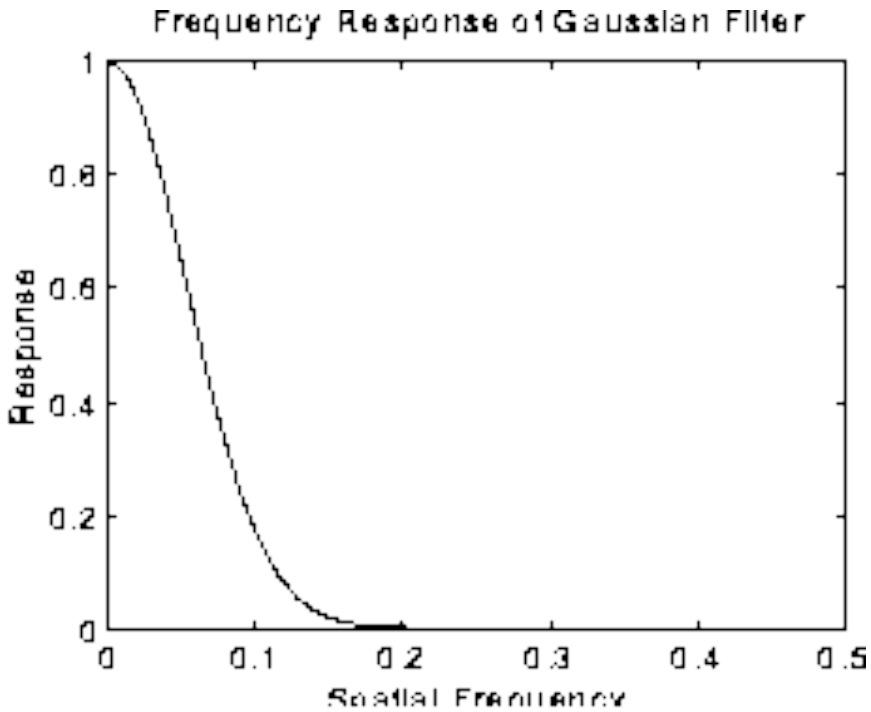
1.3	3.2	3.8	3.2	1.3

$\frac{1}{115}$

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)





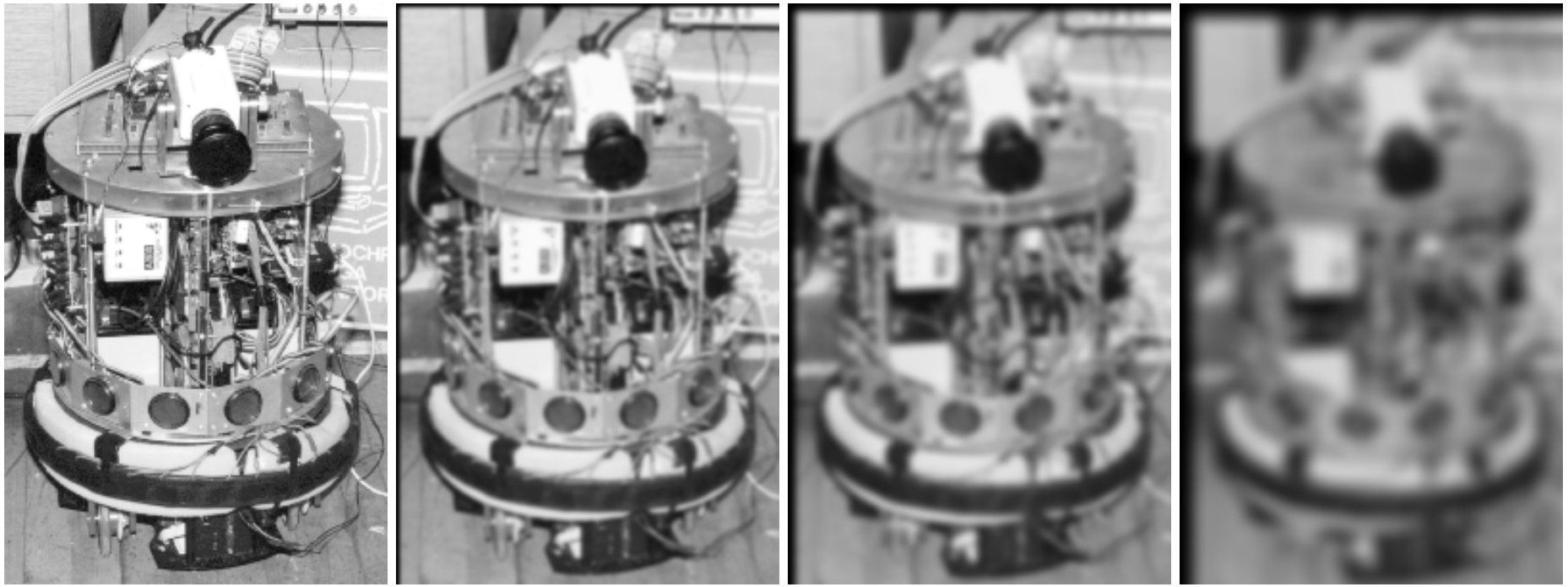
Gaussian Filtering (IV)

