# Image Filtering

Computer Vision I

CSE 252A

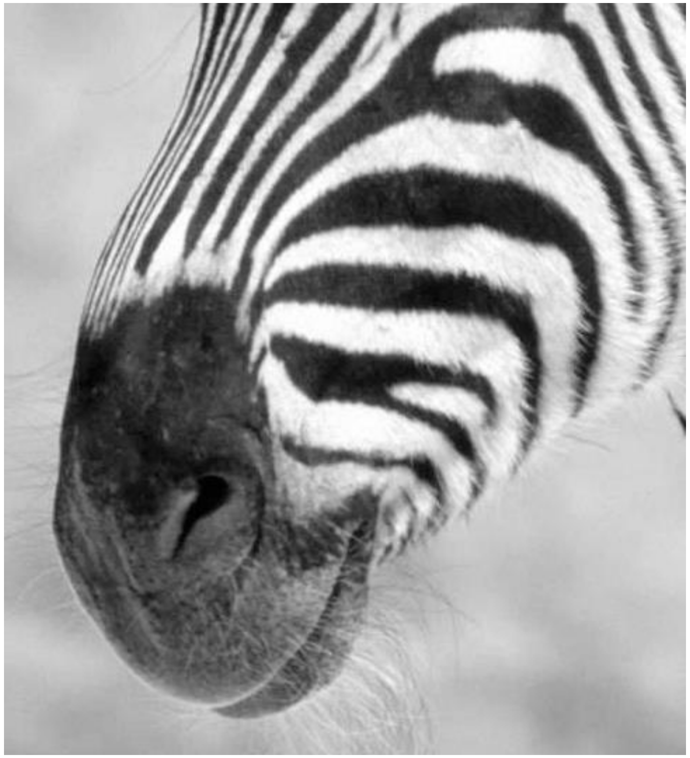
Lecture 5

# Announcements

- Assignment 1 is due Oct 25, 11:59 PM

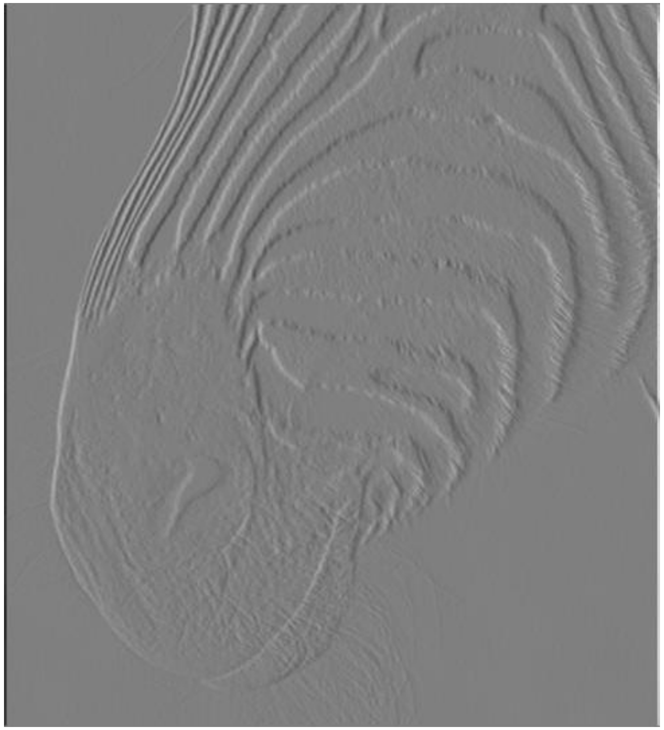
# Image Filtering Example

Input



Filter

Output



# What is image filtering?

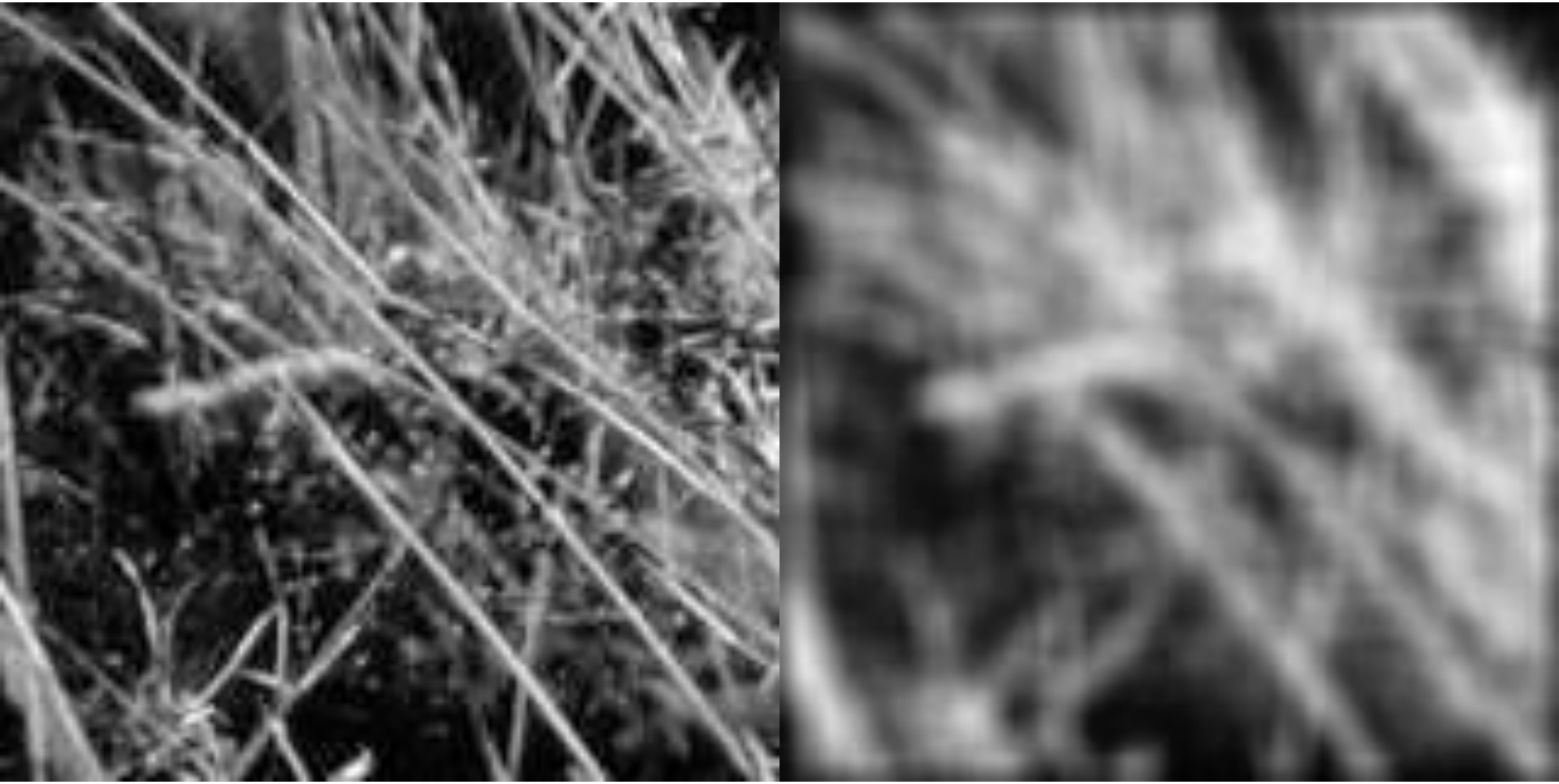
Producing a new image where the value at a pixel in the output image is a function of a neighborhood of the pixel location in the input image.

| | | |
|---|---|---|
| 10 | 5 | 3 |
| 4 | 5 | 1 |
| 1 | 1 | 7 |

→

| | | |
|---|---|---|
| | | |
| | 7 | |
| | | |

# Example: Smoothing by Averaging



# Image Filtering

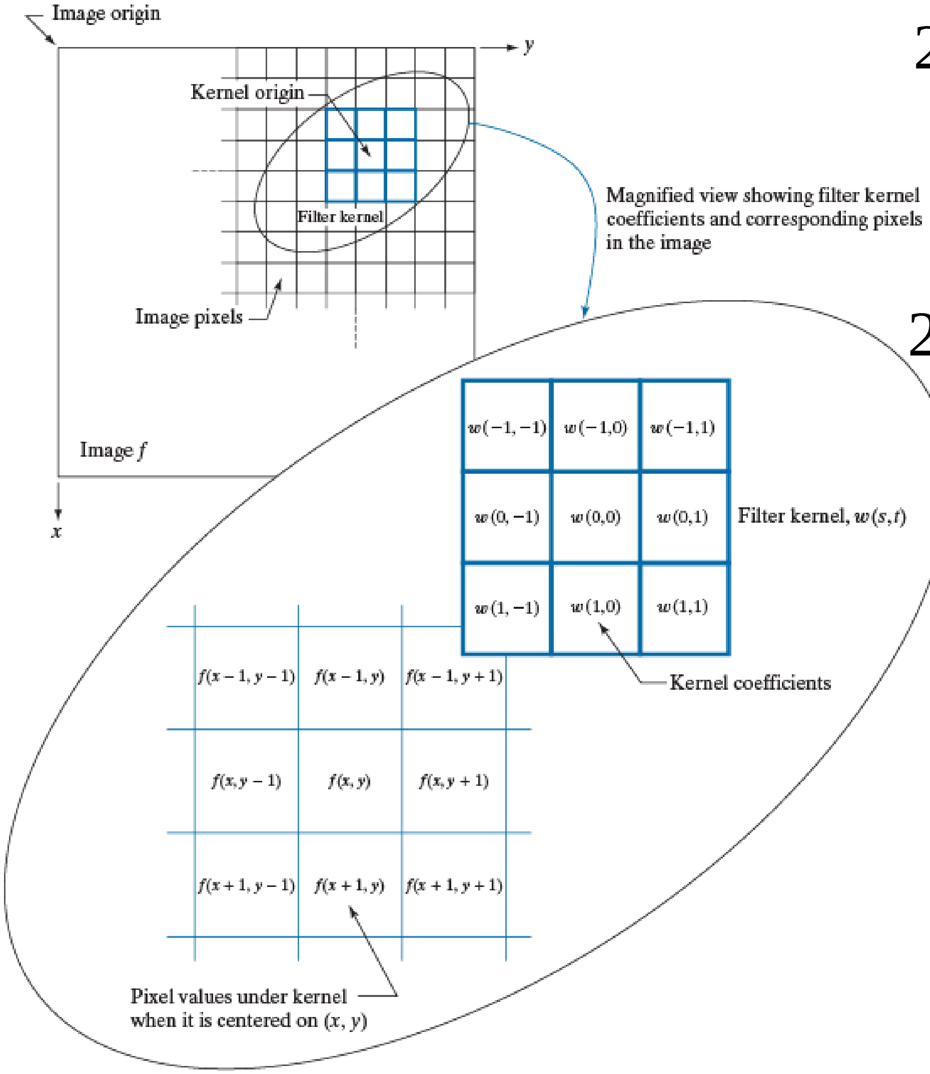
- Most common filters are linear filters and the process of applying a linear filter is called convolution
- Why filter
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching
  - The “convolution” in Convolutional Neural Networks

# 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

# Spatial filtering (2D)



2D correlation

$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

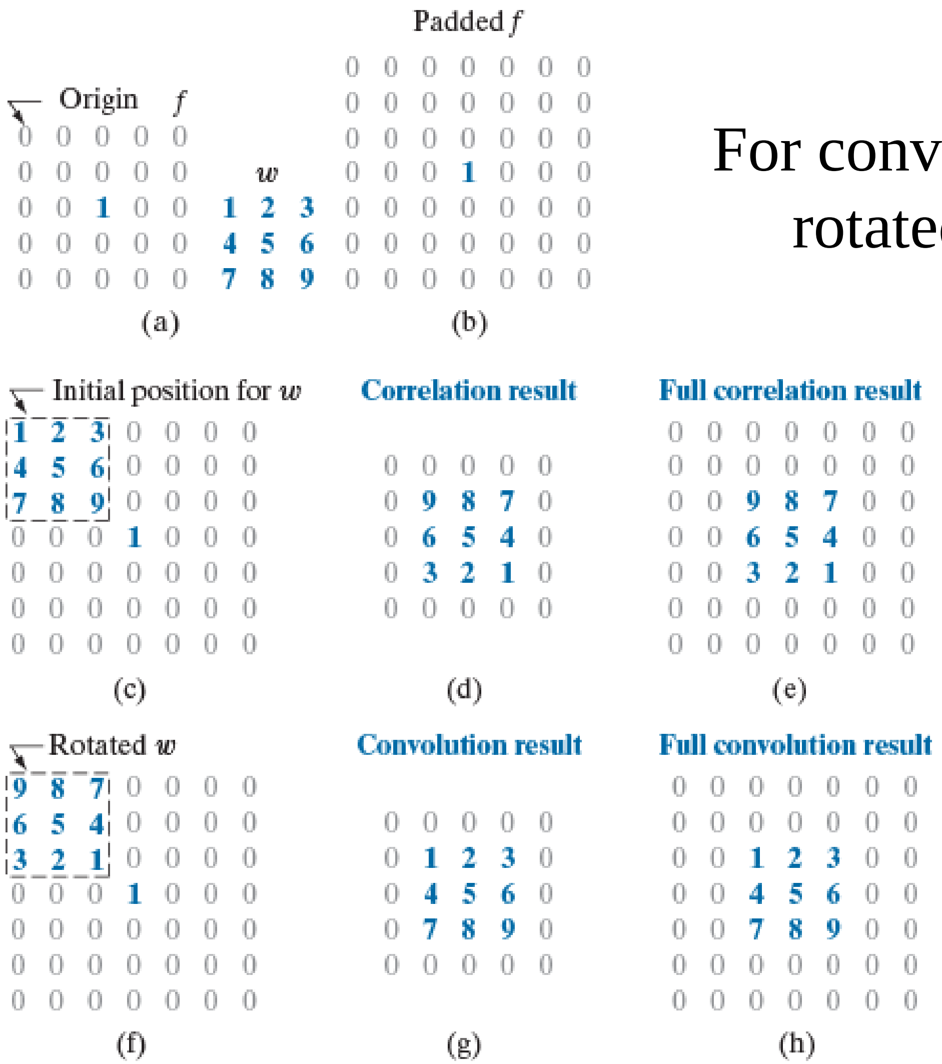
2D convolution

$$w(x,y) \bigstar f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

For convolution, kernel is rotated 180 degrees

When kernel is symmetric, convolution and correlation give the same result

# Correlation and convolution (2D)



Padded *f*

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Origin *f*

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

*w*

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

(a) (b)

Initial position for *w*

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 0 | 0 | 0 | 0 |
| 4 | 5 | 6 | 0 | 0 | 0 | 0 |
| 7 | 8 | 9 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Correlation result

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 9 | 8 | 7 | 0 |
| 0 | 6 | 5 | 4 | 0 |
| 0 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Full correlation result

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 9 | 8 | 7 | 0 | 0 |
| 0 | 0 | 6 | 5 | 4 | 0 | 0 |
| 0 | 0 | 3 | 2 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

(c) (d) (e)

Rotated *w*

| | | | | | | |
|---|---|---|---|---|---|---|
| 9 | 8 | 7 | 0 | 0 | 0 | 0 |
| 6 | 5 | 4 | 0 | 0 | 0 | 0 |
| 3 | 2 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Convolution result

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 2 | 3 | 0 |
| 0 | 4 | 5 | 6 | 0 |
| 0 | 7 | 8 | 9 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Full convolution result

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 2 | 3 | 0 | 0 |
| 0 | 0 | 4 | 5 | 6 | 0 | 0 |
| 0 | 0 | 7 | 8 | 9 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

(f) (g) (h)

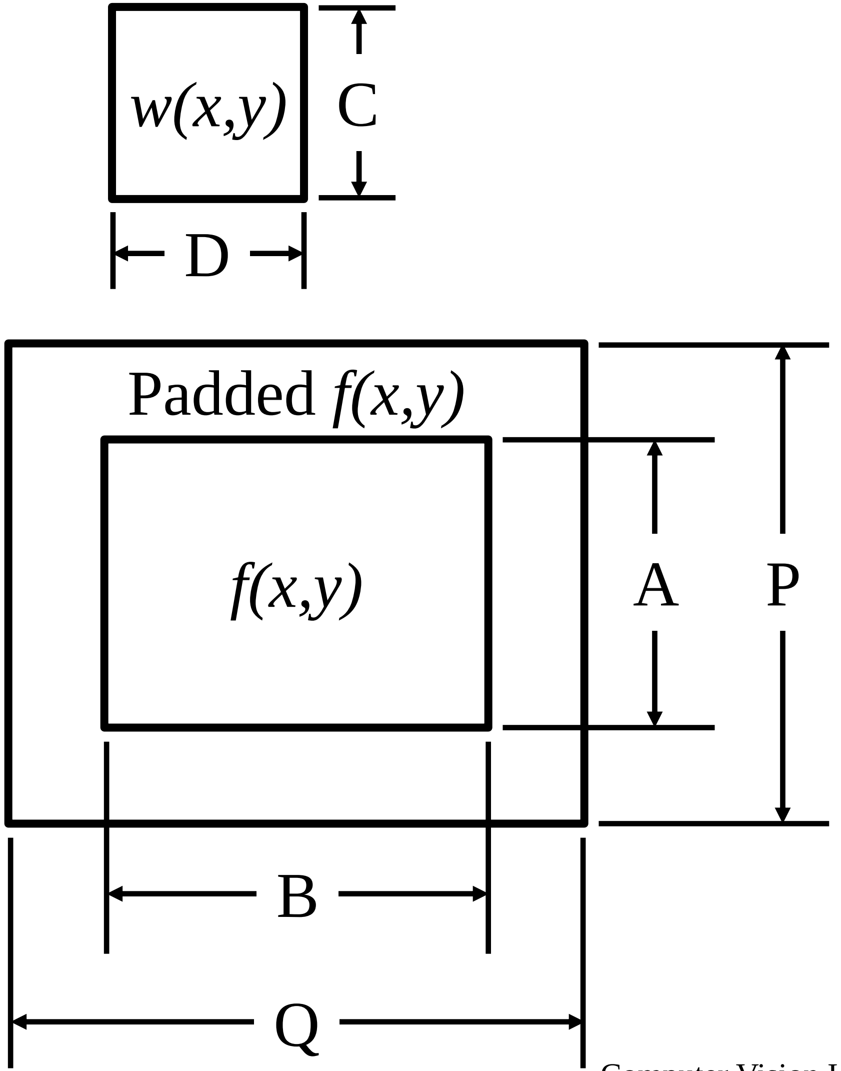
**FIGURE 3.36** Correlation (middle row) and convolution (last row) of a 2-D kernel with an image consisting of a discrete unit impulse. The 0's are shown in gray to simplify visual analysis. Note that correlation and convolution are functions of *x* and *y*. As these variable change, they *displace* one function with respect to the other. See the discussion of Eqs. (3-45) and (3-46) regarding full correlation and convolution.

For convolution, kernel is rotated 180 degrees

# “Shape” of correlation/convolution

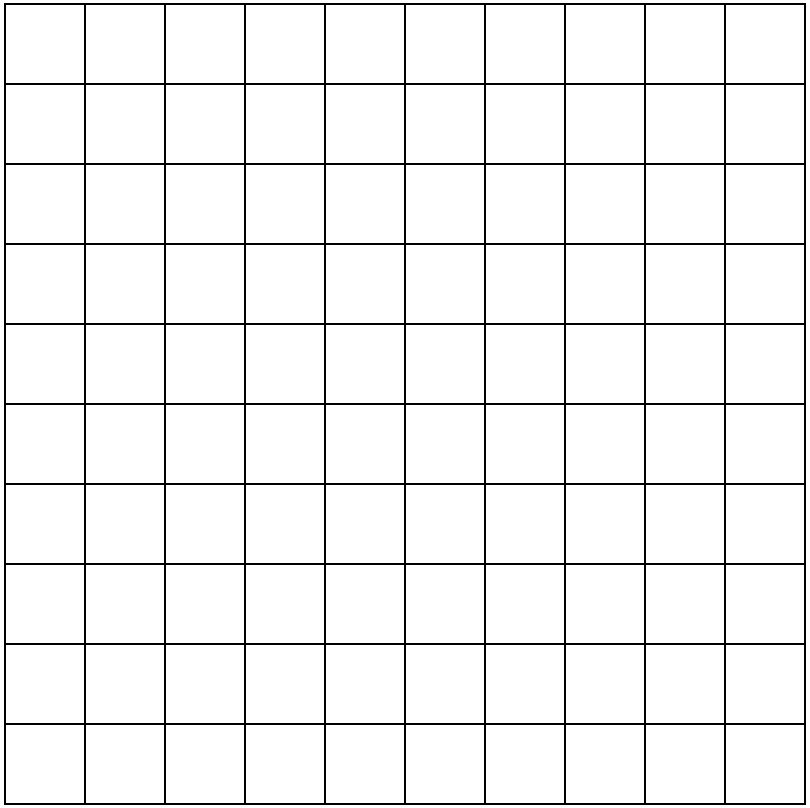
- Full
  - $w(x,y)$ and $f(x,y)$ have at least 1 pixel overlap
  - P = A + 2(C – 1)
  - Q = B + 2(D – 1)
  - Output $g(x,y)$
    - Width is B + D – 1, height is A + C – 1
- Same
  - P = A + C – 1
  - Q = B + D – 1
  - Output $g(x,y)$
    - Width is B, height is A
- Valid
  - $w(x,y)$ must be completely inside $f(x,y)$
  - P = A
  - Q = B
  - Output $g(x,y)$
    - Width is B – D + 1, height is A – C + 1

Convolution $g(x,y) = w(x,y) * f(x,y)$



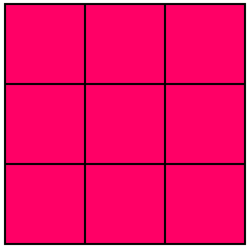
# Convolution

$$w(x,y)\bigstar f(x,y) = \sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)f(x-s,y-t)$$



Input image *f*(*x*,*y*)

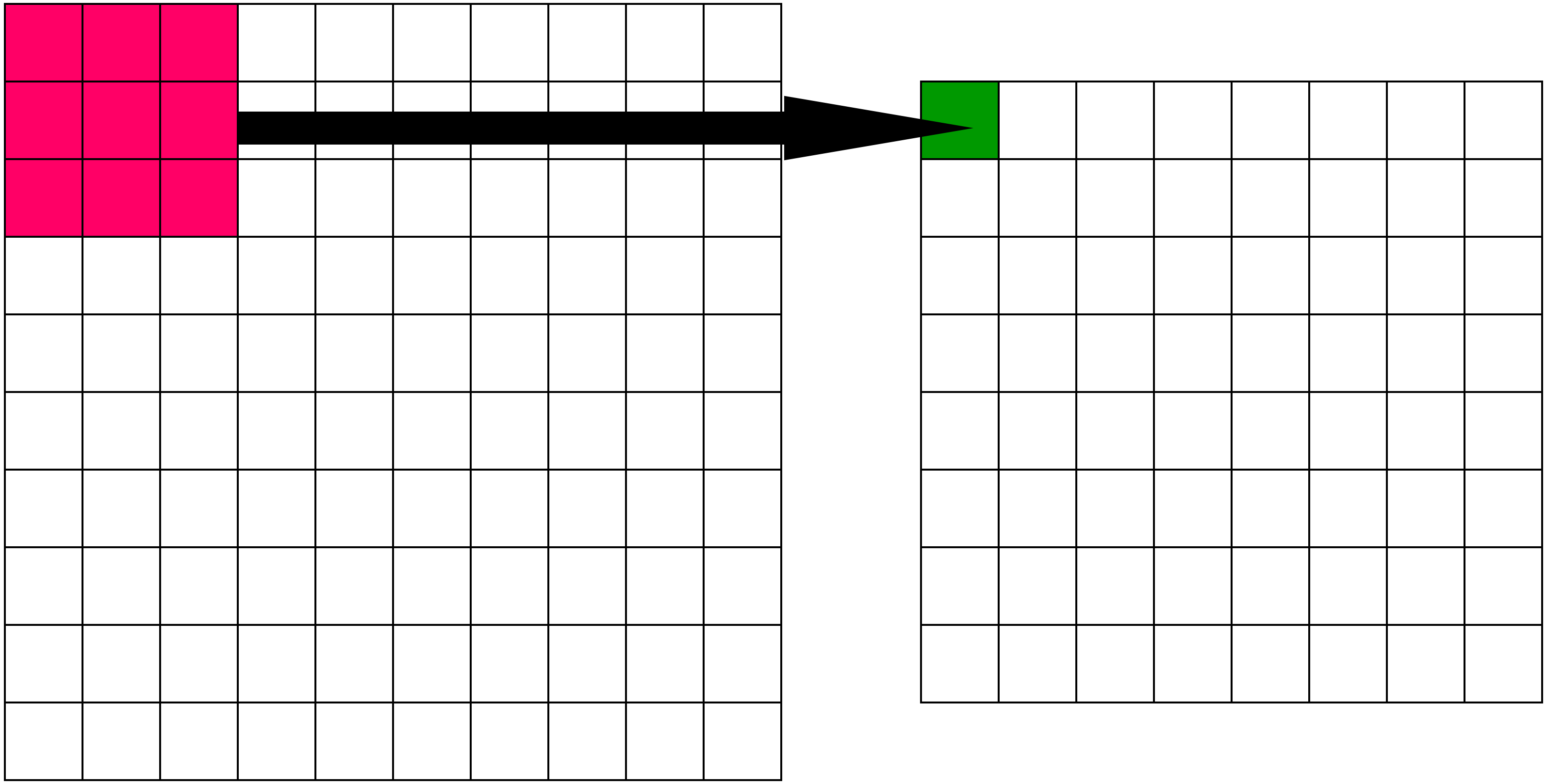
*



Kernel *w*(*x*,*y*)

Note: Typically kernel is relatively small in vision applications.

