Image Filtering

Computer Vision I

CSE 252A

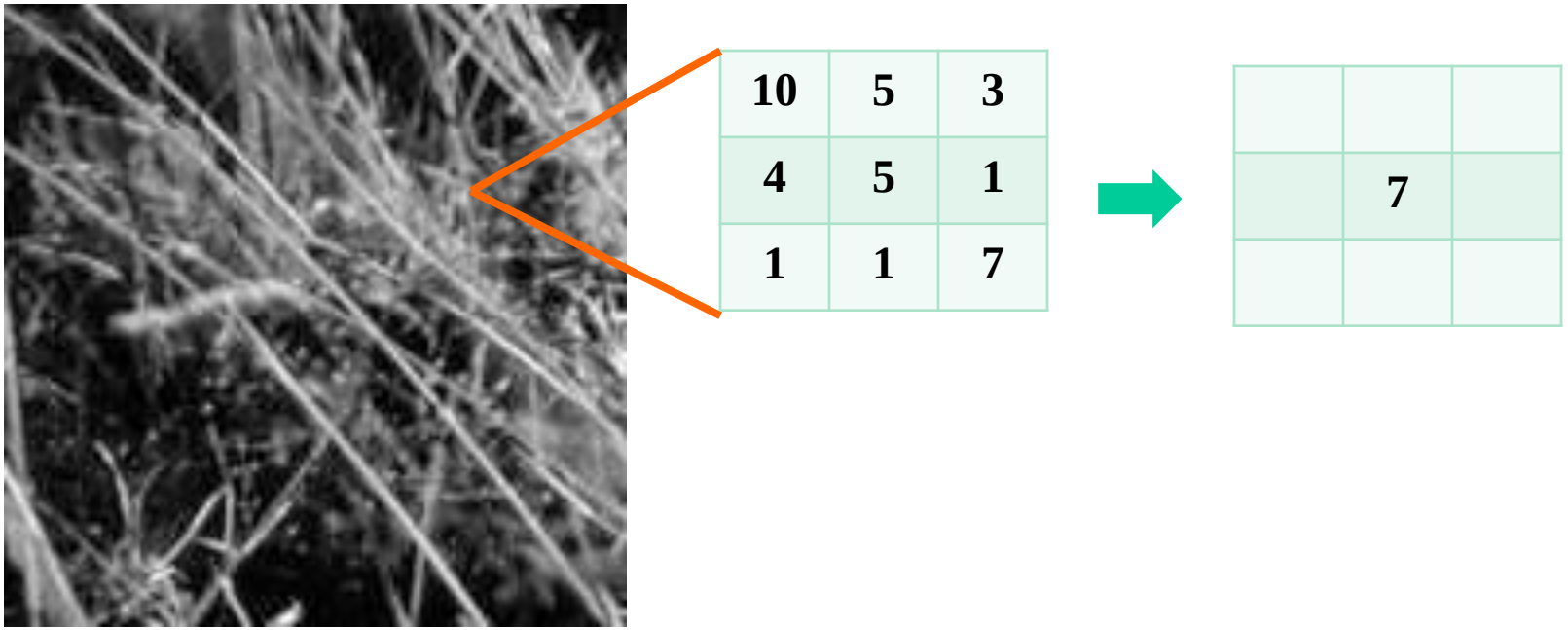
Lecture 6

Announcements

- Homework 2 is due Oct 22, 11:59 PM
- Reading:
  - Chapter 4: Linear Filters

What is image filtering?

Producing a new image where the value at a pixel in the output image is a function of a neighborhood of the pixel location in the input image.



Example: Smoothing by Averaging

Linear Filters

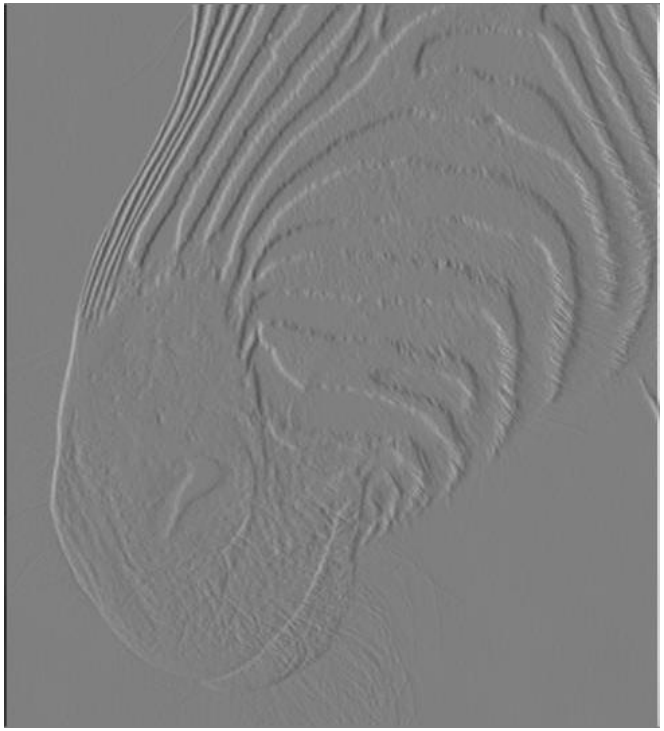
- General process:
  - Form new image whose pixels are a weighted sum of original pixel values, using the same set of weights at each point.
- Properties
  - Output is a linear function of the input
  - Output is a shift-invariant function of the input (i.e. shift the input image two pixels to the left,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

Image Filtering

Input

Output

Filter



Convolution

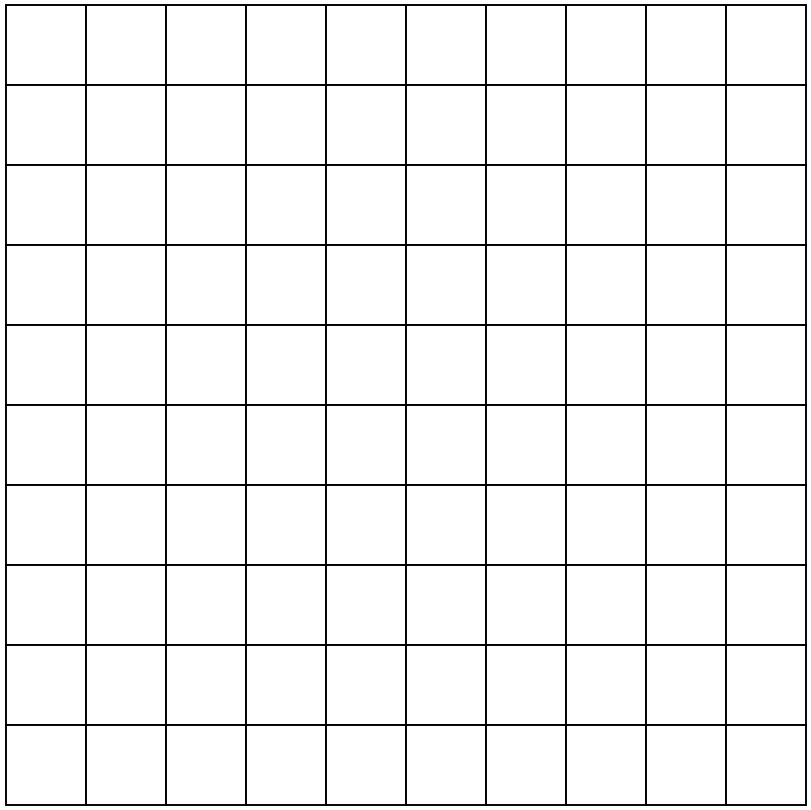


Image (I)

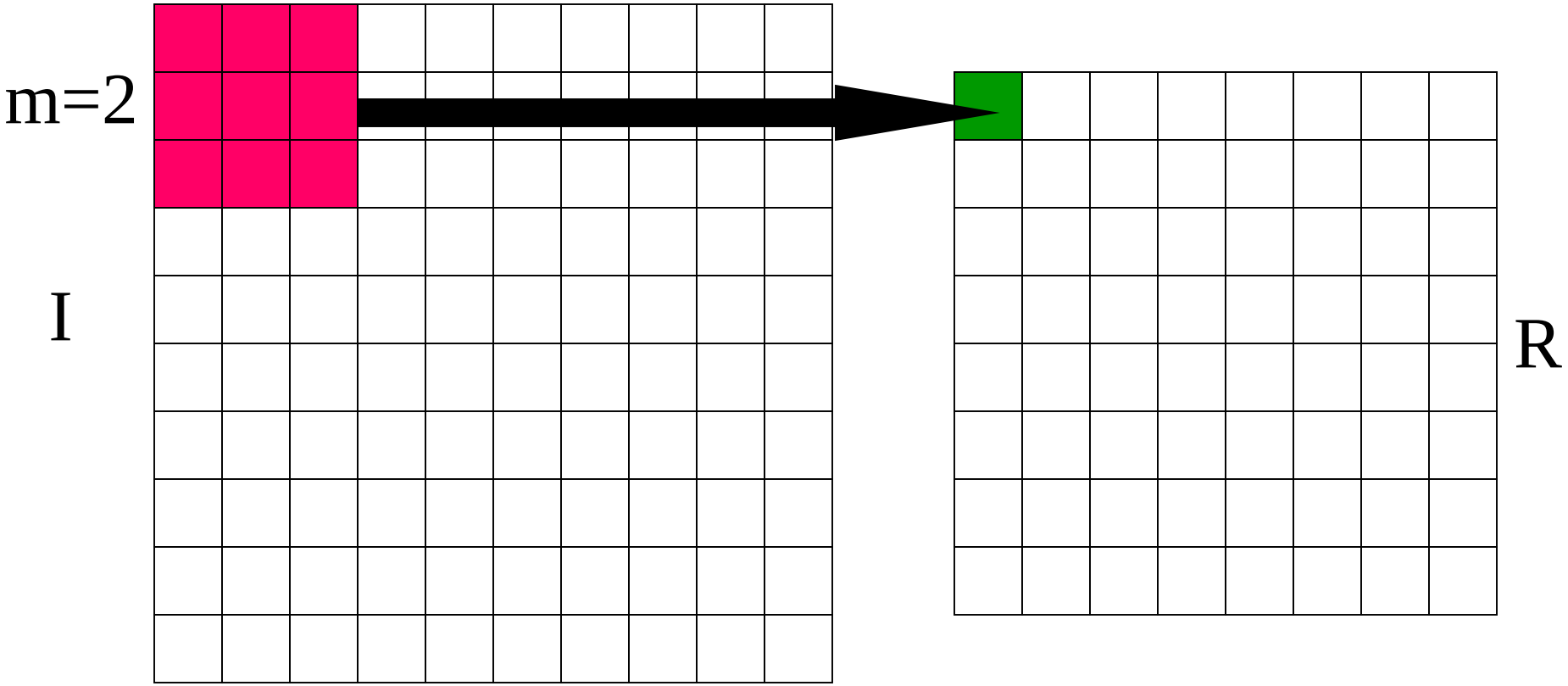
*

1	2	1
-1	-2	-1

Kernel (K)

Note: Typically Kernel is relatively small in vision applications.

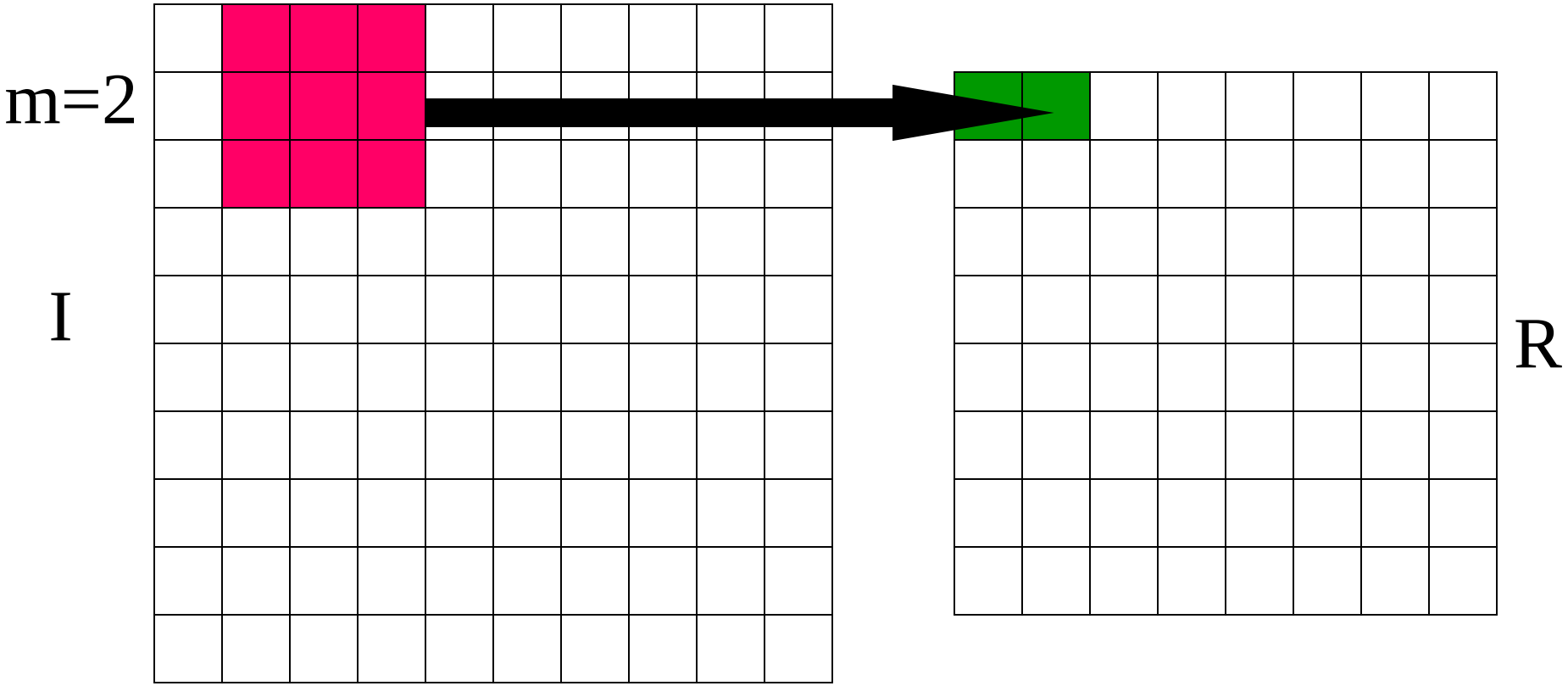
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

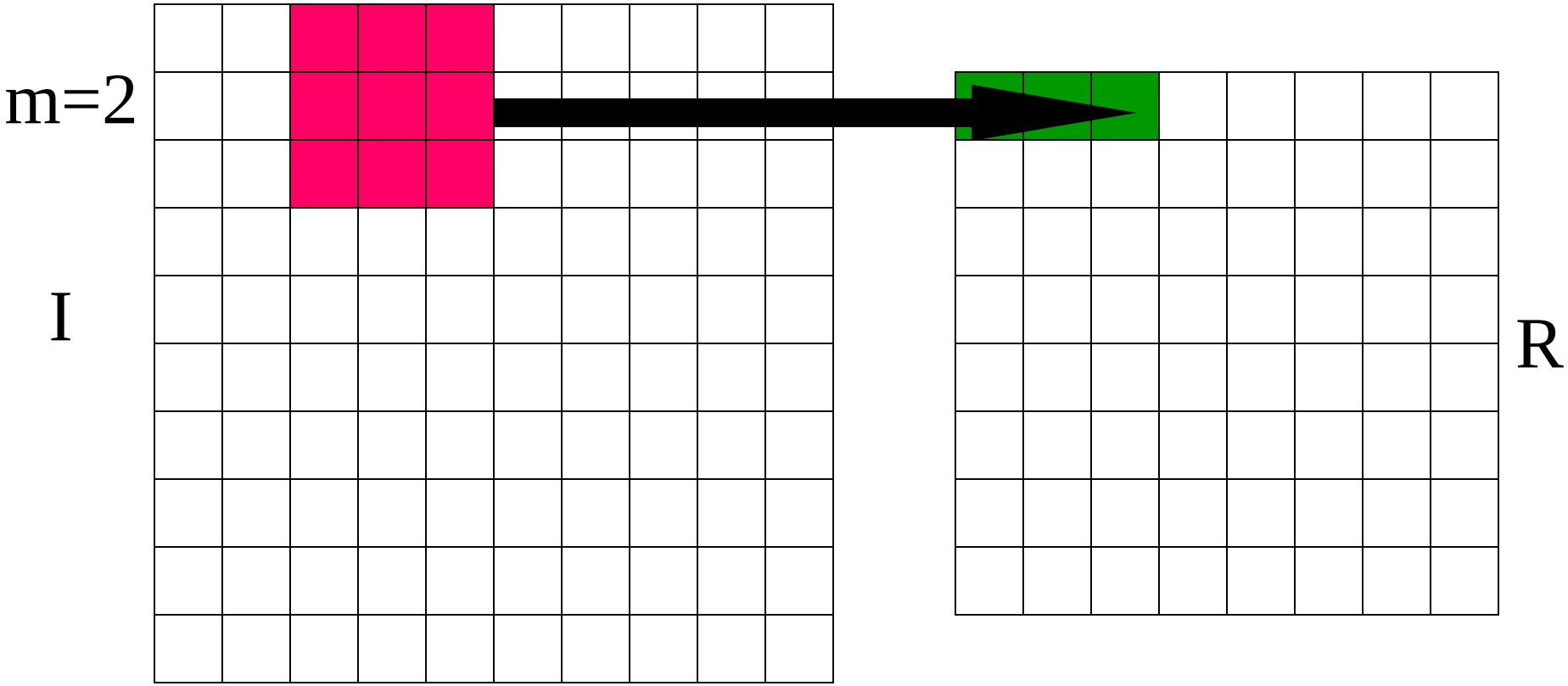
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

