COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Image Transforms and Image Enhancement in Frequency Domain

Lecture 5, Feb 25th, 2008

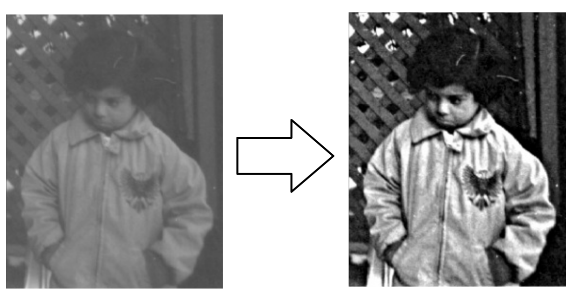
Lexing Xie

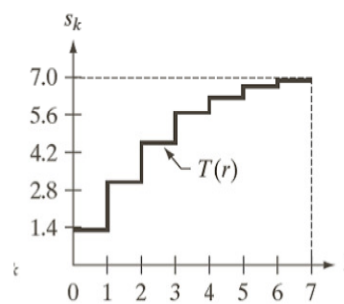
EE4830 Digital Image Processing
http://www.ee.columbia.edu/~xlx/ee4830/

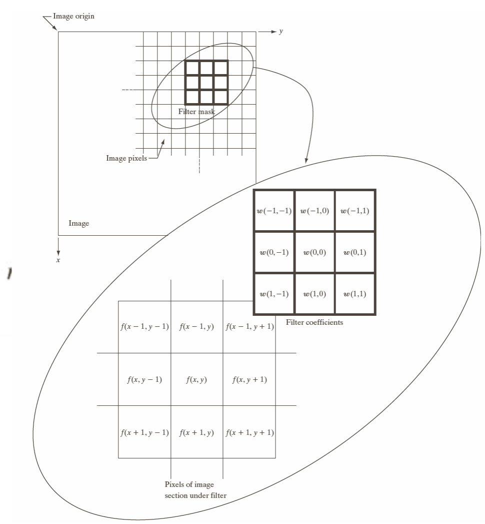
thanks to G&W website, Mani Thomas, Min Wu and Wade Trappe for slide materials

- HW clarification
  - HW#2 problem 1
    - Show: f - ∇^2 f $\approx$ A f – B blur(f)
    - A and B are constants that do not matter, it is up to you to find appropriate values of A and B, as well as the appropriate version of the blur function.

- Recap for lecture 4



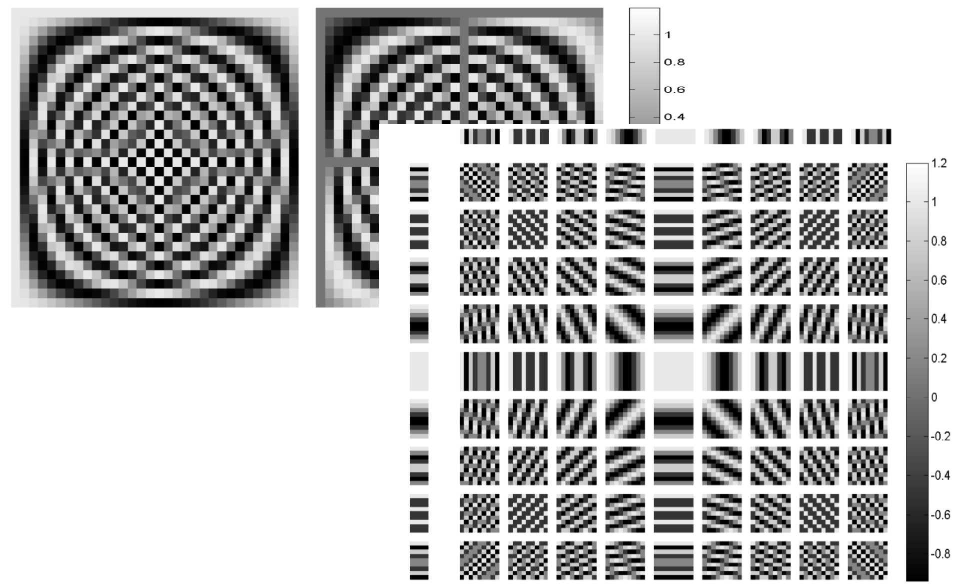




roadmap

$$e^{-j2\pi\frac{un}{N}}$$

$$e^{-j2\pi(\frac{um}{N}+\frac{vn}{N})}$$



- 2D-DFT definitions and intuitions
- DFT properties, applications
- pros and cons
- DCT

the return of DFT

- Fourier transform: a continuous signal can be represented as a (countable) weighted sum of sinusoids.