# “Valid” convolution

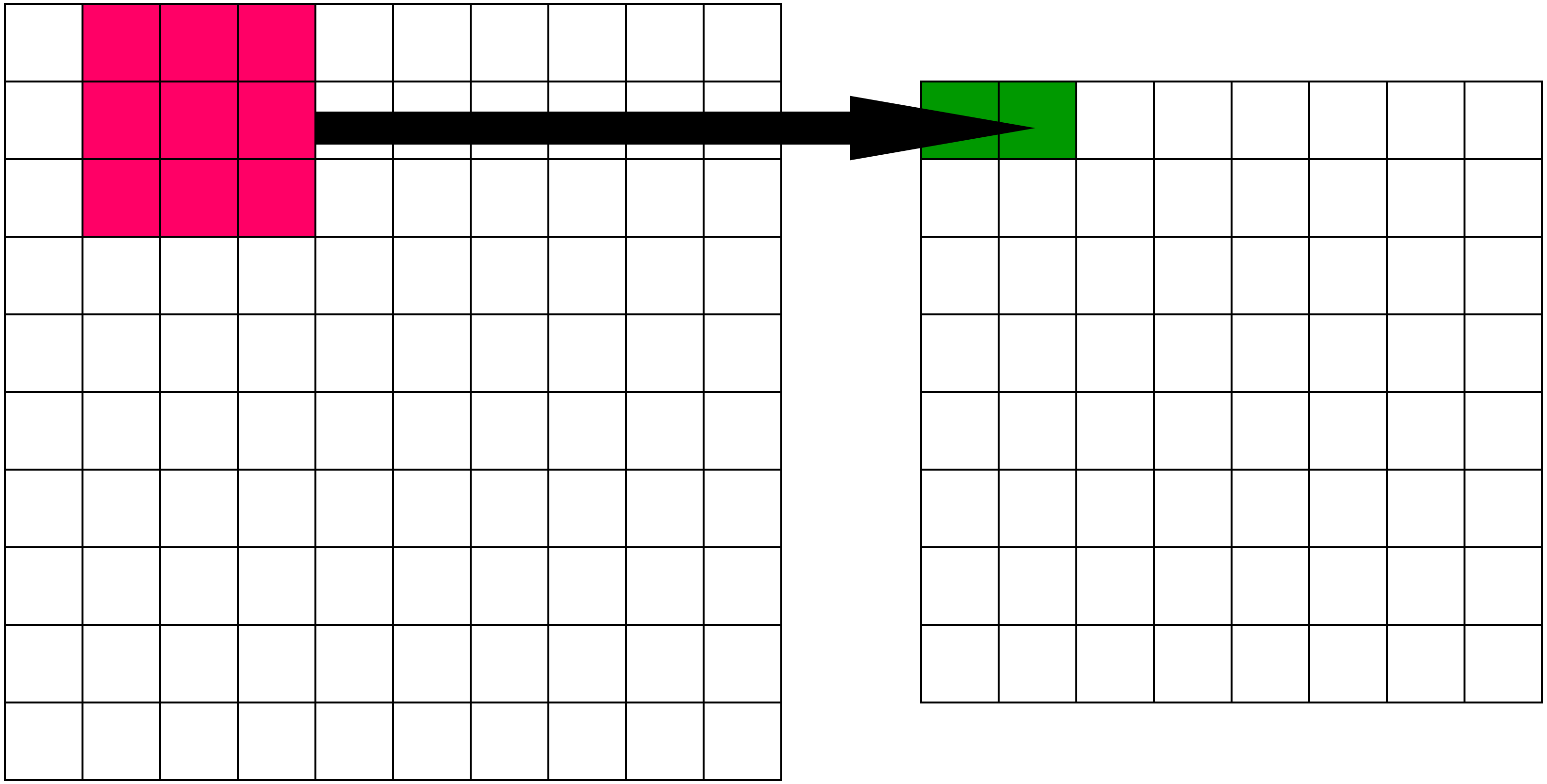


Input image

Output image

$$w(x,y) \bigstar f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

# “Valid” convolution

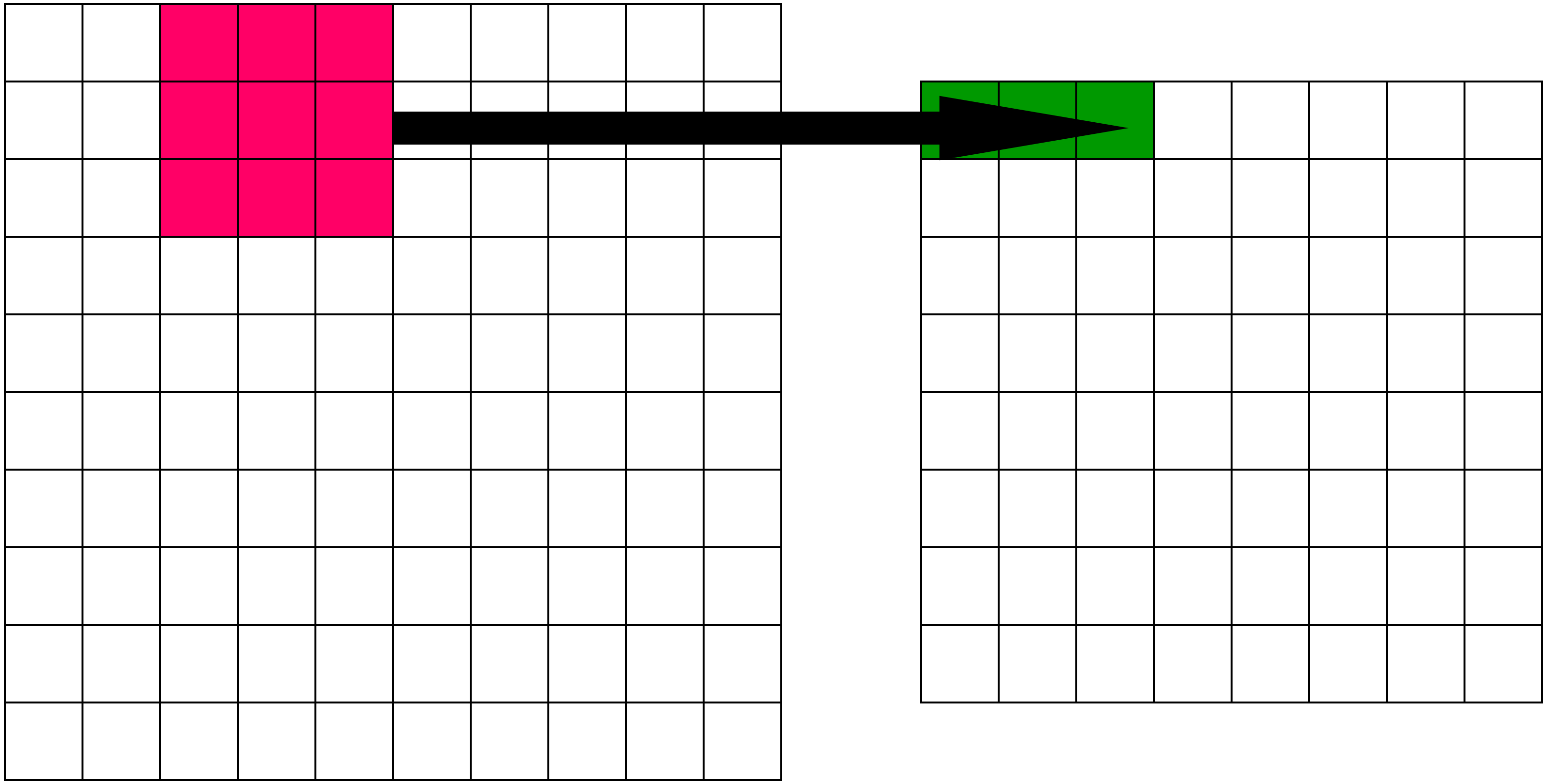


Input image

Output image

$$w(x,y)\bigstar f(x,y) = \sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)f(x-s,y-t)$$

# “Valid” convolution

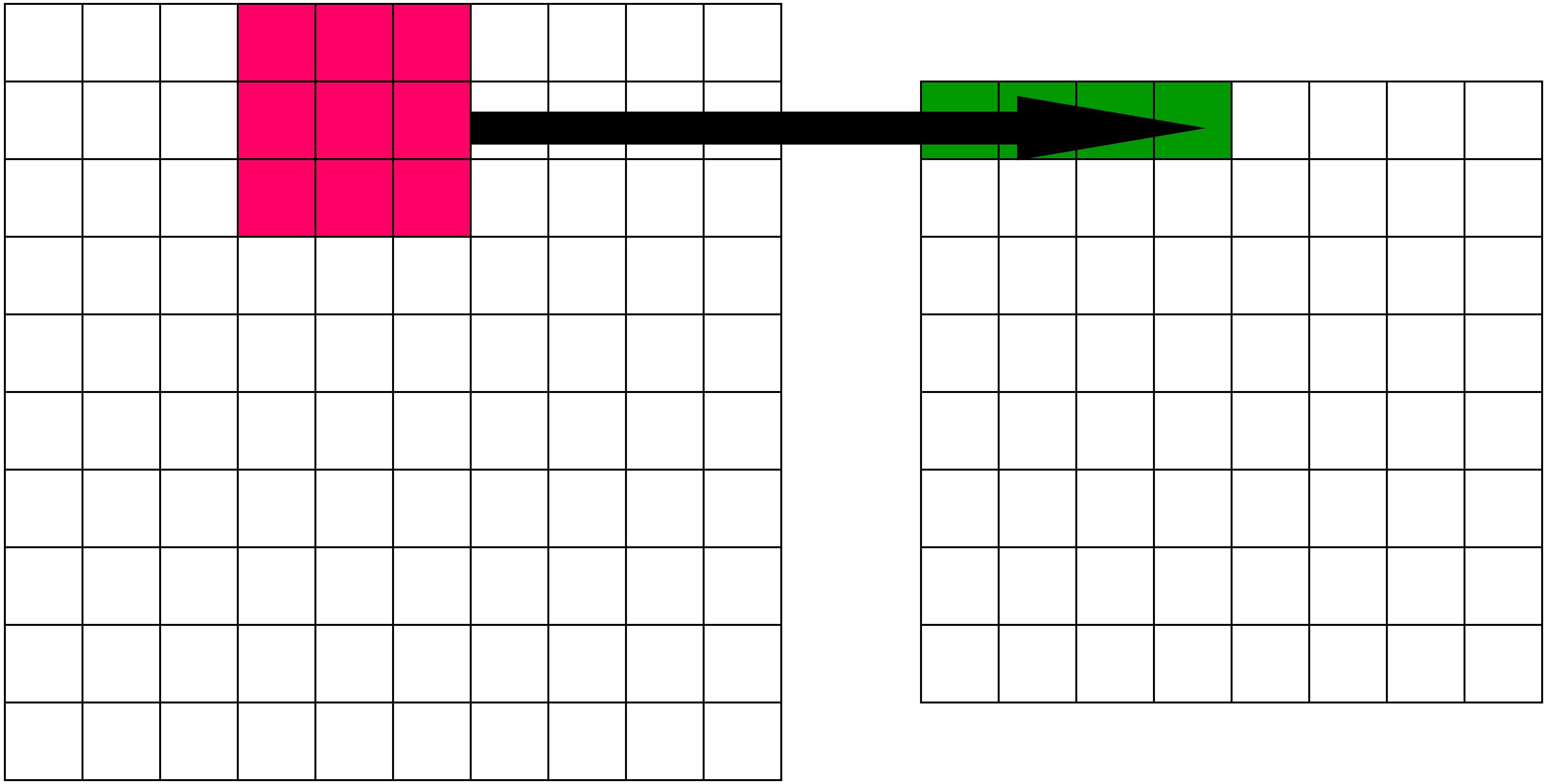


Input image

Output image

$$w(x,y)\bigstar f(x,y) = \sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)f(x-s,y-t)$$

# “Valid” convolution

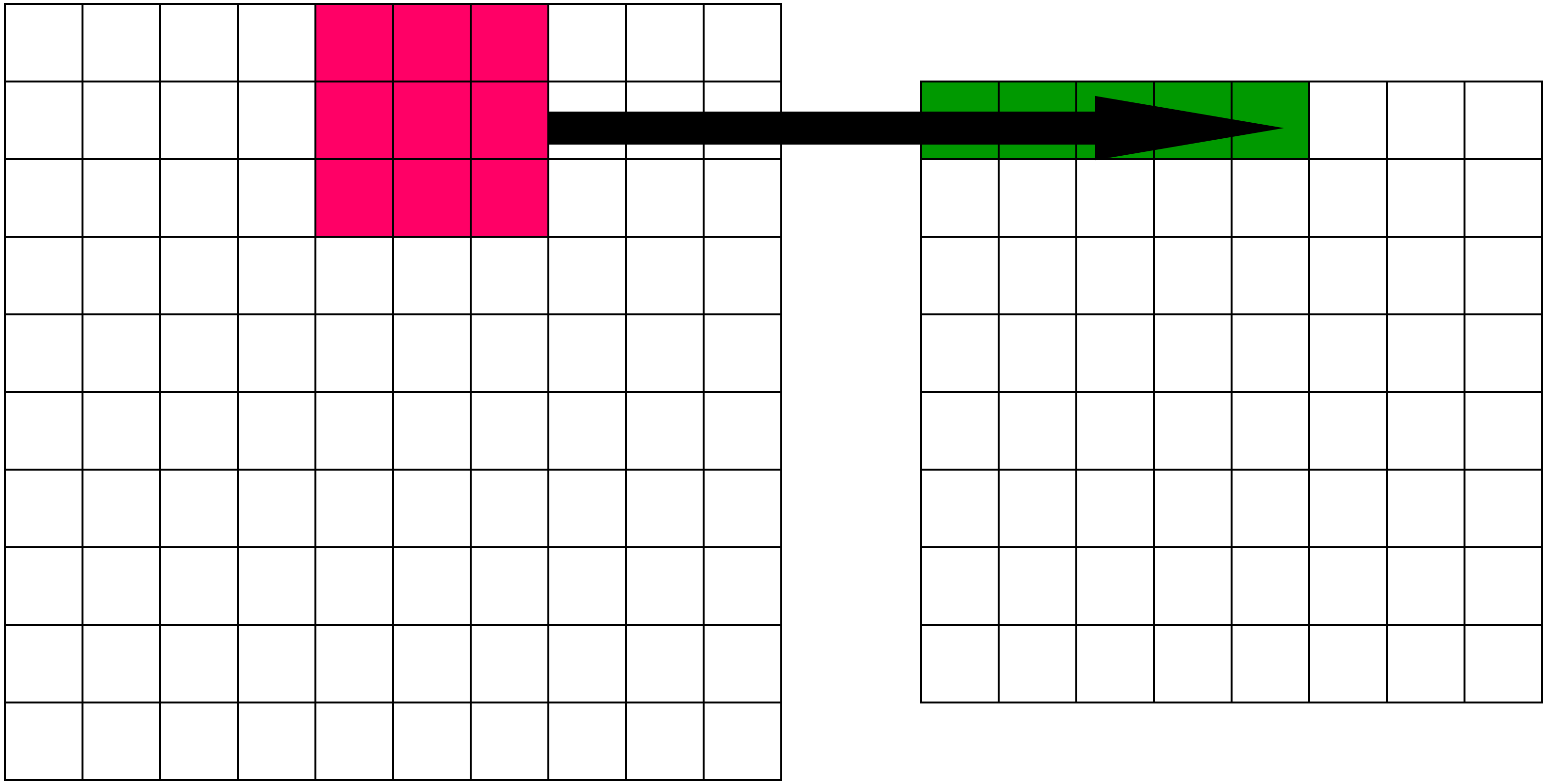


Input image

Output image

$$w(x,y)\bigstar f(x,y) = \sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)f(x-s,y-t)$$

# “Valid” convolution

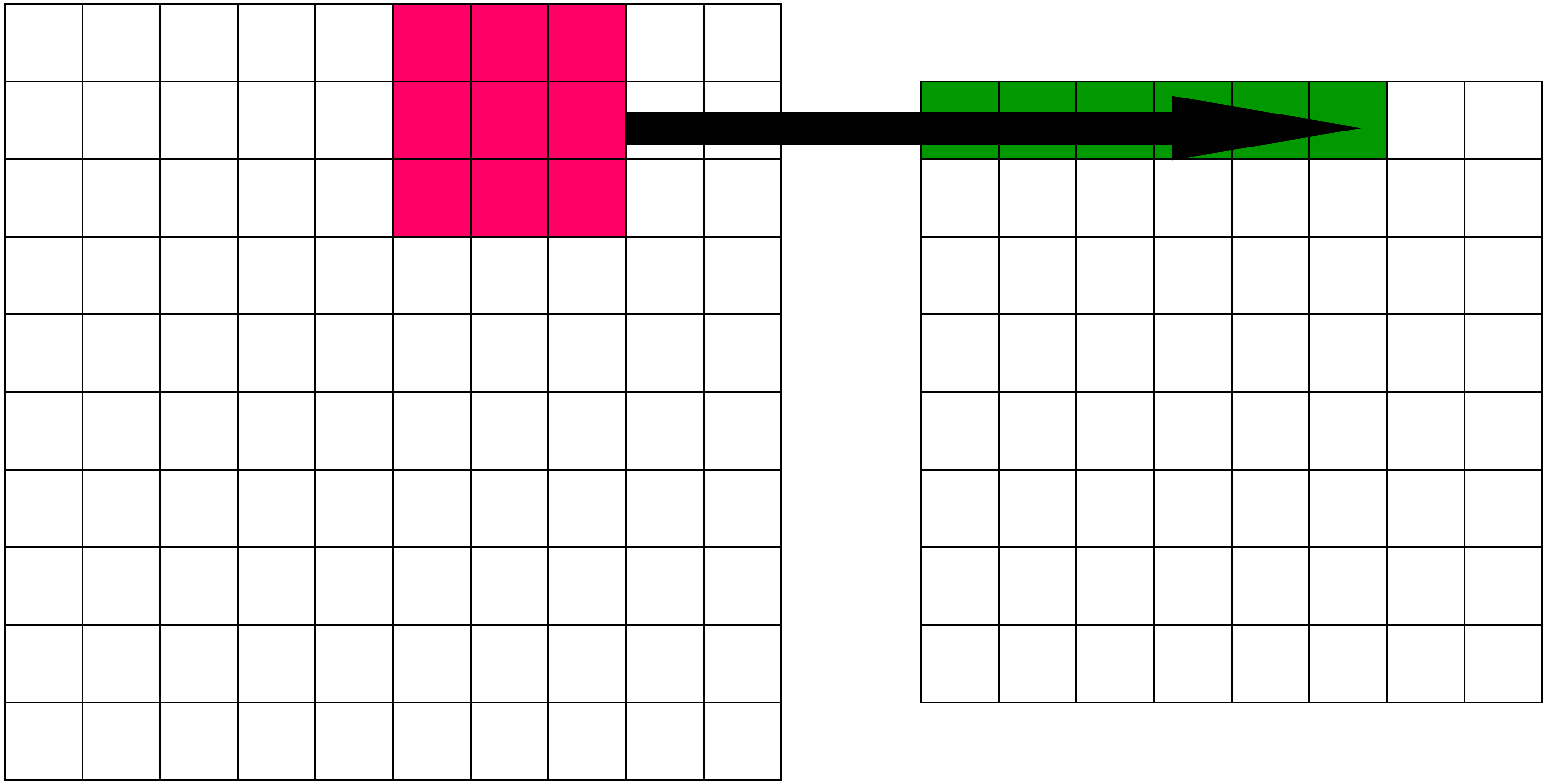


Input image

Output image

$$w(x,y)\bigstar f(x,y) = \sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)f(x-s,y-t)$$

