Computer Vision

4. Filtering

I-Chen Lin

College of Computer Science,
National Yang Ming Chiao Tung University

Outline

- Impulse response and convolution.
- Linear filter and image pyramid.

Textbook:

• David A. Forsyth and Jean Ponce, Computer Vision: A Modern Approach, Prentice Hall, New Jersey, 2003 or 2012.

Some contents are from the reference lecture notes:

• Prof. D. Lowe, Computer Vision, UBC, CA.

• Prof. D.A. Forsyth, Computer Vision, UIUC.

• Prof. J. Rehg, Computer Vision, Georgia Inst. of Tech.

• Prof. T. Darrell, Computer Vision and Applications, MIT.

What's linear filtering?

- Construct a new image whose pixels are a weighted sum of the original pixel values.
  - Using the same set of weights at each point.
  - Mainly based on neighborhood of the target pixels.

- For example,

6	5	3
4	6	3
1	2	7

$\xrightarrow{\text{Some functions}}$

	5/4	

Linear filtering

- The simplest and most useful case:
  - Replace each pixel with a linear combination of its neighbors.

- The prescription for the linear combination is called the **kernel**.

$$
\begin{array}{|c|c|c|}\hline 6 & 5 & 3 \\ \hline 4 & 6 & 3 \\ \hline 1 & 2 & 7 \\ \hline \end{array}
\;*\;
\underbrace{\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline -1/4 & 1/2 & -1/4 \\ \hline 0 & 0 & 0 \\ \hline \end{array}}_{\text{kernel}}
\;=\;
\begin{array}{|c|c|c|}\hline & & \\ \hline & 5/4 & \\ \hline & & \\ \hline \end{array}
$$

Correlation

I

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	99	11
10	9	9	11	10	10

F

1/9

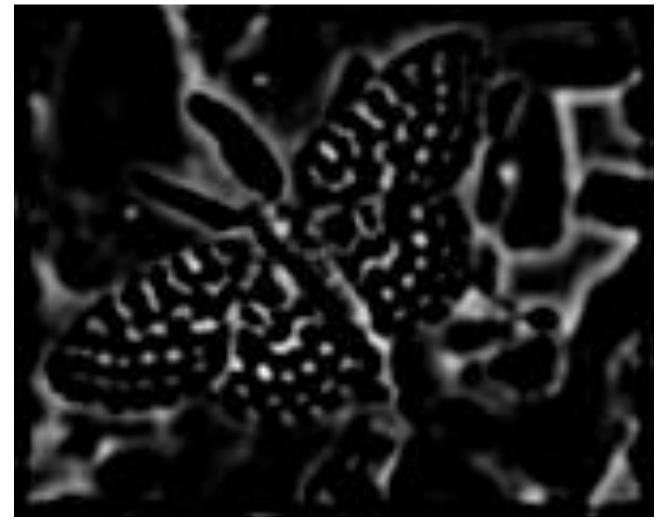
1	1	1
1	1	1
1	1	1

O

X	X	X	X	X	X
X	10				X
X					X
X					X
X					X
X	X	X	X	X	X

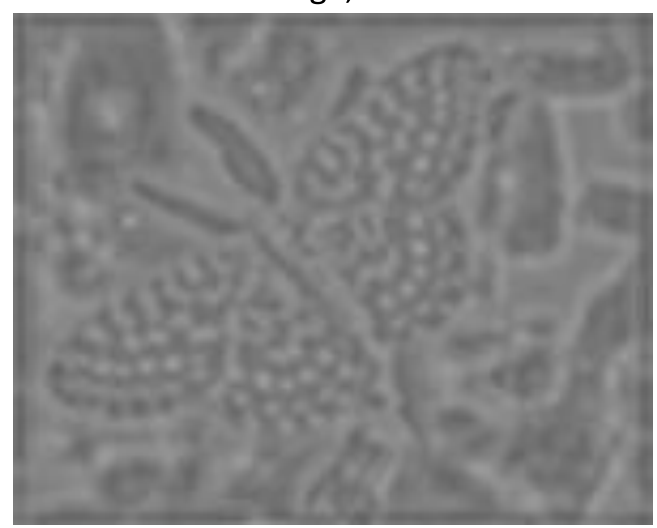
1/9.(10x1 + 11x1 + 10x1 + 9x1 + 10x1 + 11x1 + 10x1 + 9x1 + 10x1) = 1/9.(90) = 10