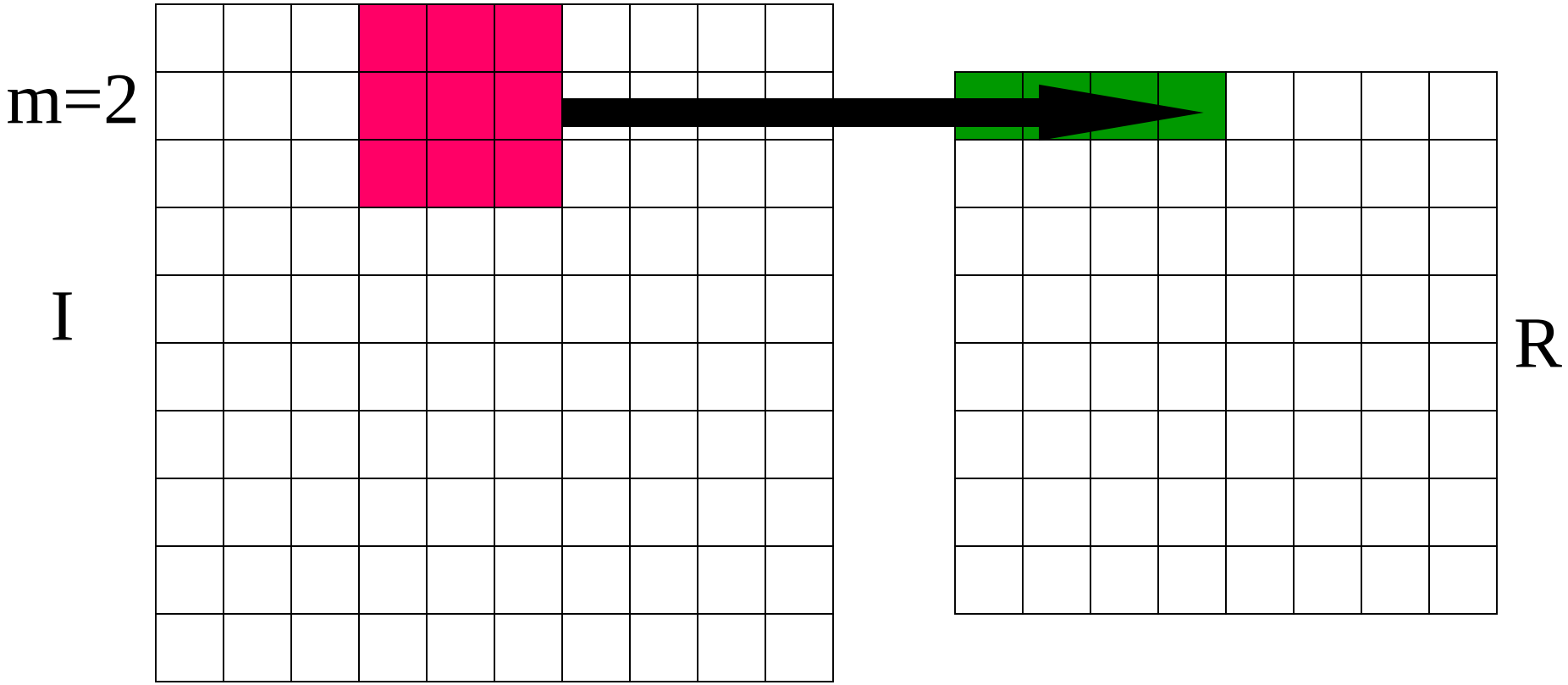
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

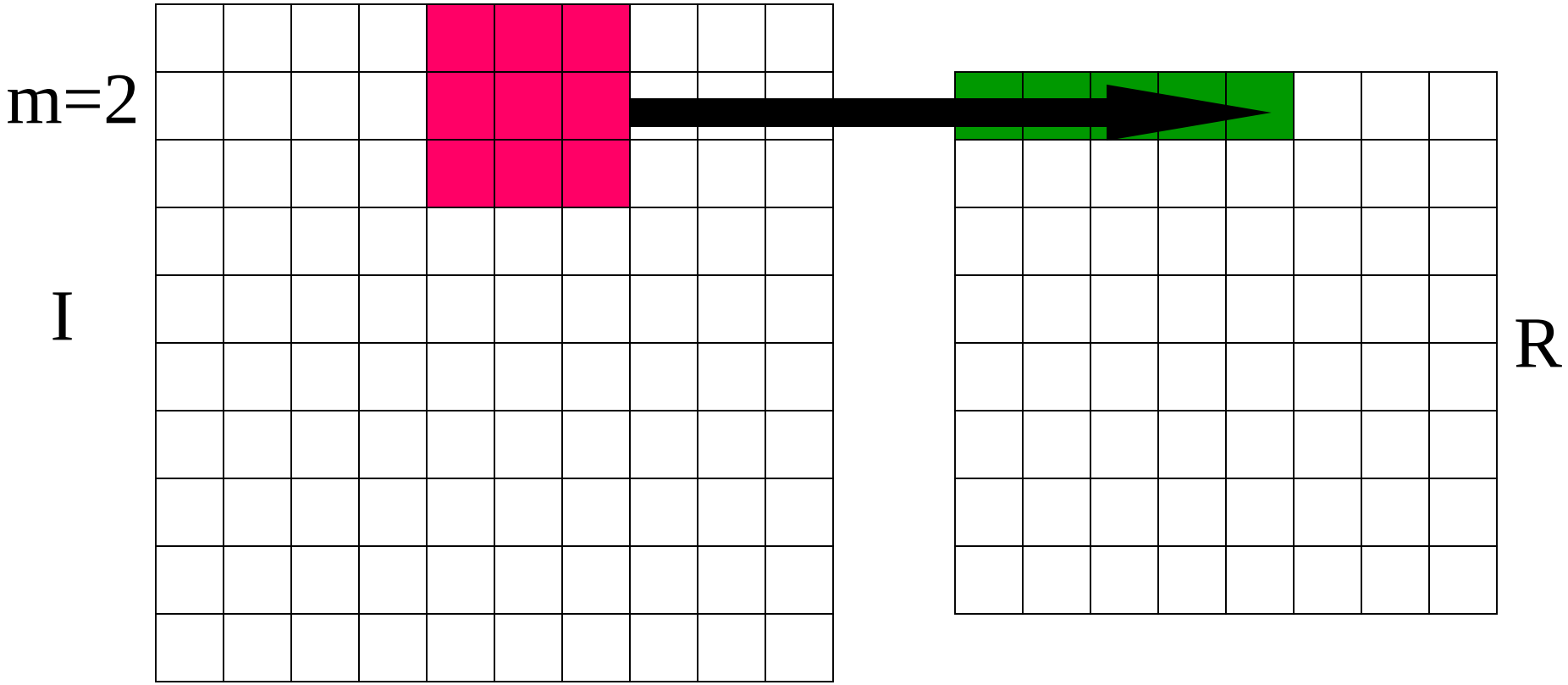
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

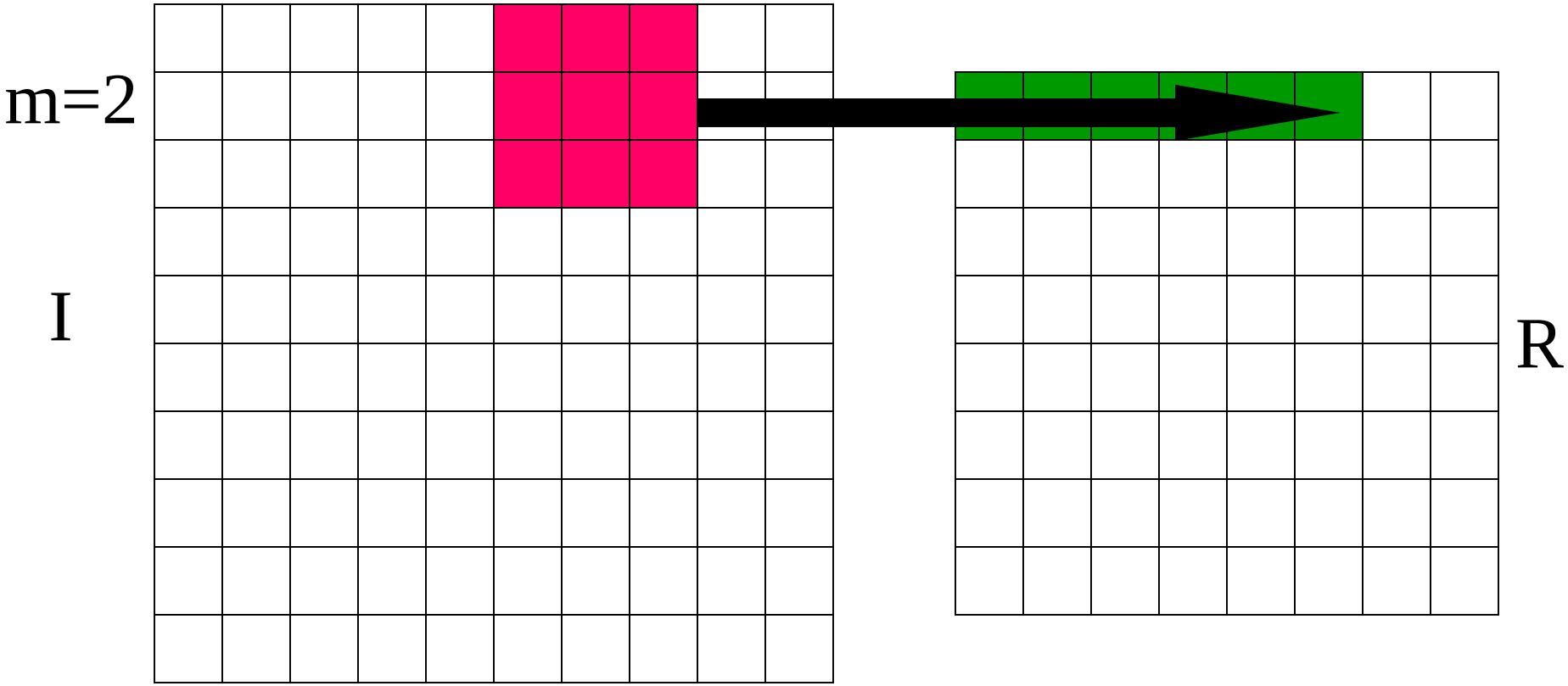
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

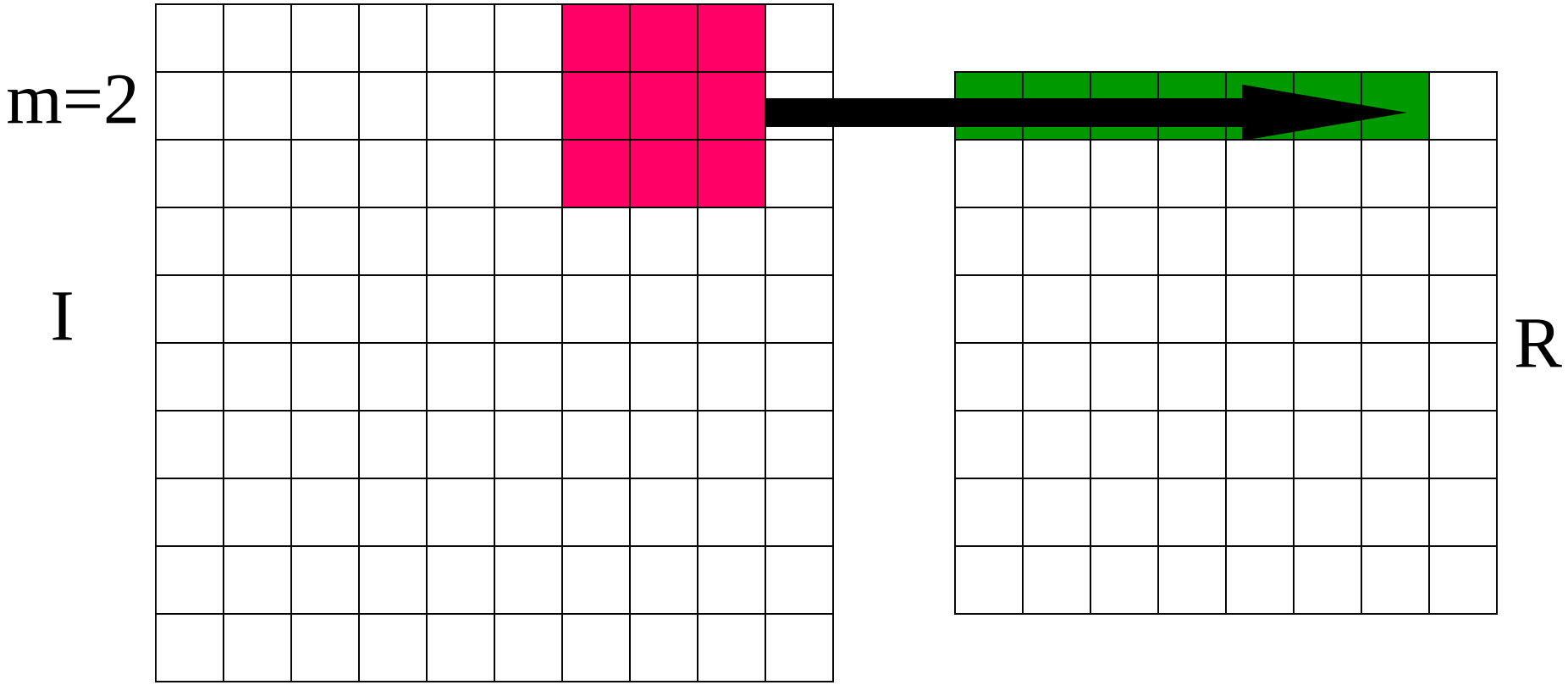
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

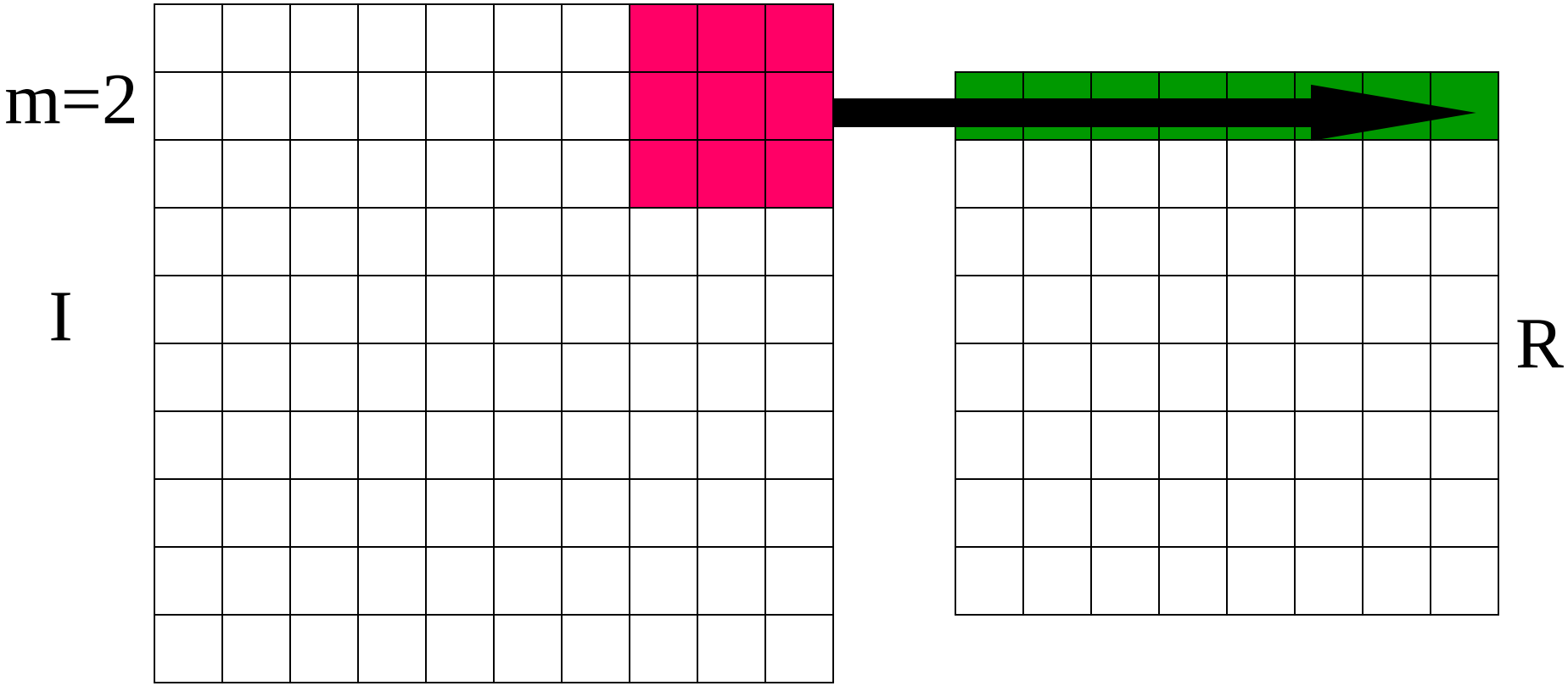
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