# “Valid” convolution

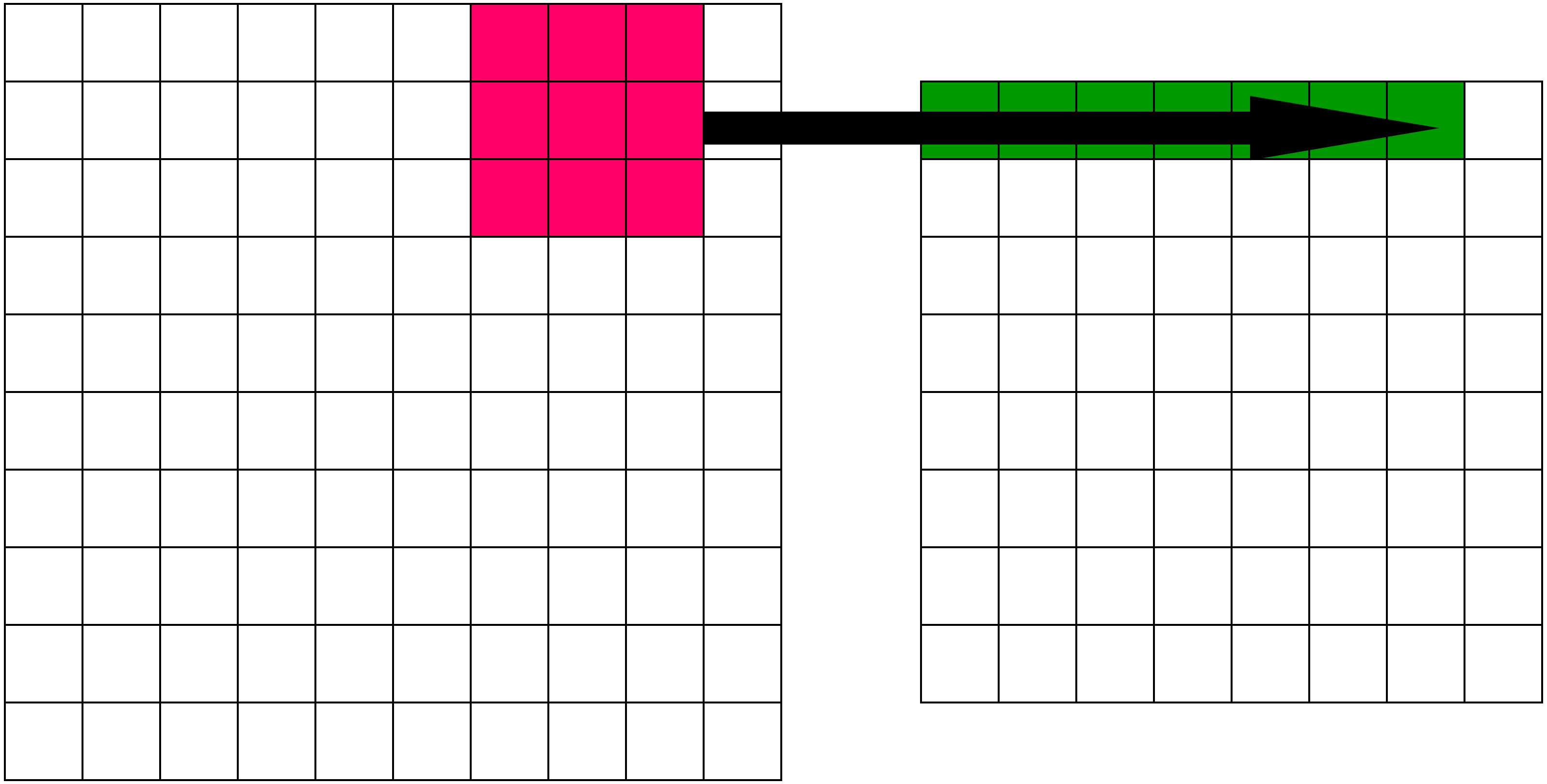


Input image

Output image

$$w(x,y)\bigstar f(x,y) = \sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)f(x-s,y-t)$$

# “Valid” convolution

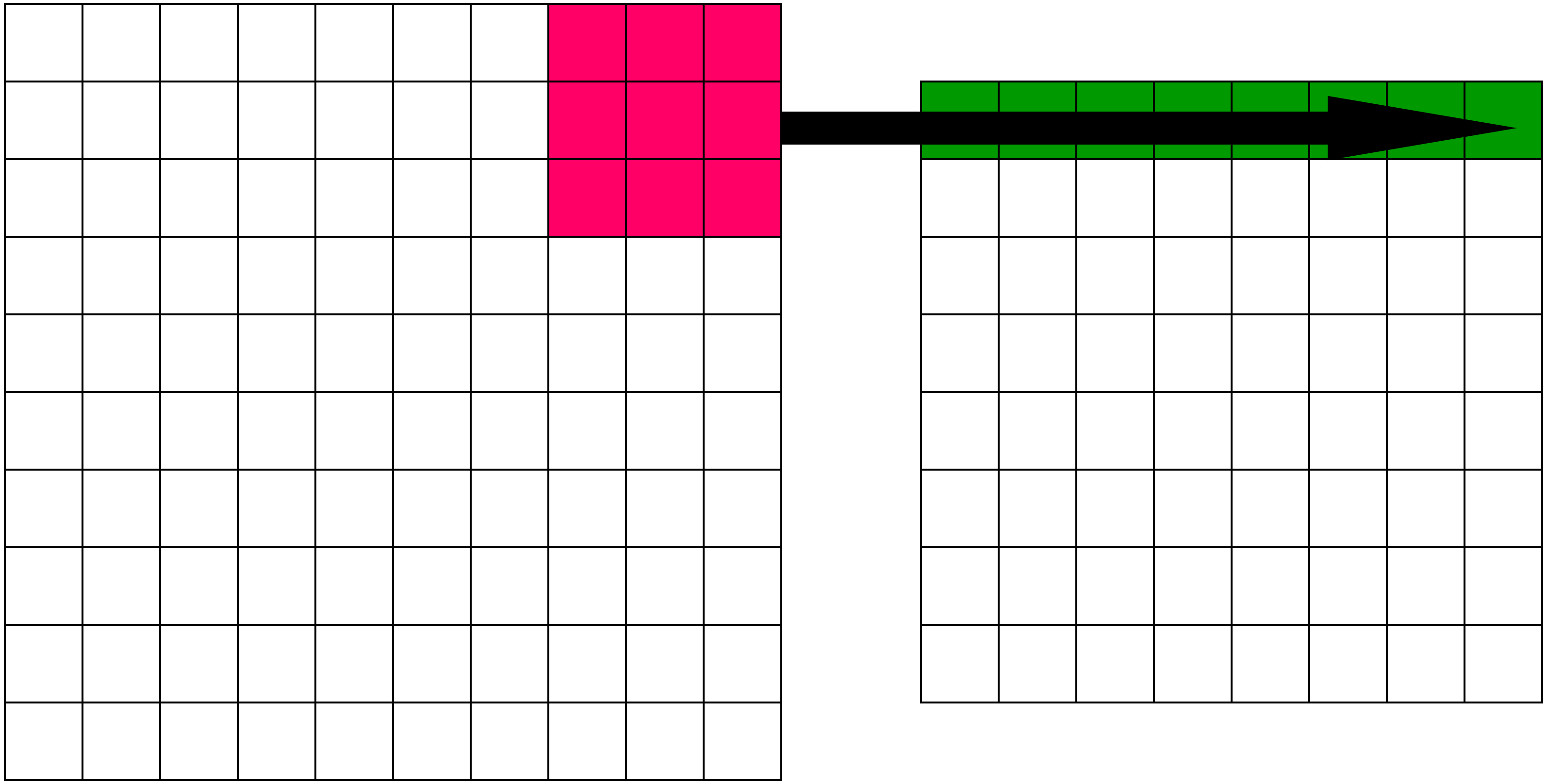


Input image

Output image

$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

# "Valid" convolution

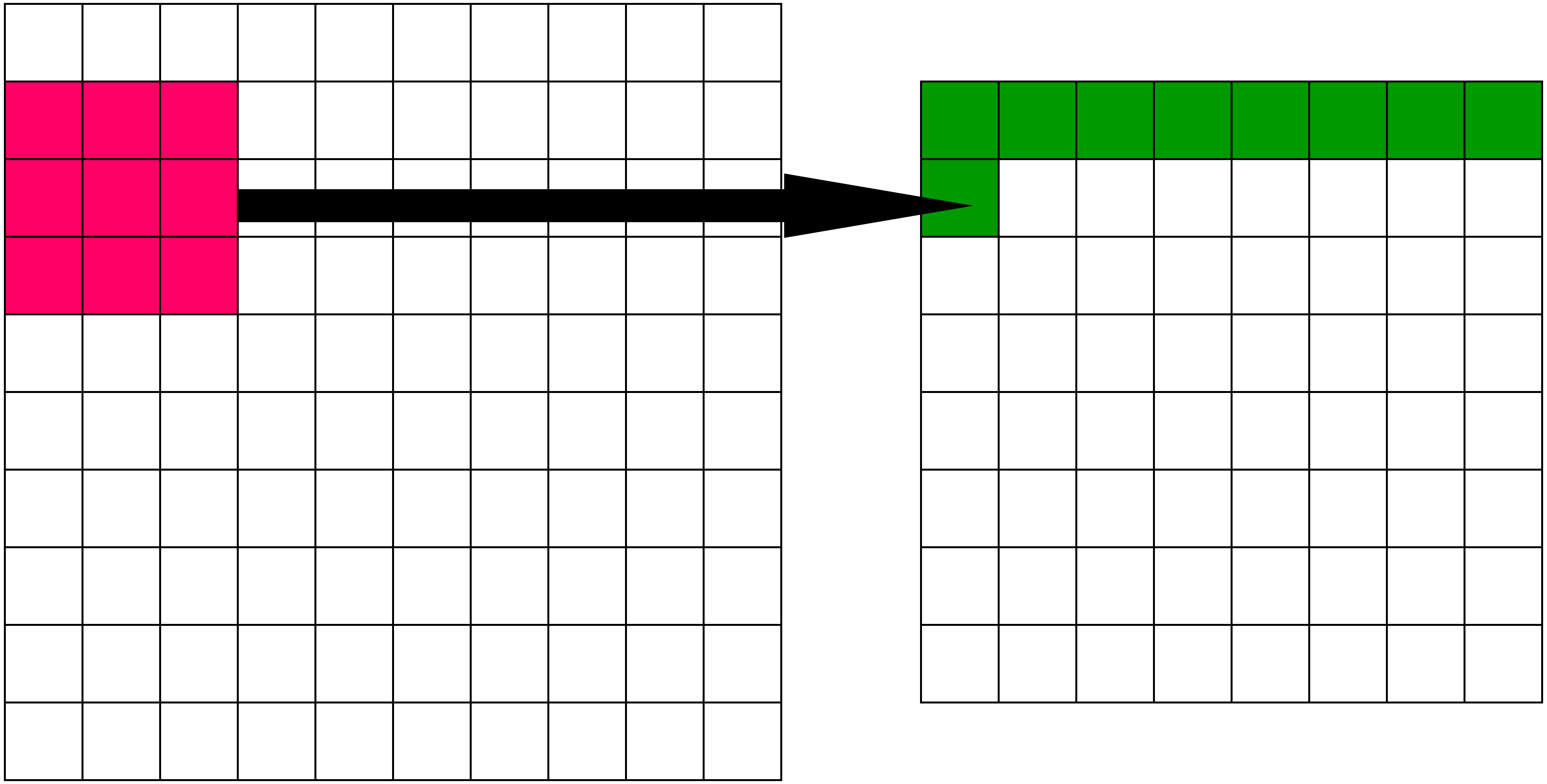


Input image

Output image

$$w(x,y)\bigstar f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

# “Valid” convolution

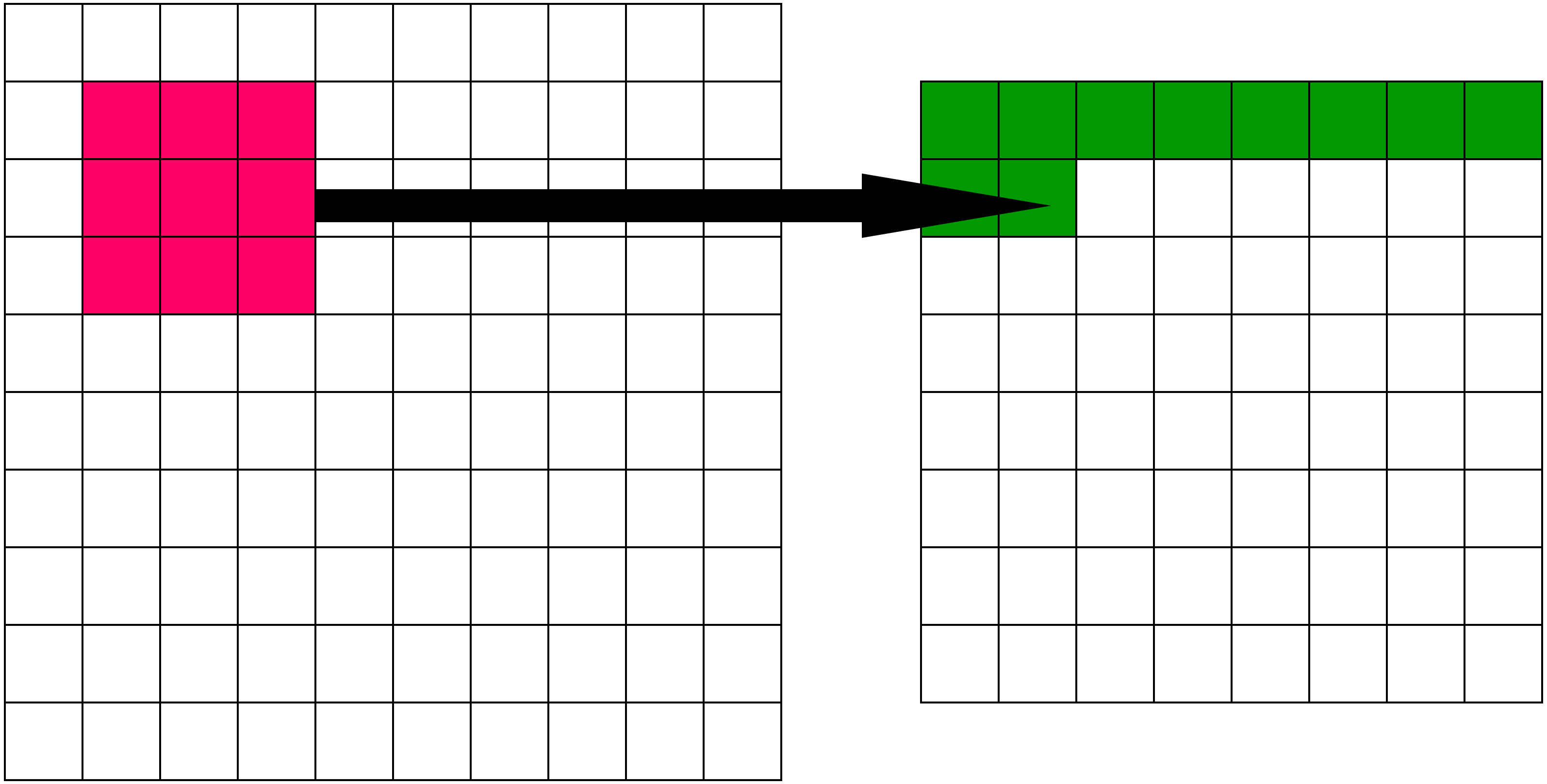


Input image

Output image

$$w(x,y)\bigstar f(x,y) = \sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

# “Valid” convolution

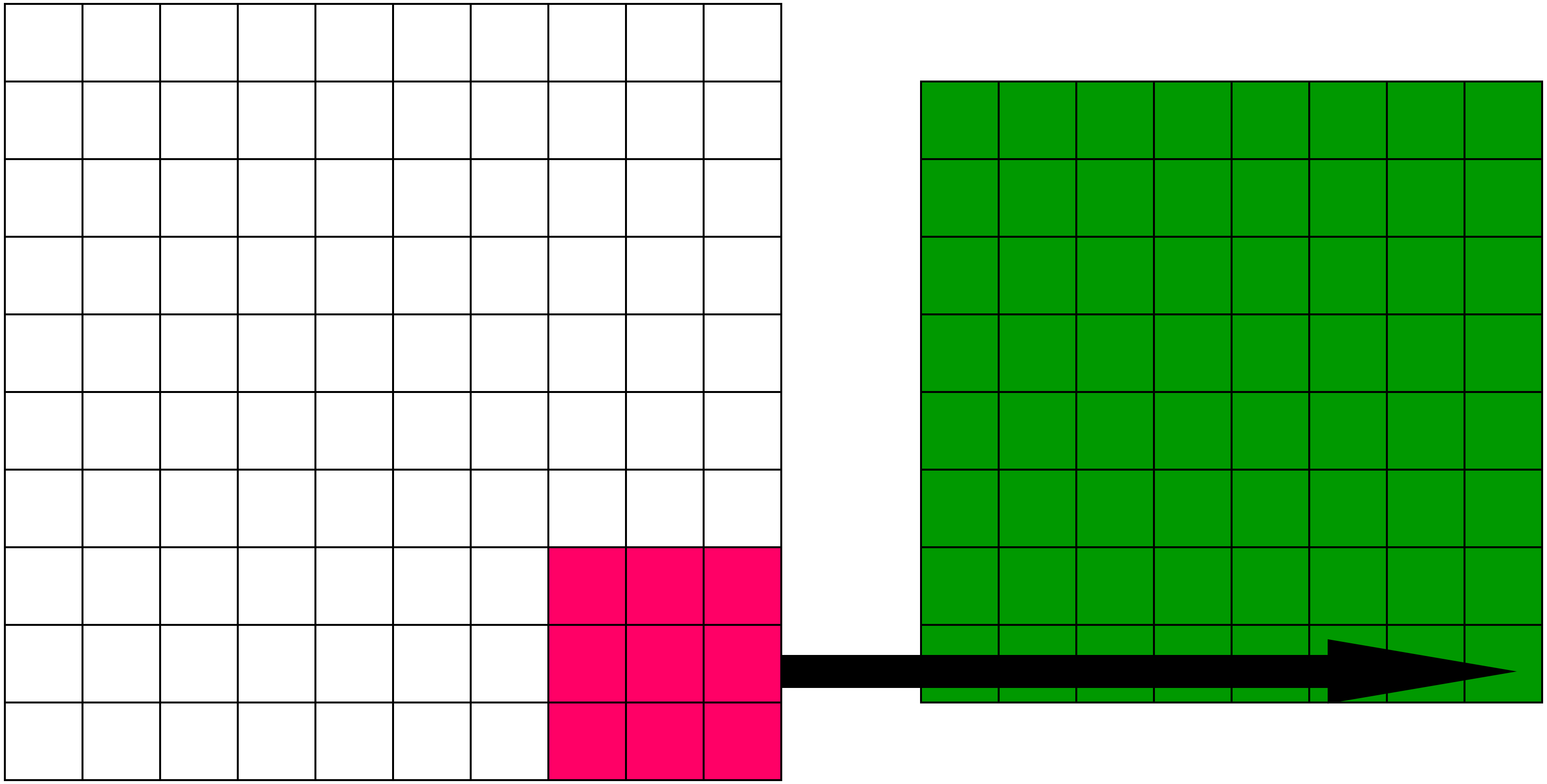


Input image

Output image

$$w(x,y)\bigstar f(x,y)=\sum_{s=-a}^{a}\sum_{t=-b}^{b}w(s,t)f(x-s,y-t)$$

# “Valid” convolution



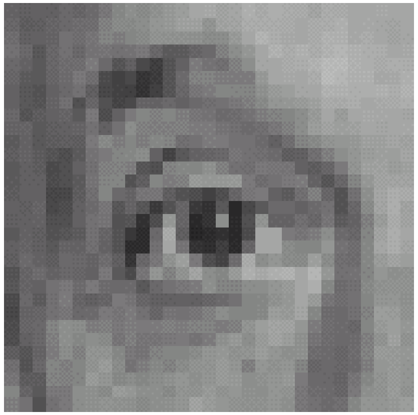
Input image

Output image

$$w(x,y)\bigstar f(x,y) = \sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)f(x-s,y-t)$$

# Correlation

## Linear filtering (warm-up slide)



original

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

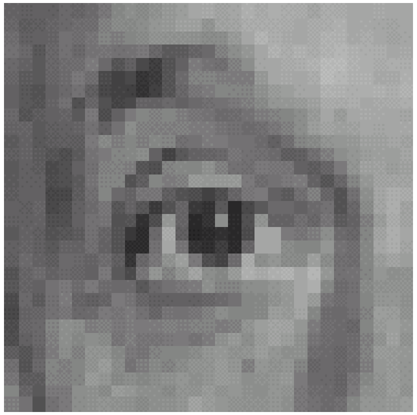
Pixel offset

?

(Bill Freeman)

# Correlation

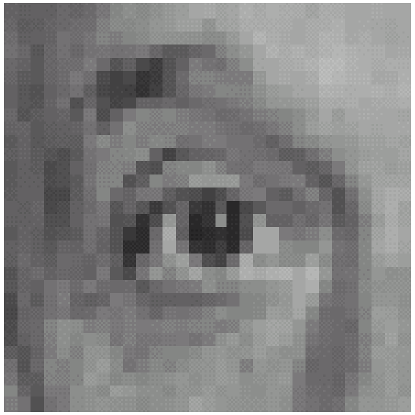
## Linear filtering (warm-up slide)



original

|   |   |   |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

Pixel offset

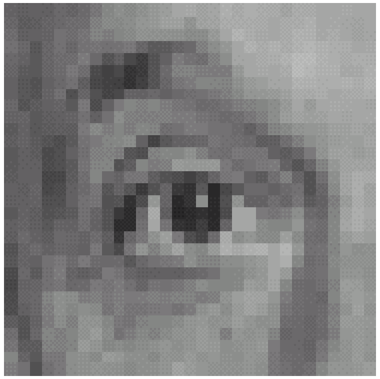


Filtered
(no change)

(Bill Freeman)

# Correlation

## Linear filtering



original

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 0 |

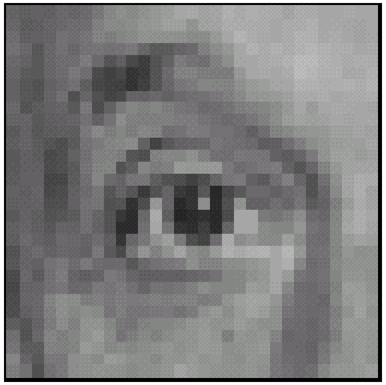
Pixel offset

?