- Sampling theorem
- To keep the energy in 98.76% area

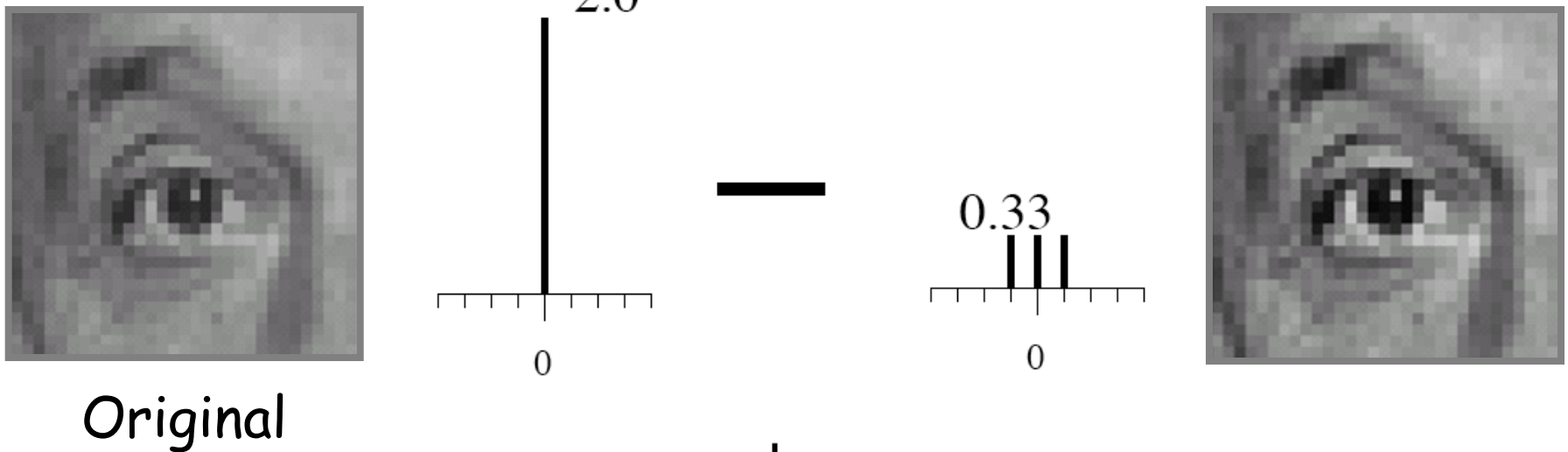
  sigma >0.8
- Repeated averaging

Example

- σ =1.0, σ =2.0, σ =4.0



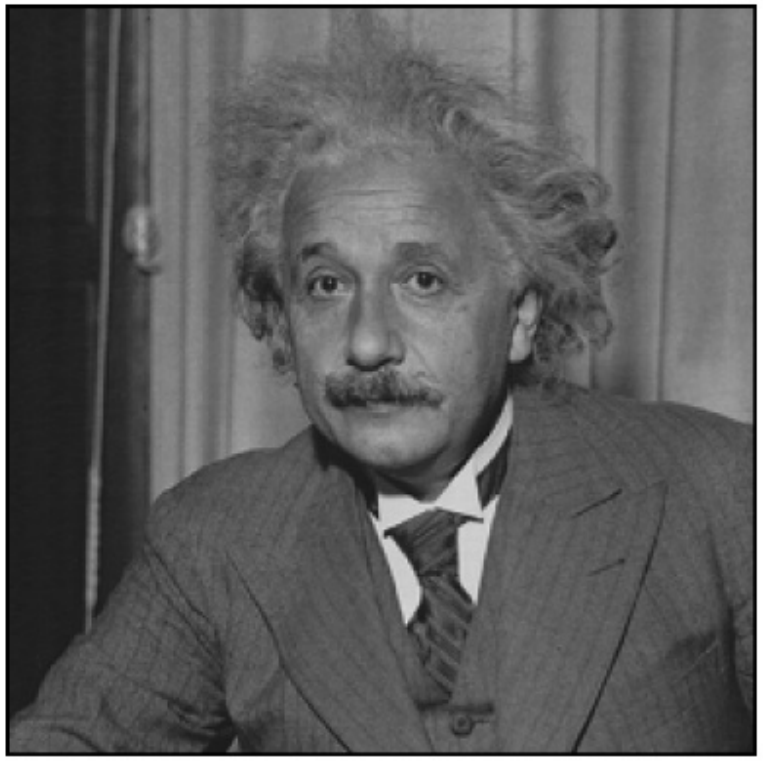
Sharpening Filter