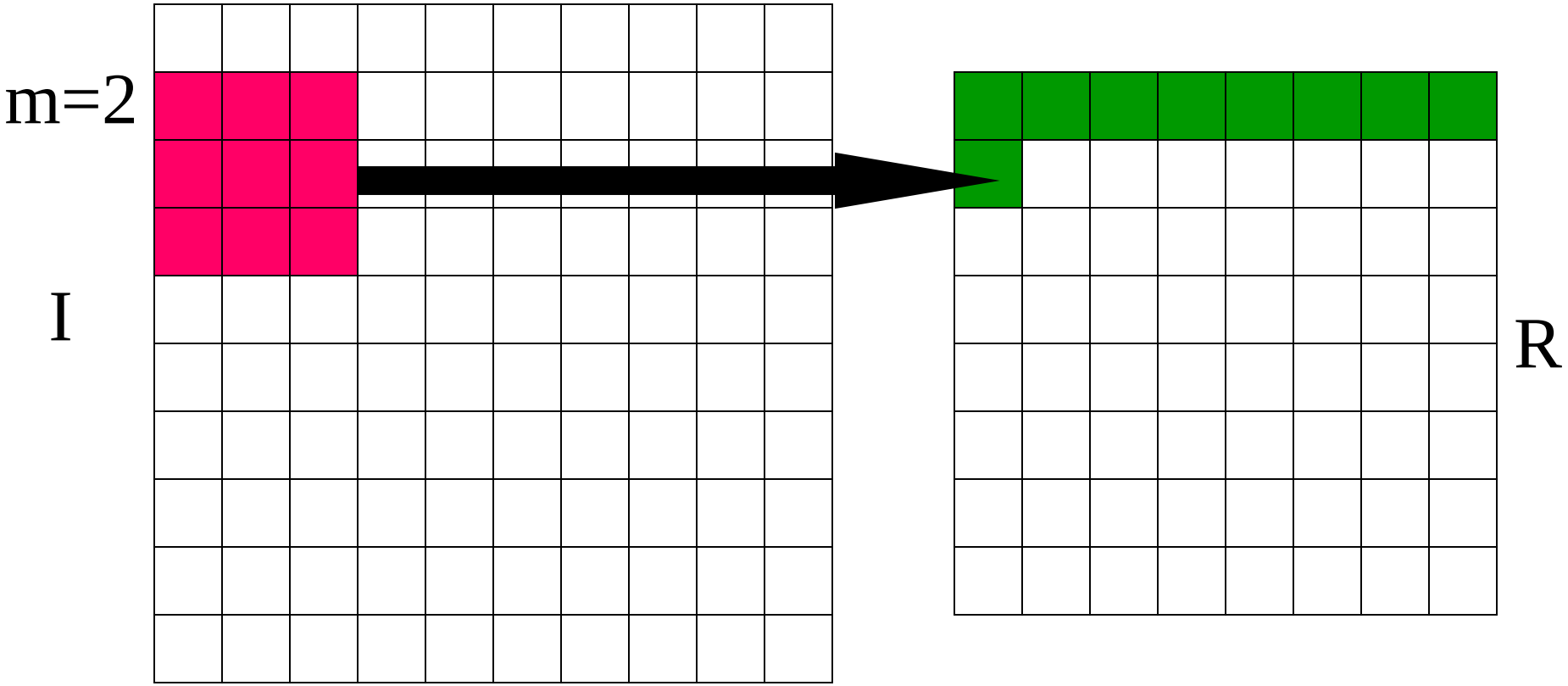
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

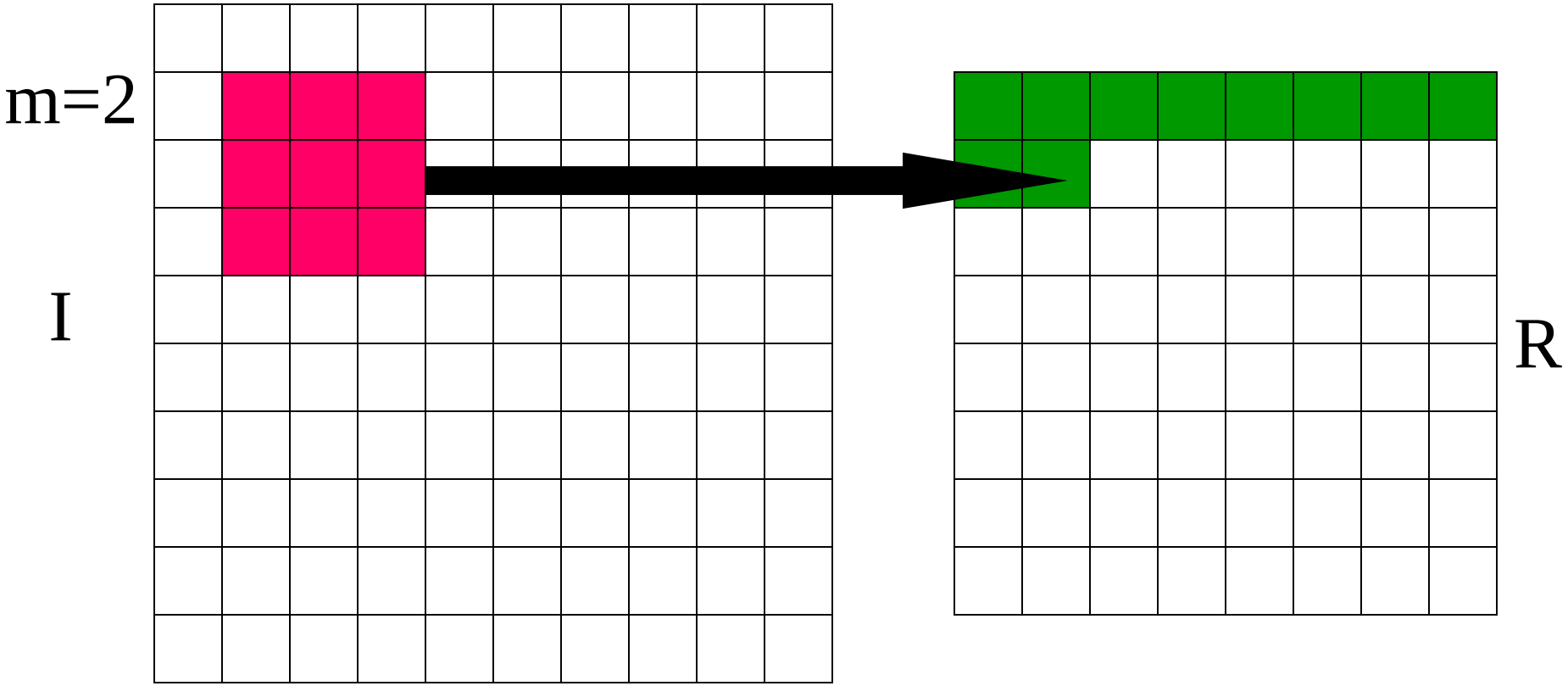
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

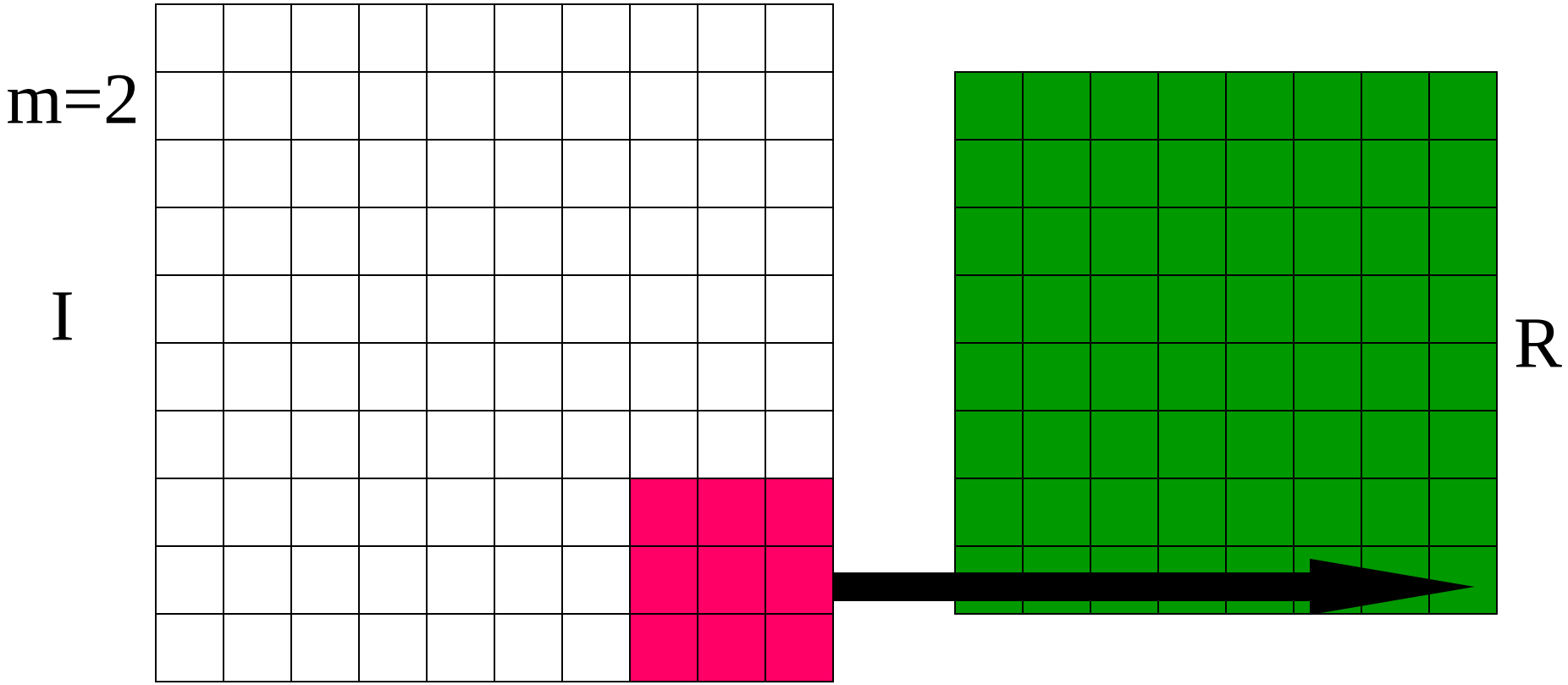
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

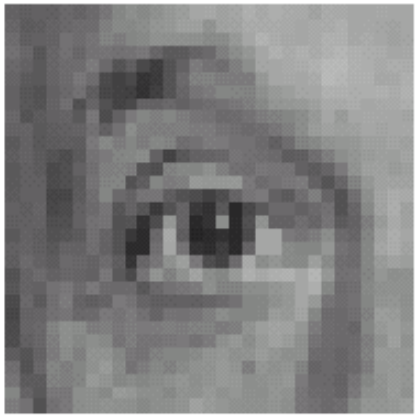
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

Linear filtering (warm-up slide)



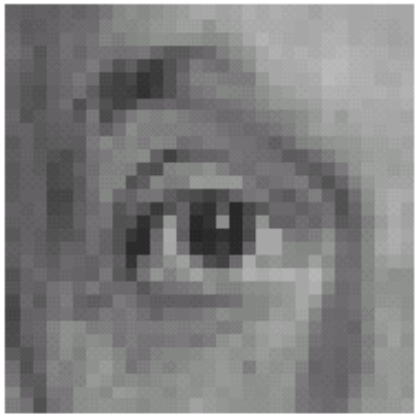
original

0	0	0
0	1	0
0	0	0

Pixel offset

?

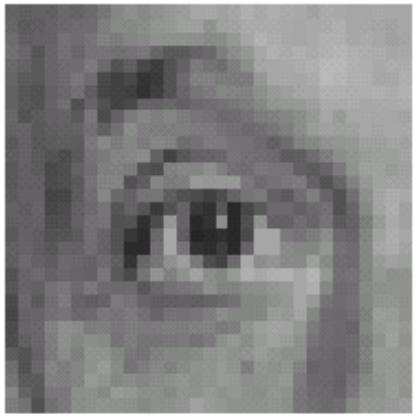
Linear filtering (warm-up slide)



original

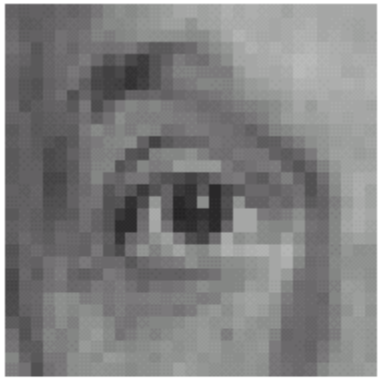
0	0	0
0	1	0
0	0	0

Pixel offset



Filtered
(no change)

Linear filtering



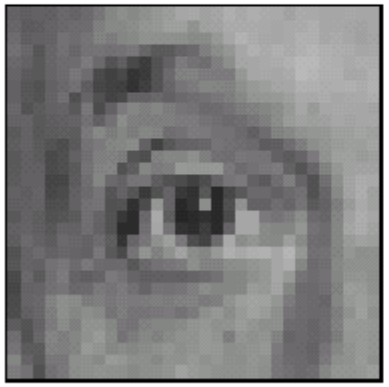
original

0	0	0
0	0	1
0	0	0

Pixel offset

?

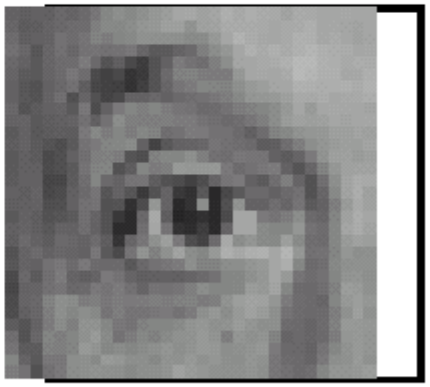
Shift



original

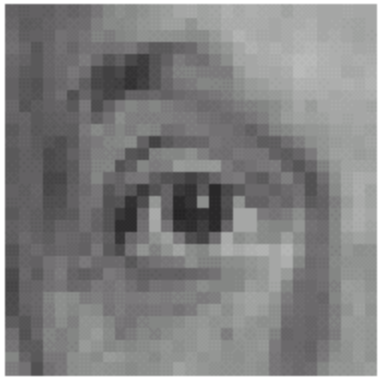
0	0	0
0	0	1
0	0	0

Pixel offset



Shifted one
Pixel to the left

Linear filtering

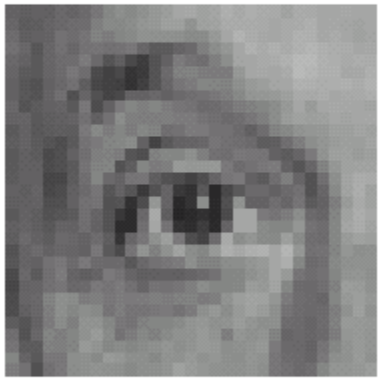


original

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

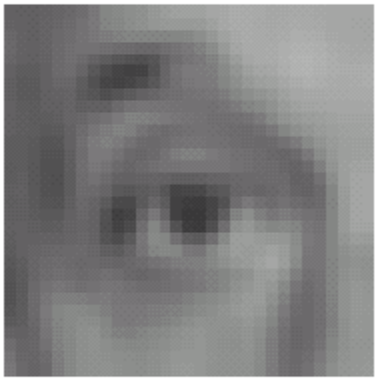
?