(Bill Freeman)

# Correlation

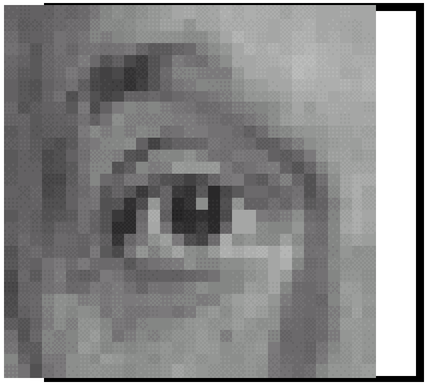
## Shift



original

|  |  |  |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 0 |

Pixel offset

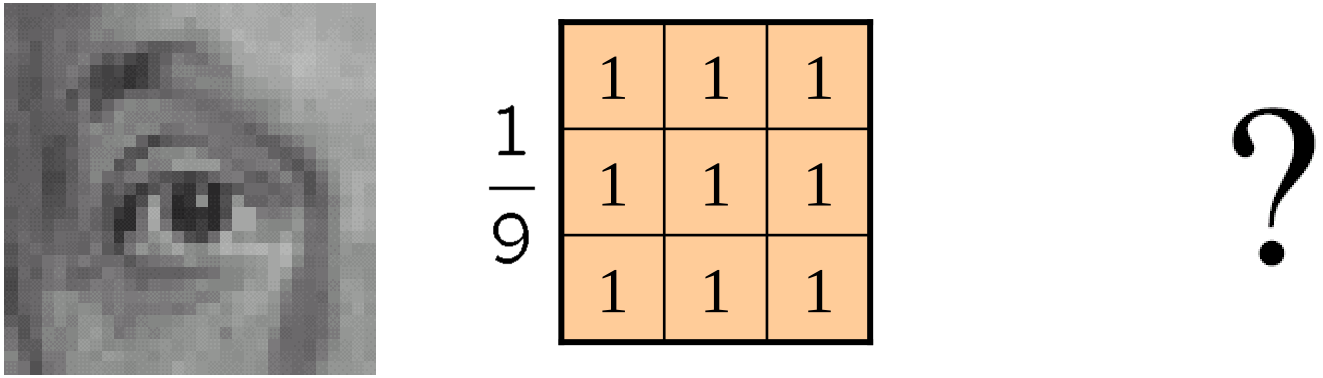


Shifted one
Pixel to the left

(Bill Freeman)

# Correlation

## Linear filtering

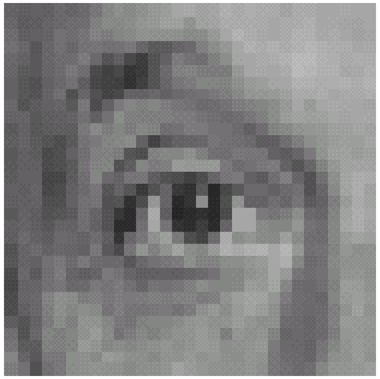


original

(Bill Freeman)

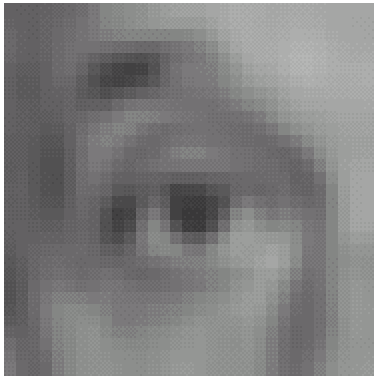
# Correlation

## Blurring



original

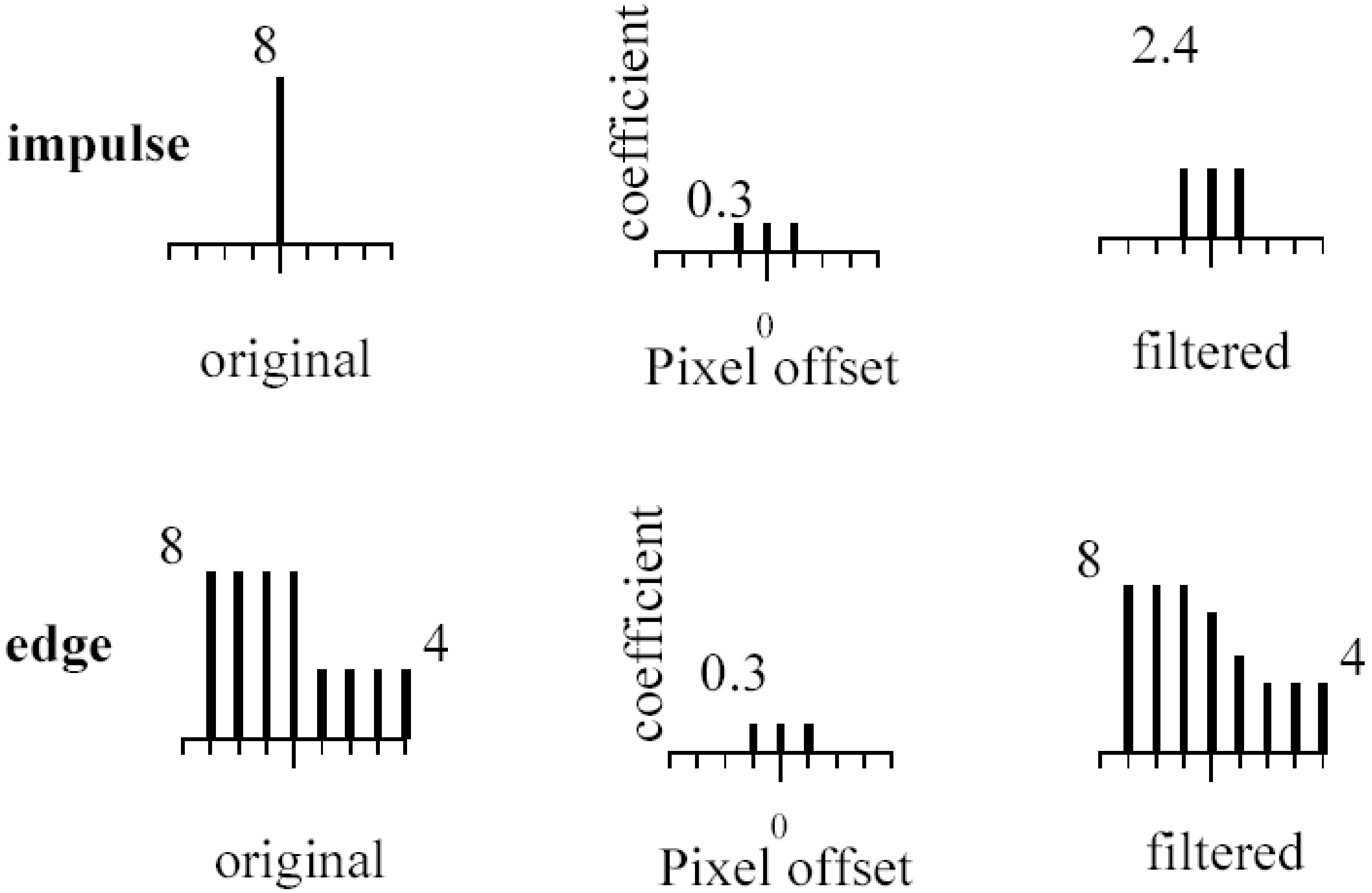
$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$



Blurred (filter applied in both dimensions).

(Bill Freeman)

# Blur Examples
# 1-Dimensional Correlation



# Practice with linear filters

Original

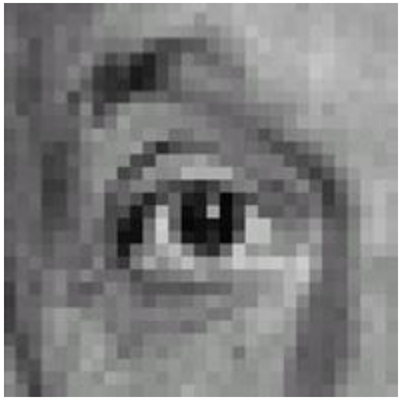
$$\left[ \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \right] \qquad ?$$

Source: D. Lowe

# Practice with linear filters

Original

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

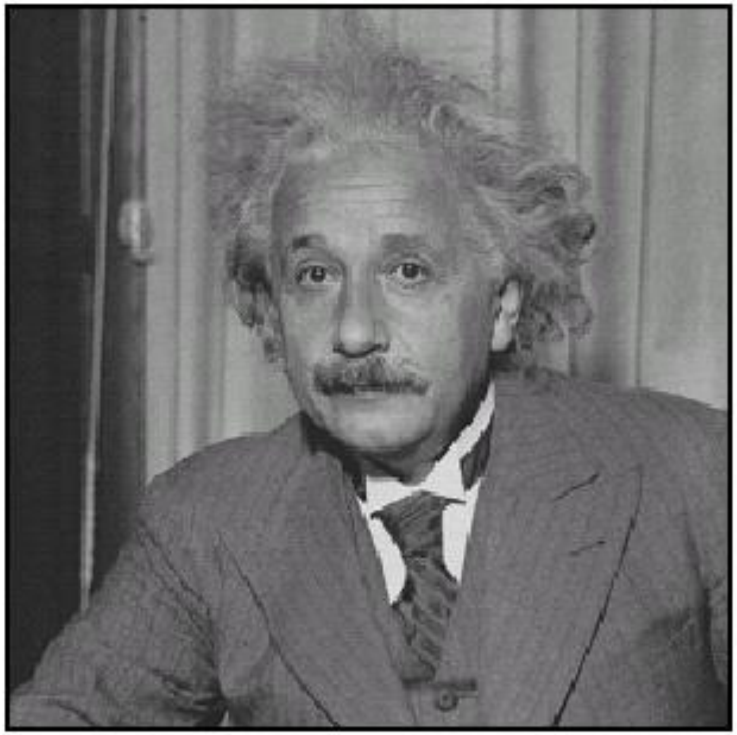


Sharpening filter

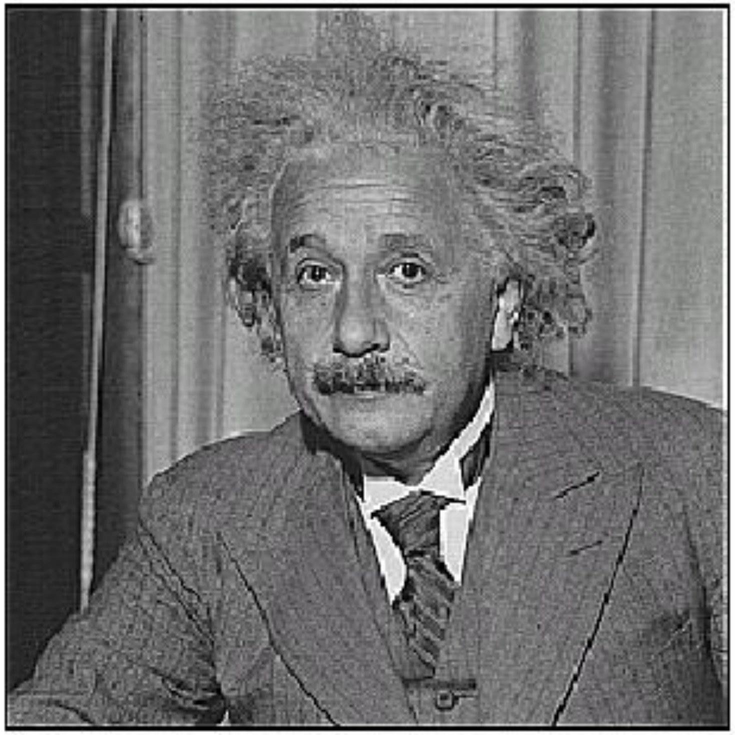
- Accentuates differences with local average

Source: D. Lowe

# Sharpening

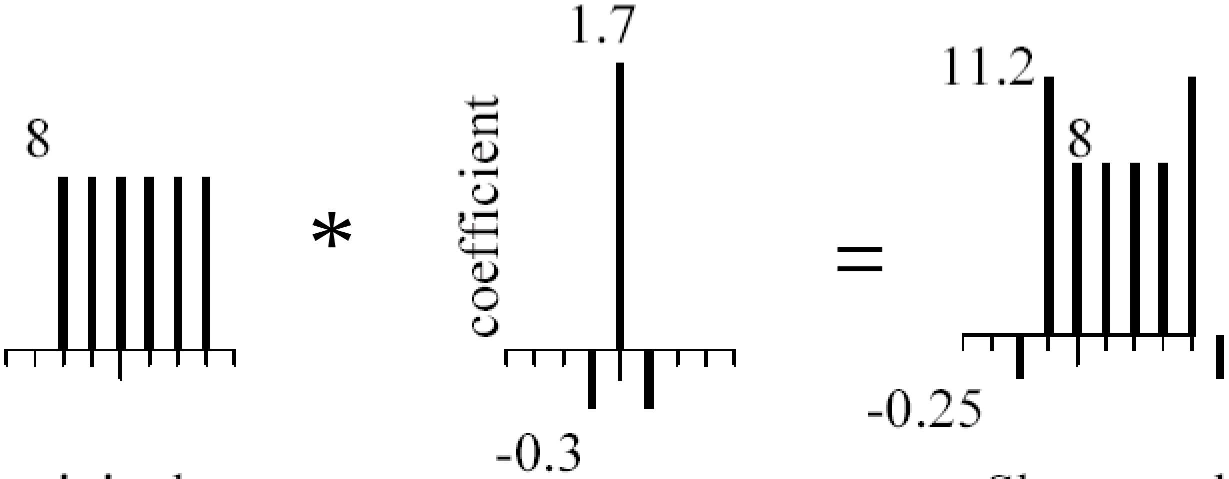


**before**



**after**

# Sharpening example



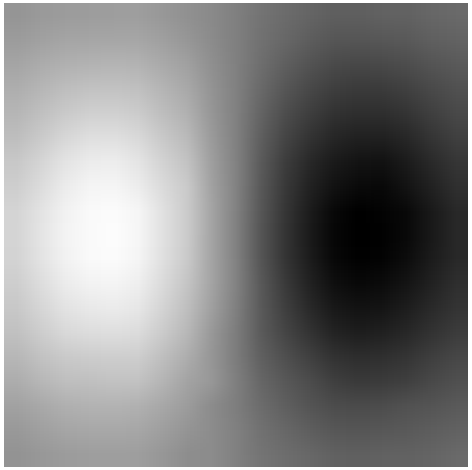
original

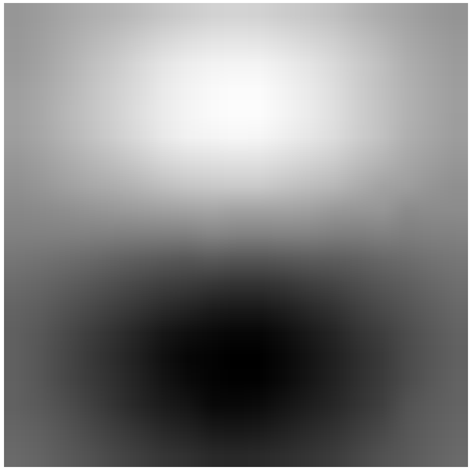
Sharpened (differences are accentuated; constant areas are left untouched).

# 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

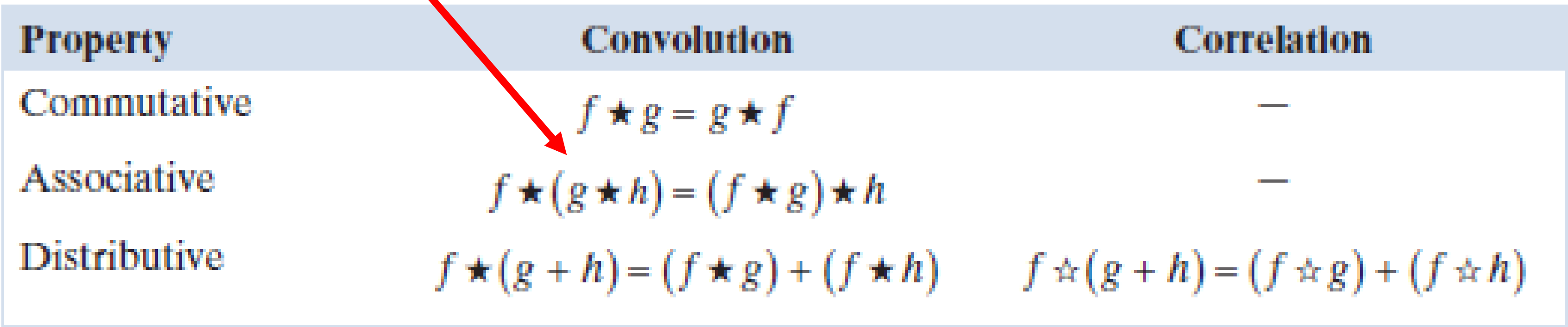
- Insight, for corelation
  - filters look like the effects they are intended to find
  - filters find effects they look like





# Correlation and convolution

This is equivalent to applying one filter



| Property | Convolution | Correlation |
|---|---|---|
| Commutative | $f \bigstar g = g \bigstar f$ | — |
| Associative | $f \bigstar (g \bigstar h) = (f \bigstar g) \bigstar h$ | — |
| Distributive | $f \bigstar (g + h) = (f \bigstar g) + (f \bigstar h)$ | $f \star (g + h) = (f \star g) + (f \star h)$ |

- Convolution is commutative and associative, correlation is not

# Convolution properties (cont.)

- Distributes over addition:

  $I * (k_1 + k_2) = (I * k_1) + (I * k_2)$

- Scalars factor out:
  - For scalar s

  $s\,(f * I) = (sf) * I = f * (sI)$

- Identity:

  unit impulse $e = [0, 0, 1, 0, 0]$

  $I * e = I$

Source: S. Lazebnik

# Properties of Continuous Convolution

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

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

# Convolutional Neural Networks

- Core operation in CNN is, not surprisingly, convolution.
- During training of a CNN, the weights of the convolution kernels are learned.
- 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

# Filtering to reduce noise

- Noise is what we are not interested in
  - Usually think of simple, low-level noise:
    - Light fluctuations; Sensor noise; Quantization effects; Finite precision
  - Complex noise: shadows; extraneous objects
- A pixel’s neighborhood contains information about its color and intensity
- Averaging noise reduces its effect

# Additive noise

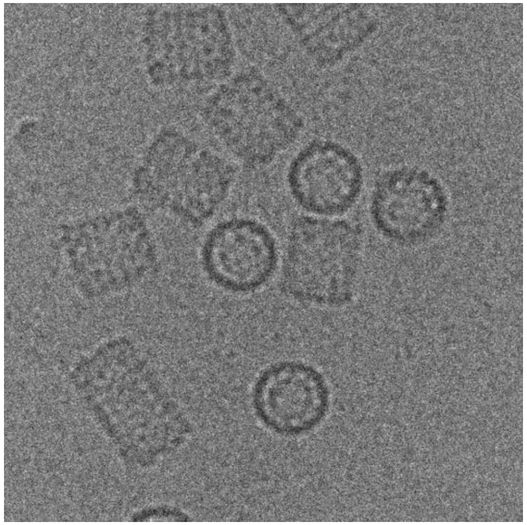
- I = S + N. Noise doesn’t depend on signal.
- We’ll consider:

$I_i = s_i + n_i$ 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

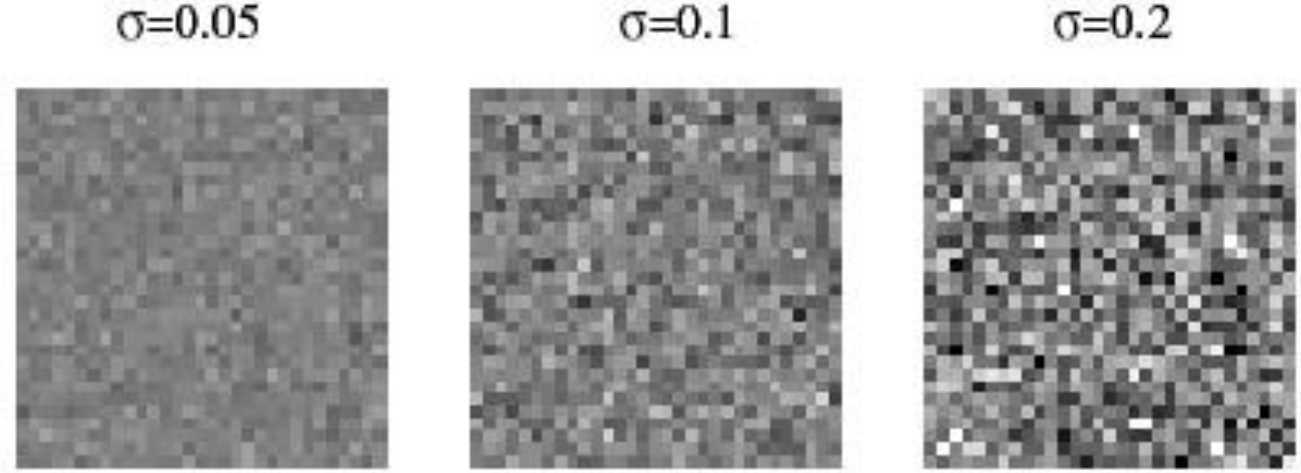
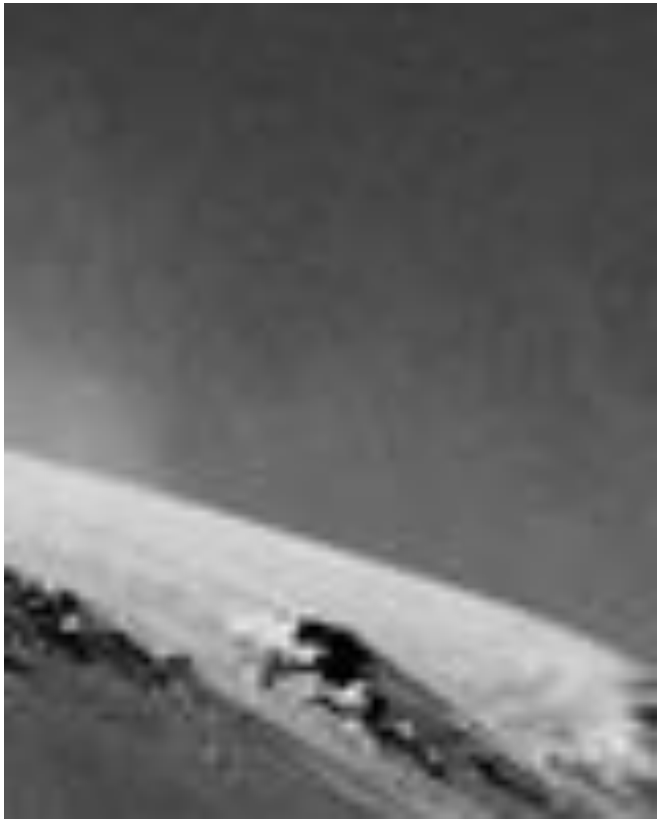


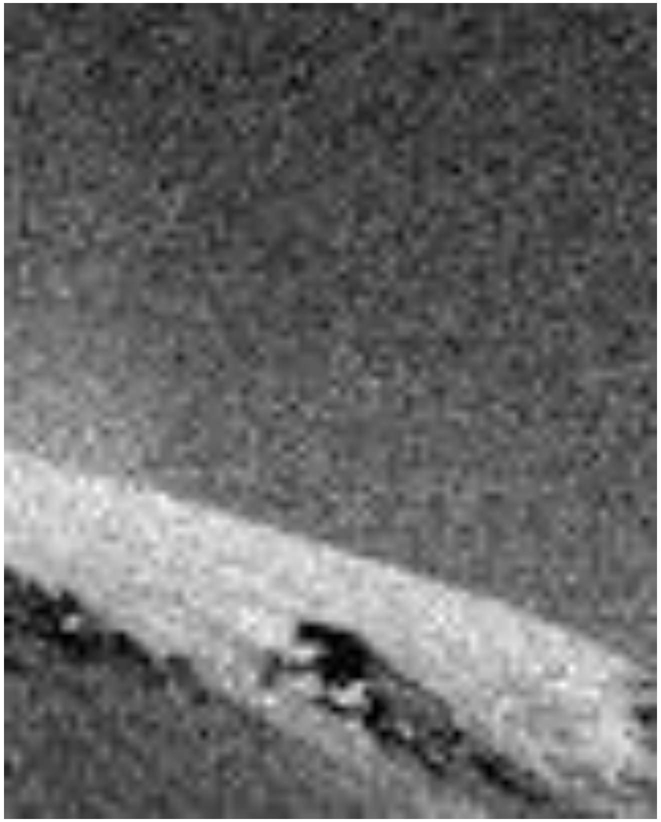
Image is constant with $I_i = 0.5$

Gaussian noise, $n_i$ drawn from Gaussian distribution with zero mean and standard deviation $\sigma$

Gaussian Noise:
sigma=1



Gaussian Noise:
sigma=16



# Averaging Filter

- Mask with positive entries, that sum 1
- Replaces each pixel with an average of its neighborhood
- If all weights are equal, it is called a Box filter