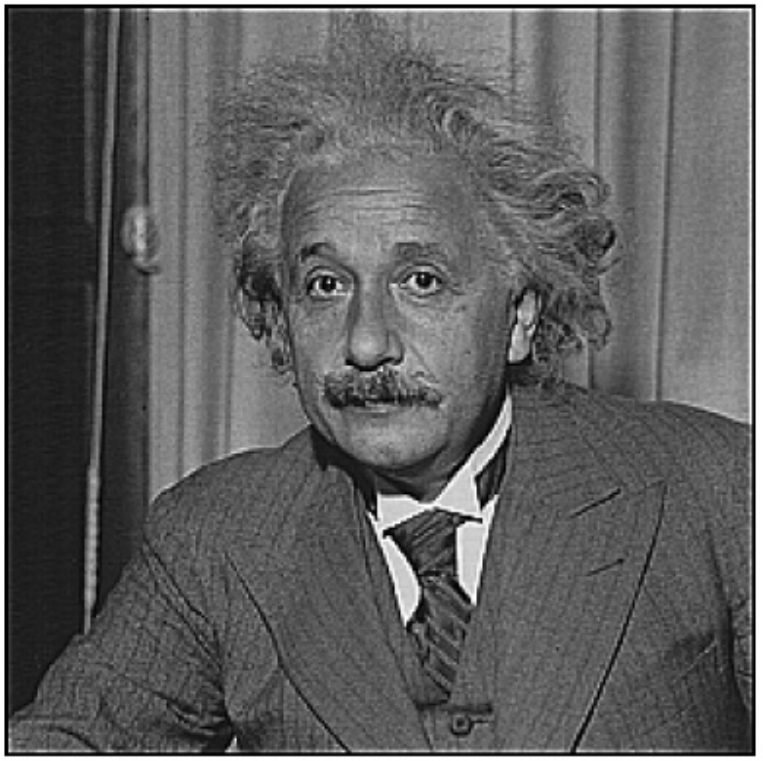
Original

Sharpening filter

Sharpening Filter



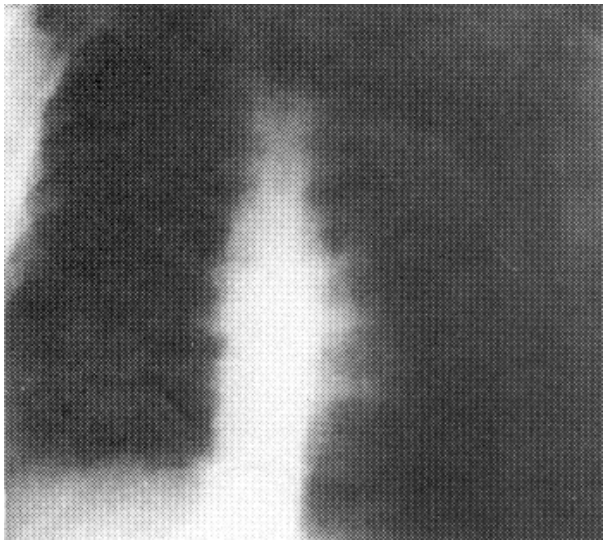
before

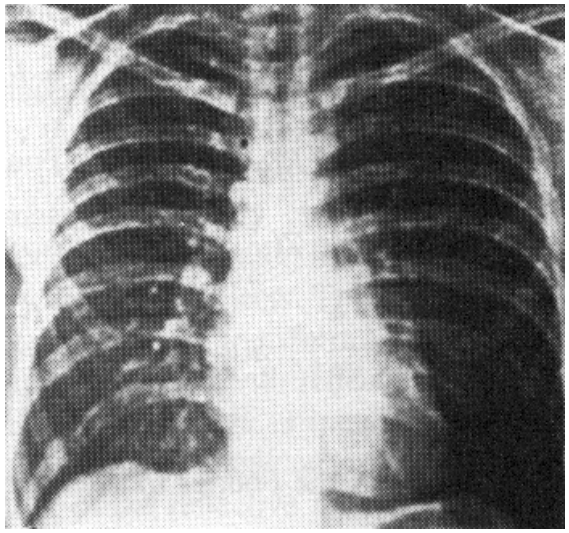


after

Application: High Frequency Emphasis