Normalized correlation



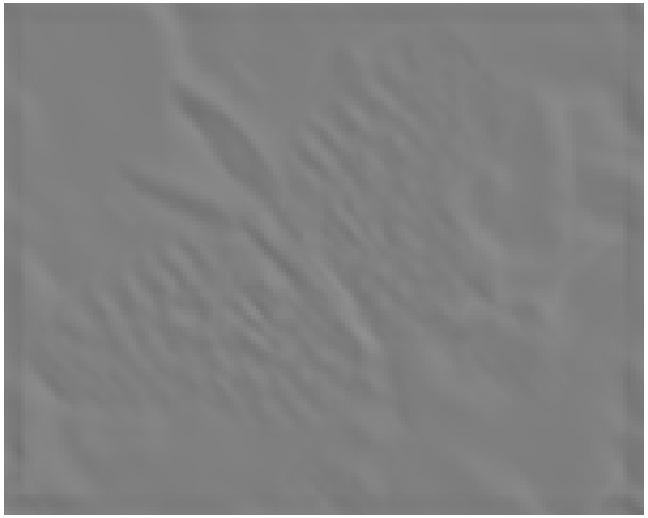
Zero mean image, -1:1 scale

Zero mean image, -max:max scale

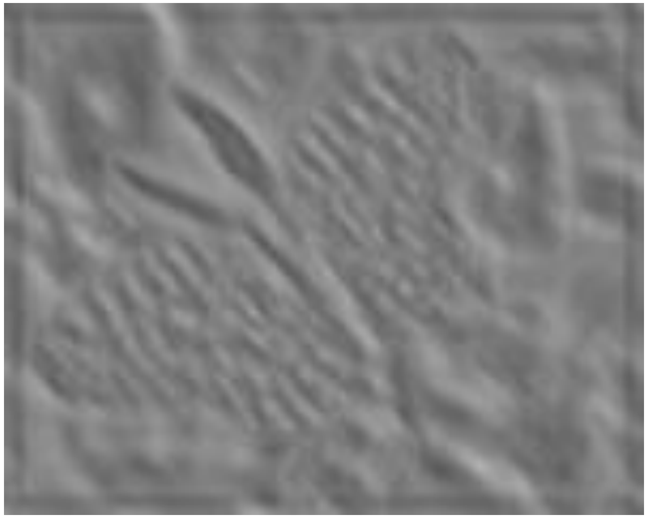


Normalized correlation

Zero mean image, -1:1 scale



Zero mean image, -max:max scale



Finding hands

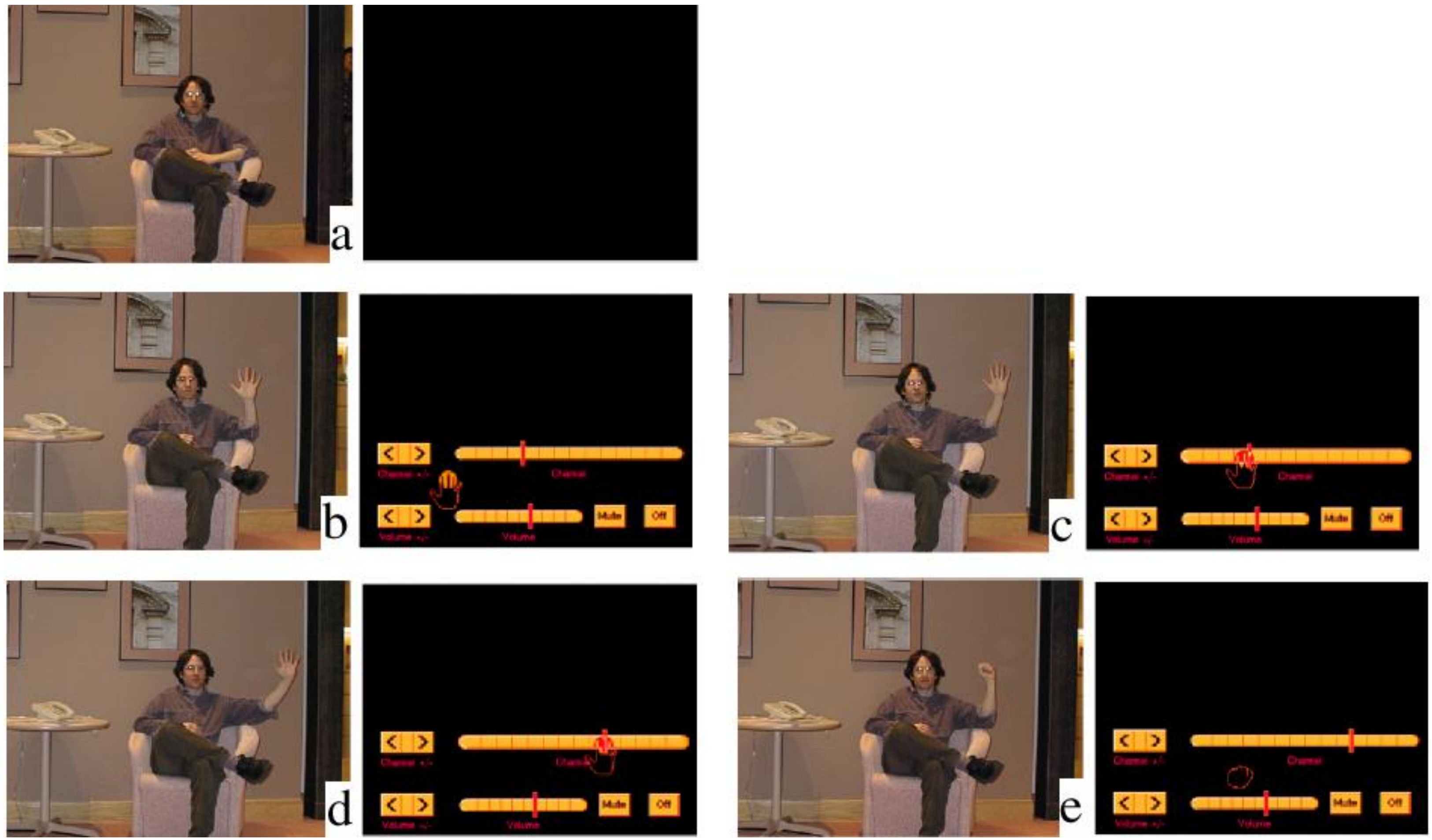


Figure from “Computer Vision for Interactive Computer Graphics,” W.Freeman et al, IEEE Computer Graphics and Applications, 1998.

Convolution

- The same as correlation with reversed kernels.
- Represent the linear weights as an image, *F*, called the kernel.
- Convolution operation:
  - Center origin of the kernel *F* at each pixel location
  - Multiply weights by corresponding pixels
  - Set resulting value for each pixel
- Image, *R*, resulting from convolution of *F* with image *H*, where *u*,*v* range over kernel pixels:

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

Warning: the textbook mixes up H and F

Correlation compared to convolution

- Correlation

$$O(x,y)=\sum_{j=-k}^{k}\sum_{i=-k}^{k}F(i,j)I(x+i,y+j)$$

- Convolution

$$O(x,y)=\sum_{j=-k}^{k}\sum_{i=-k}^{k}F(i,j)I(x-i,y-j)$$

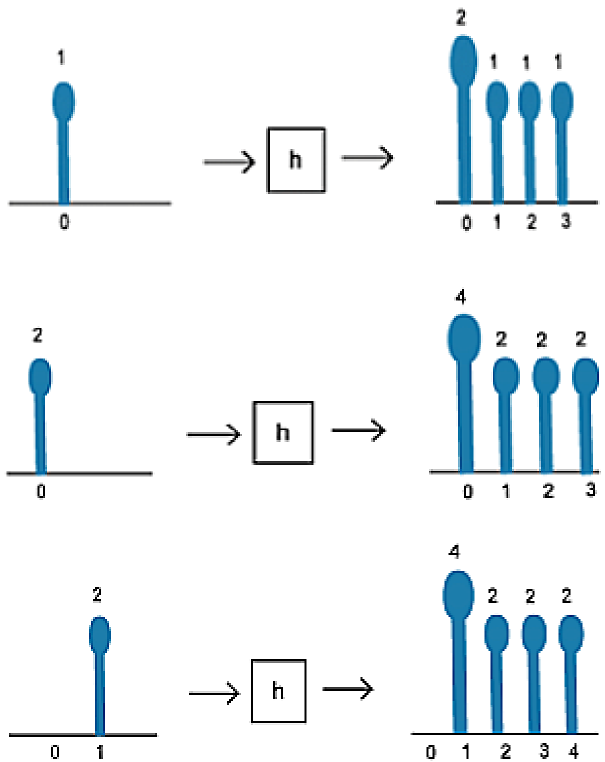
$$=\sum_{j=-k}^{k}\sum_{i=-k}^{k}F(-i,-j)I(x+i,y+j)$$

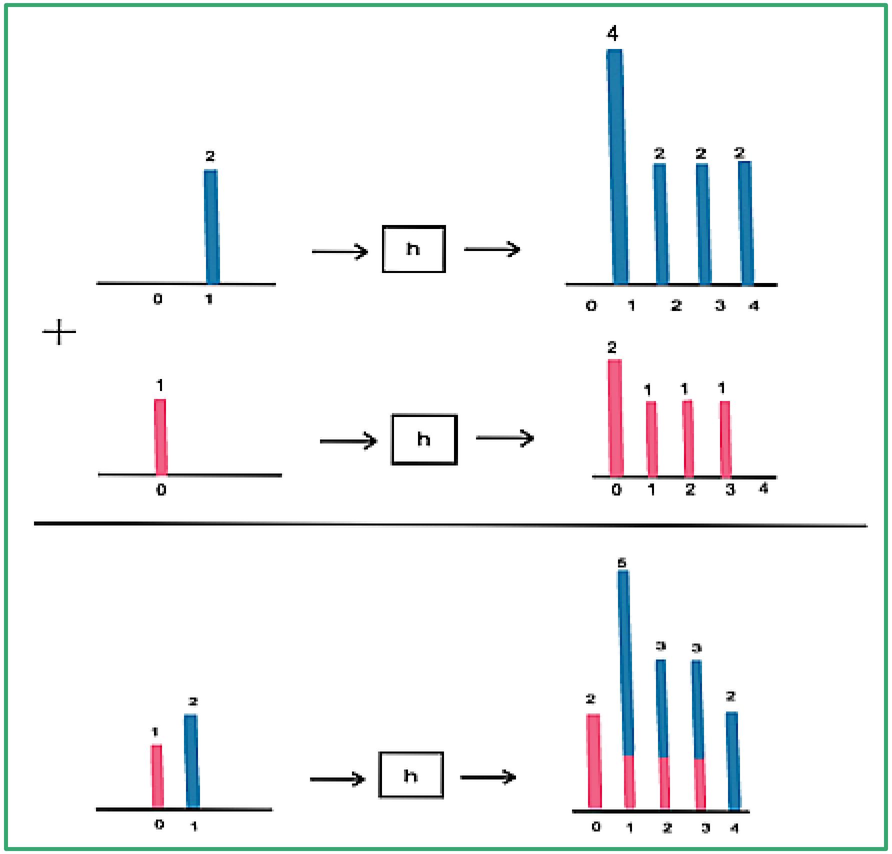
If $F(i, j) == F(-i, -j)$, then convolution is equivalent to correlation.

Convolution and impulse response

- Convolution can be viewed as a sequence of impulse responses.

$$O(x)=\sum_{i=-k}^{k}h(i)I(x-i)$$





Figures from cnx.org/content/m11539/latest/

Examples

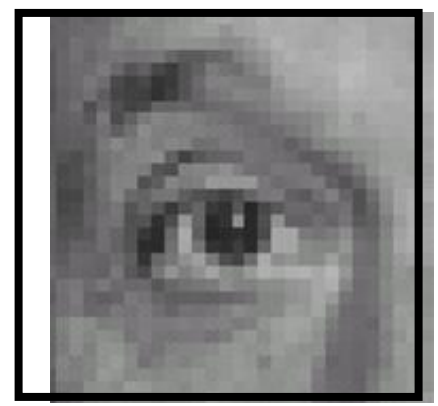
Original

0	0	0
0	1	0
0	0	0

Filtered (no change)

Original

0	0	0
0	0	1
0	0	0

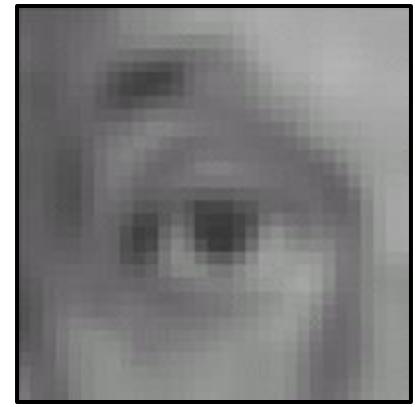


Shifted right By 1 pixel

Examples (cont.)