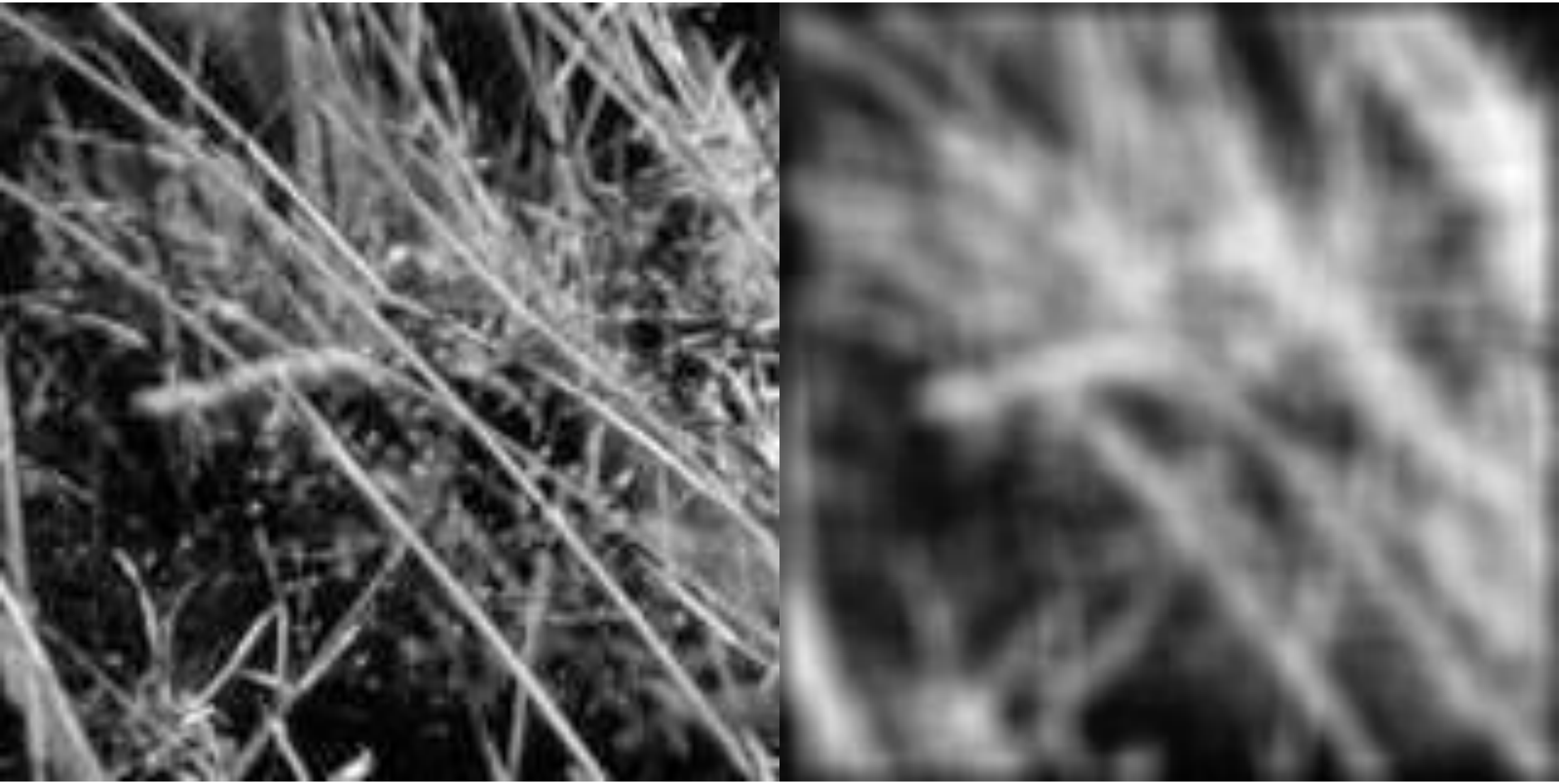
F

$$1/9 \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

(Camps)

# Smoothing by Averaging

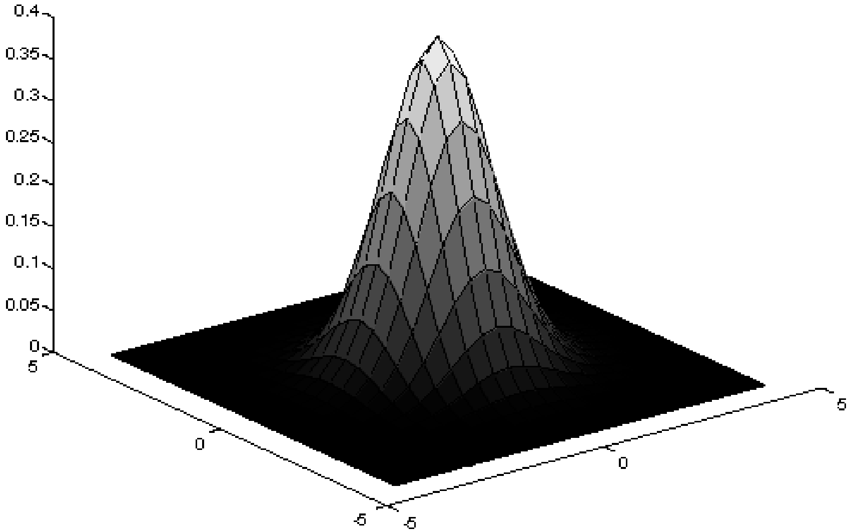
Kernel: ■

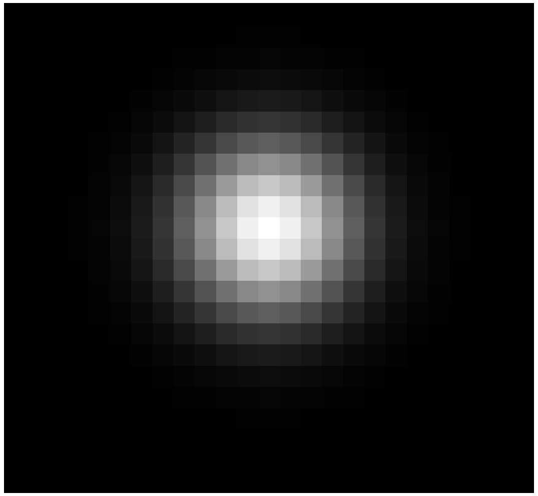


# An Isotropic Gaussian Kernel

- Circularly symmetric Gaussian with variance $\sigma^2$

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

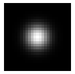


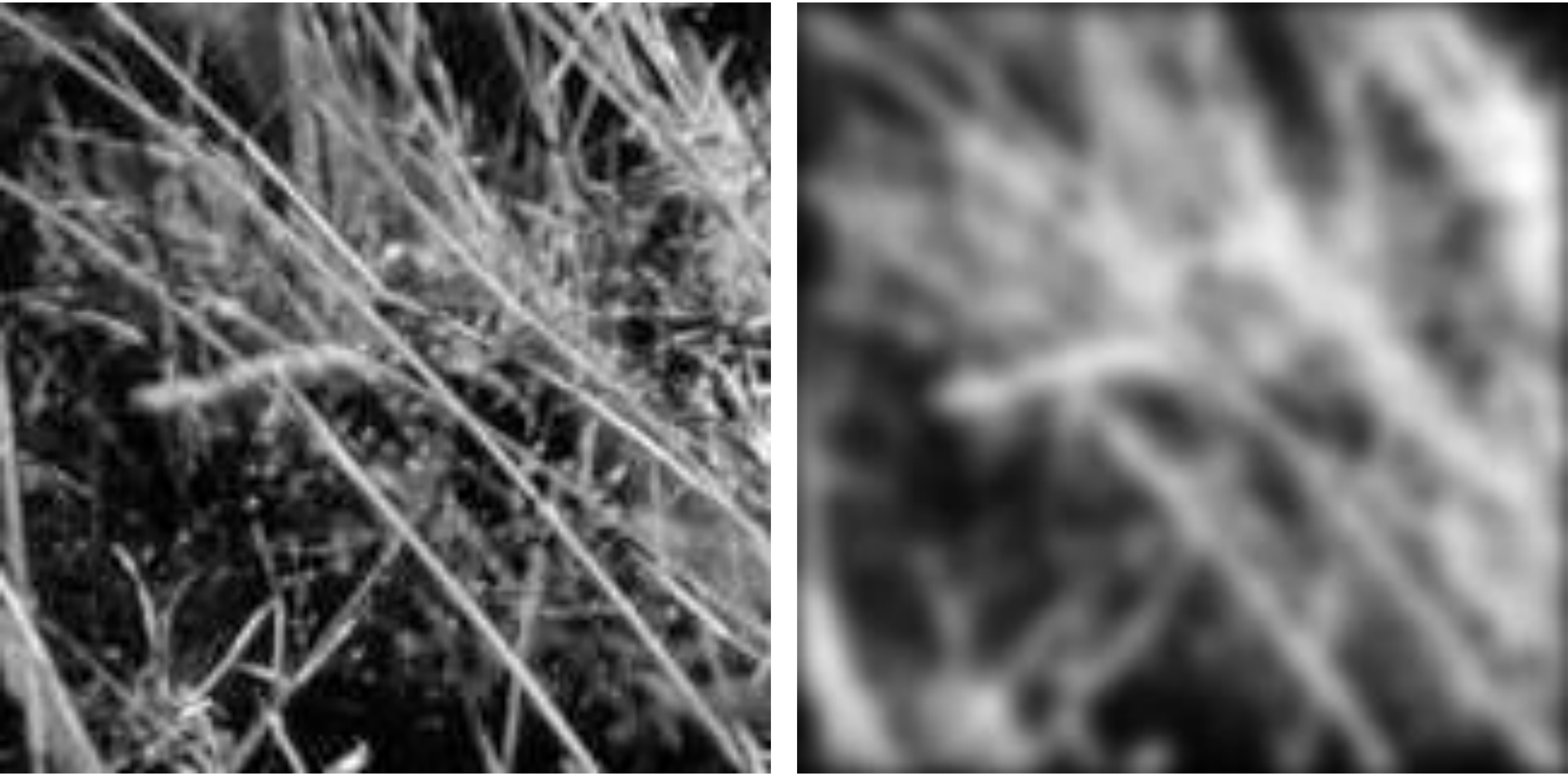


$\frac{1}{273}$

| | | | | |
|---|---|---|---|---|
| 1 | 4 | 7 | 4 | 1 |
| 4 | 16 | 26 | 16 | 4 |
| 7 | 26 | 41 | 26 | 7 |
| 4 | 16 | 26 | 16 | 4 |
| 1 | 4 | 7 | 4 | 1 |

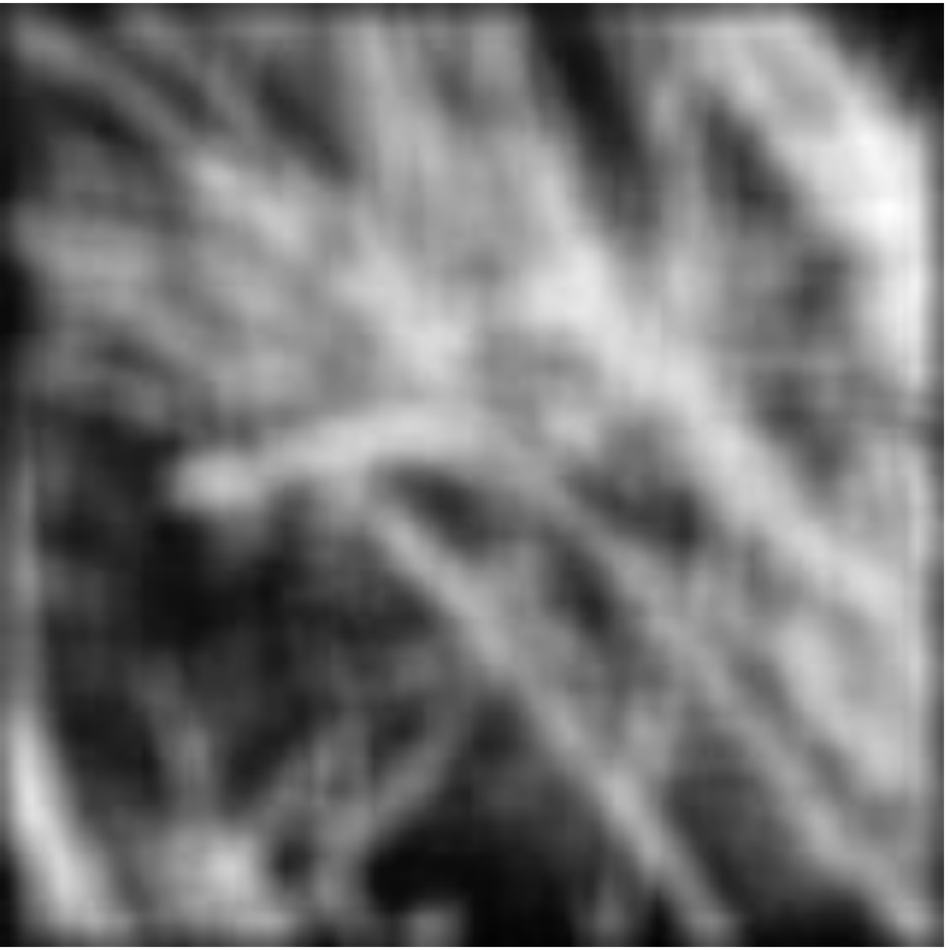
# Smoothing with a Gaussian

Kernel: 

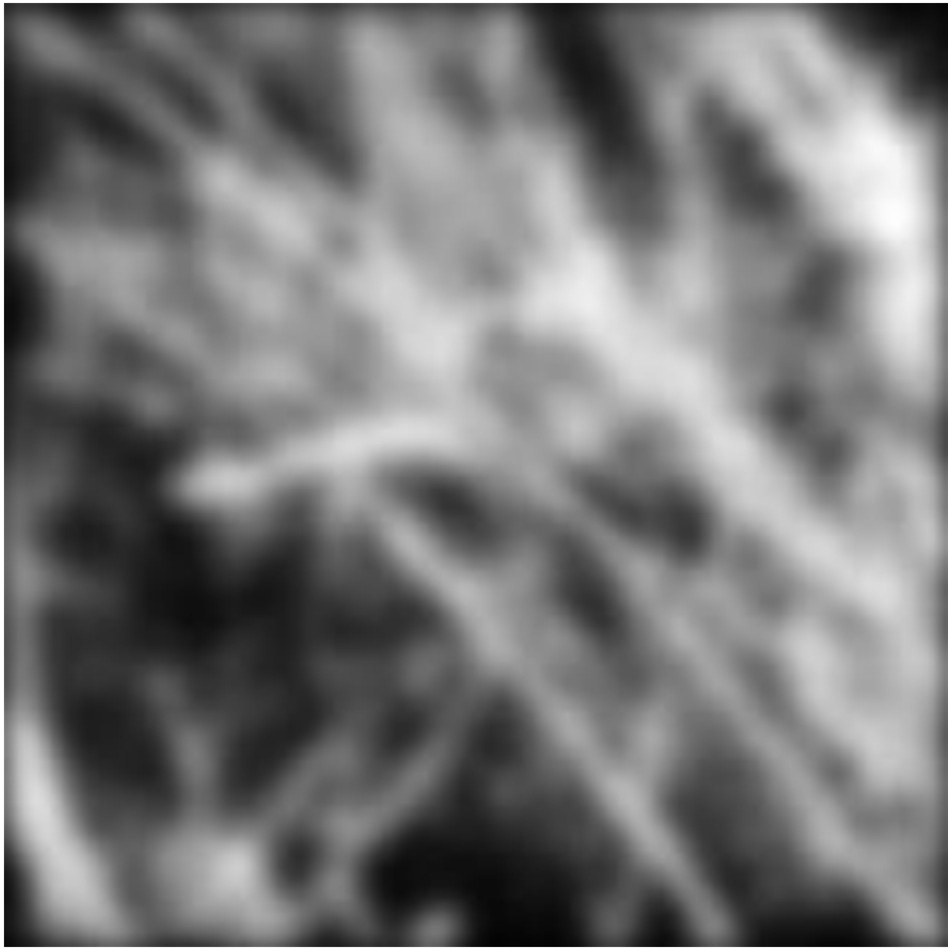


# Smoothing

Box filter

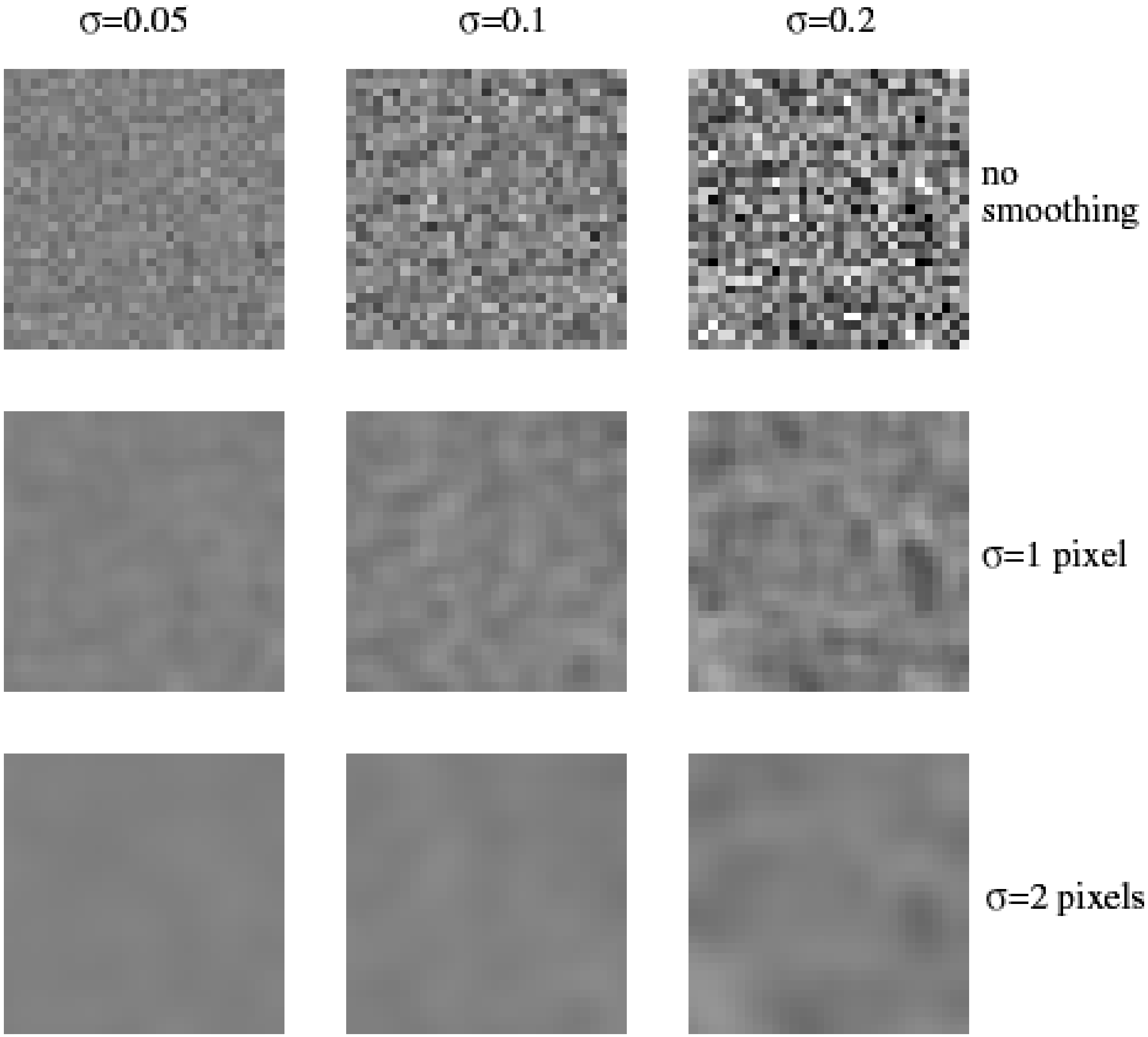


Gaussian filter



# The effects of smoothing

Increased Noise →



Increased Smoothing ↓

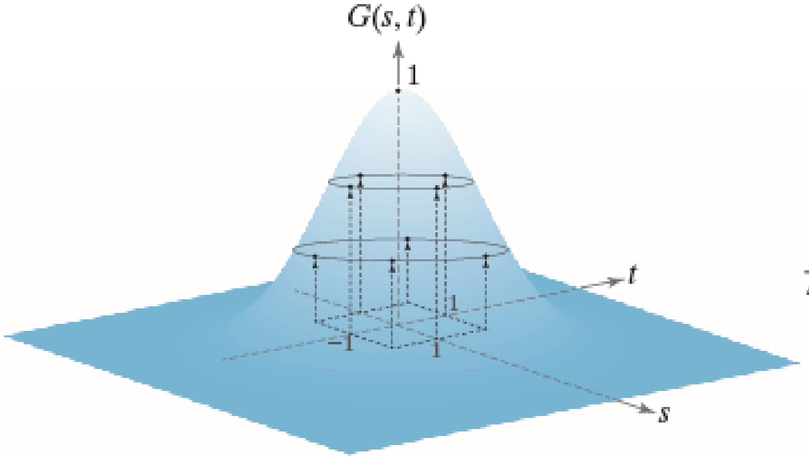
**An image with constant intensity + noise** :

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 $3 \times 3$ kernel [this is the same as Fig. 3.37(b)].



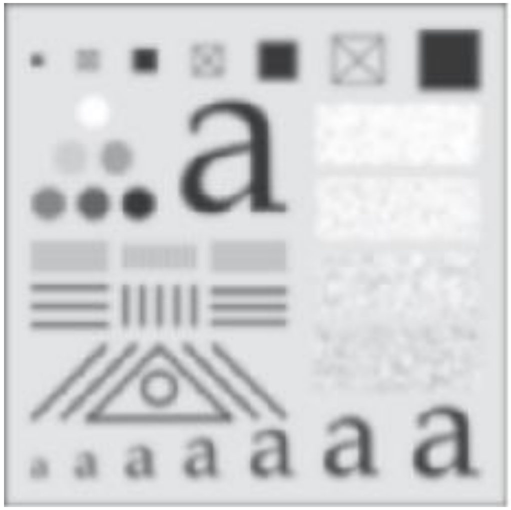
$$\frac{1}{4.8976} \times \begin{array}{|c|c|c|} \hline 0.3679 & 0.6065 & 0.3679 \\ \hline 0.6065 & 1.0000 & 0.6065 \\ \hline 0.3679 & 0.6065 & 0.3679 \\ \hline \end{array}$$

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

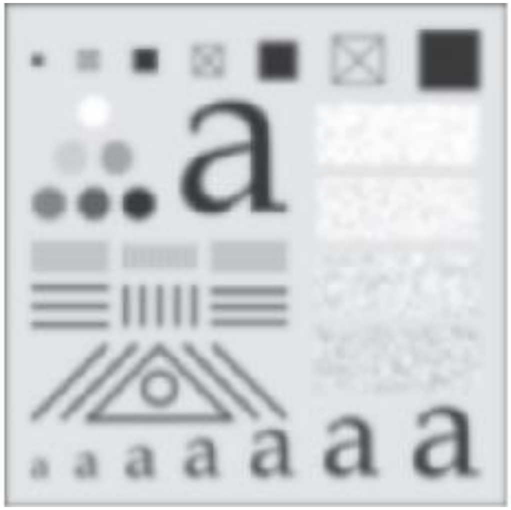
Volume under surface greater than 3σ is negligible

# Smoothing with Gaussian kernel

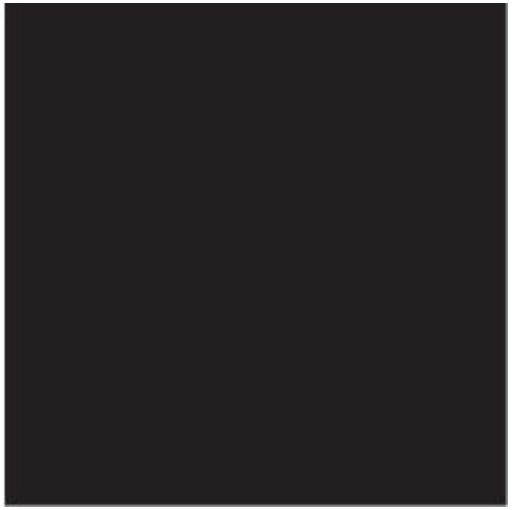
$$\text{minimum size is } \begin{cases} \lceil 6\sigma \rceil & \text{if } \lceil 6\sigma \rceil \text{ is odd} \\ \lceil 6\sigma \rceil + 1 & \text{otherwise} \end{cases}$$