Original

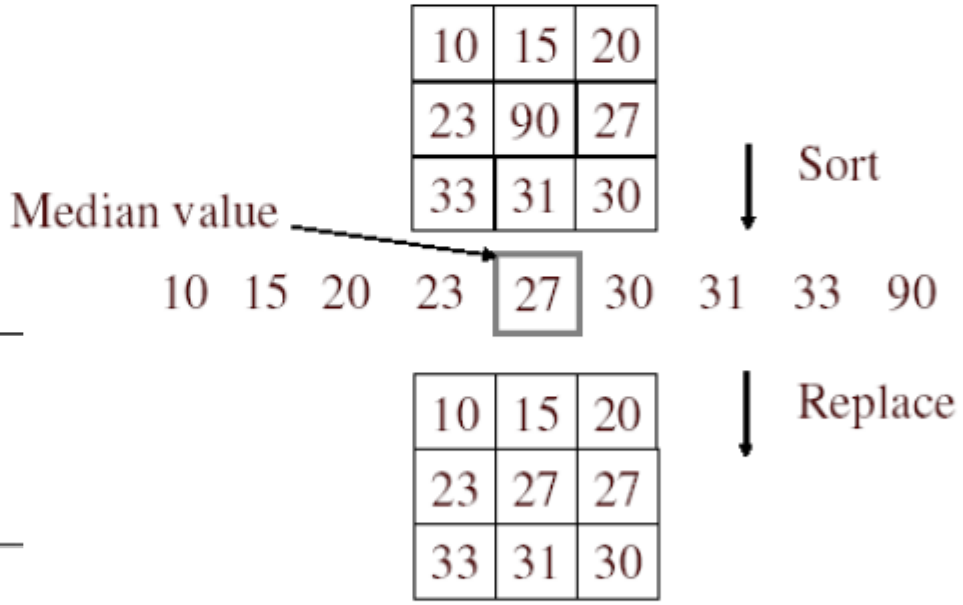




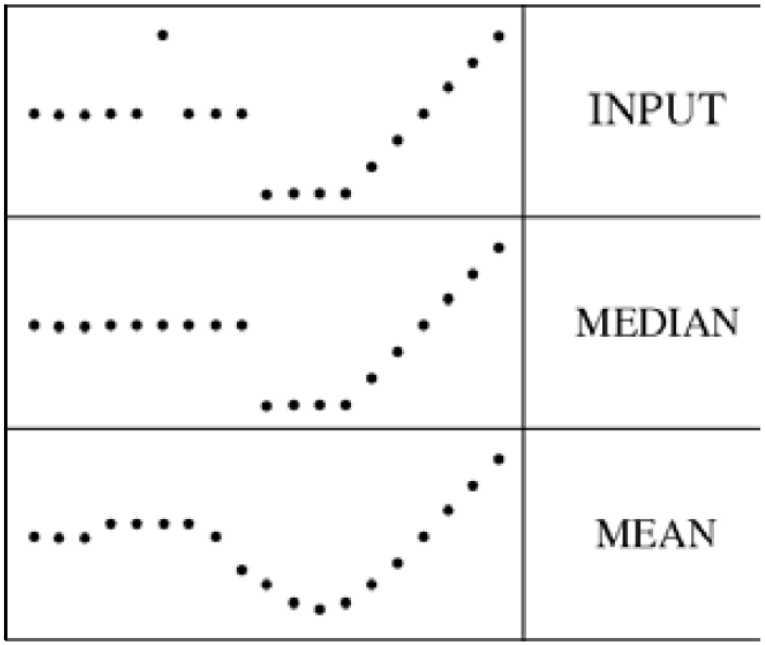
High Frequency Emphasis
+
Histogram Equalization

Non-Linear Filters: Median Filter

- Replace each pixel by the median of its neighbors.