Original

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



Blur (with a box filter)

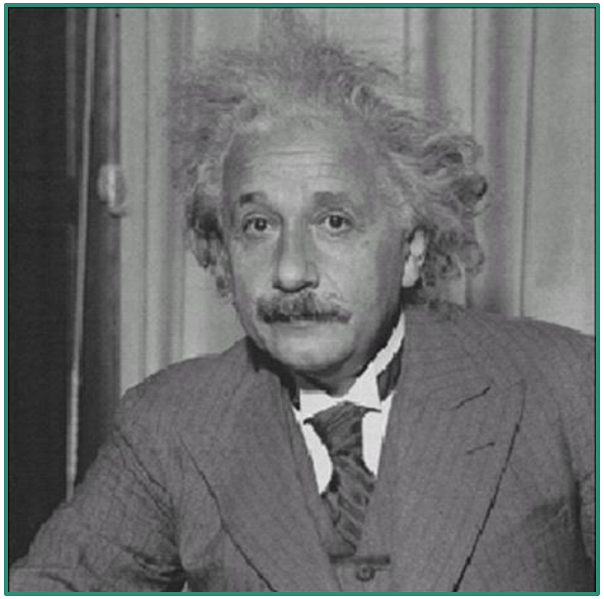
Original

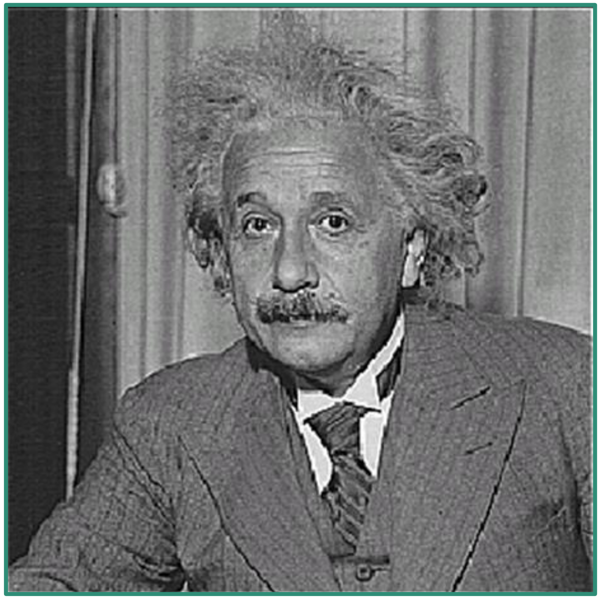
$$\begin{bmatrix}0 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 0\end{bmatrix} - \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Sharpening filter

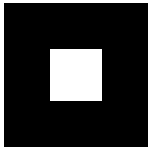
Sharpening

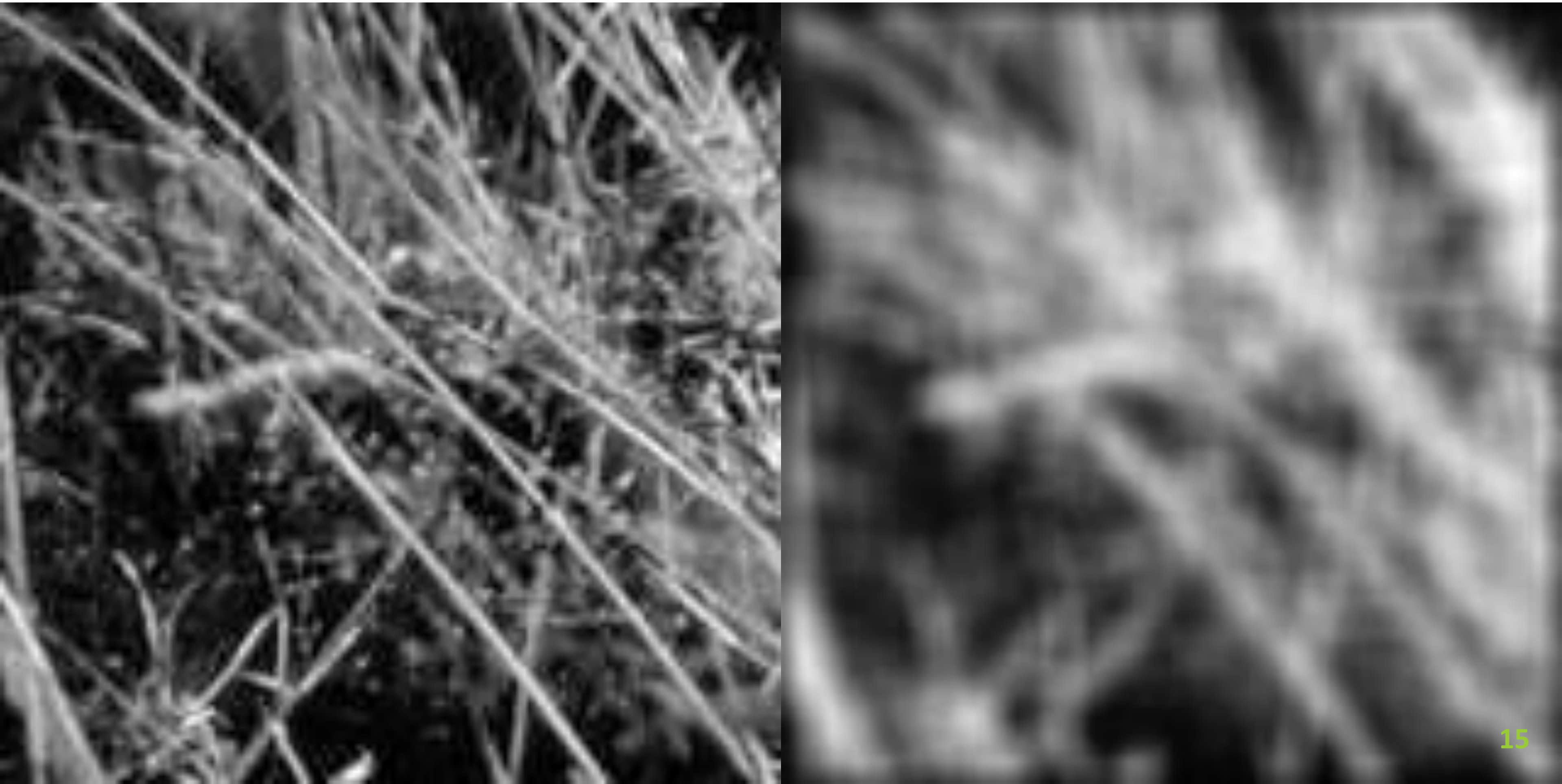
- Sharpening filter
  - Accentuates differences with local average





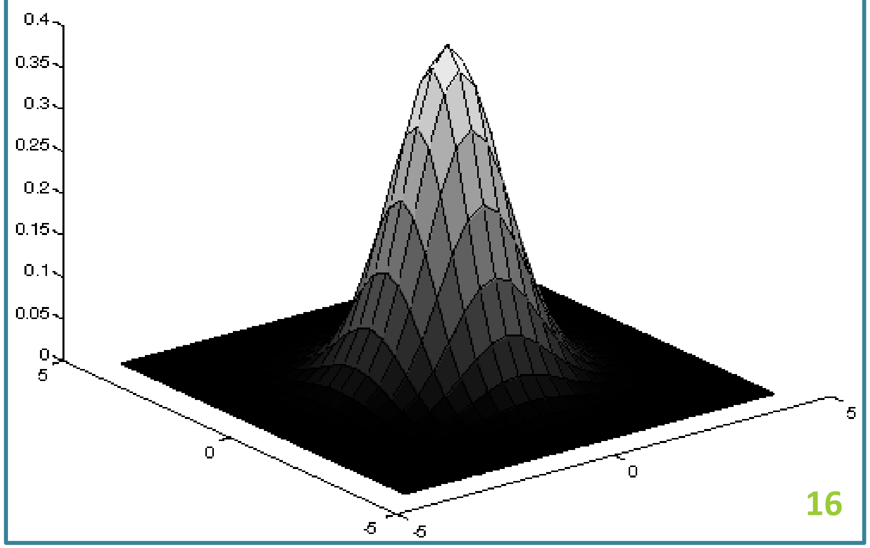
Examples: Smoothing with a box filter





Smoothing with a Gaussian

- Averaging does not model defocussed lens well
- Impulse response should be fuzzy blob, but a box filter would give a little square.
- Gaussian gives a better model of a fuzzy blob.



Gaussian kernel

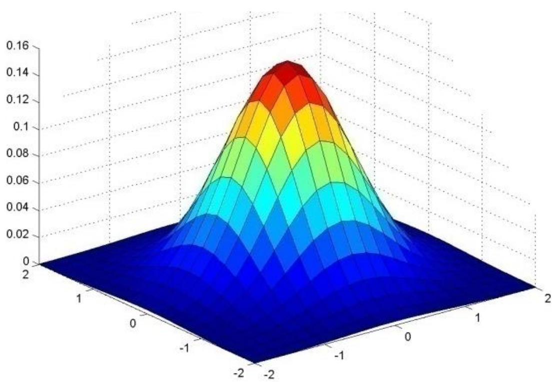
- Weight contributions of neighboring pixels by nearness.