σ = 7
43x43

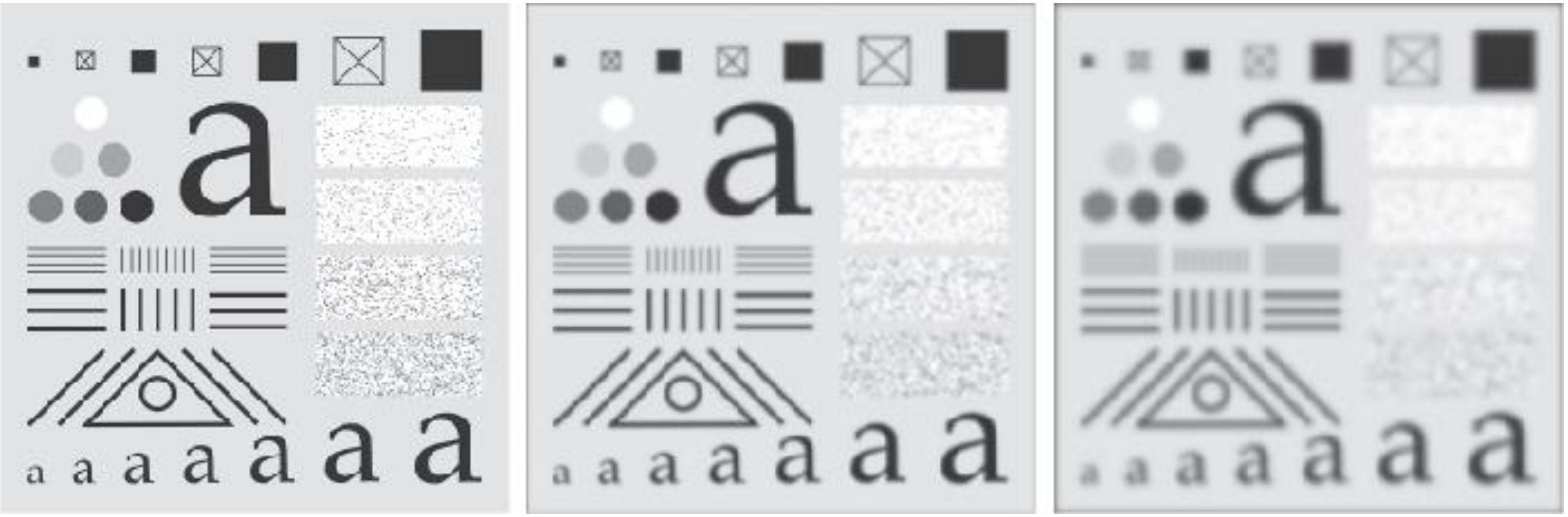


σ = 7
85x85



Difference

# Smoothing with Gaussian kernel



Input image

σ = 3.5
21x21

σ = 7
43x43

# Border padding

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| v | v | v | v | v | v | v | v | v |
| v | v | v | v | v | v | v | v | v |
| v | v | 1 | 2 | 3 | 4 | 5 | v | v |
| v | v | 6 | 7 | 8 | 9 | 10 | v | v |
| v | v | 11 | 12 | 13 | 14 | 15 | v | v |
| v | v | 16 | 17 | 18 | 19 | 20 | v | v |
| v | v | v | v | v | v | v | v | v |

Zero padding
when v = 0

Constant padding

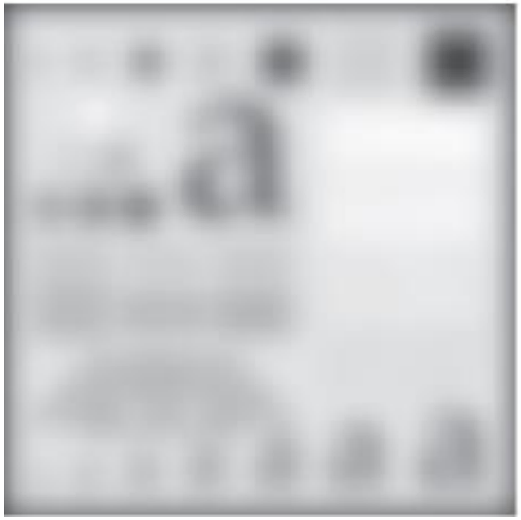
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 2 | 3 | 4 | 5 | 5 | 5 |
| 1 | 1 | 1 | 2 | 3 | 4 | 5 | 5 | 5 |
| 1 | 1 | 1 | 2 | 3 | 4 | 5 | 5 | 5 |
| 6 | 6 | 6 | 7 | 8 | 9 | 10 | 10 | 10 |
| 11 | 11 | 11 | 12 | 13 | 14 | 15 | 15 | 15 |
| 16 | 16 | 16 | 17 | 18 | 19 | 20 | 20 | 20 |
| 16 | 16 | 16 | 17 | 18 | 19 | 20 | 20 | 20 |

Replicate padding

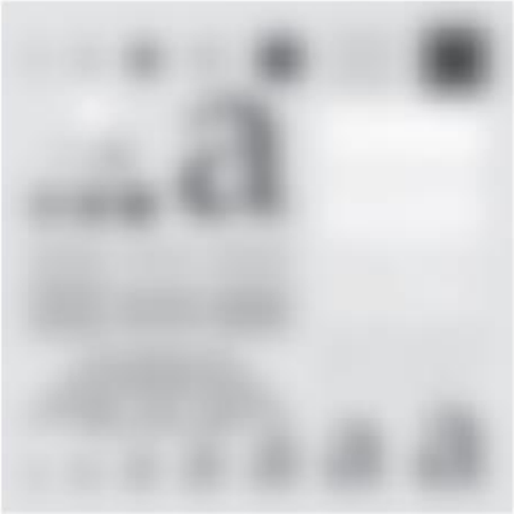
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 13 | 12 | 11 | 12 | 13 | 14 | 15 | 14 | 13 |
| 8 | 7 | 6 | 7 | 8 | 9 | 10 | 9 | 8 |
| 3 | 2 | 1 | 2 | 3 | 4 | 5 | 4 | 3 |
| 8 | 7 | 6 | 7 | 8 | 9 | 10 | 9 | 8 |
| 13 | 12 | 11 | 12 | 13 | 14 | 15 | 14 | 13 |
| 18 | 17 | 16 | 17 | 18 | 19 | 20 | 19 | 18 |
| 13 | 12 | 11 | 12 | 13 | 14 | 15 | 14 | 13 |

Mirror padding

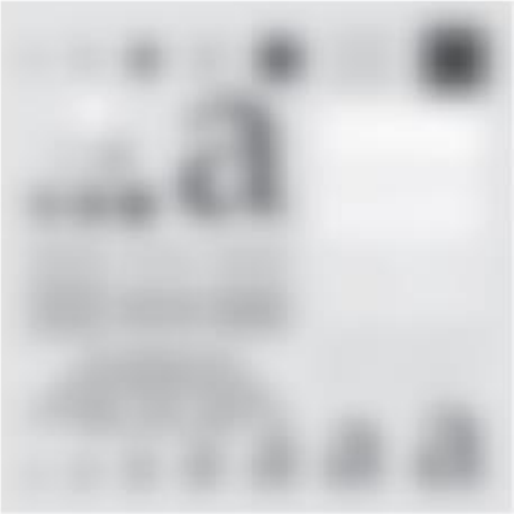
# Border padding



Zero padding

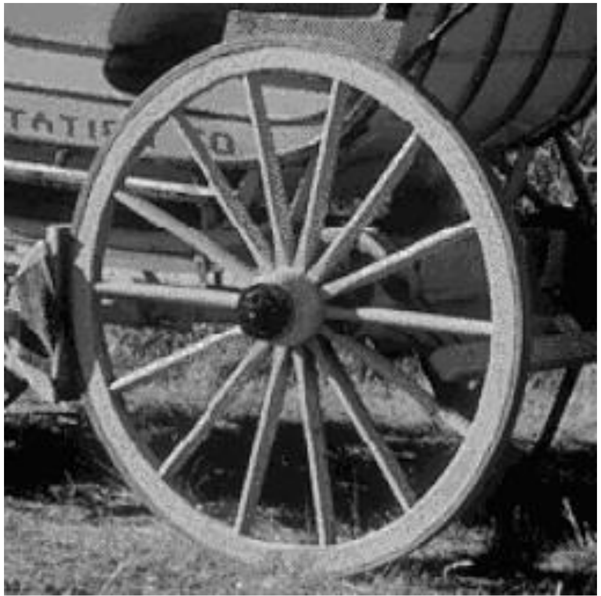


Mirror padding

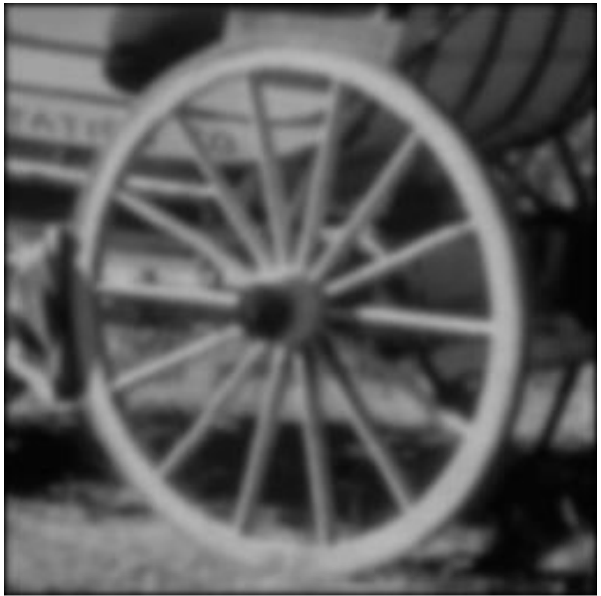


Replicate padding

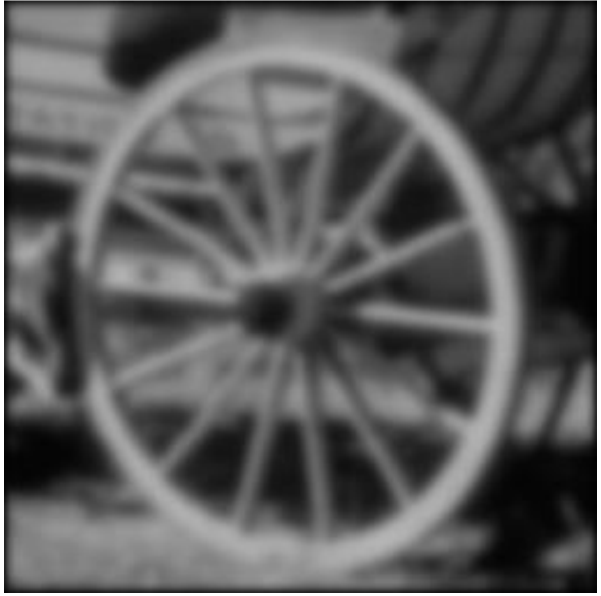
# Gaussian Smoothing



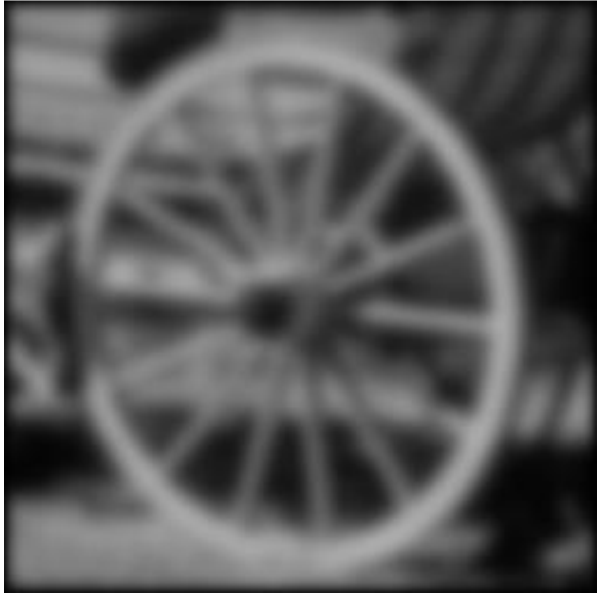
original



$s = 2$



$s = 2.8$



$s = 4$

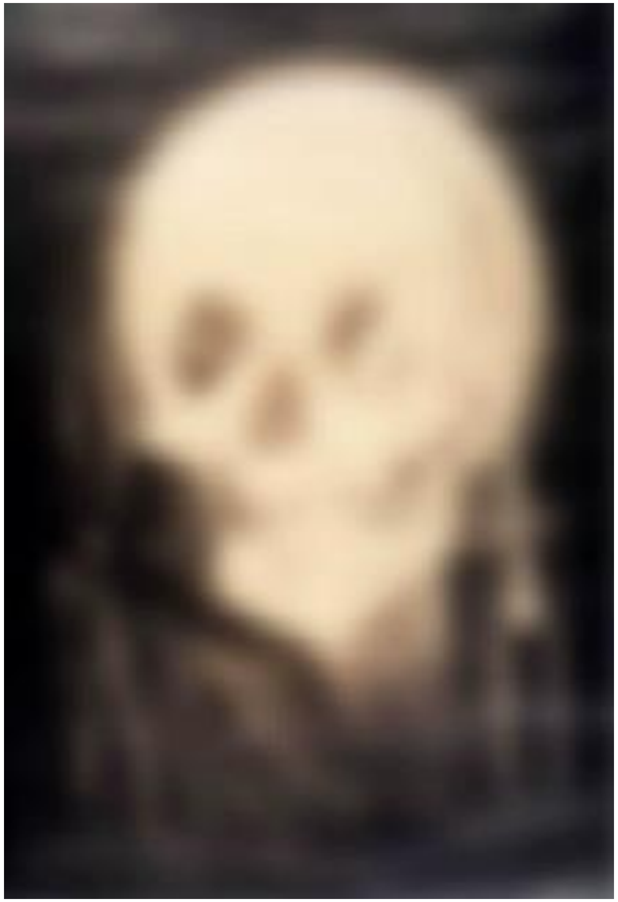
# Gaussian Smoothing

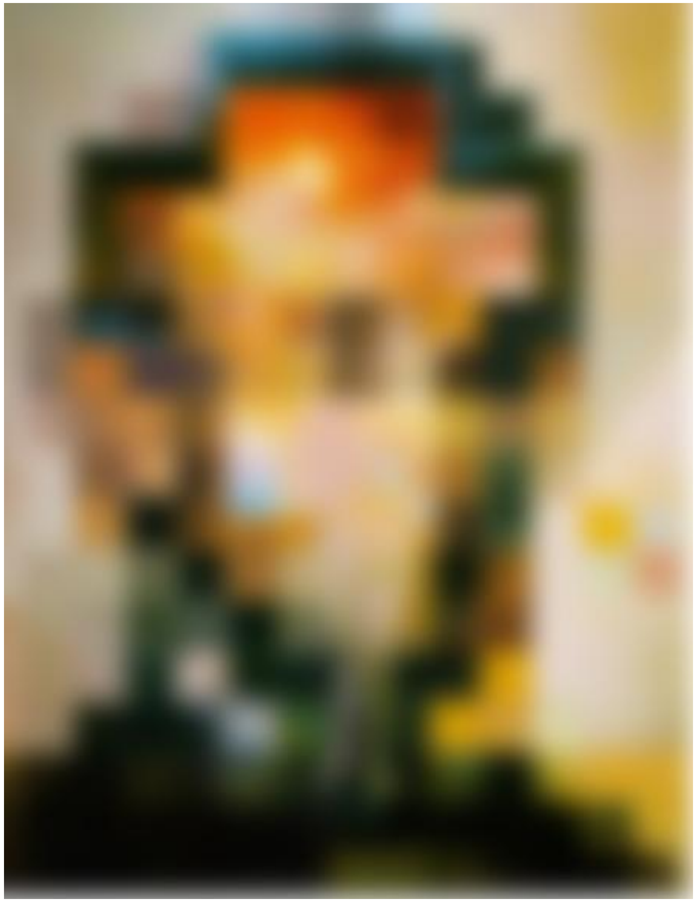
by Charles Allen Gillbert

by Harmon & Julesz

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

# Gaussian Smoothing





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

# Efficient Implementation

- Both, the Box filter and the Gaussian filter are separable:
  - First convolve each row with a 1D horizontal kernel
  - Then convolve each column with a 1D vertical kernel

# Overview: Image processing in the frequency domain

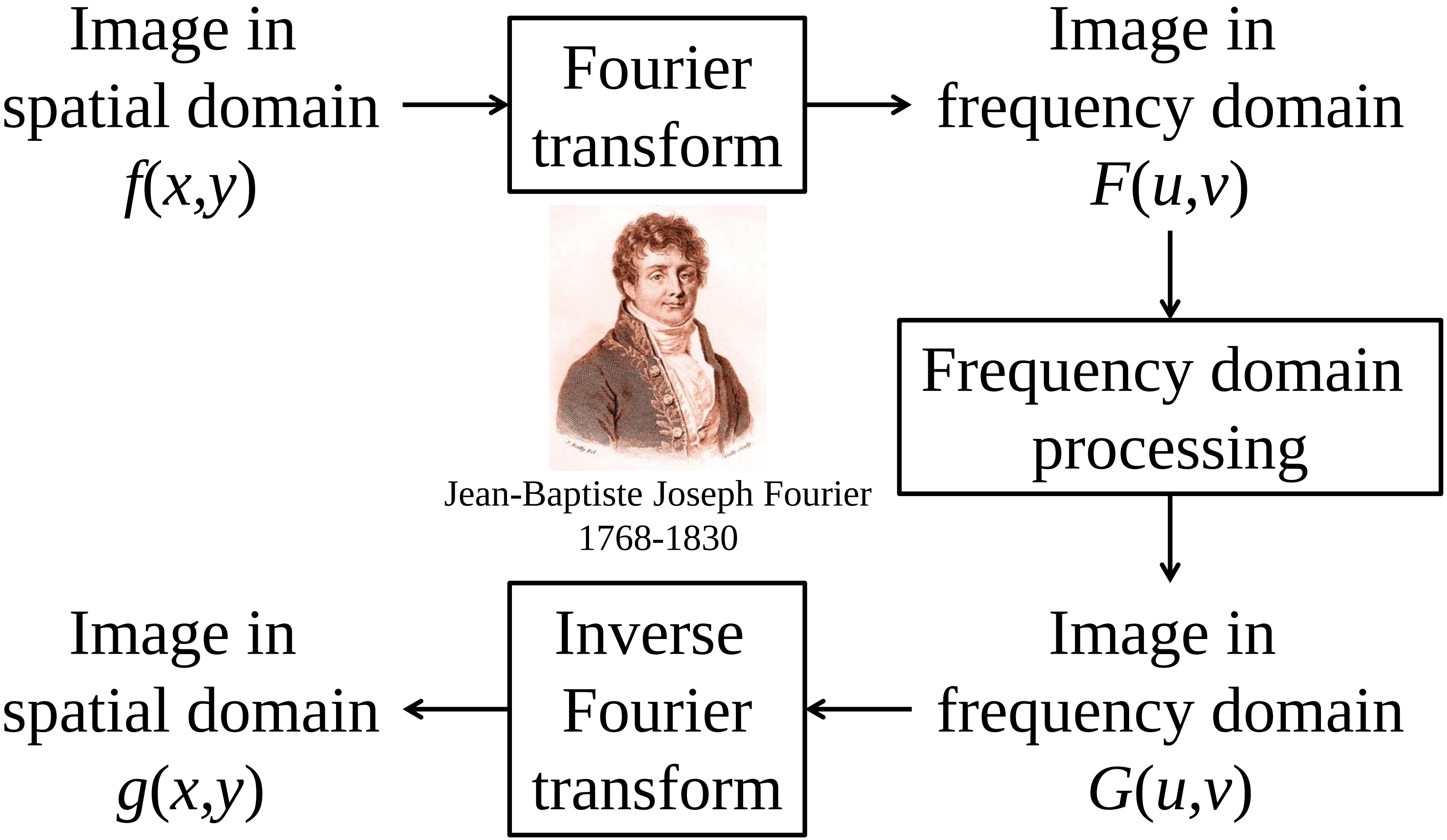


Image in spatial domain $f(x,y)$ → Fourier transform → Image in frequency domain $F(u,v)$

↓ Frequency domain processing ↓

Image in frequency domain $G(u,v)$ → Inverse Fourier transform → Image in spatial domain $g(x,y)$

Jean-Baptiste Joseph Fourier
1768-1830

# Fourier Transform

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

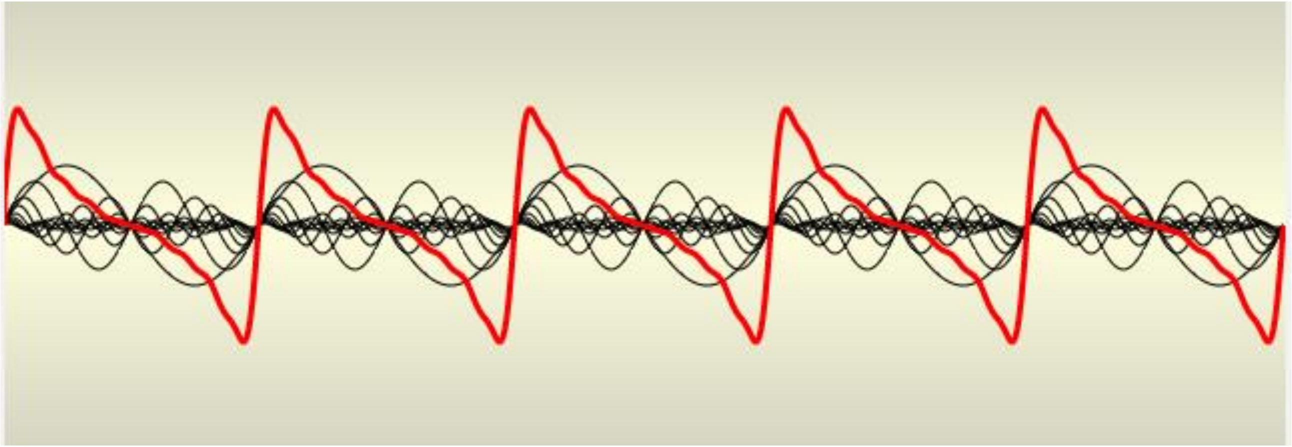
Time domain ←→ Frequency Domain

Real ←→Complex

- Consider time domain signal to be expressed as weighted sum of sinusoid. A sinusoid cos(ut+$\phi$) is characterized by its phase $\phi$ 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)

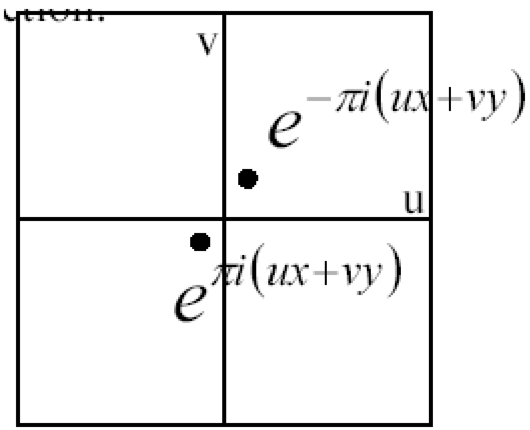
Fourier basis element

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

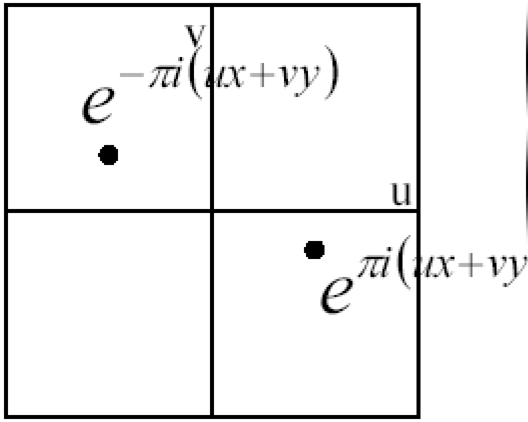
Transform is sum of orthogonal basis functions

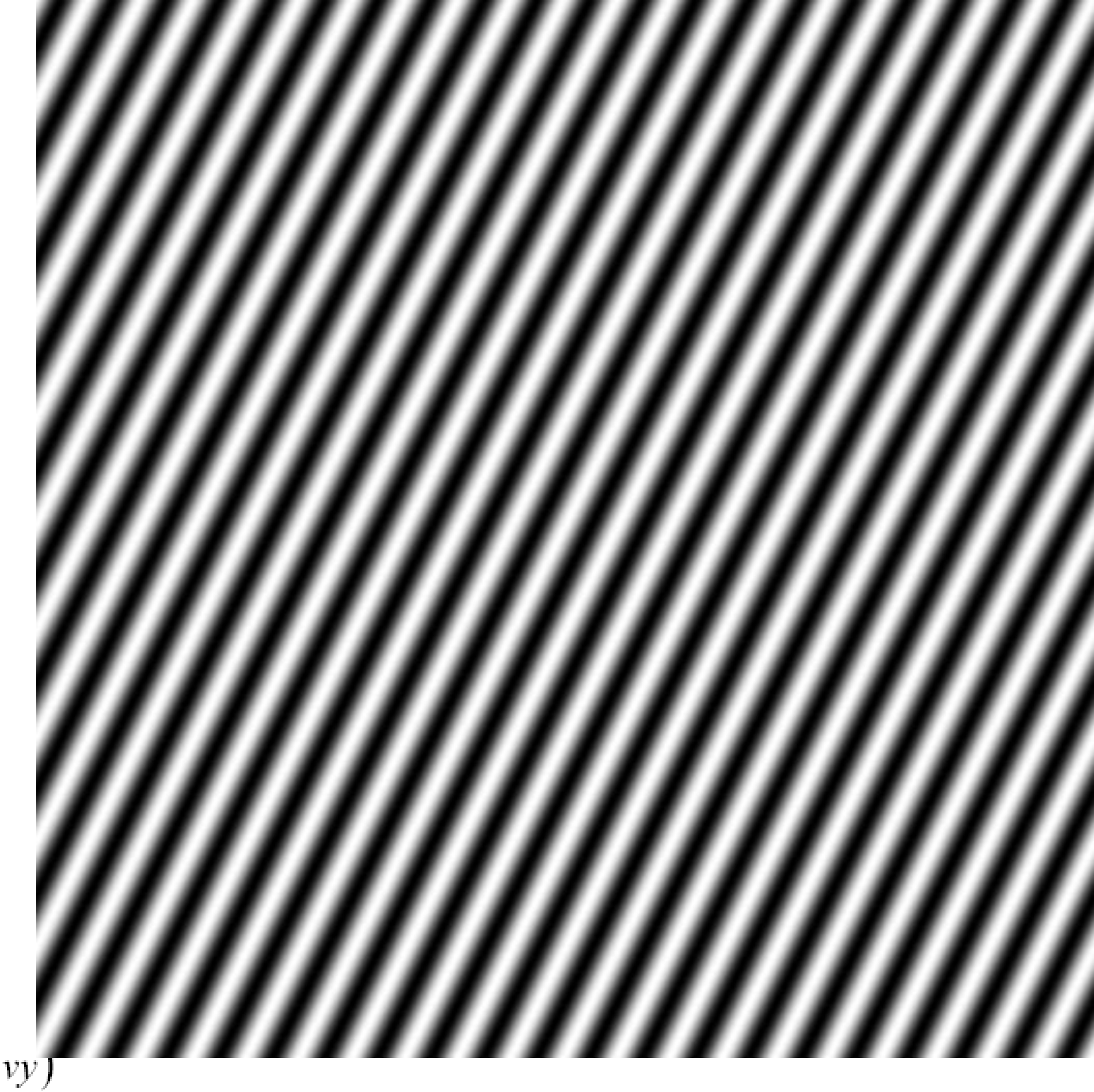
Vector (u,v)

- Magnitude gives frequency
- Direction gives orientation.