Blurring



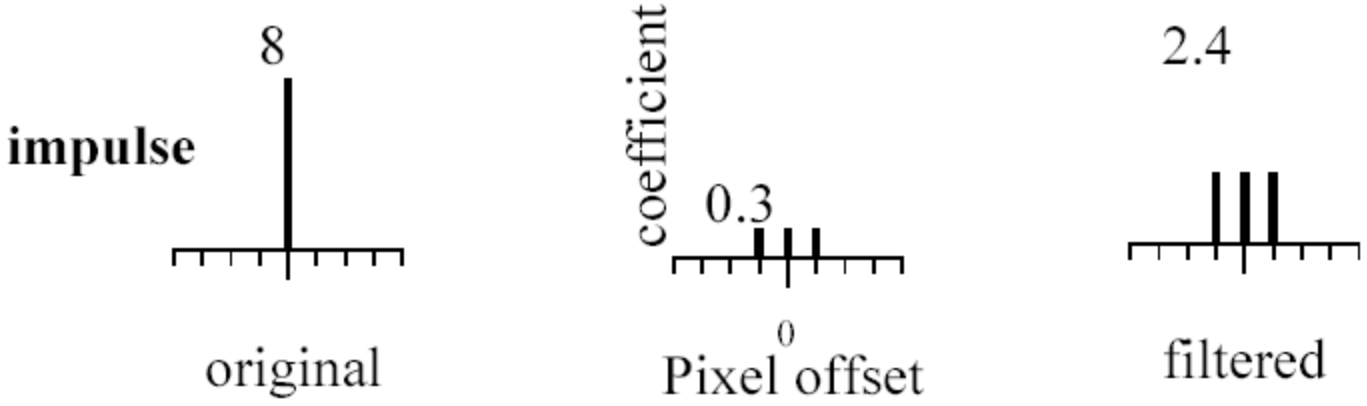
original

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

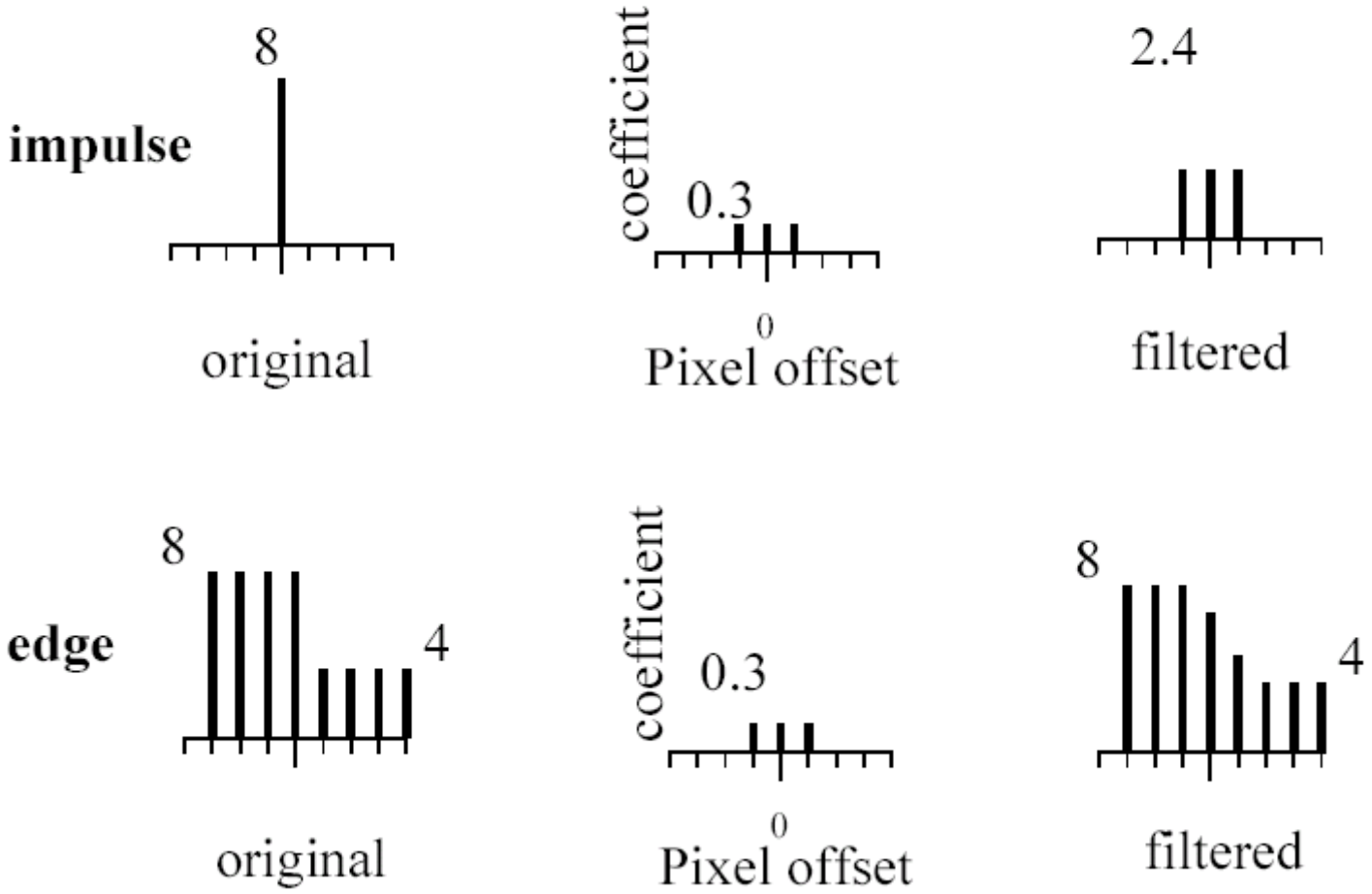


Blurred (filter applied in both dimensions).

Blur Examples
1-Dimensional



Blur Examples
1-Dimensional



Practice with linear filters

Original

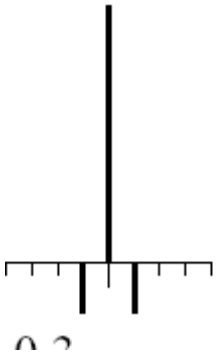
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad ?$$

Source: D. Lowe

Practice with linear filters

$$\left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{array}\right] - \frac{1}{9}\begin{array}{|ccc|}\hline 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ \hline \end{array}$$

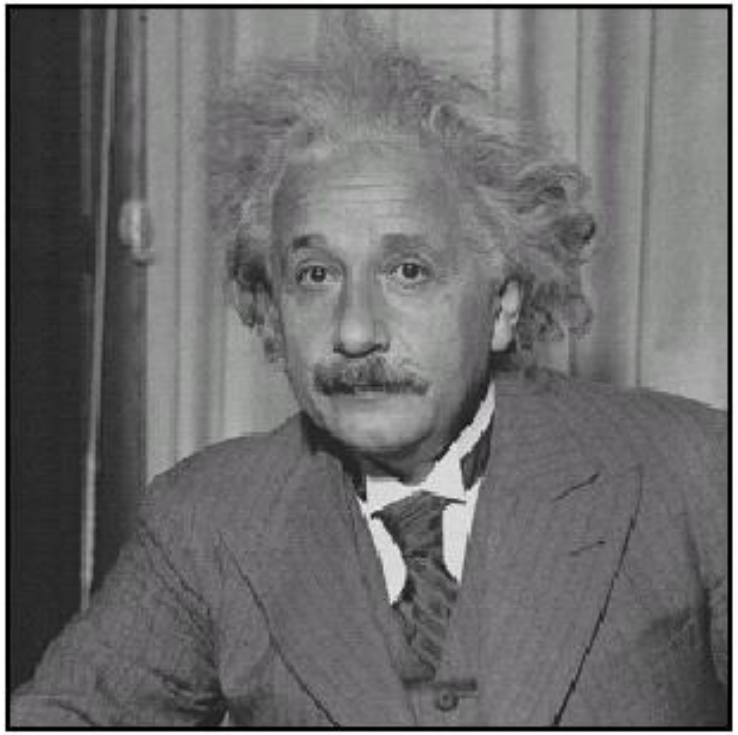
Original



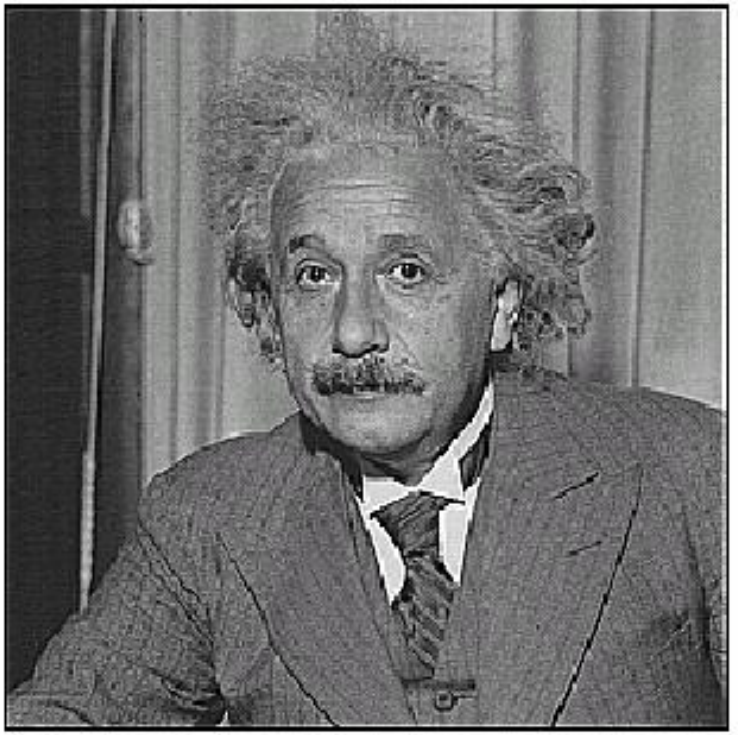
Sharpening filter
- Accentuates differences with local average

Source: D. Lowe

Sharpening



before

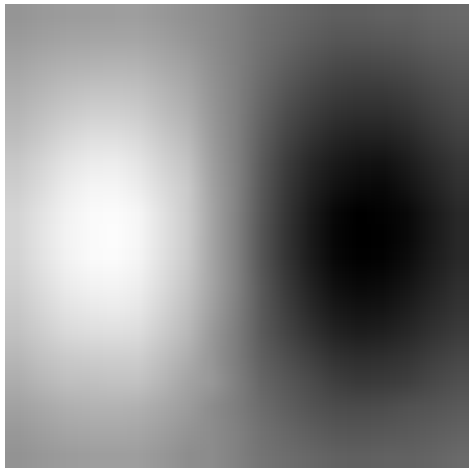


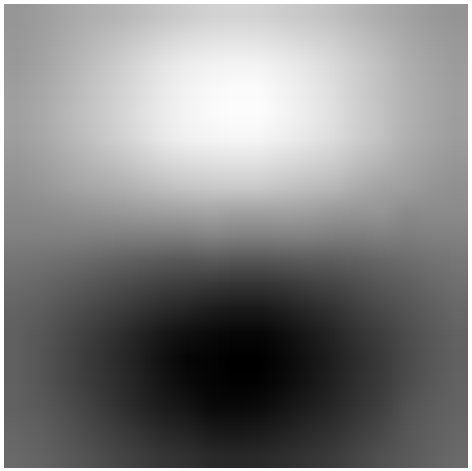
after

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





Convolutional Neural Networks

- Core operation is, not surprisingly, convolution.
- Can be extended to 3D – e.g.,
  - image and R,G,B as channels (N x N x 3)
  - Volumetric data such as MRI, CT
- During training of a CNN, the weights of the convolution kernels are learned.

Properties of Continuous Convolution

(Holds for discrete too)

Let f,g,h be images and * denote convolution

$$f * g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-u, y-v)g(u,v)dudv$$

- Commutative: f*g=g*f
- Associative: f*(g*h)=(f*g)*h
- Linear: for scalars a & b and images f,g,h

  (af+bg)*h=a(f*h)+b(g*h)
- Differentiation rule

$$\frac{\partial}{\partial x}(f * g) = \frac{\partial f}{\partial x} * g = f * \frac{\partial g}{\partial x}$$

Filtering to reduce noise

- Noise is what we are not interested in.
  - We will discuss simple, low-level noise today: Light fluctuations; Sensor noise; Quantization effects; Finite precision
  - Not complex: shadows; extraneous objects.
- A pixel's neighborhood contains information about its intensity.
- Averaging noise reduces its effect.

Additive noise

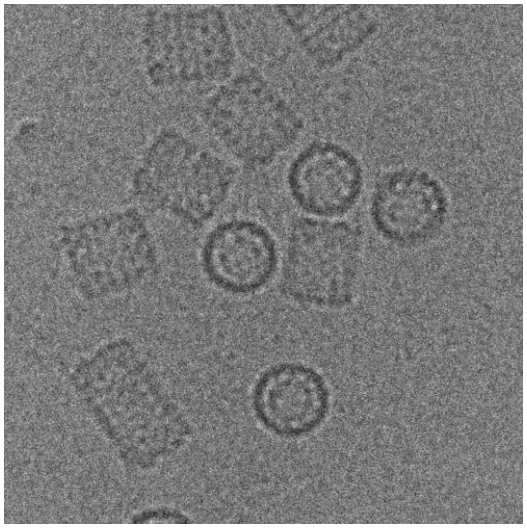
- I = S + N. Noise doesn't depend on signal.
- We'll consider:

$I_i = s_i + n_i$ with $E(n_i) = 0$

s_i deterministic. n_i a random var.

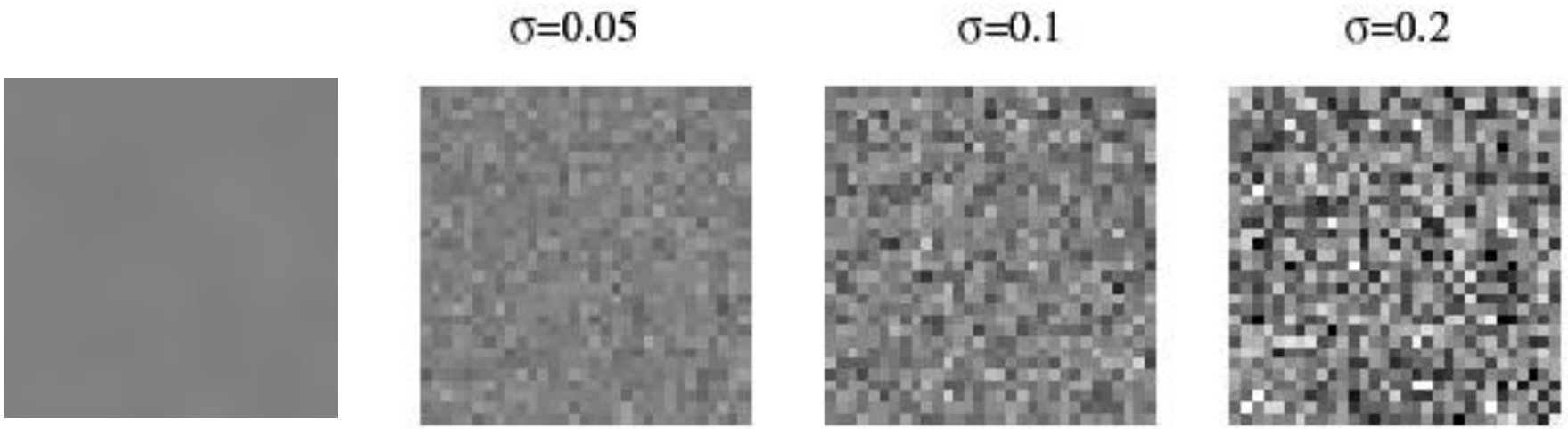
n_i, n_j independent for $i \neq j$

n_i, n_j identically distributed



- Gaussian noise, n_i drawn from Gaussian.

Gaussian noise



Gaussian noise, n_i drawn from Gaussian distribution with zero mean and standard deviation σ

Gaussian Noise:
sigma=1

Gaussian Noise:
sigma=16

Averaging Filter

- Convolution Kernel with positive entries, that sum 1.
- Average of the neighborhood.
- For efficiency, 8 adds and one multiply.
- If all weights are equal, it is called a Box filter.