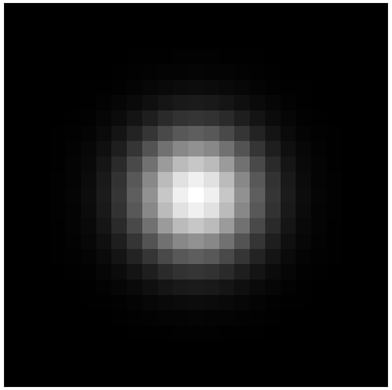
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

- For efficiency, usually use integers as weights, e.g.: 0.125 *

	1	
1	4	1
	1	

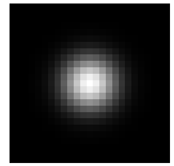
  - Don’t forget normalizing the weight sum to 1.

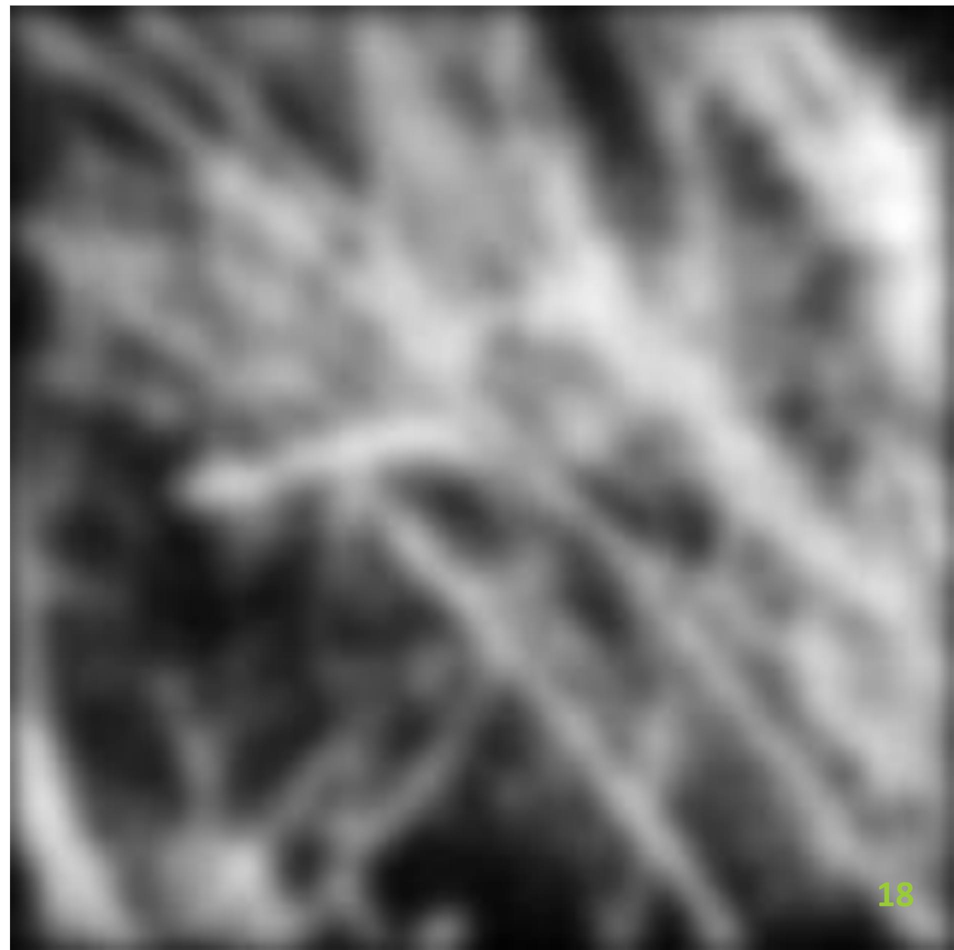




0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

Smoothing with a Gaussian





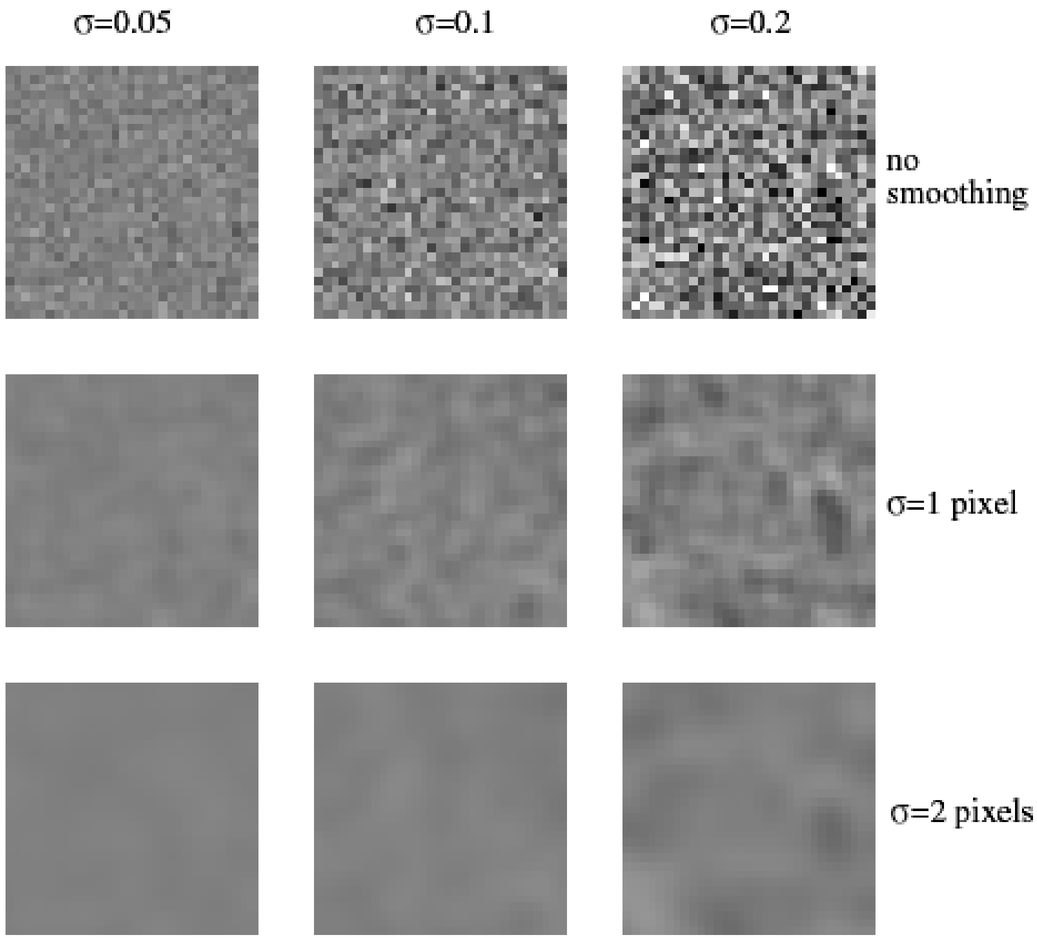
Additive Gaussian noise

$\sigma = 1$

$\sigma = 16$

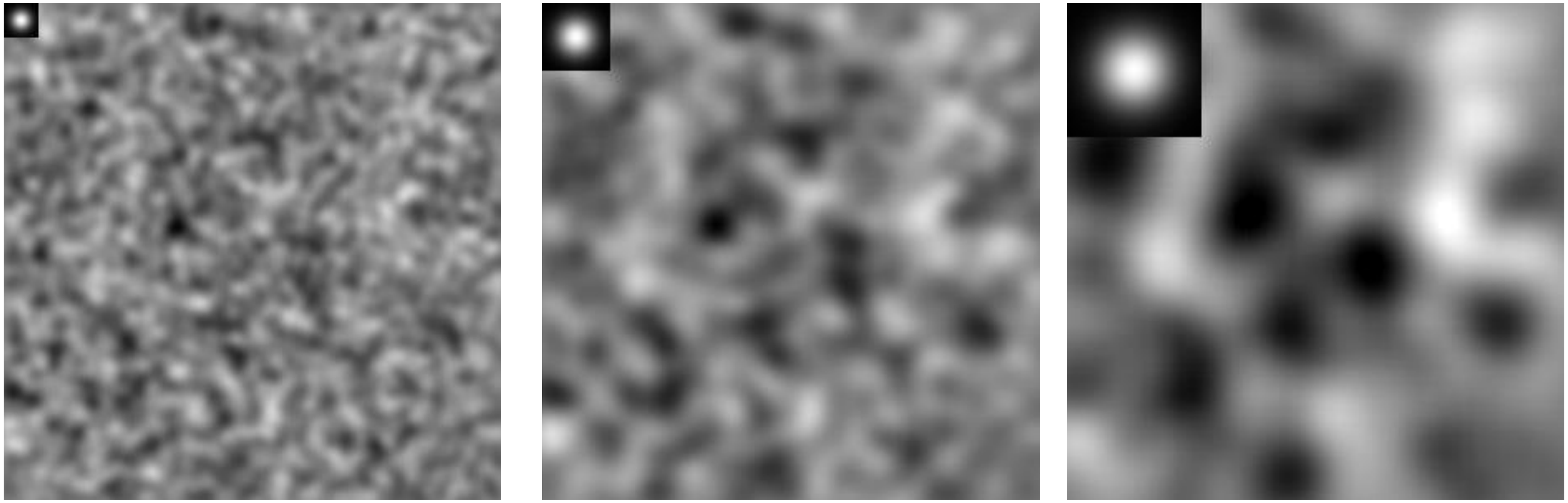
Noise and smoothing

- Smoothing reduces pixel noise:
  - Each row shows smoothing with Gaussians of different width
  - Each column shows different amounts of Gaussian noise.