- Comparison with mean:

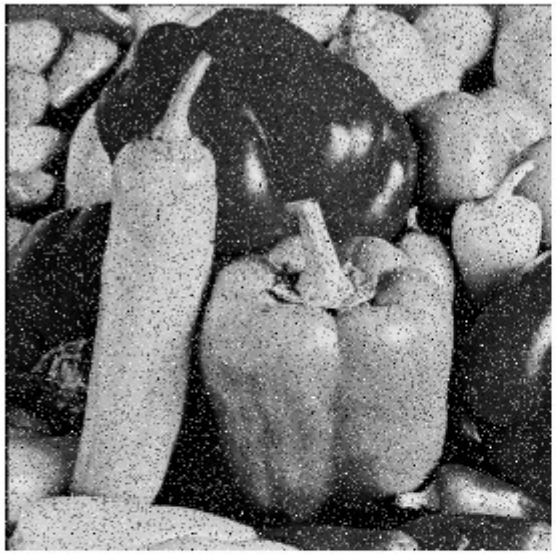


Median Filtering

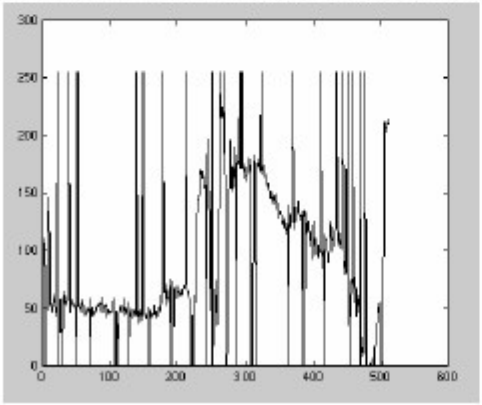
- Median filtering is a non-linear operation that is particularly effective at removing salt and pepper noise.