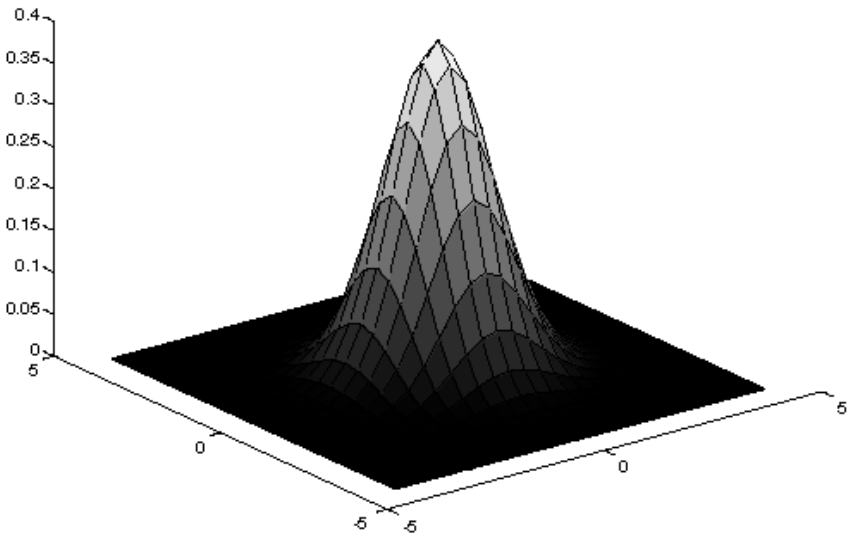
F

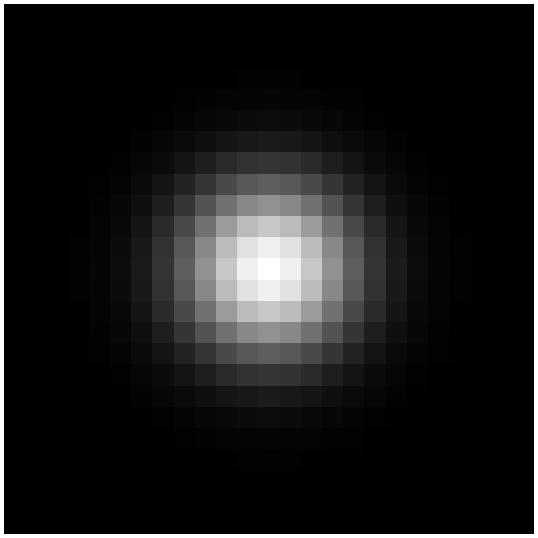
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Smoothing by Averaging

Kernel: 

An Isotropic Gaussian





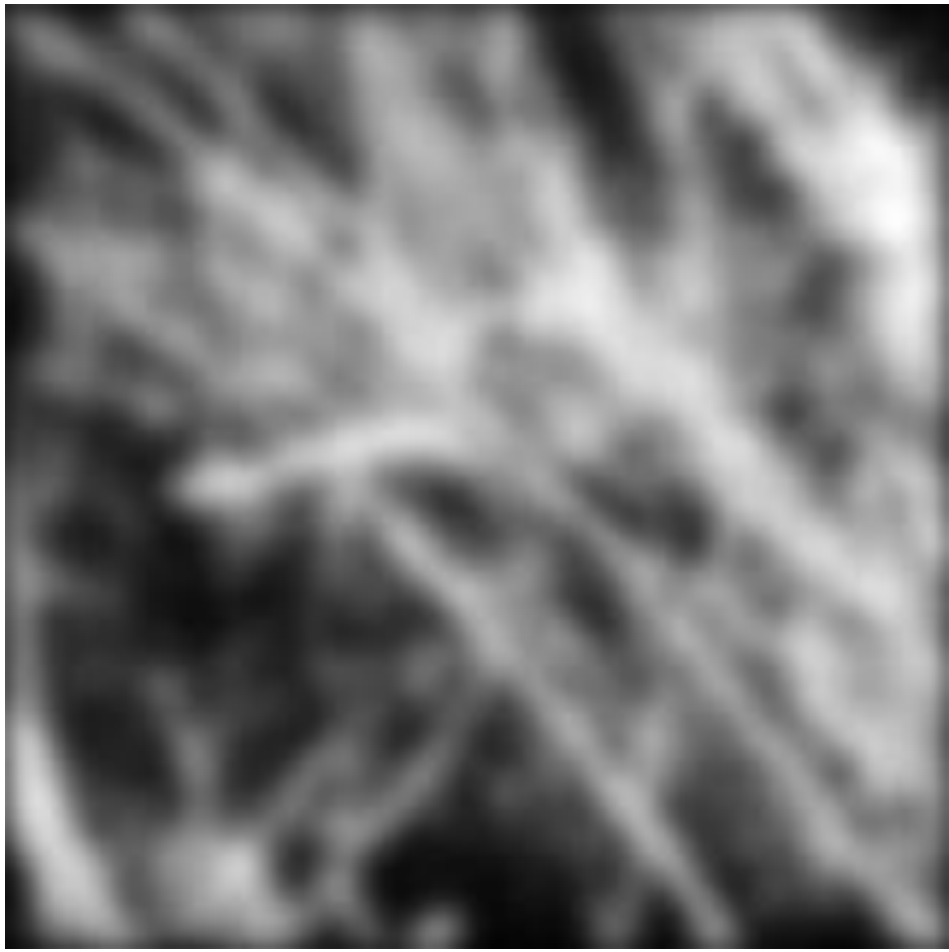
- Smoothing kernel proportional to

$$e^{-\frac{x^2+y^2}{2s^2}}$$

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

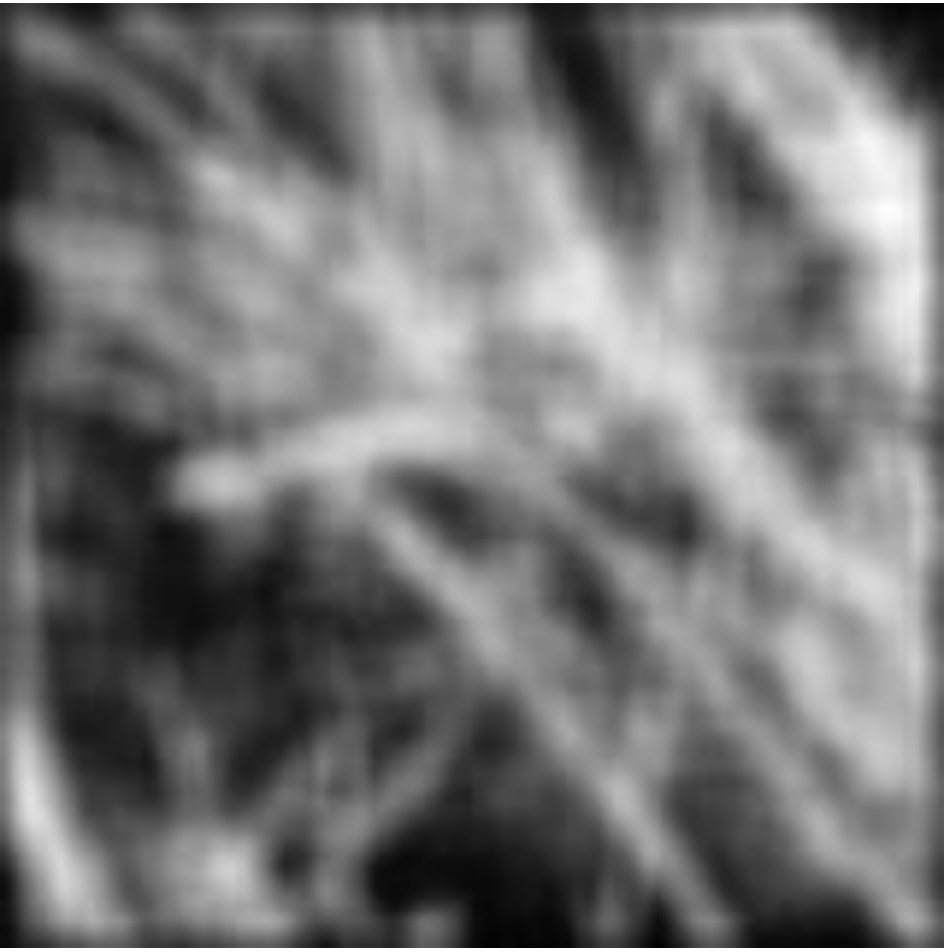
Smoothing with a Gaussian

Kernel: 

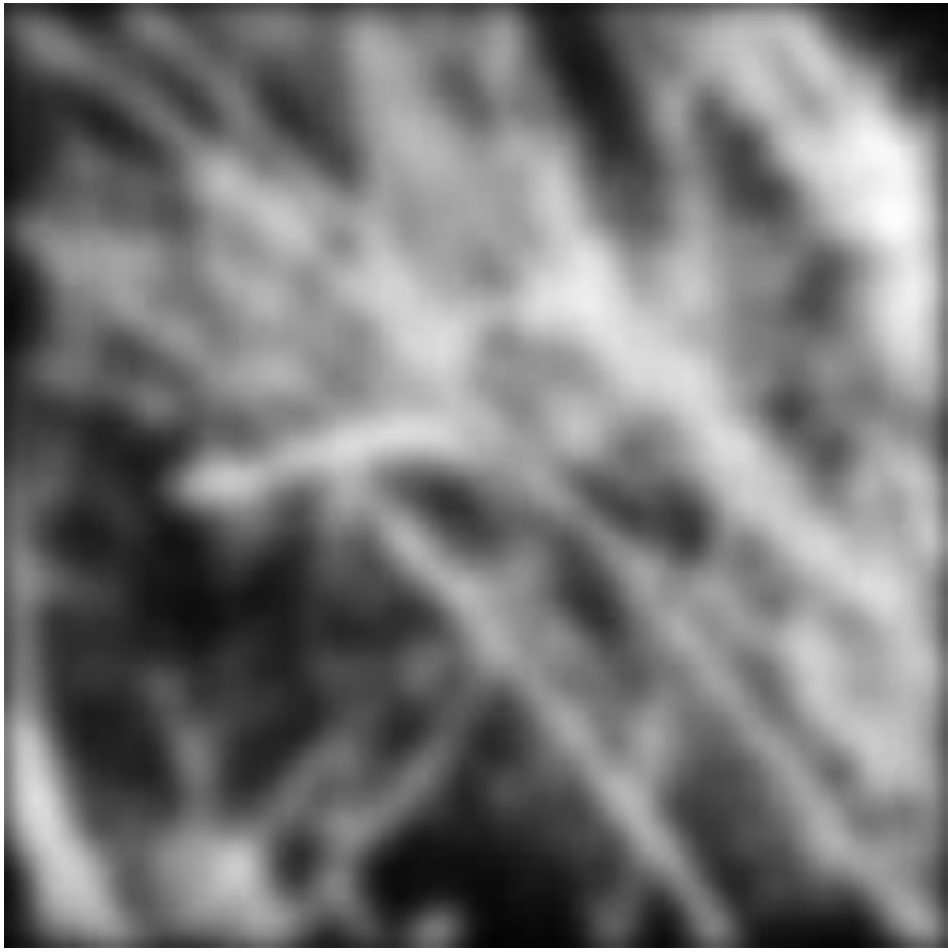


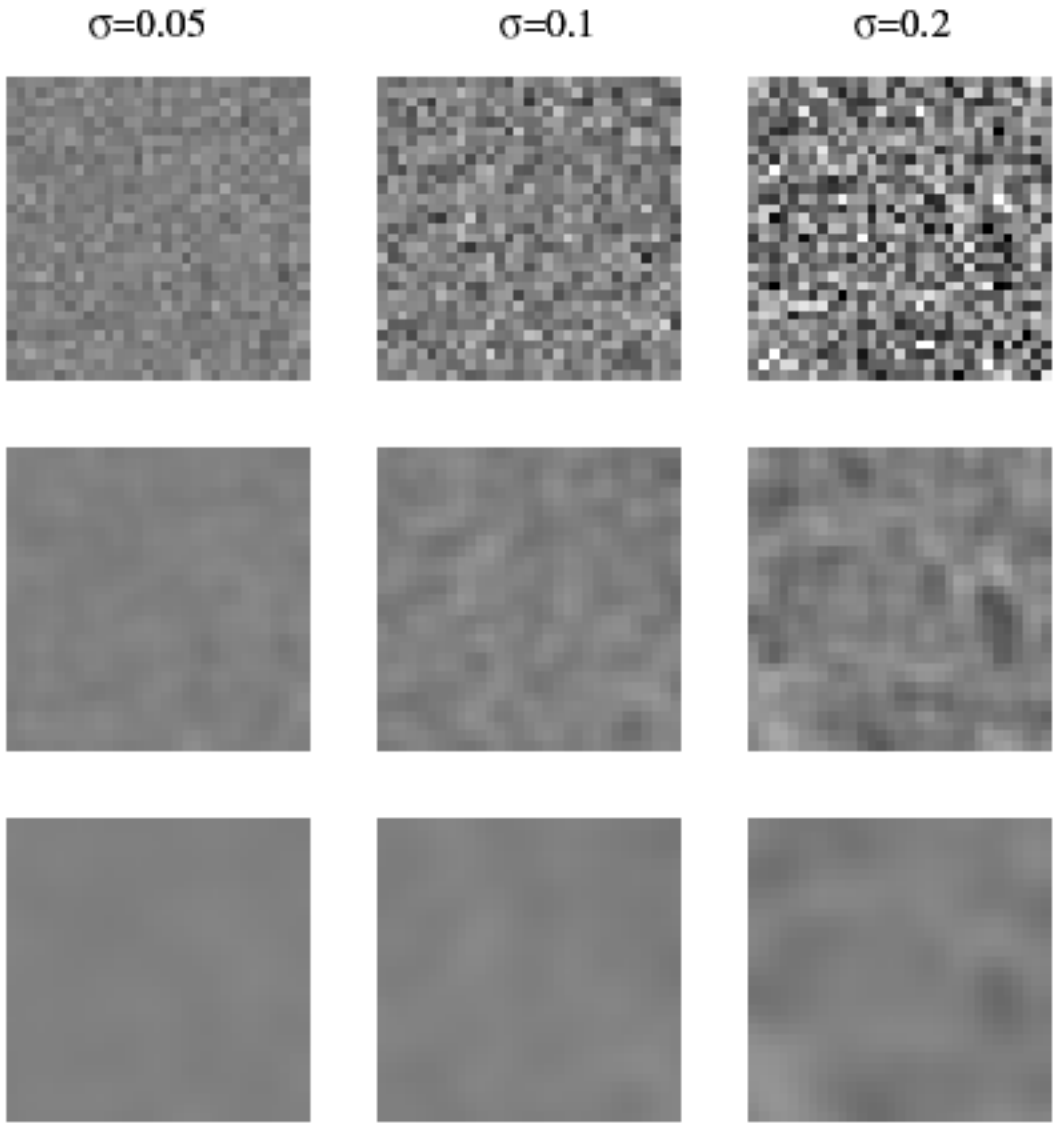
Smoothing

Box filter



Gaussian filter



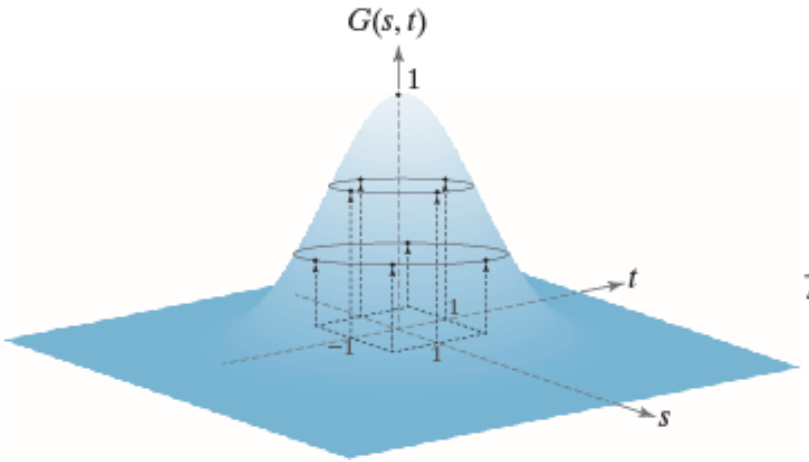


The effects of smoothing
Each row shows smoothing with gaussians of different width; each column shows different realizations of an image of gaussian noise.