Noise and smoothing

- Filtered noise is sometimes useful
  - E.g. in textures synthesis.



Efficient implementation

- Both the box filter and the Gaussian filter are separable into two 1D convolutions:
  - First convolve each row with a 1D filter
  - Then convolve each column with a 1D filter.

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

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

- A 2D Gaussian can be expressed as the product of Gaussian of *X* and *Y*.

Other noise and smoothing

Could be salt-and-pepper noise.

Figures from lecture notes D.A. Forsyth, Computer Vision, UIUC.

Differentiation

- Recall, for 2D function, f(*x*, *y*):

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

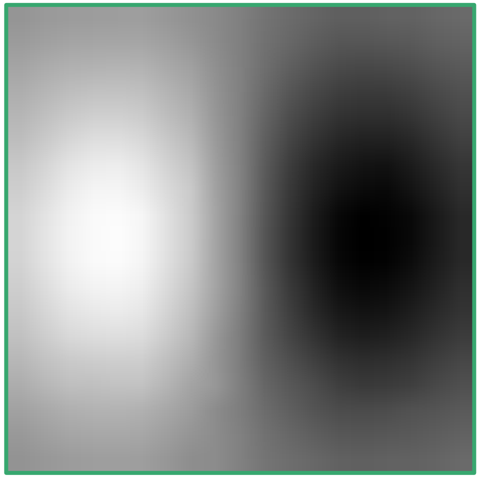
  - This is linear and shift invariant, so must be the result of a convolution.
  - We could approximate this as symmetric finite difference:

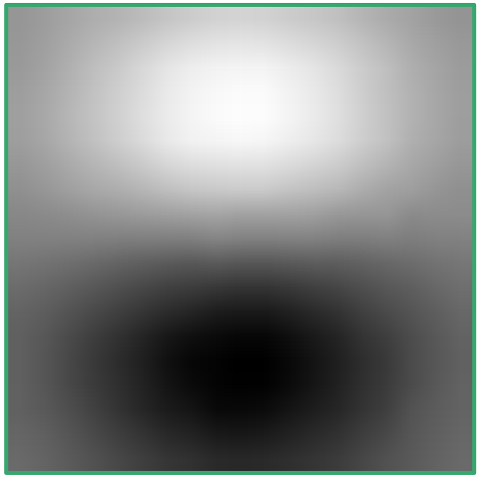
$$\frac{\partial h}{\partial x} \approx \frac{h_{i+1,j} - h_{i-1,j}}{2} \qquad H = \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}$$

- Issue: noise
  - Smooth before differentiation
  - Two convolutions to smooth, then differentiate?

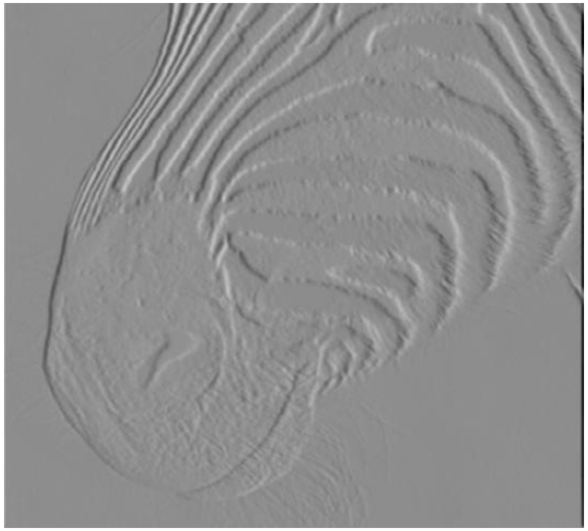
Differentiation

- Because differentiation is convolution, and convolution is associative.
  - $((f**g)**h)=(f**(g**h))$
- We can use a derivative of Gaussian filter.