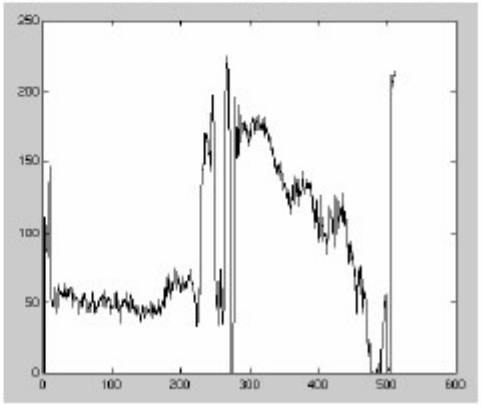
Median Filter

Salt and pepper noise



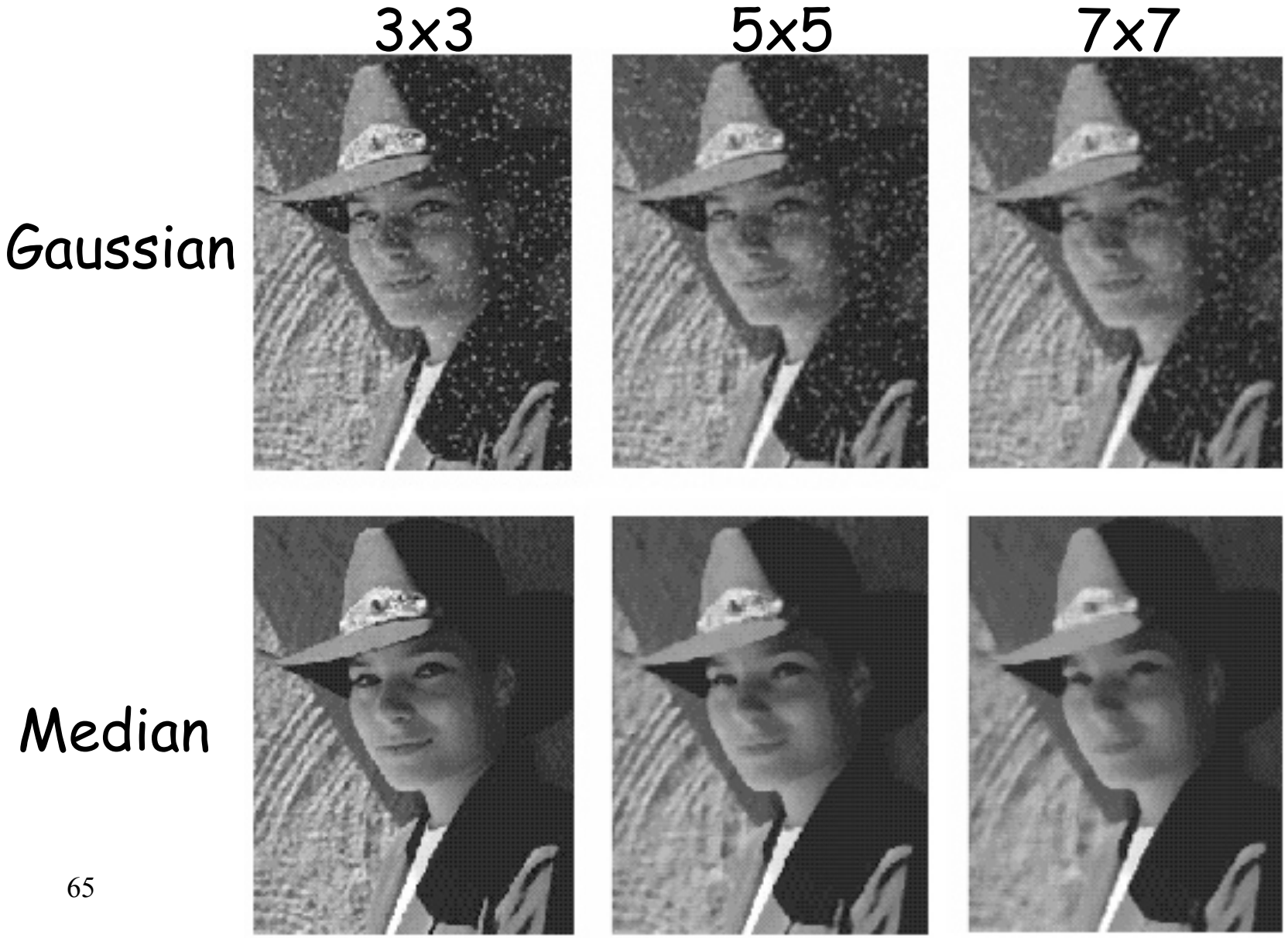
Median filtered



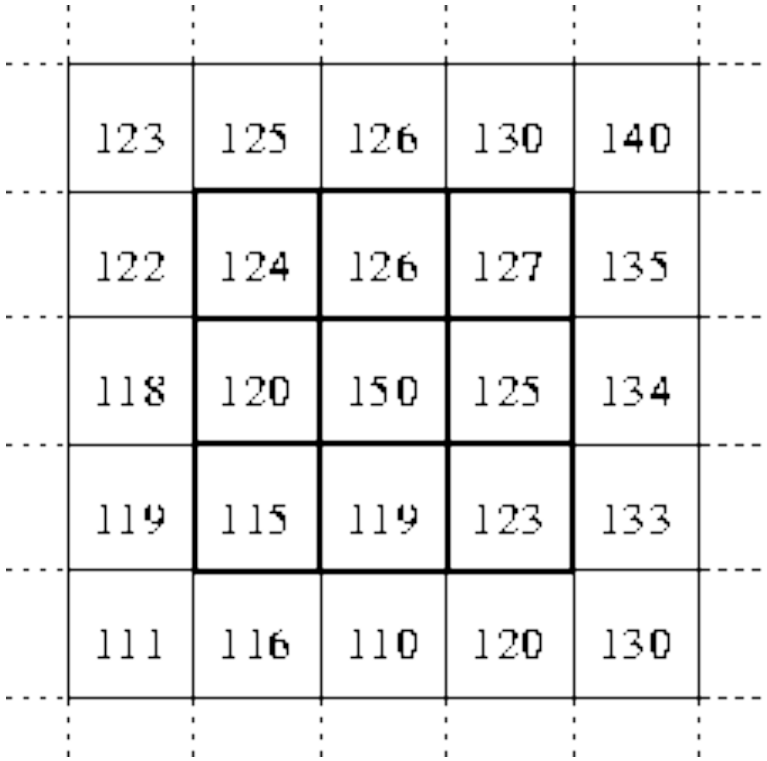


Plots of a row of the image

Median vs. Gaussian Filtering



3x3 Median Filter



Neighbourhood values:

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

Median value: 124

Example