Smoothing with Gaussian kernel

a b

FIGURE 3.41
(a) Sampling a Gaussian function to obtain a discrete Gaussian kernel. The values shown are for $K = 1$ and $\sigma = 1$. (b) Resulting 3×3 kernel [this is the same as Fig. 3.37(b)].



$$\frac{1}{4.8976} \times \begin{bmatrix} 0.3679 & 0.6065 & 0.3679 \\ 0.6065 & 1.0000 & 0.6065 \\ 0.3679 & 0.6065 & 0.3679 \end{bmatrix}$$

Standard deviation σ	Percent of total volume under surface
1	39.35
2	86.47
3	98.89

Volume under surface greater than 3σ is negligible

Smoothing with Gaussian kernel

σ = 7
43x43

σ = 7
85x85

Difference

Smoothing with Gaussian kernel

Input image

σ = 3.5
21x21

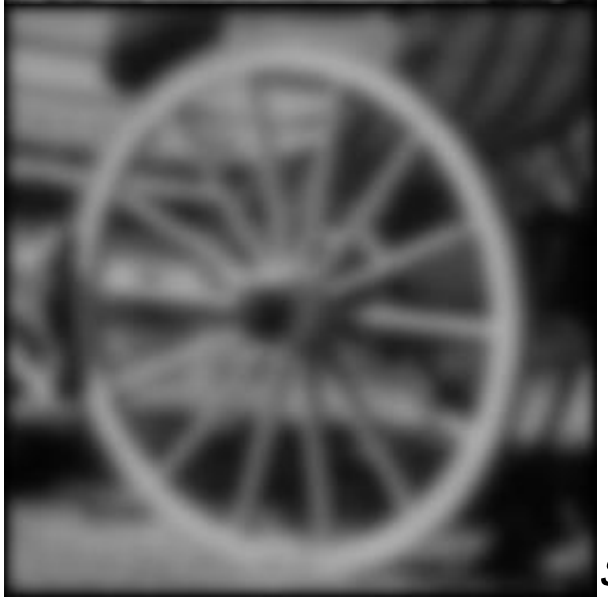
σ = 7
43x43

Gaussian Smoothing

original

$s = 2$

$s = 2.8$



$s = 4$

Gaussian Smoothing

by Charles Allen Gillbert

by Harmon & Julesz

http://www.michaelbach.de/ot/cog_blureffects/index.html

Gaussian Smoothing

http://www.michaelbach.de/ot/cog_blureffects/index.html

Halftone image (binary)

Gaussian Blurred

Efficient Implementations

Both, the Box filter and the Gaussian filter are separable:

- First convolve each row with a 1-D filter
- Then convolve each column with a 1-D filter.

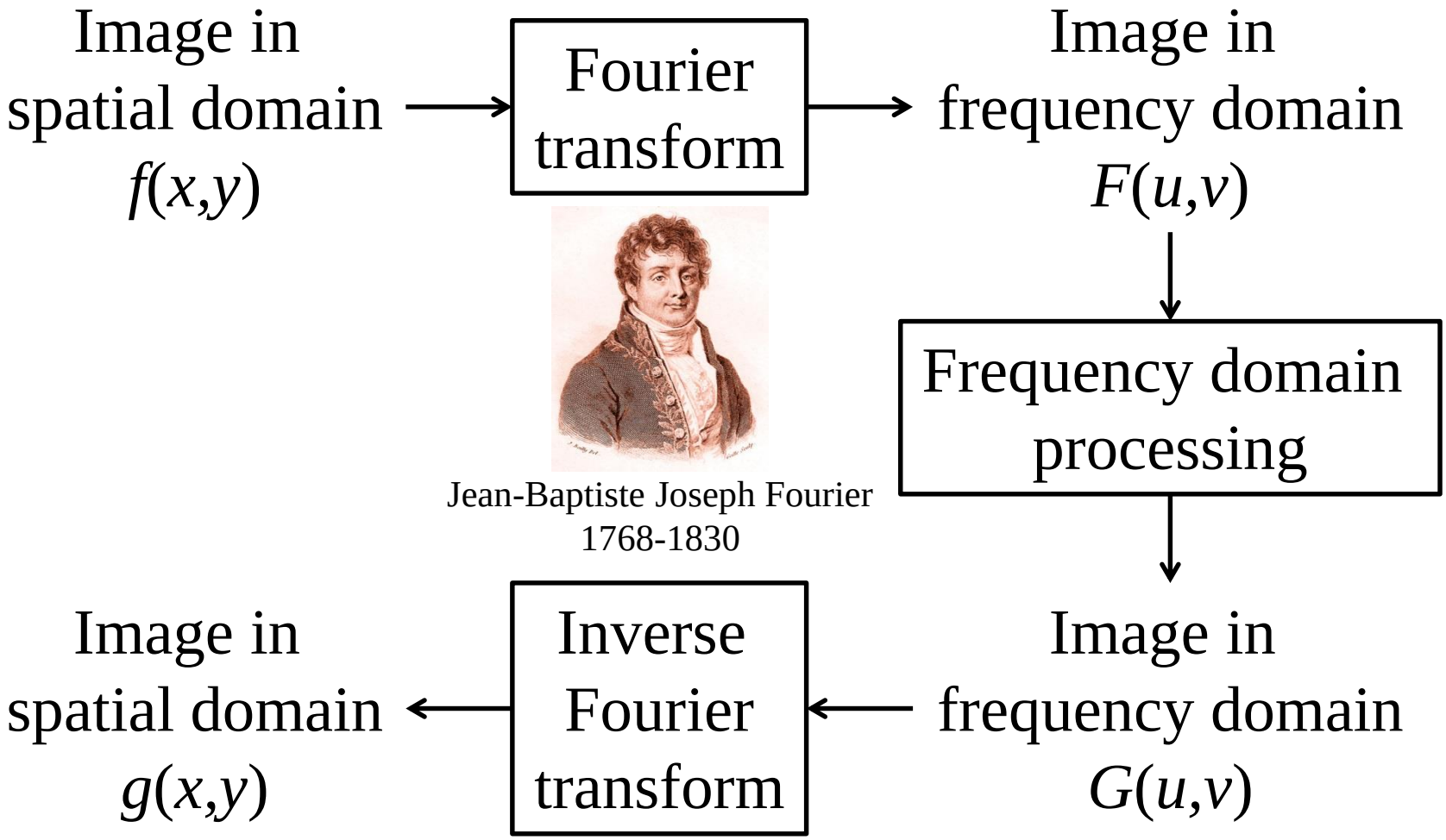
$$K_r = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$$K_c = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

For Gaussian kernels $g_1(x)$ and $g_2(x)$,

- If g_1 & g_2 respectively have variance σ_1^2 & σ_2^2
- Then $g_1 * g_2$ has variance $\sigma_1^2 + \sigma_2^2$

Overview: Image processing in the frequency domain



Jean-Baptiste Joseph Fourier
1768-1830

Fourier Transform

- 1-D transform (signal processing)
- 2-D transform (image processing)
- Consider 1-D

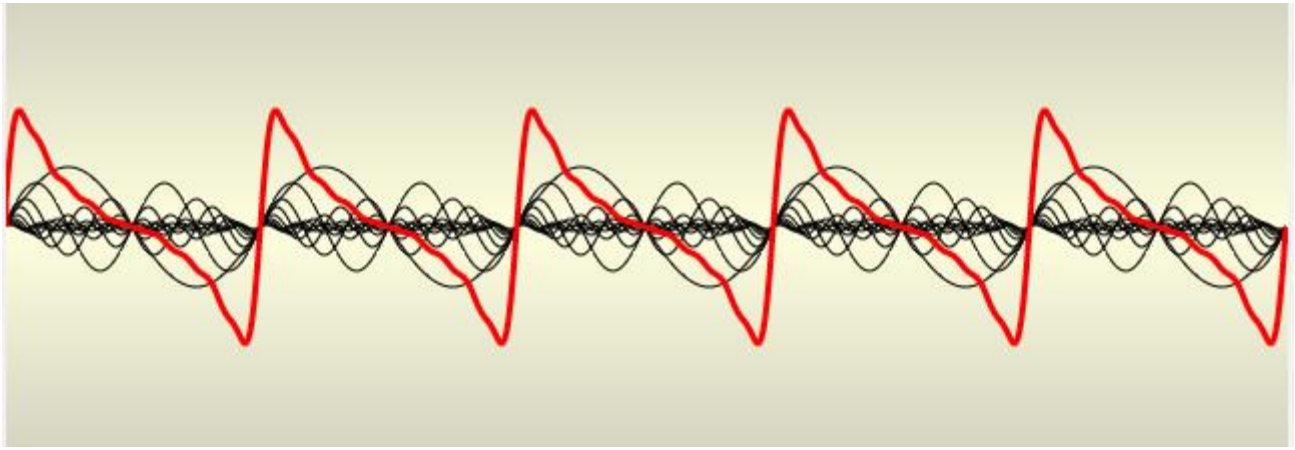
Time domain $\leftrightarrow$ Frequency Domain

Real $\leftrightarrow$ Complex

- Consider time domain signal to be expressed as weighted sum of sinusoid. A sinusoid $\cos(ut+\phi)$ is characterized by its phase ϕ and its frequency u
- The Fourier transform of the signal is a function giving the weights (and phase) as a function of frequency u.

Fourier Transform

- 1D example
  - Sawtooth wave
    - Combination of harmonics



Fourier Transform

Discrete Fourier Transform (DFT) of I[x,y]

$$F[u,v] \equiv \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I[x,y] e^{\frac{-2\pi j}{N}(xu+yv)}$$

Inverse DFT

$$I[x,y] \equiv \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F[u,v] e^{\frac{+2\pi j}{N}(ux+vy)}$$

x,y: spatial domain

u,v: frequence domain

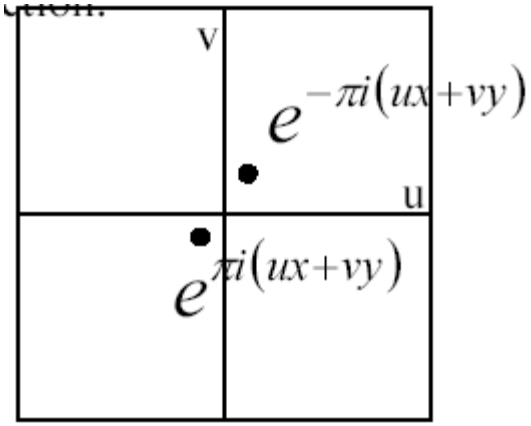
Implemented via the "Fast Fourier Transform" algorithm (FFT)

Fourier basis element

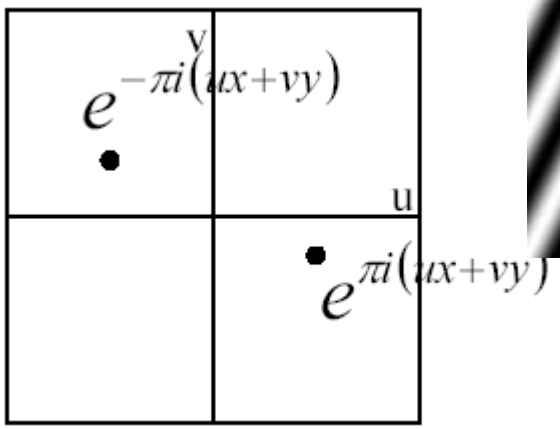
$$e^{-i2\pi(ux+vy)}$$

Transform is sum of orthogonal basis functions

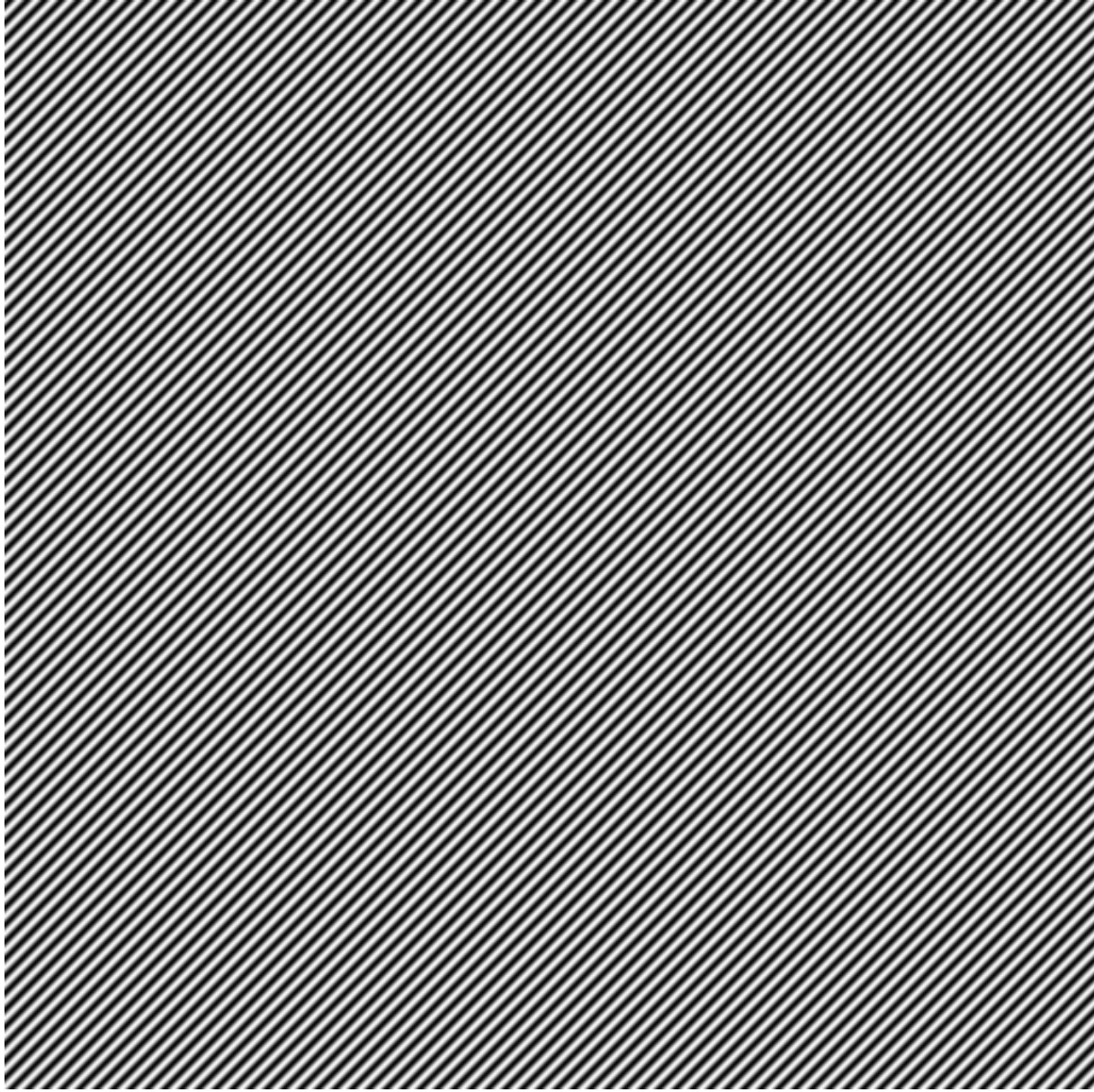
Vector (u,v)
- Magnitude gives frequency
- Direction gives orientation.

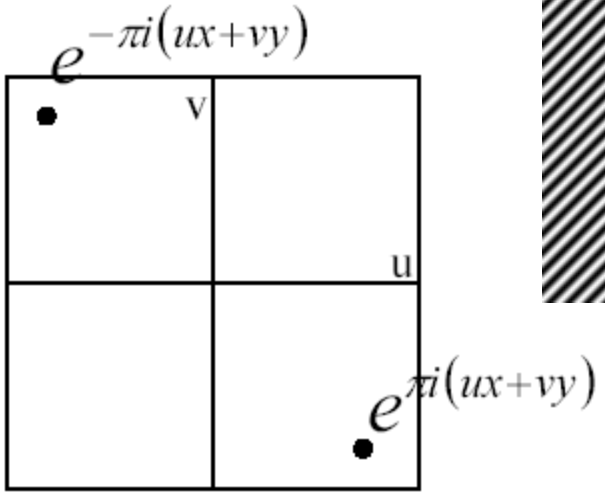


Here u and v are larger than in the previous slide.