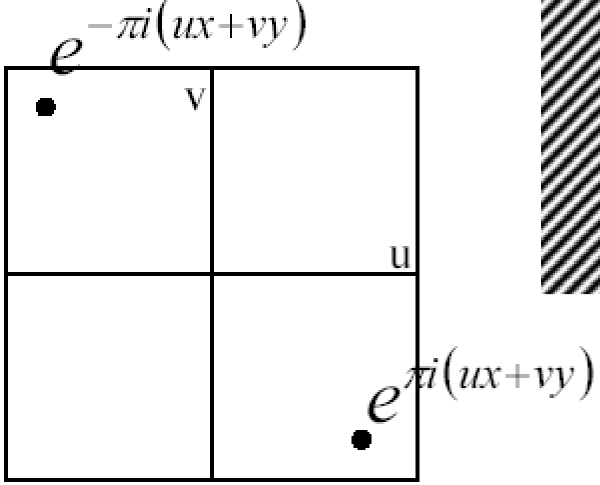


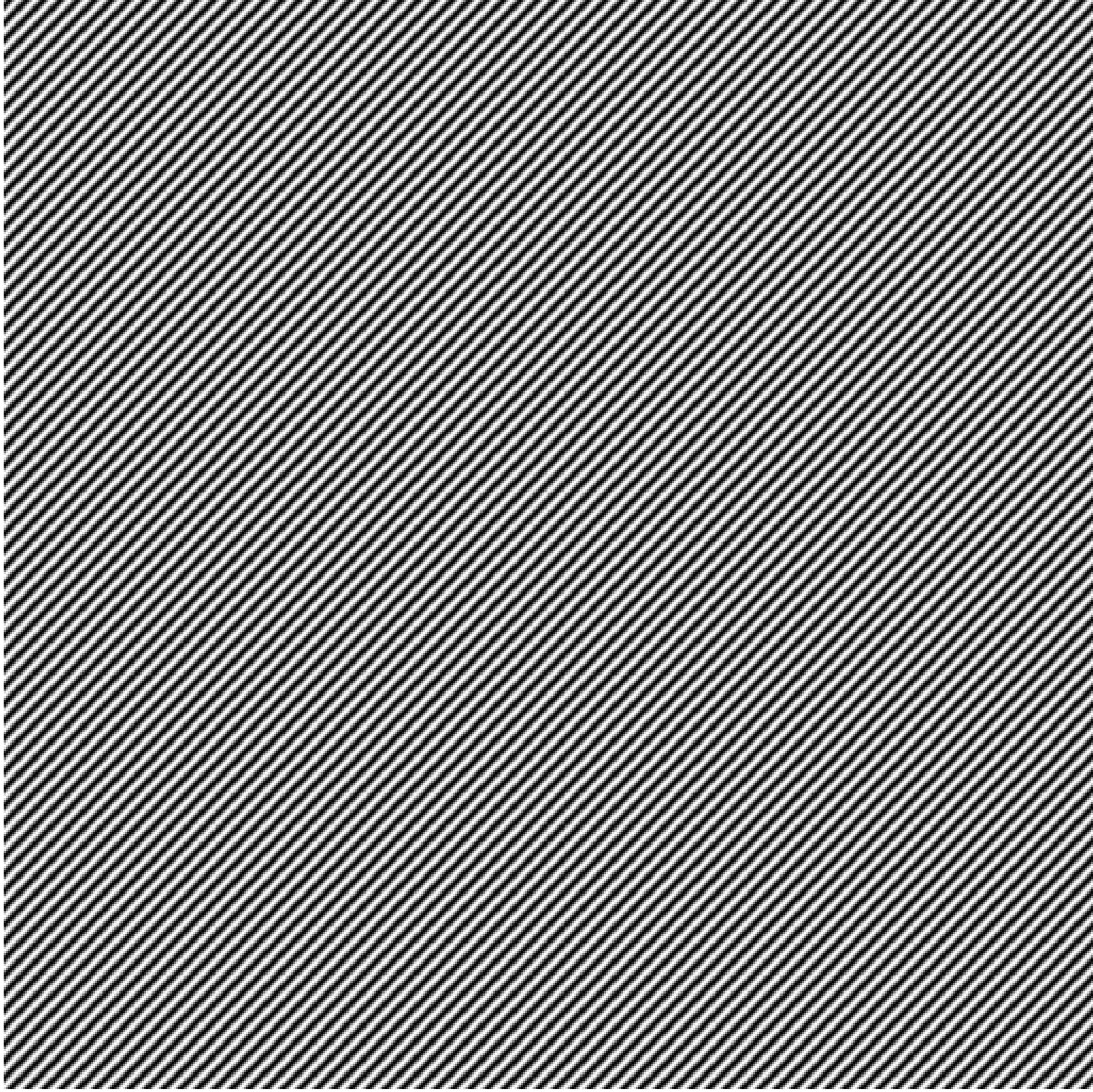
Here u and v are larger than in the previous slide.





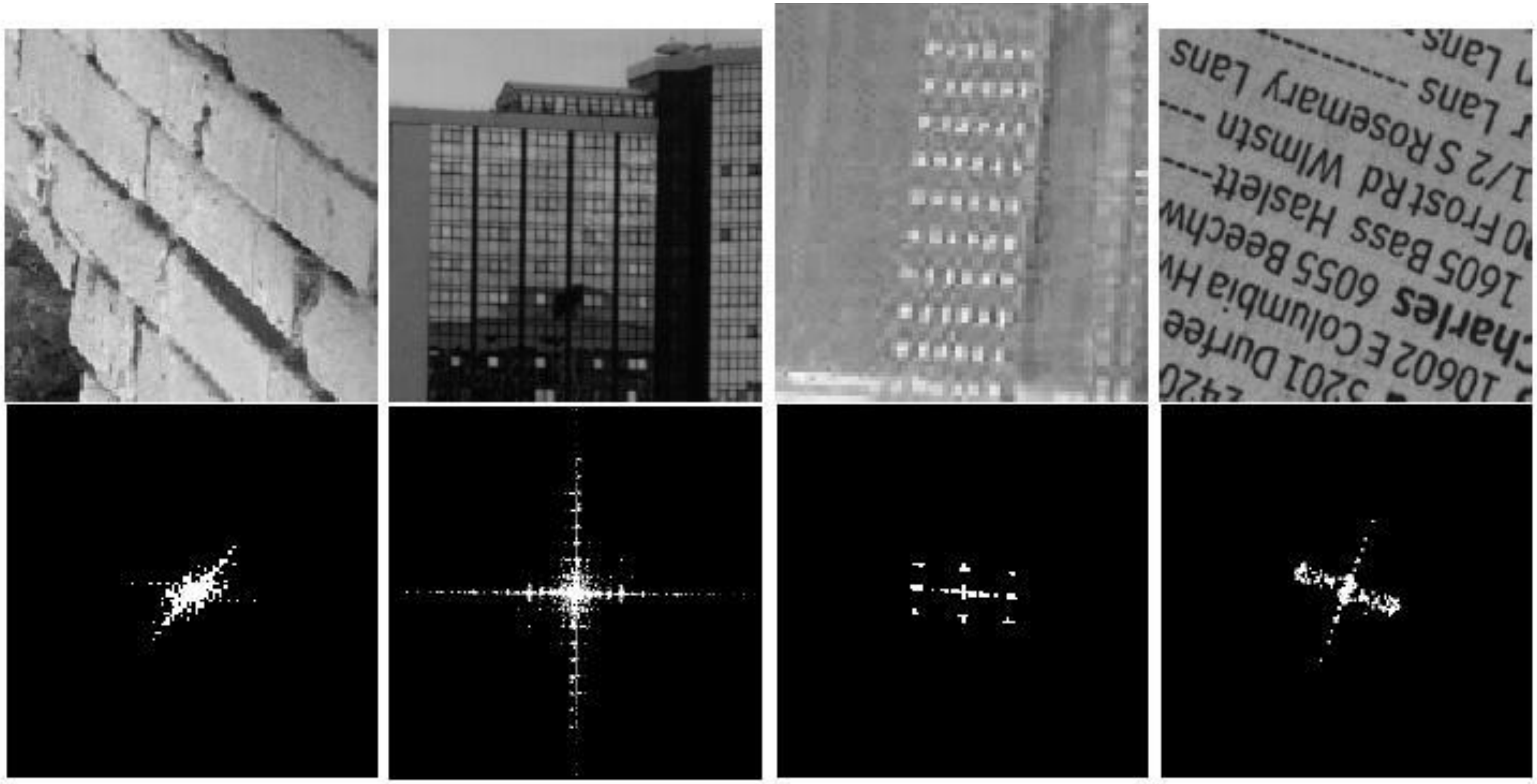
# And larger still...





# Using Fourier Representations

Dominant Orientation



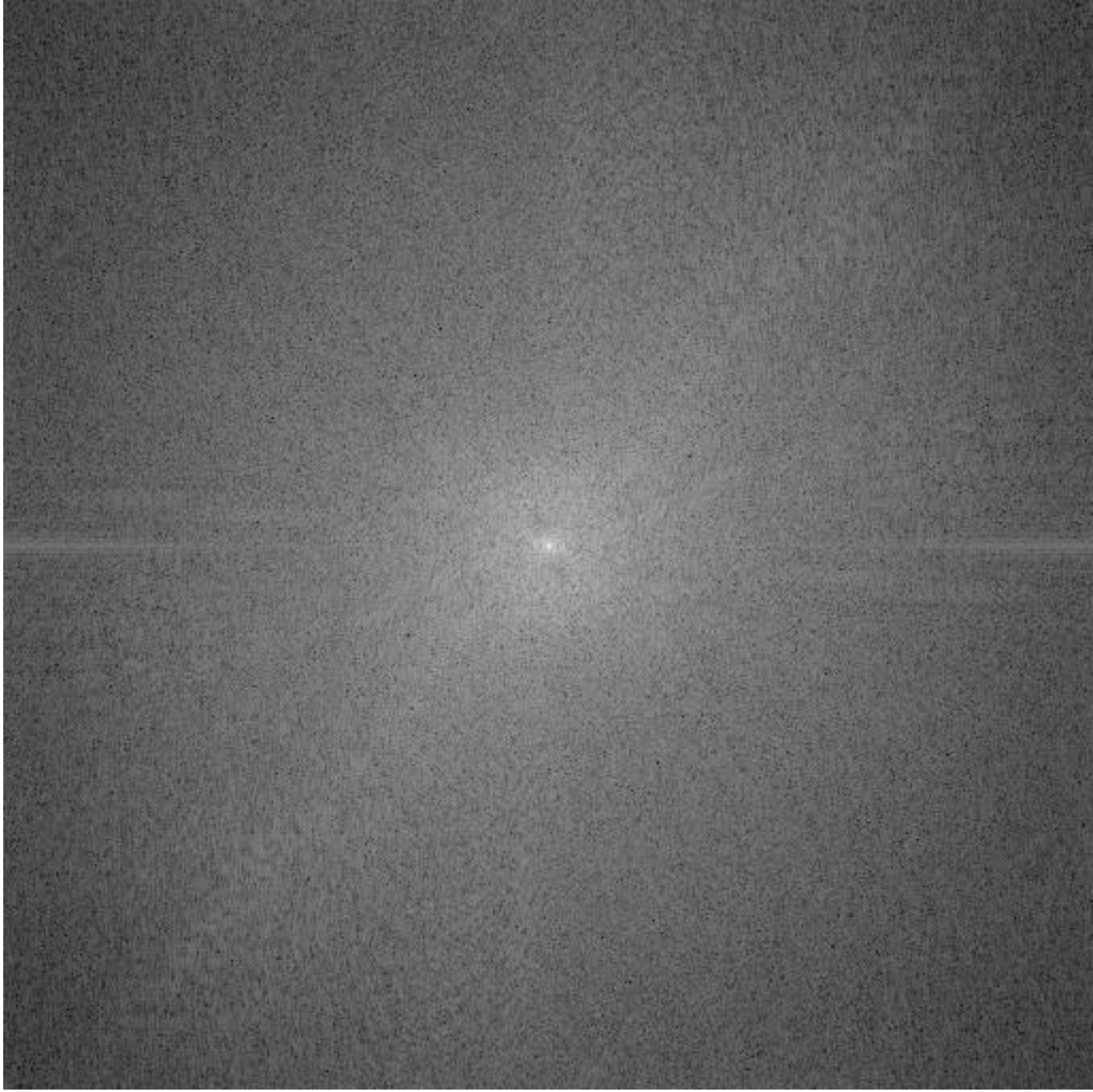
<u>Limitations</u>: not useful for local segmentation

# Phase and Magnitude

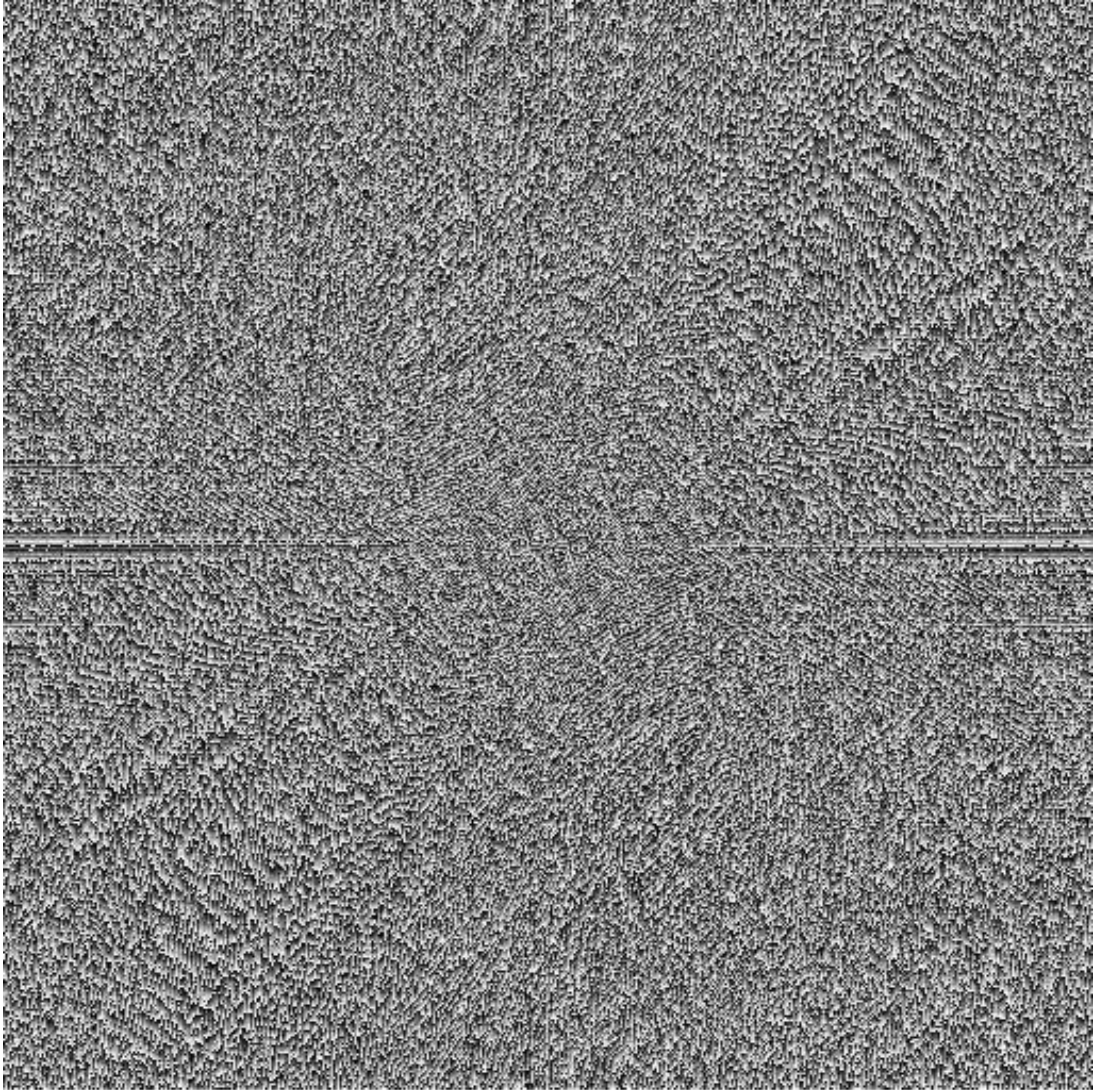
$$e^{i\theta} = \cos\theta + i \sin \theta$$

- 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?

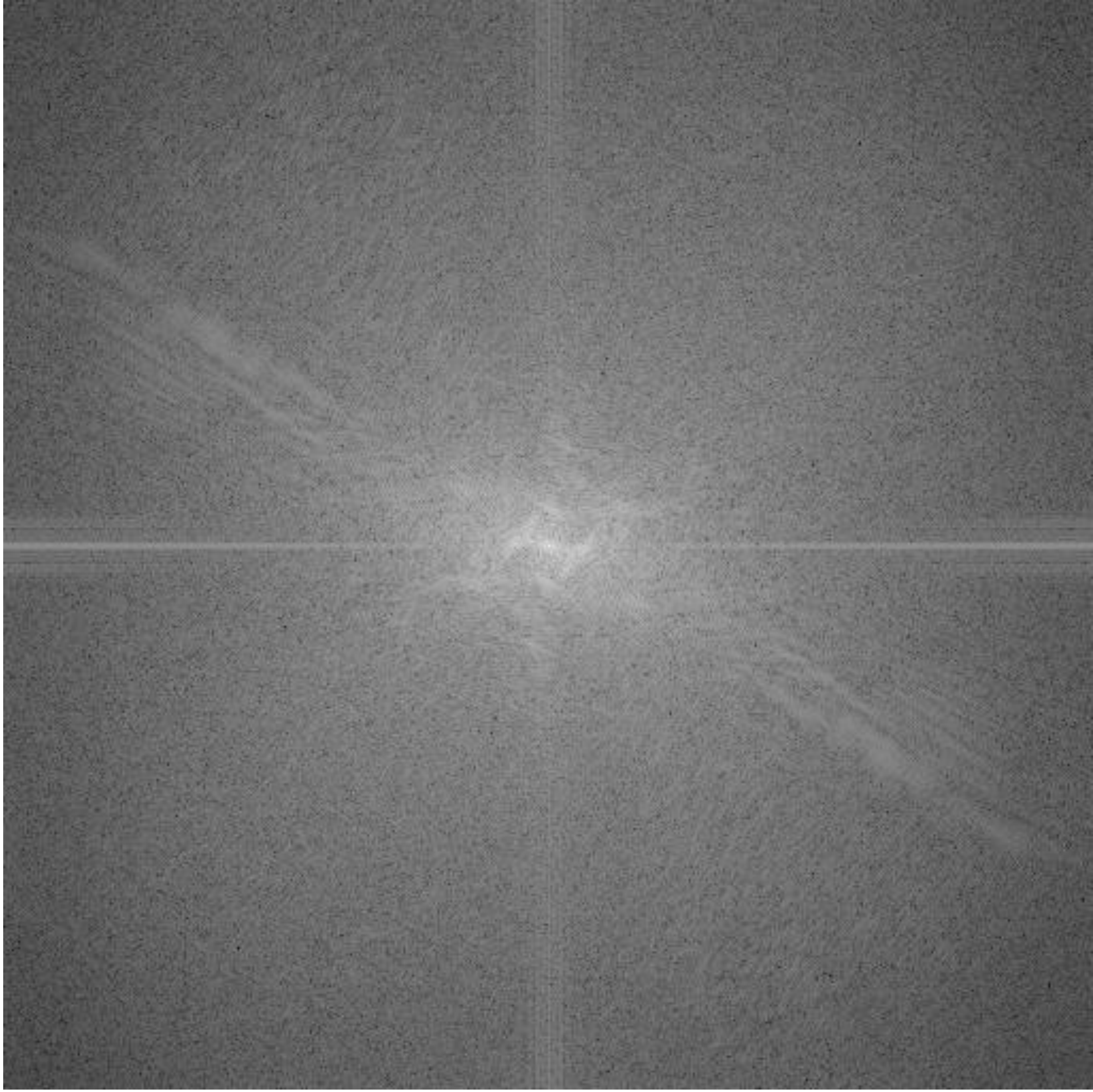
This is the magnitude transform of the cheetah pic