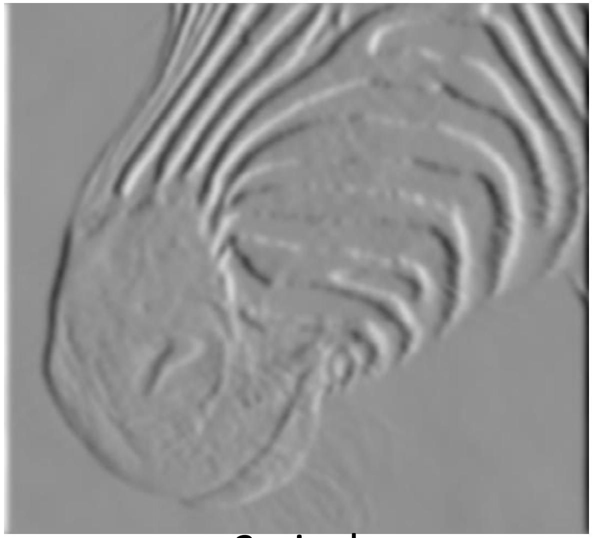




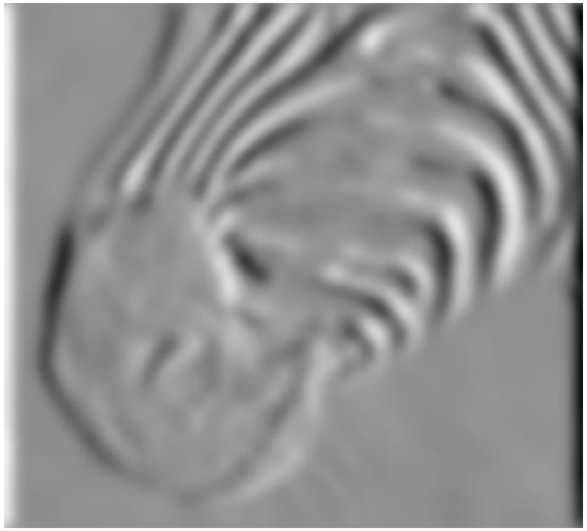
Scale affects derivatives



1 pixel



3 pixel



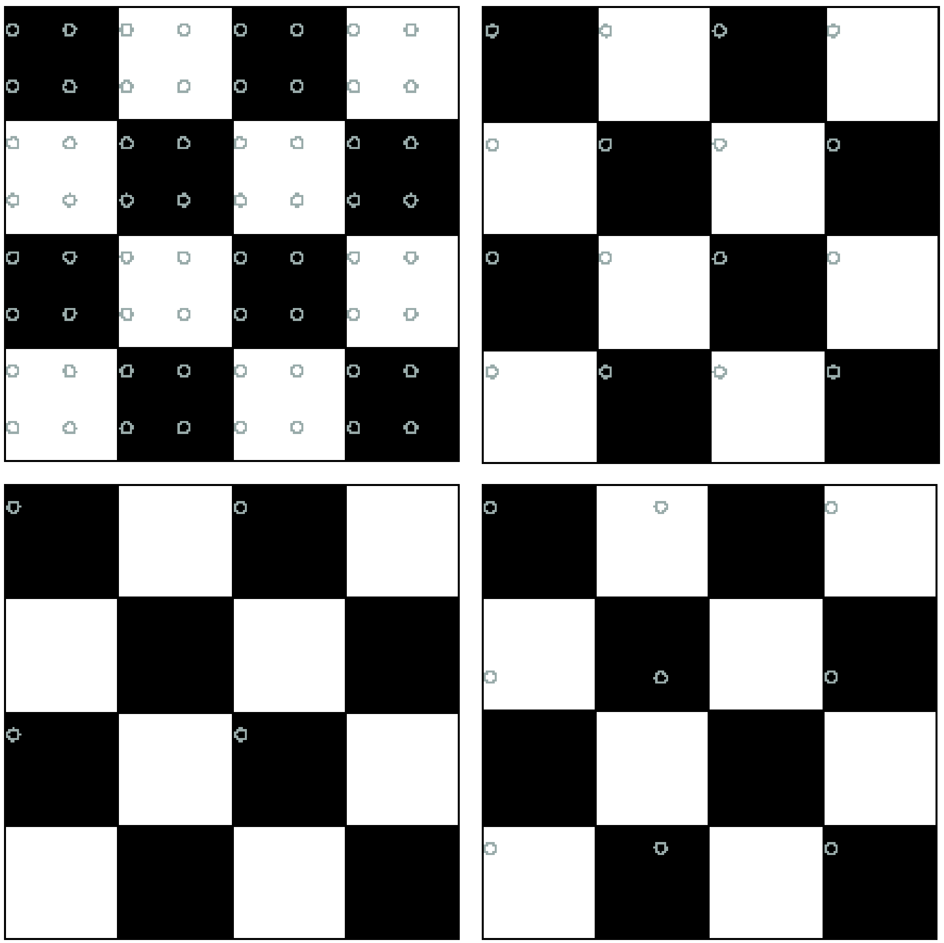
7 pixel

Aliasing

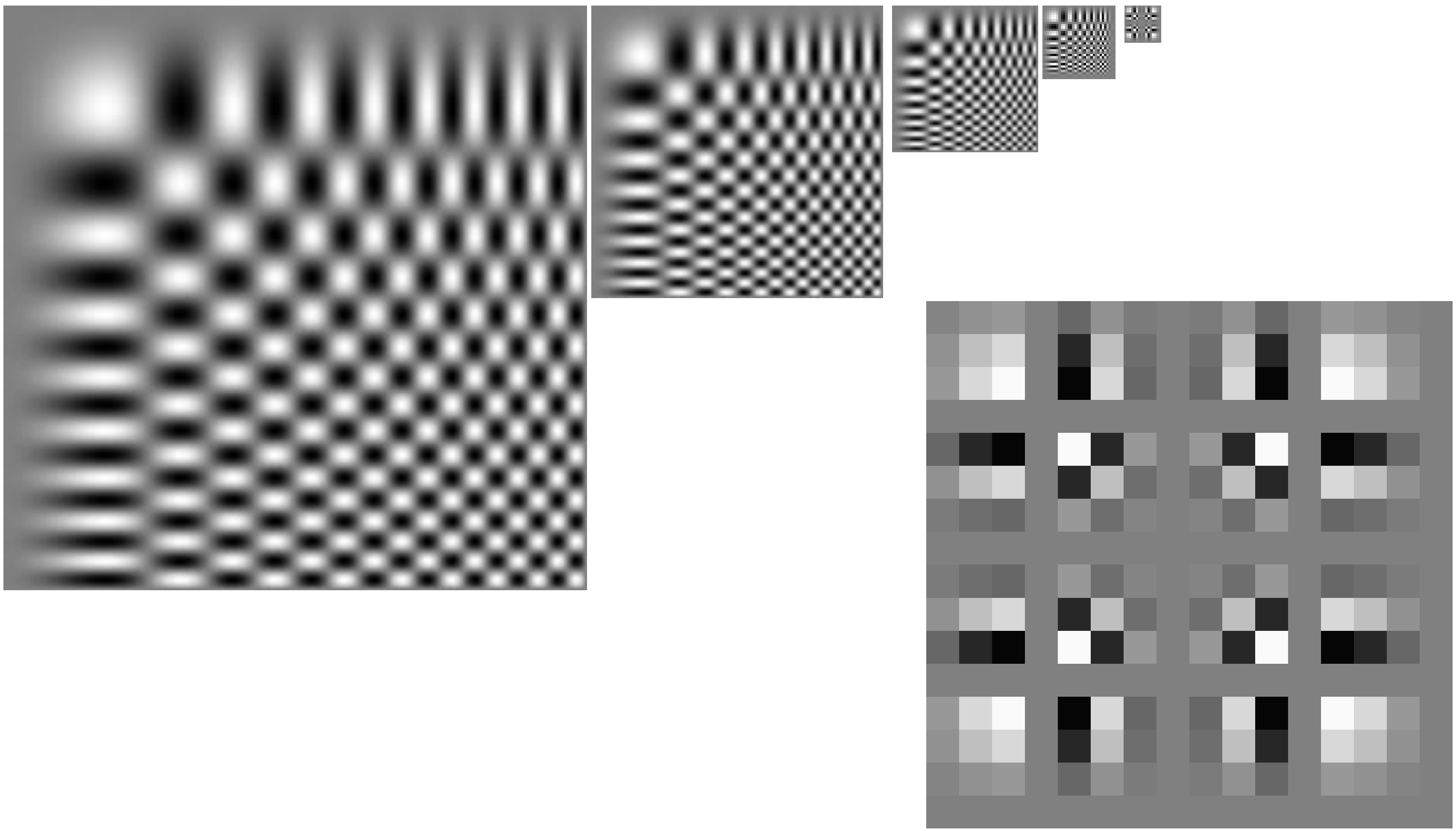
- Can't shrink an image by taking every second pixel
- If we do so, characteristic errors appear
  - Typically, small phenomena look bigger; fast phenomena can look slower
  - Common phenomenon
    - Wagon wheels rolling the wrong way in movies
    - Checkerboards misrepresented in ray tracing
    - Striped shirts look funny on color television

Re-sampling

- Resample the checkerboard by taking one sample at each circle.

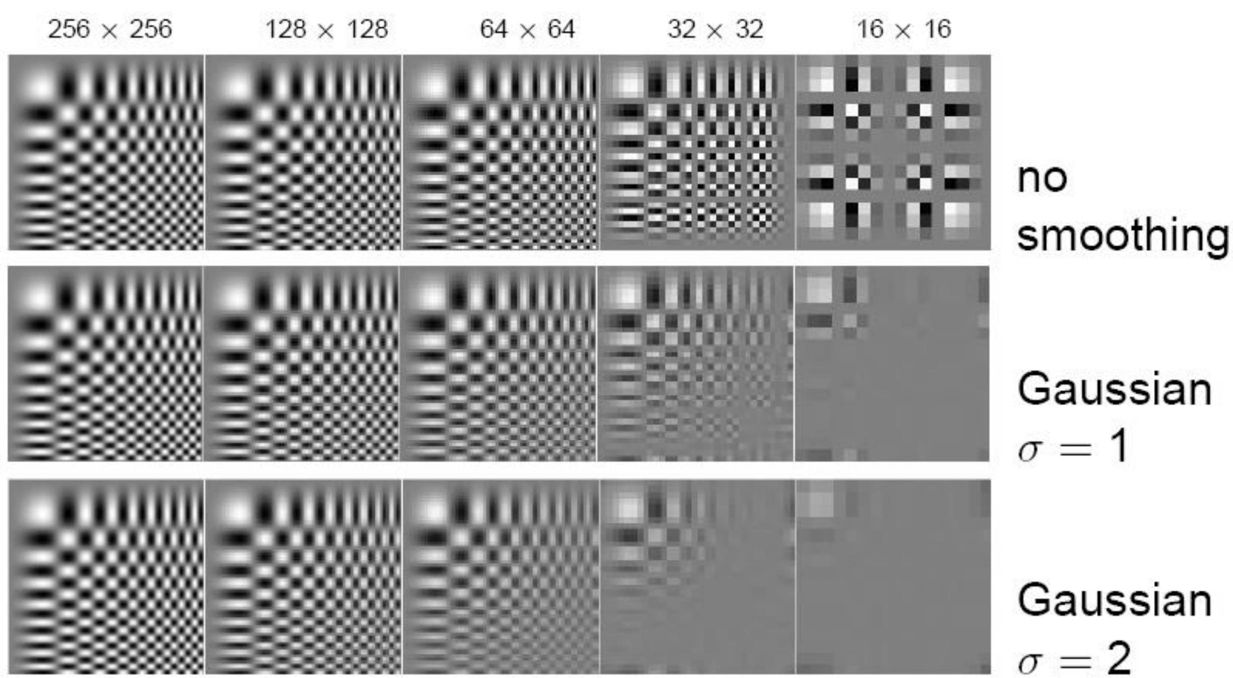


Simple sampling, but …



Smoothing as low-pass filtering

- High frequencies lead to trouble with sampling.
- Suppress high frequencies before sampling !
- Truncate high frequencies in FT or convolve with a low-pass filter.

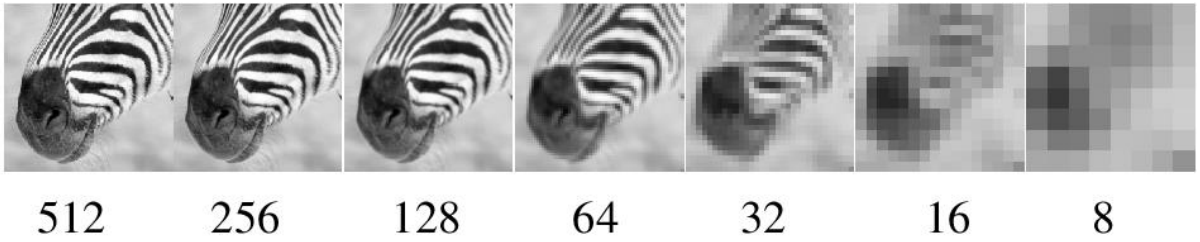


Forsyth & Ponce Figures 7.12–7.14 (top rows)