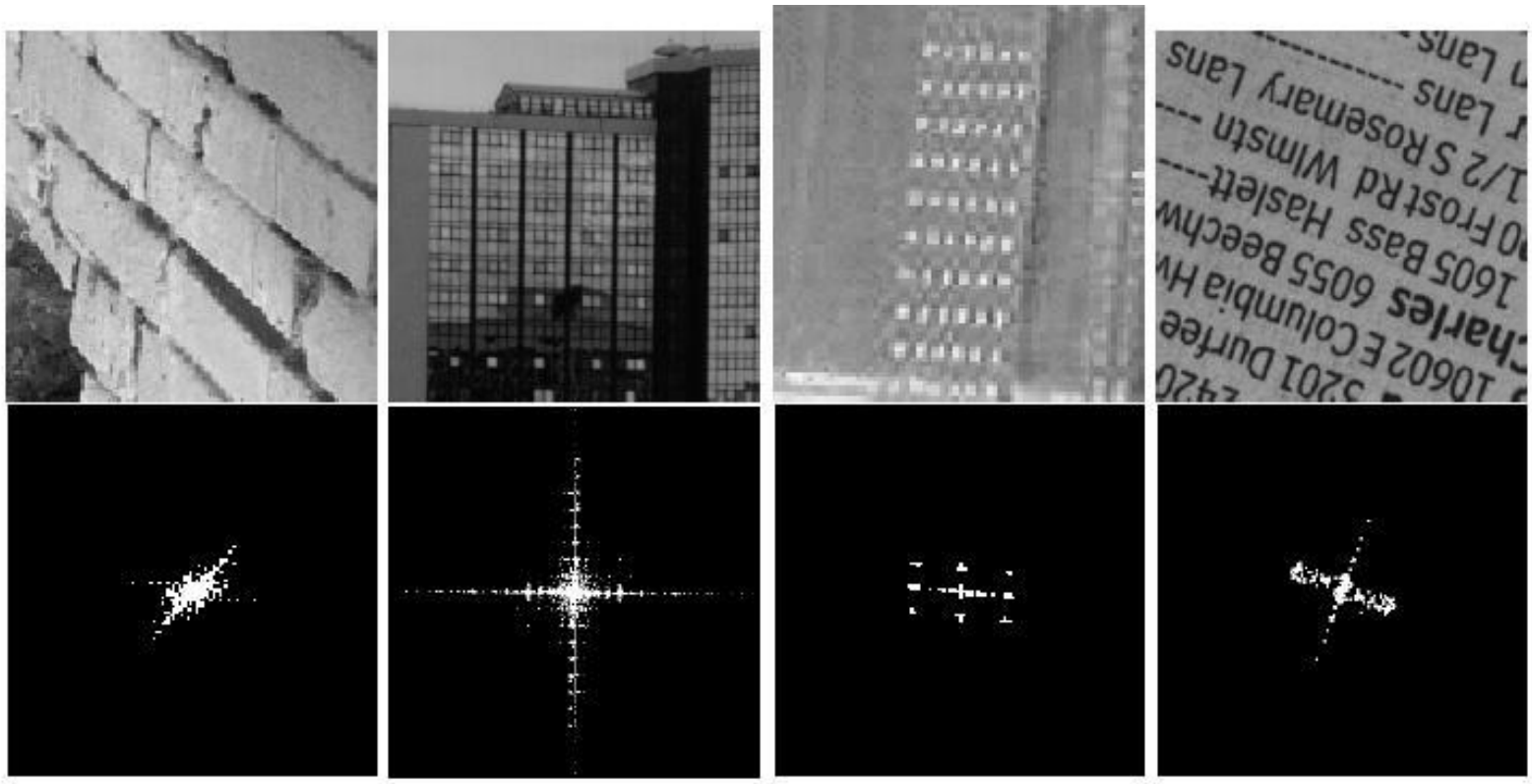
And larger still...





Using Fourier Representations

Dominant Orientation



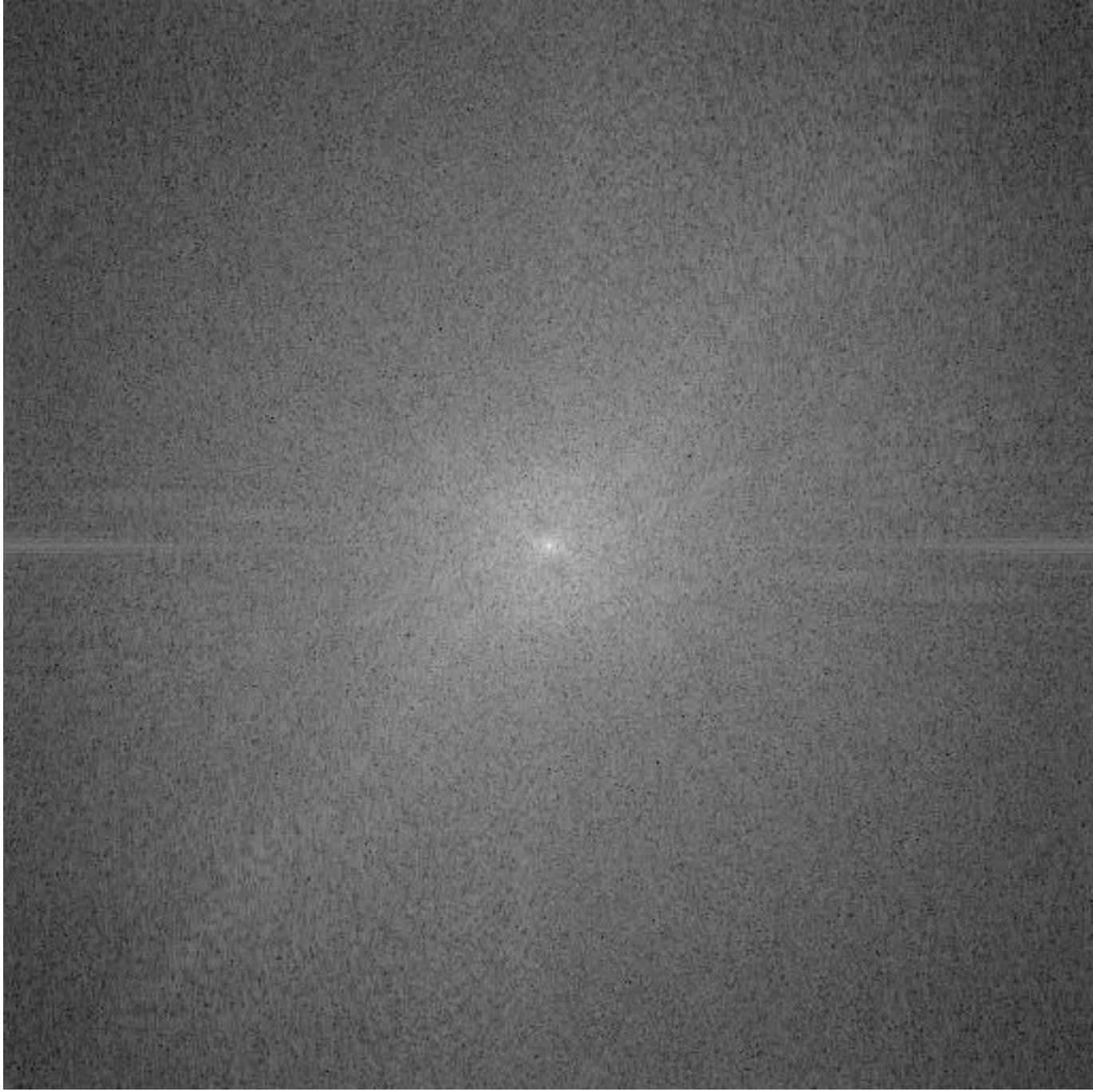
Limitations: not useful for local segmentation

Phase and Magnitude

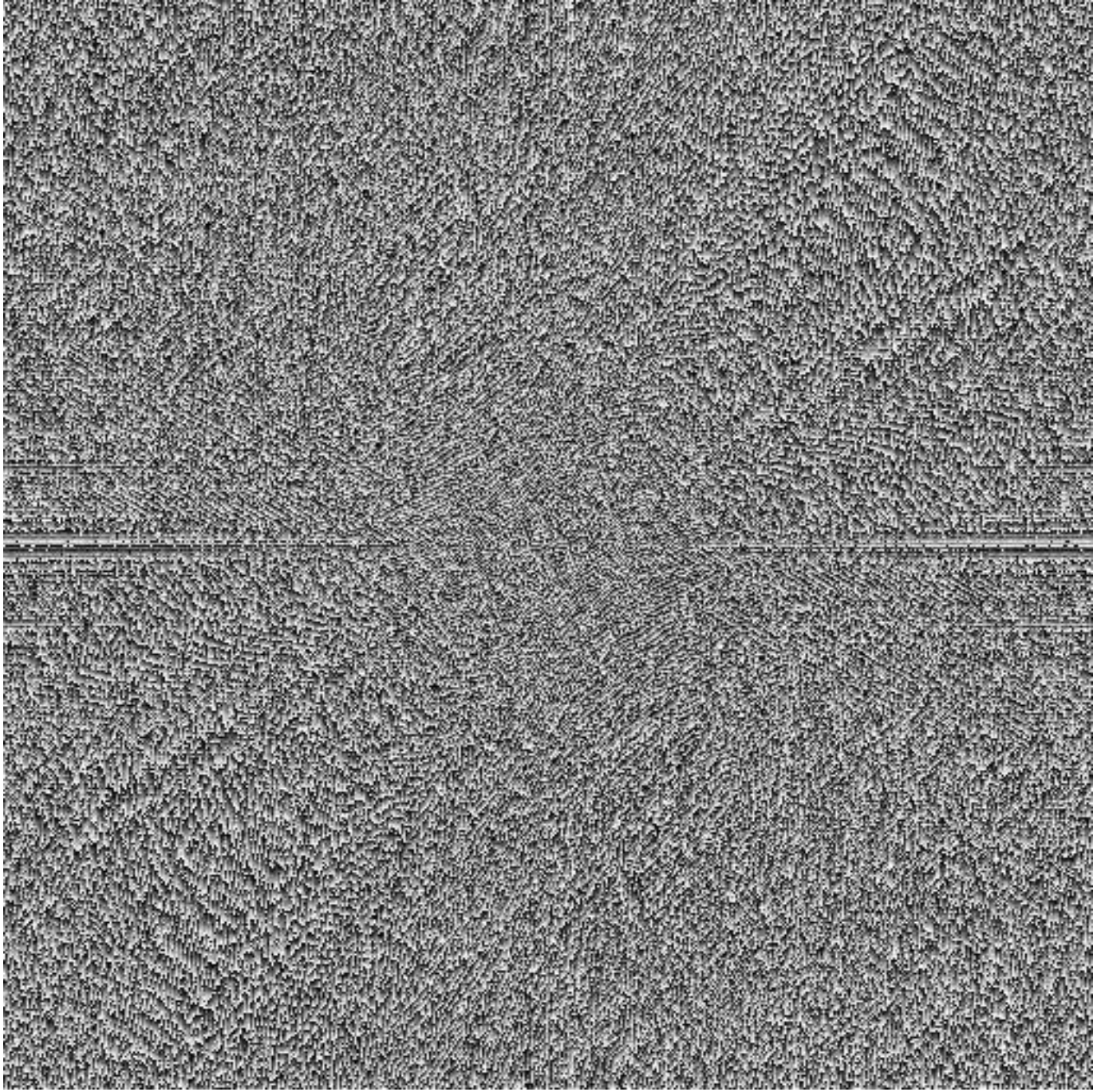
$$e^{i\theta} = \cos\theta + i \sin\theta$$

- Fourier transform of a real function is complex
  - difficult to plot, visualize
  - instead, we can think of the phase and magnitude of the transform
- Phase is the phase of the complex transform
- Magnitude is the magnitude of the complex transform
- Curious fact
  - all natural images have about the same magnitude transform
  - hence, phase seems to matter, but magnitude largely doesn't
- Demonstration
  - Take two pictures, swap the phase transforms, compute the inverse - what does the result look like?

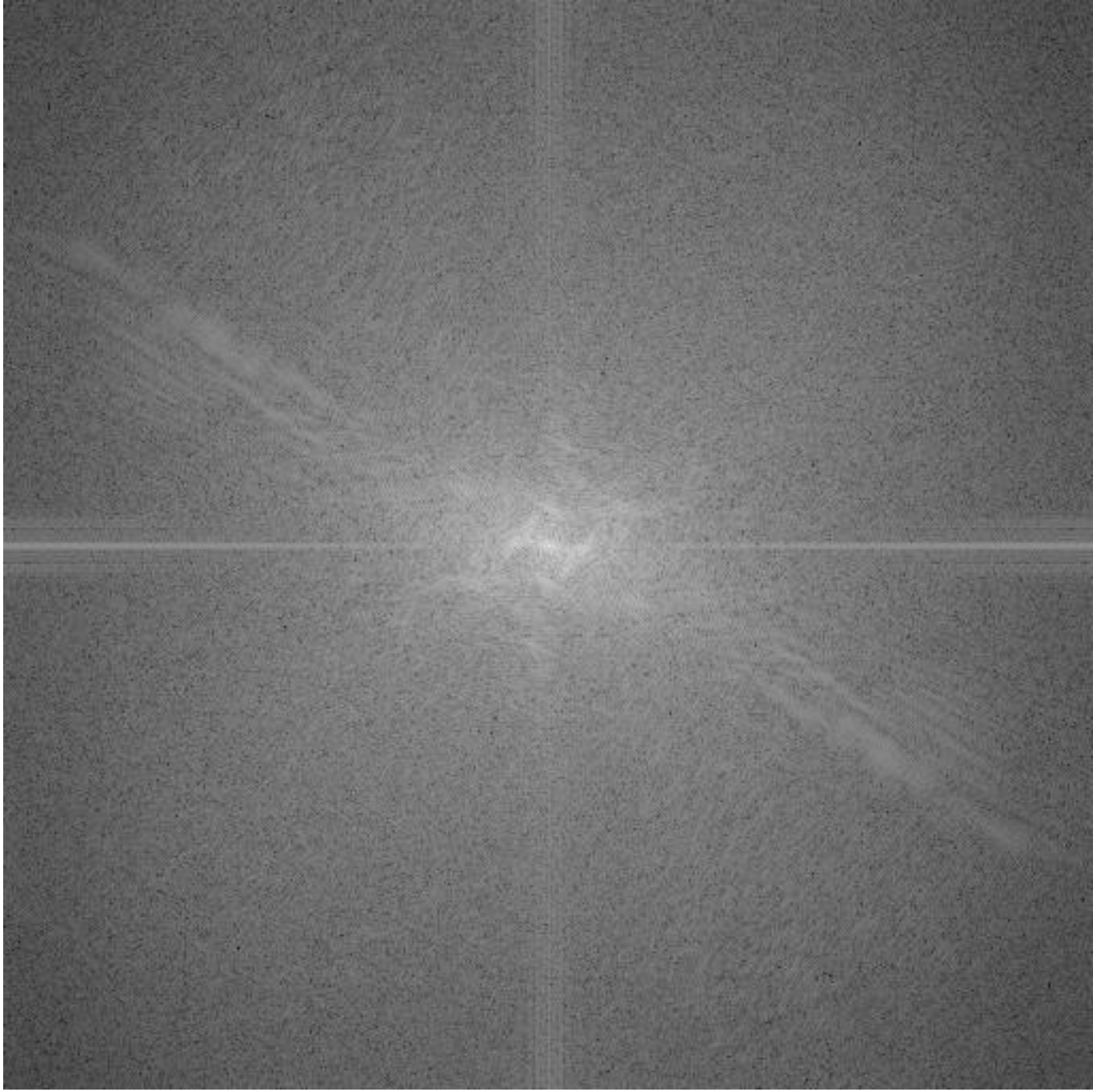
This is the magnitude transform of the cheetah pic