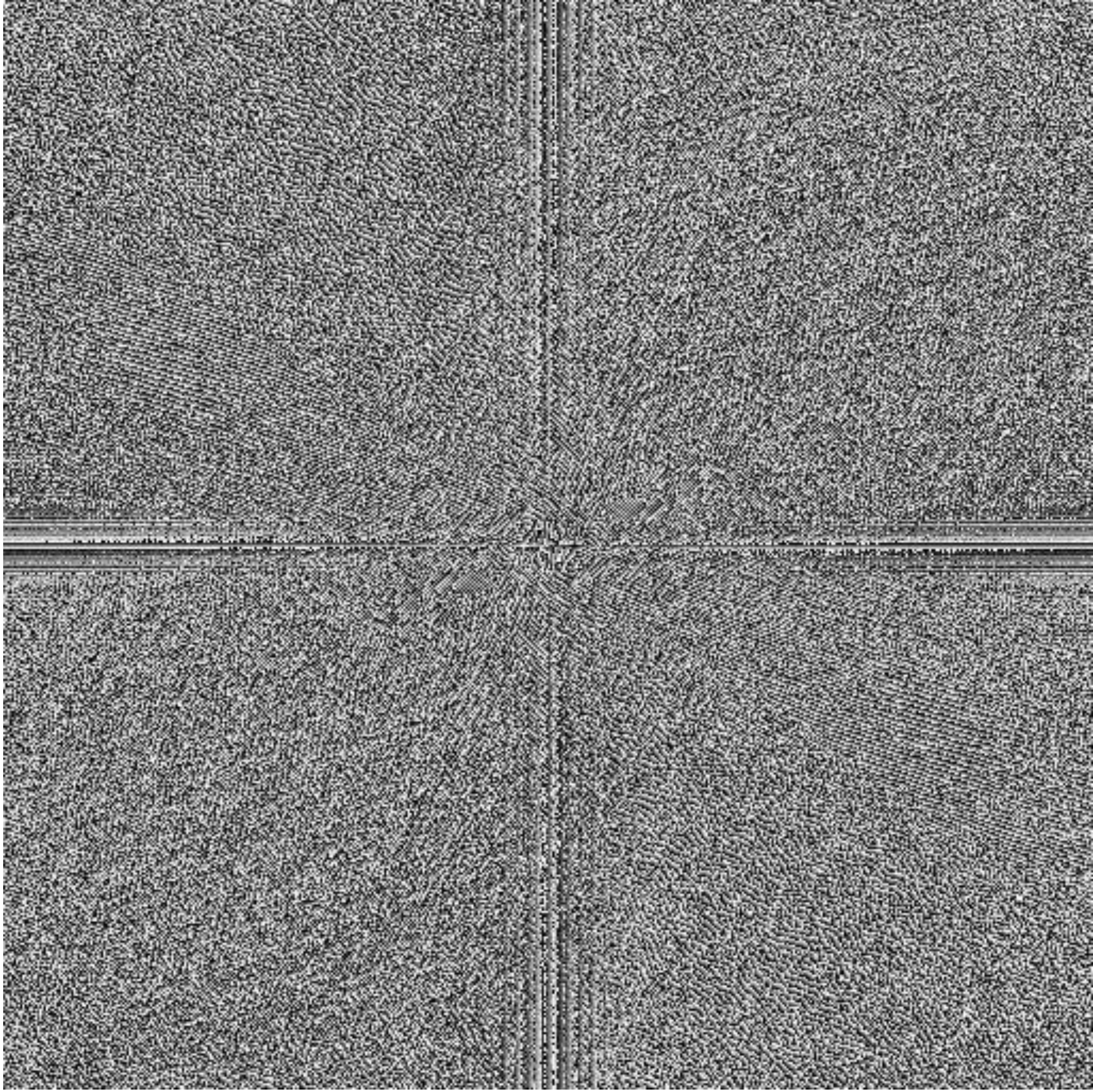
This is the phase transform of the cheetah pic



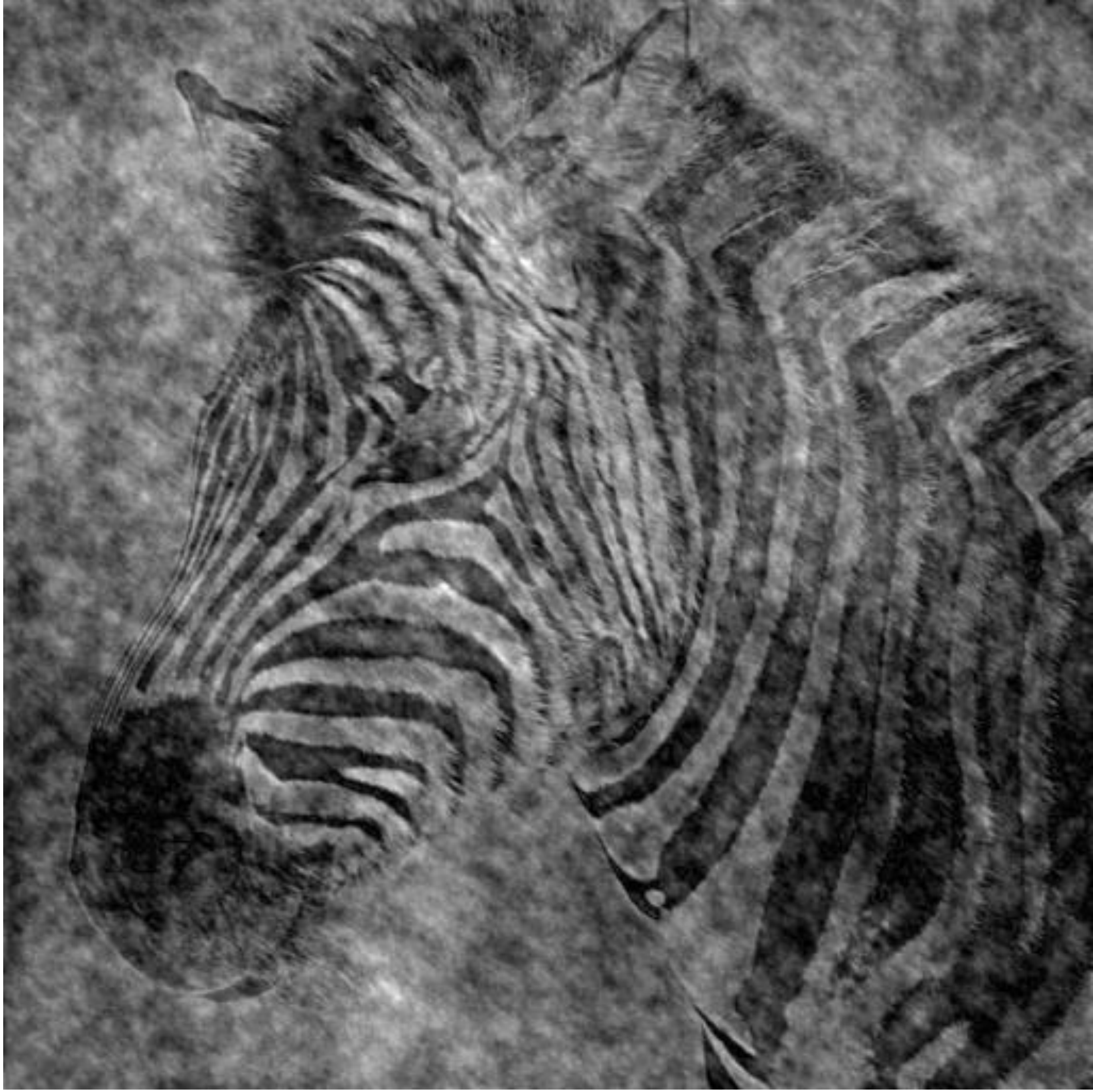
This is the magnitude transform of the zebra pic



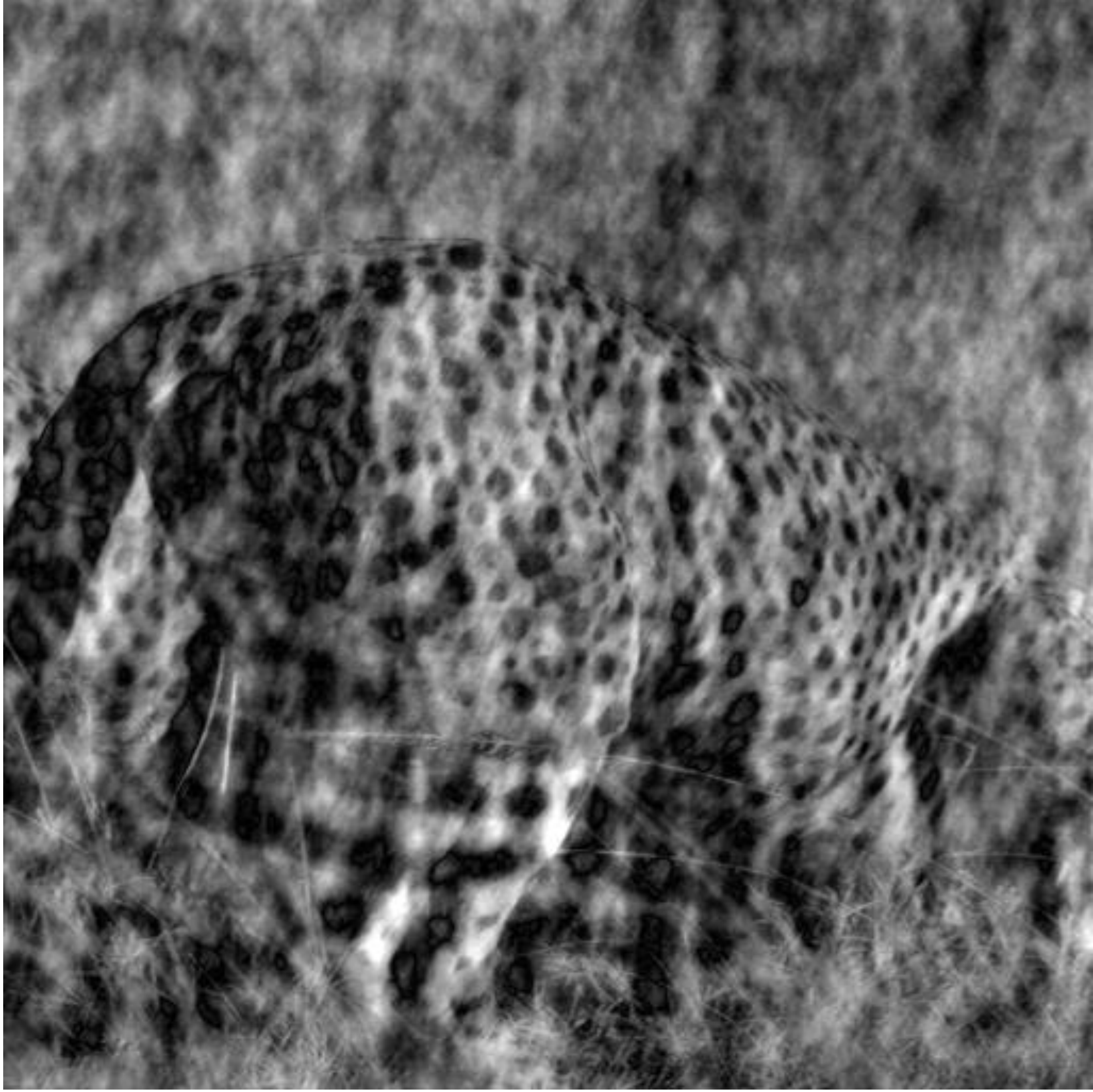
This is the phase transform of the zebra pic



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

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

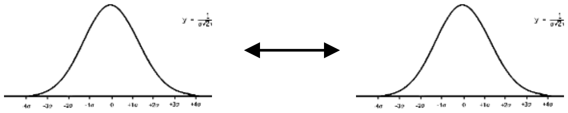
- Thus, one way of thinking about the properties of a convolution is by thinking of how it modifies the frequencies of the image to which it is applied.

- 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.

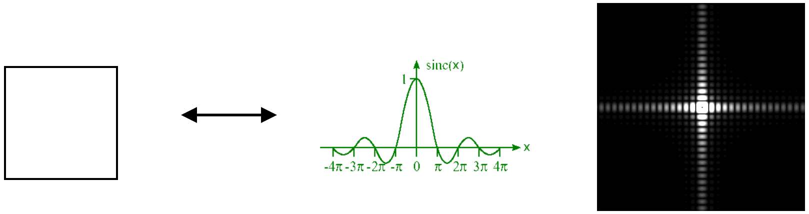
- This is referred to as the **Convolution Theorem**

# 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

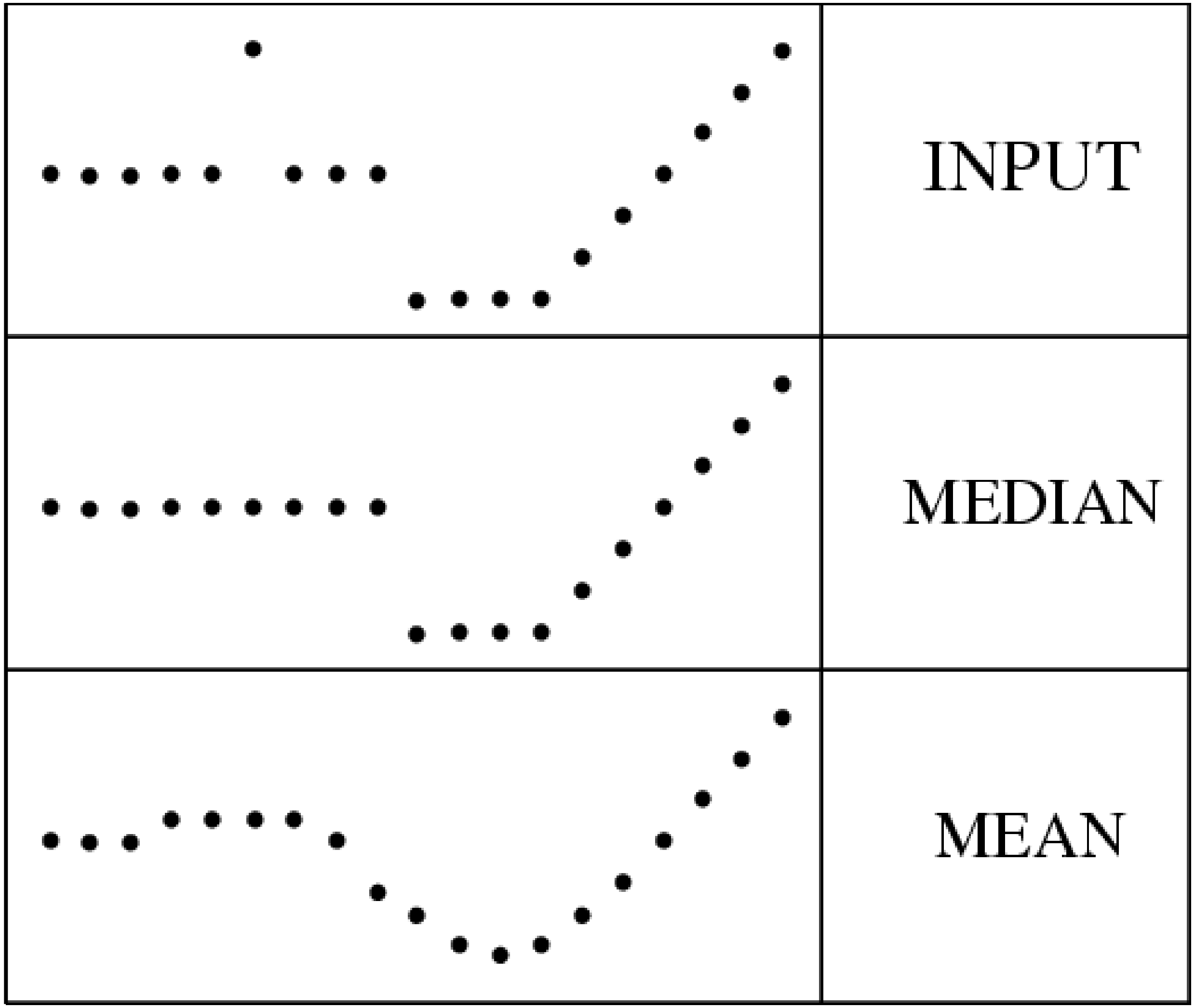
↓

?,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

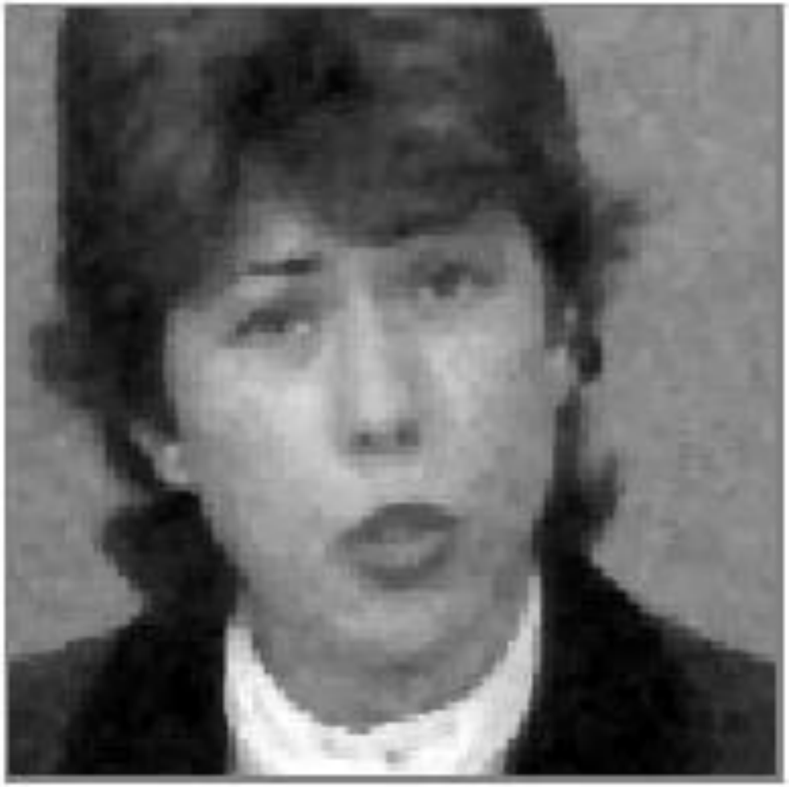
Median completely discards the spike,
linear filter always responds to all aspects

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

Median filters can destroy detail

# Median filter : images

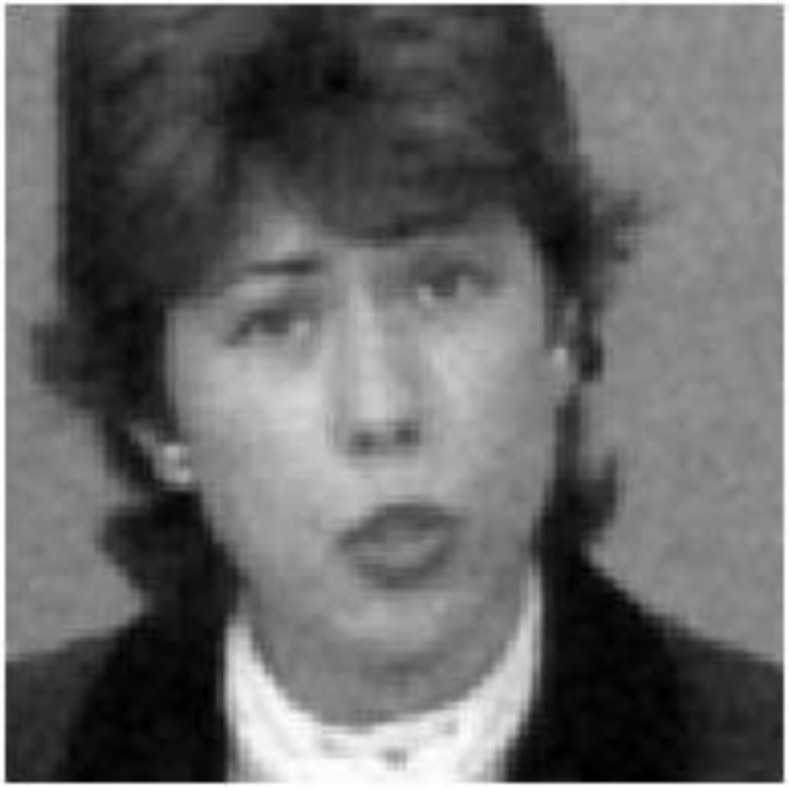
3 x 3 median filter :



sharpens edges, destroys edge cusps and protrusions

# Median filters : Gauss revisited

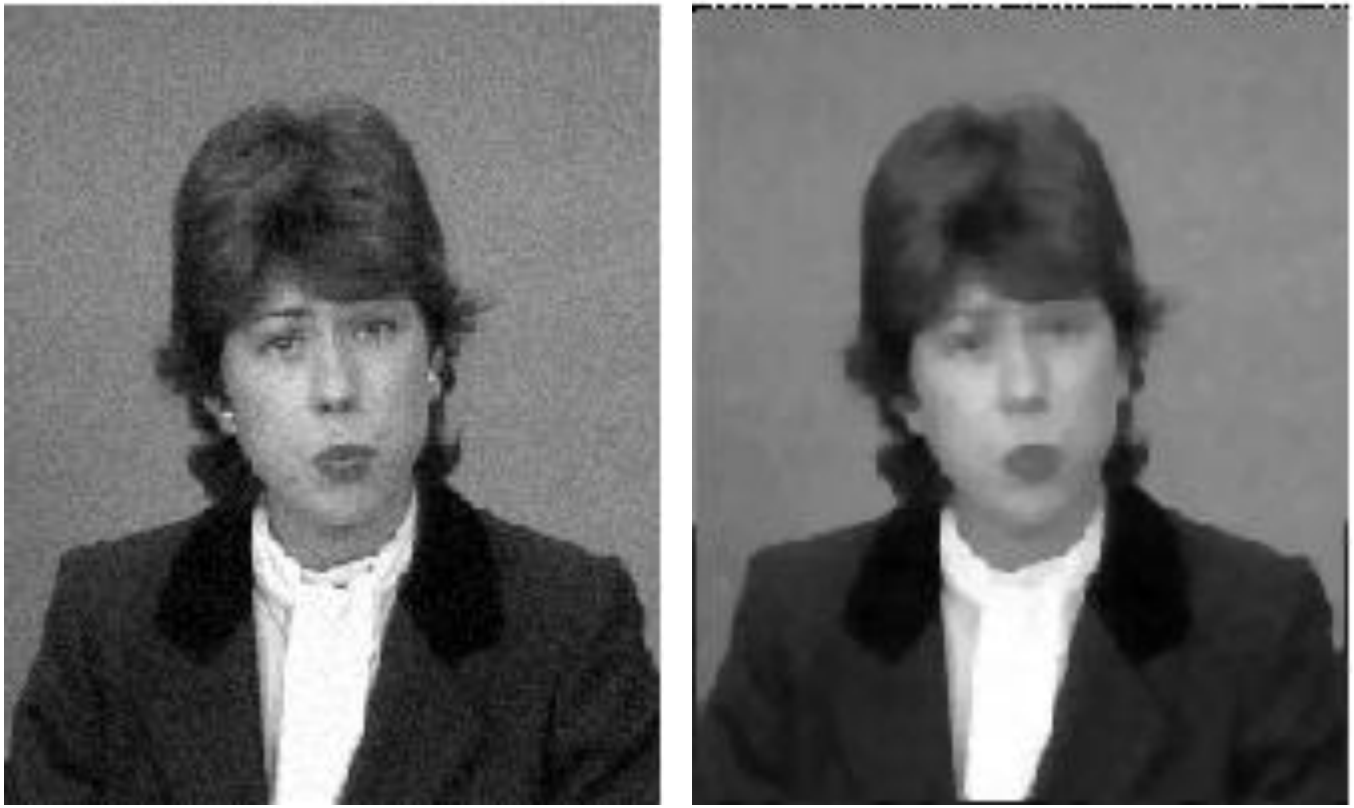
Comparison with Gaussian :



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

- Edge detection and corner detection