The Gaussian pyramid

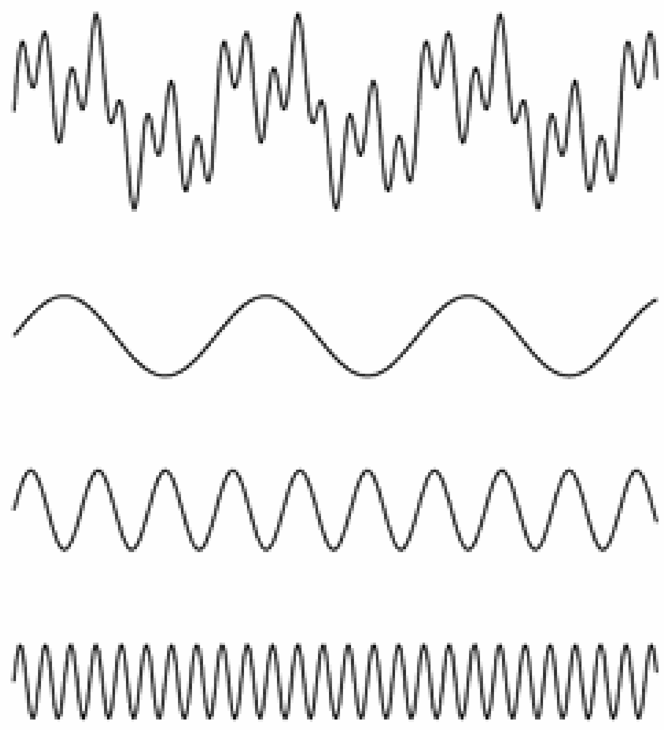
- Create each level from previous one:
  - smooth and sample
- Smooth with Gaussians, in part because
  - a Gaussian*Gaussian = another Gaussian
  - $G(x) * G(y) = G(\mathrm{sqrt}(x^2 + y^2))$
- Gaussians are low pass filters, so the representation is redundant once smoothing has been performed

The Gaussian pyramid



1D Fourier Transform

- Represent functions on a new basis .
- Think of functions as vectors, with many components.



www.science.org.au/nova/029/029img/wave1.gif

2D Fourier Transform

- Represent function on a new basis .
  - Think of functions as vectors, with many components.

- We now apply a linear transformation to transform the basis dot product with each basis element
- Basis elements have the form

$$\boxed{e^{-i\theta} = \cos\theta + i\sin\theta} \qquad e^{-i2\pi(ux+vy)}$$

$$F(\boldsymbol{f}(x,y))(u,v) = \iint_{\mathrm{R}^2} \boldsymbol{f}(x,y) e^{-i2\pi(ux+vy)} dxdy$$

Discrete Fourier Transform

discrete domain

Forward transform

$$F[m,n]=\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} f[k,l]e^{-2\pi i\left(\frac{km}{M}+\frac{\ln}{N}\right)}$$

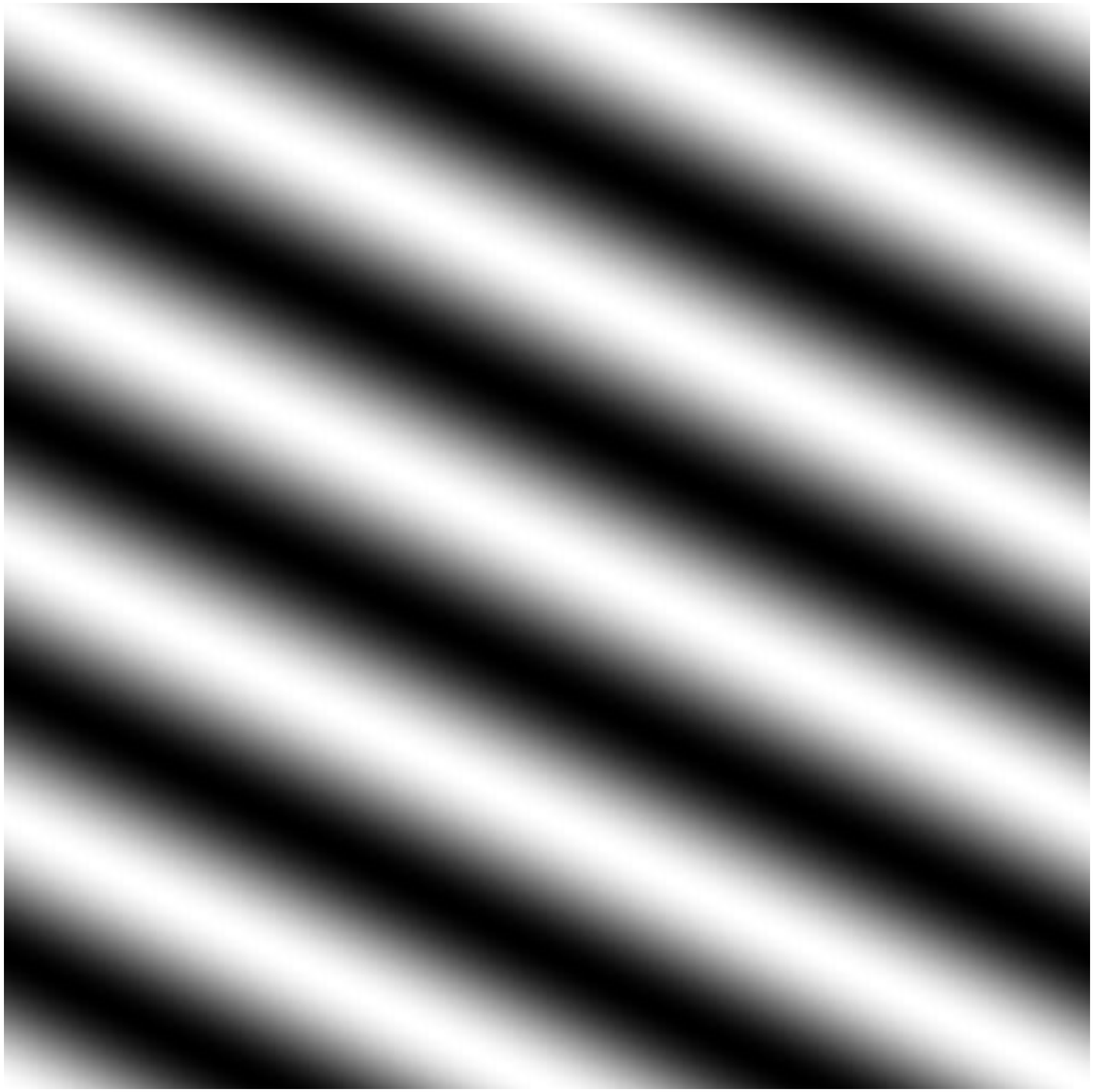
Inverse transform

$$f[k,l]=\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} F[m,n]e^{2\pi i\left(\frac{km}{M}+\frac{\ln}{N}\right)}$$

Figures modified from lecture notes of Prof. T. Darrell, Computer Vision and Applications, MIT.

Real part of basis element

- The magnitude of the vector (u, v) gives a frequency
- Its direction gives an orientation



Phase and magnitude

- $f(x,y)$ is the image and is REAL
- $F(u,v)$ (abbreviate as F) is in general, COMPLEX.

- Fourier transform is complex and difficult for visualization.

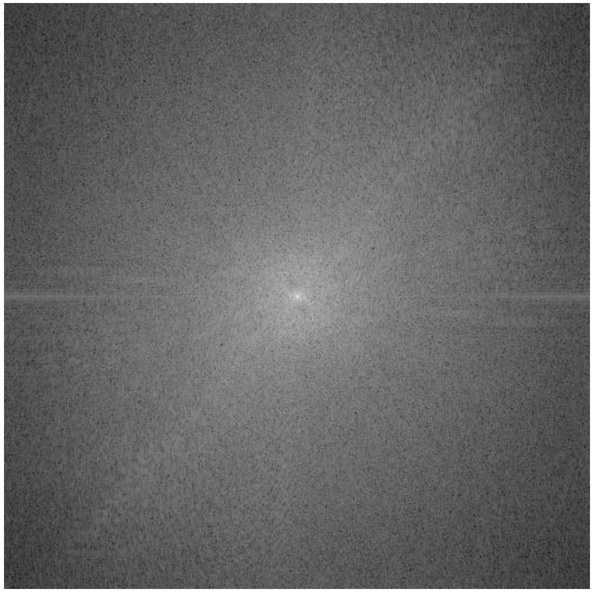
- Instead, think of phase and magnitude
  - Magnitude = magnitude of complex transform

    = *sqrt*($\text{REAL}(F)^2+\text{IMAGINARY}(F)^2$)

  - Phase = phase of the complex transform

    = *atan* (IMAGINARY(F)/REAL(F))