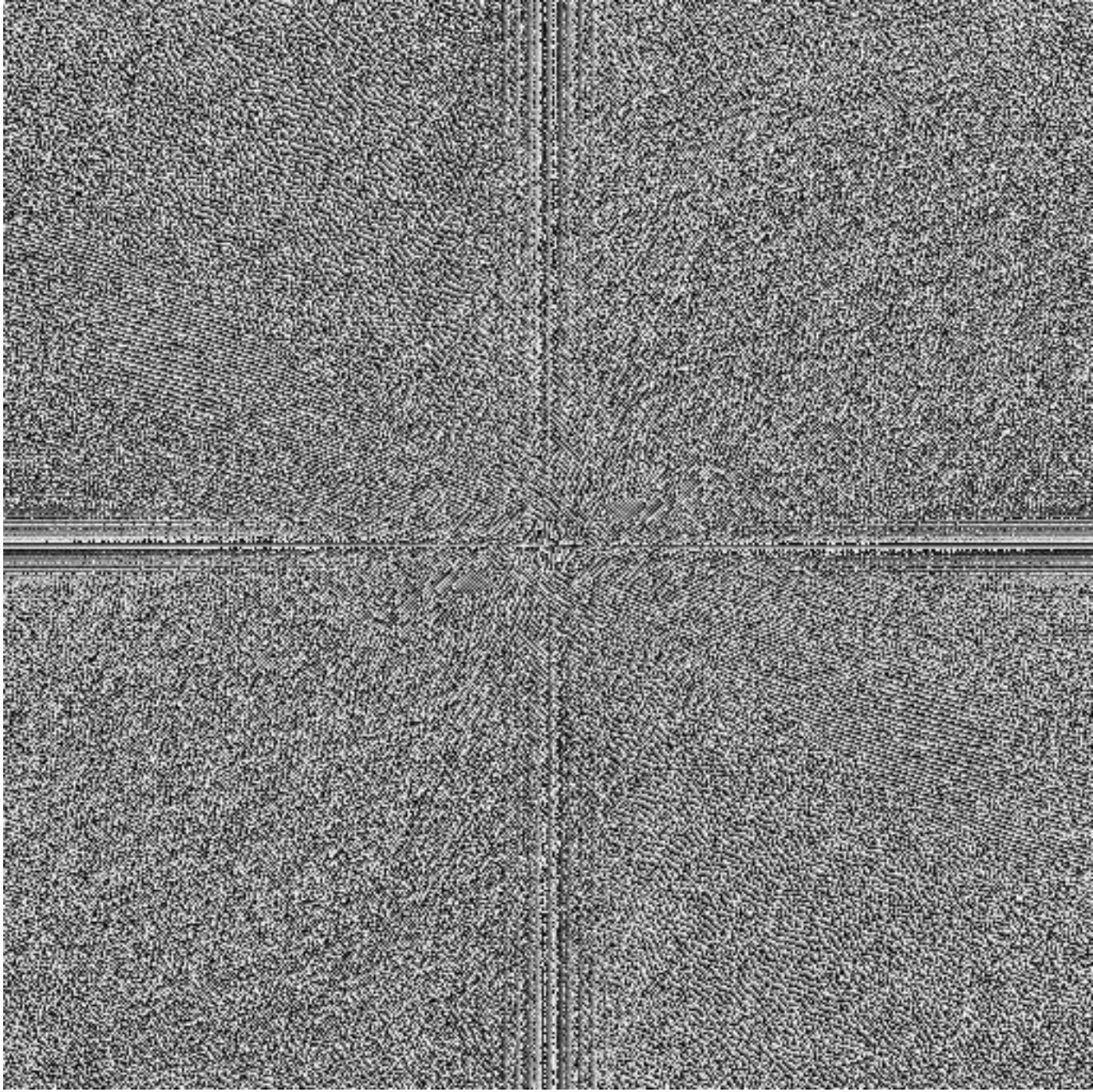
This is the phase transform of the cheetah pic



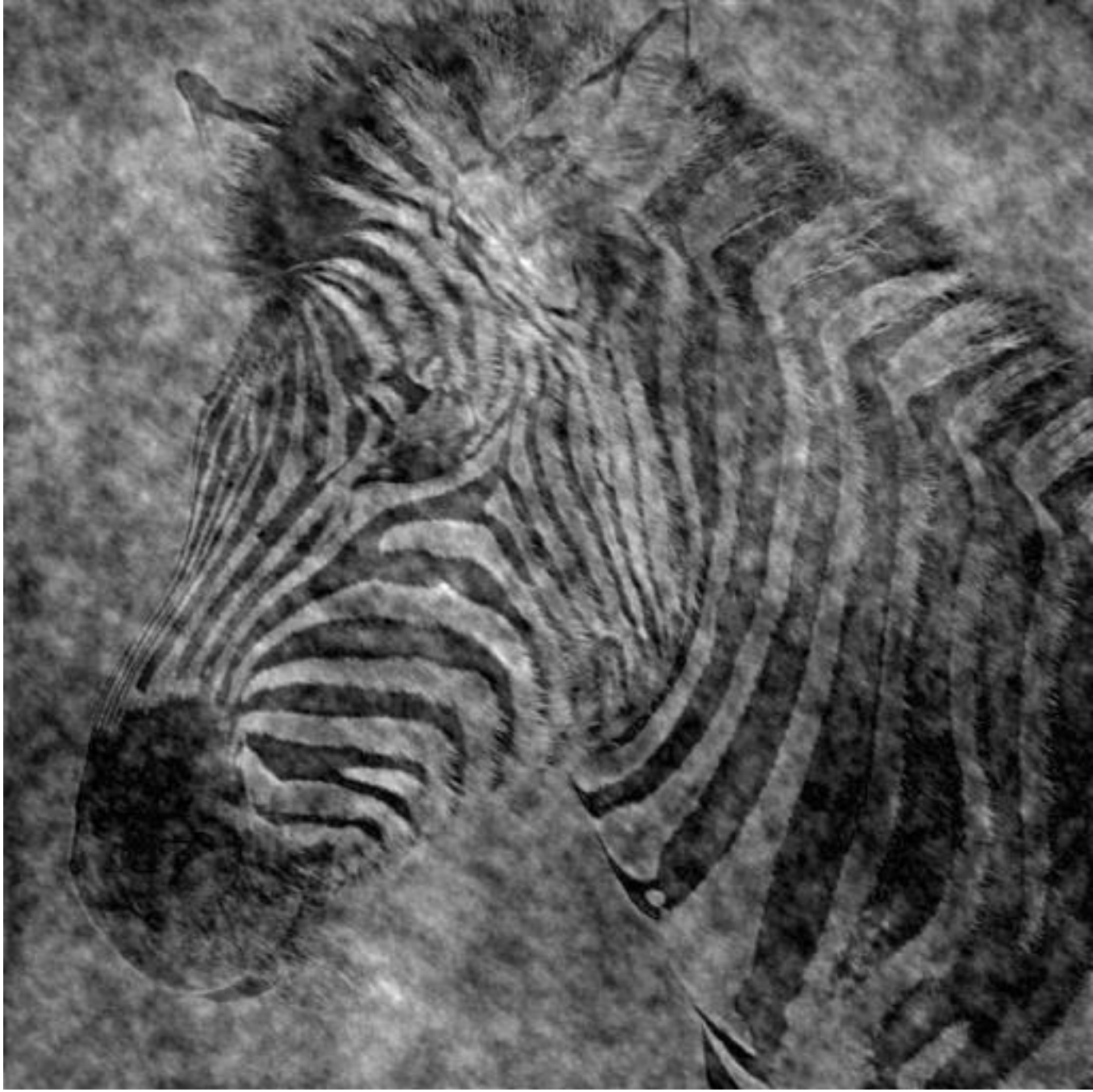
This is the magnitude transform of the zebra pic



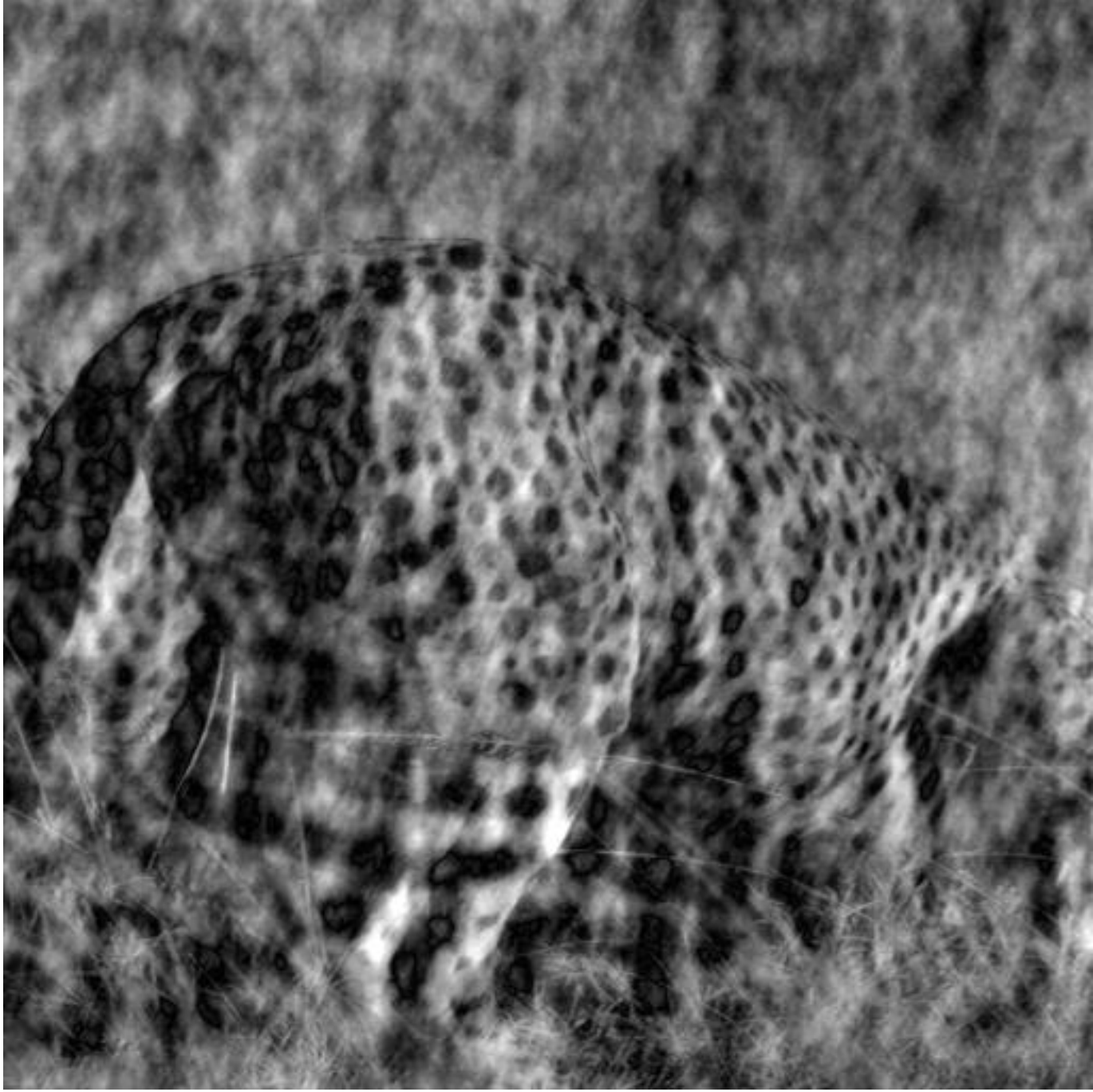
This is the phase transform of the zebra pic



Reconstruction with zebra phase, cheetah magnitude



Reconstruction with cheetah phase, zebra magnitude



The Fourier Transform and Convolution

- If H and G are images, and F(.) represents Fourier transform, then

$$F(H*G) = F(H)F(G)$$

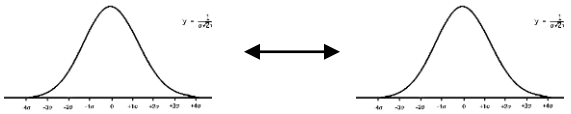
- Thus, one way of thinking about the properties of a convolution is by thinking of how it modifies the frequencies of the image to which it is applied.

- In particular, if we look at the power spectrum, then we see that convolving image H by G attenuates frequencies where G has low power, and amplifies those which have high power.

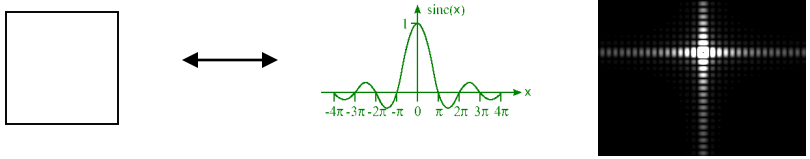
- This is referred to as the **Convolution Theorem**

Various Fourier Transform Pairs

- Important facts
  - scale function down ⇔ scale transform up
    i.e. high frequency = small details
  - The Fourier transform of a Gaussian is a Gaussian.



compare to box function transform



Other Types of Noise

- Impulsive noise
  - randomly pick a pixel and randomly set to a value
  - saturated version is called salt and pepper noise
- Quantization effects
  - Often called noise although it is not statistical
- Unanticipated image structures
  - Also often called noise although it is a real repeatable signal.

Some other useful filtering techniques

- Median filter
- Anisotropic diffusion

Median filters : principle

Method :

1. rank-order neighborhood intensities
2. take middle value

- non-linear filter
- no new gray levels emerge...

Median filters: Example for window size of 3

1,1,1,7,1,1,1,1

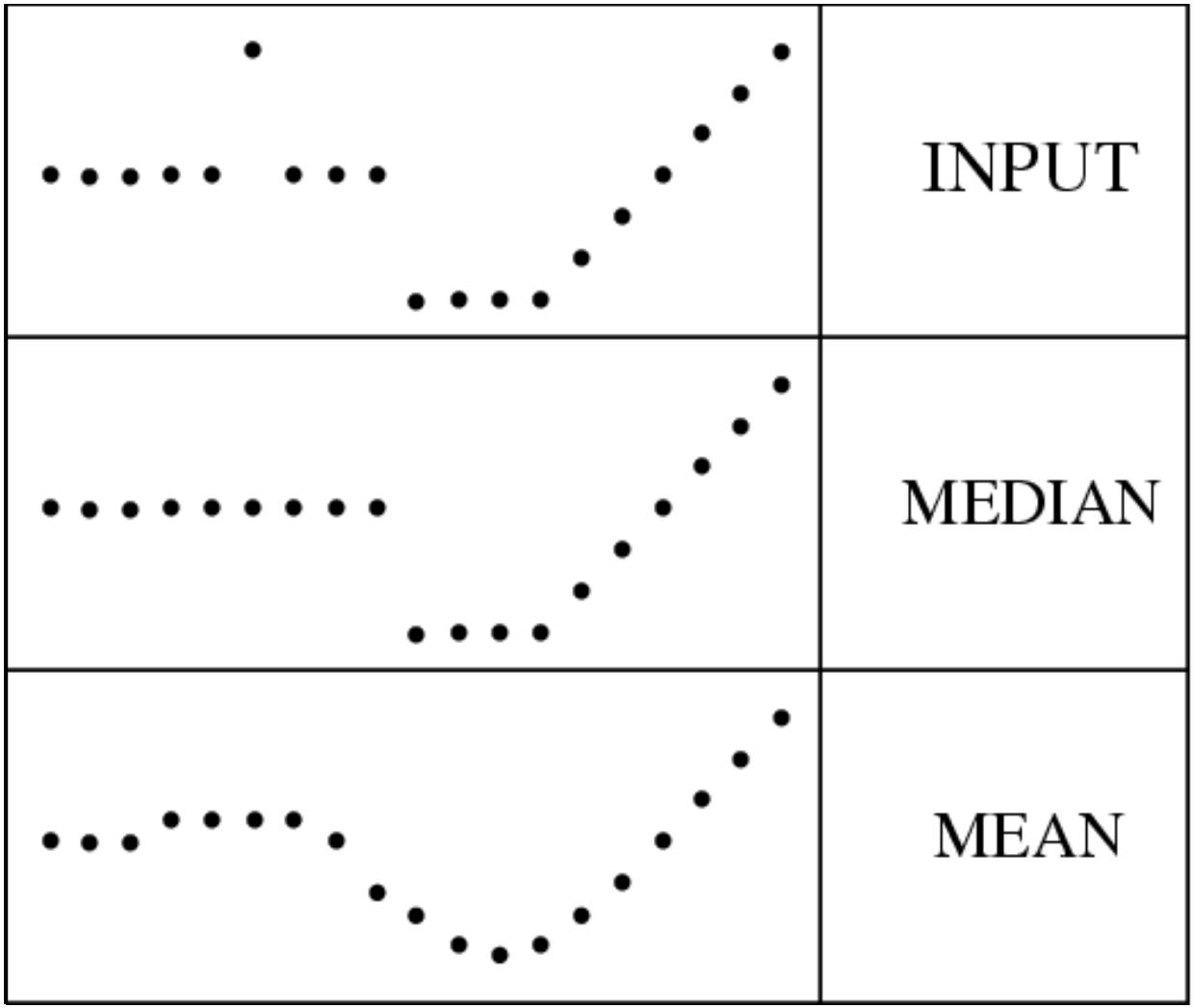
↓

?,1,1,1,1,1,1,?

The advantage of this type of filter is that it eliminates spikes (salt & pepper noise).

Median filters : example

filters have width 5 :



Median filters : analysis

median completely discards the spike,
linear filter always responds to all aspects

median filter preserves discontinuities,
linear filter produces rounding-off effects

Do not become all too optimistic

Median filter : images

3 x 3 median filter :

sharpens edges, destroys edge cusps and protrusions

Median filters : Gauss revisited

Comparison with Gaussian :

e.g. upper lip smoother, eye better preserved

Example of median

10 times 3 X 3 median

patchy effect
important details lost (e.g. earring)

Next Lecture

- Early vision
  - Edges and corners
- Reading:
  - Chapter 5: Local Image Features