Phase and magnitude



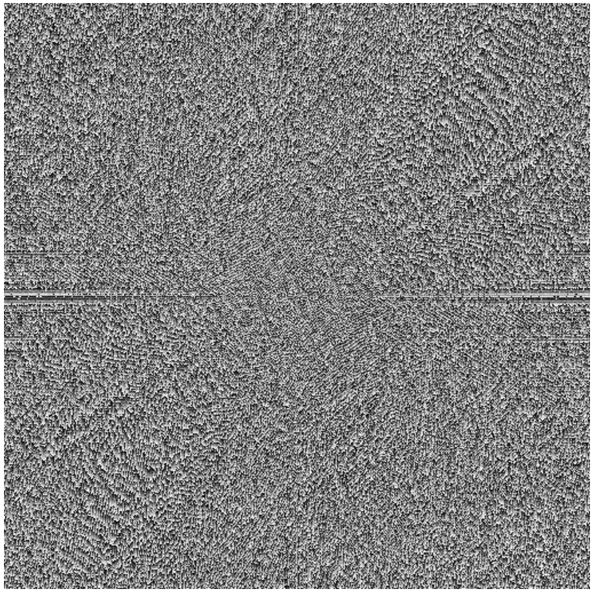
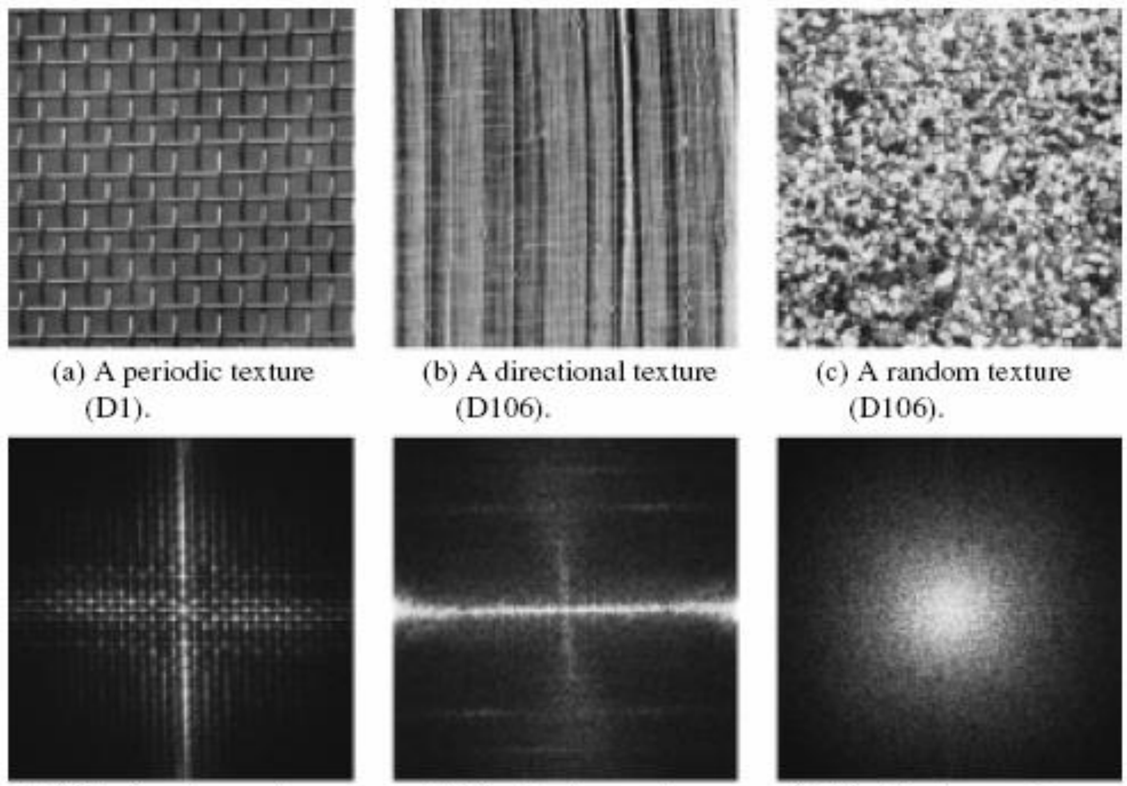


Image *f*(x,y)

Magnitude of F(u,v)

Phase of F(u,v)

Phase and magnitude



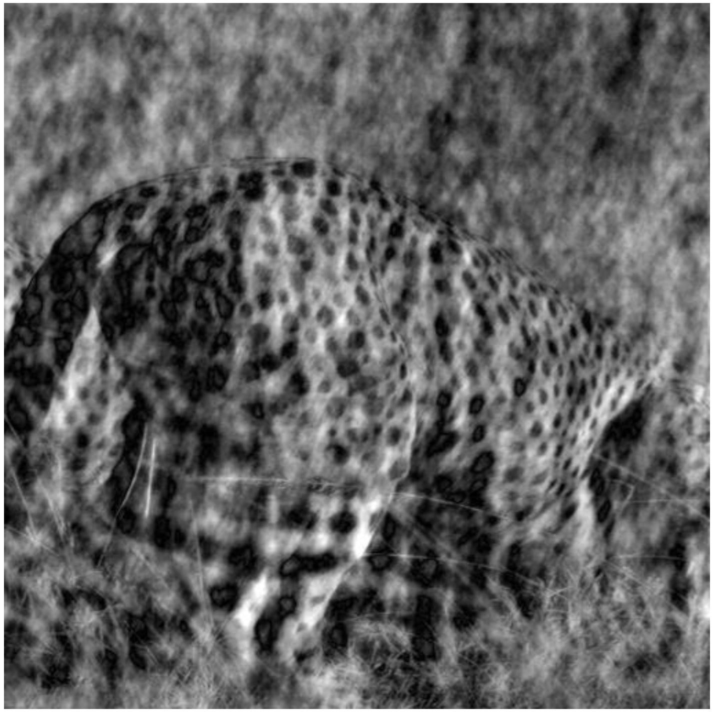
(a) A periodic texture (D1).
(b) A directional texture (D106).
(c) A random texture (D106).
(d) The Fourier spectrum of (a).
(e) The Fourier spectrum of (b).
(f) The Fourier spectrum of (c).

Figure from Lee and Chen, A New Method for Coarse Classification of Textures and Class Weight Estimation for Texture Retrieval, Pattern Recognition and Image Analysis, Vol. 12, No. 4, 2002, pp. 400–410.

Phase and magnitude

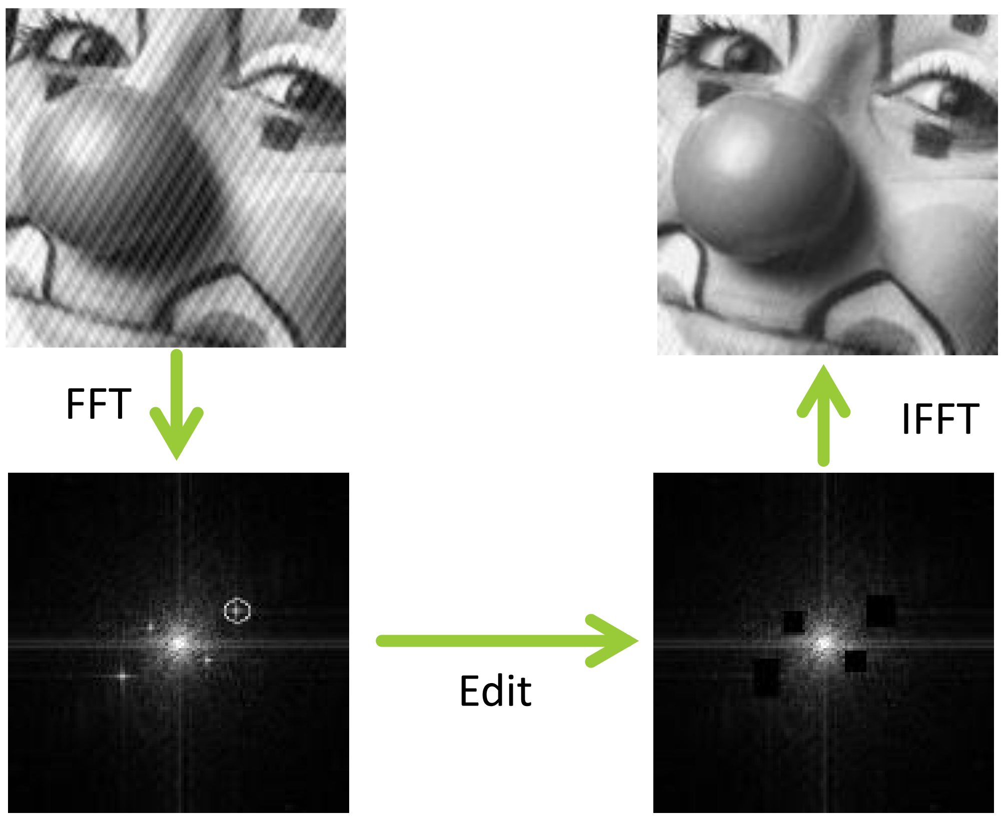
- Reconstruct with one's phase, and magnitude of the other.

Reconstruct with zebra phase, cheetah magnitude



Reconstruct with cheetah phase, zebra magnitude

The process and application of FFT



Source: www.mediacy.com/apps/fft.htm, Image Pro Plus FFT Example.

Slides from Earl F. Glynn, “Fourier Analysis and Image Processing”