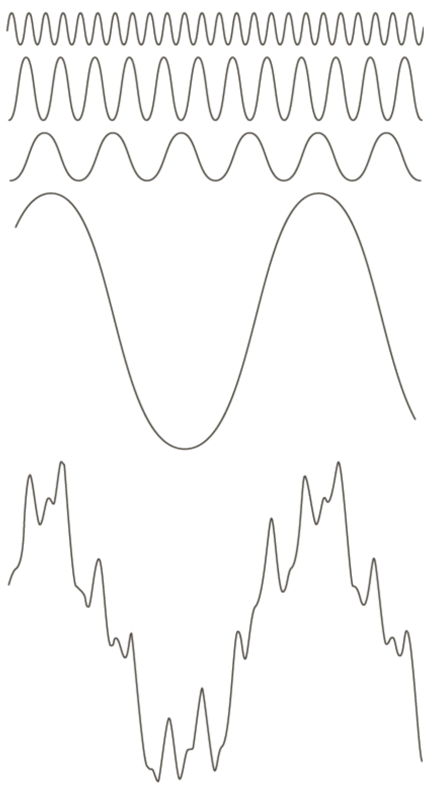
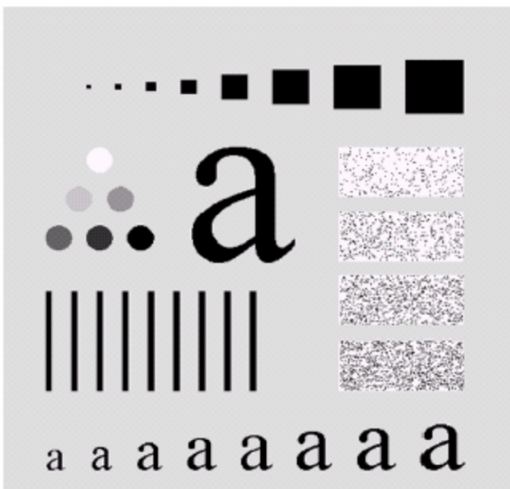


FIGURE 4.1 The function at the bottom is the sum of the four functions above it. Fourier's idea in 1807 that periodic functions could be represented as a weighted sum of sines and cosines was met with skepticism.

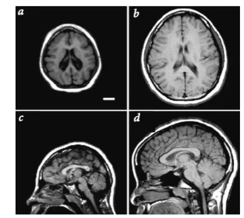
warm-up brainstorm

- Why do we need image transform?

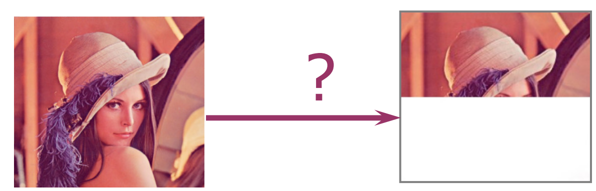


why transform?

- Better image processing
  - Take into account long-range correlations in space
  - Conceptual insights in spatial-frequency information. what it means to be "smooth, moderate change, fast change, …"
- Fast computation: convolution vs. multiplication



- Alternative representation and sensing
  - Obtain transformed data as measurement in radiology images (medical and astrophysics), inverse transform to recover image
- Efficient storage and transmission
  - Energy compaction
  - Pick a few "representatives" (basis)
  - Just store/send the "contribution" from each basis



outline

- why transform
- 2D Fourier transform
  - a picture book for DFT and 2D-DFT
  - properties
  - implementation
  - applications
- discrete cosine transform (DCT)
  - definition & visualization
  - Implementation

next lecture: transform of all flavors, unitary transform, KLT, others …

1-D continuous FT

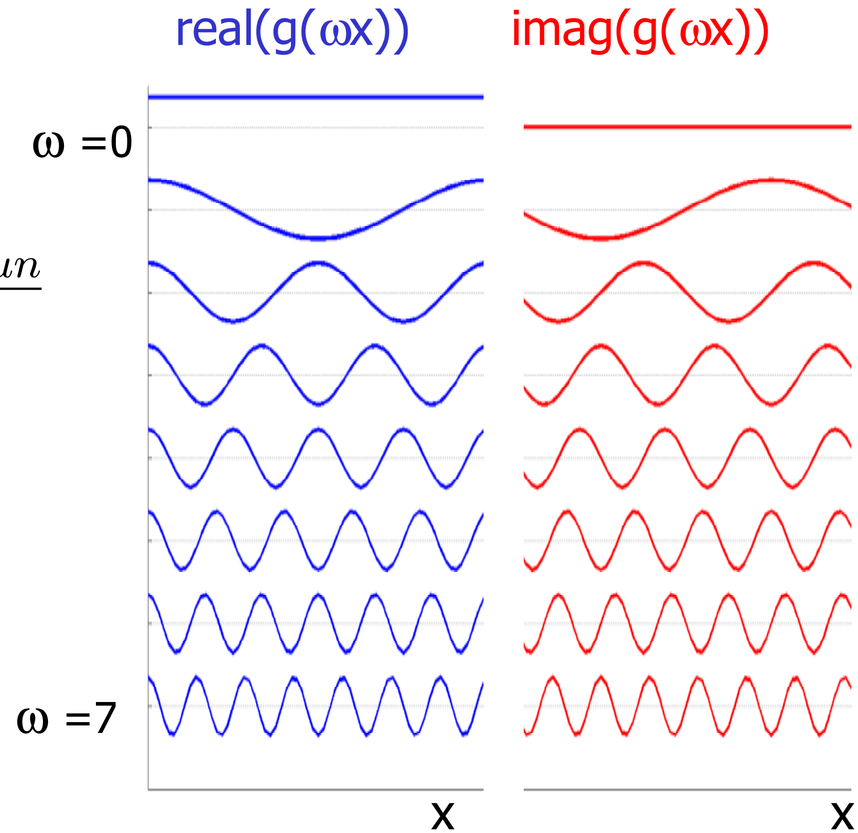
- 1D – FT

$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-j2\pi\omega x}$$

$$g(\omega x) = e^{-j2\pi\omega x}$$

- 1D – DFT of length N

$$F(u) \;\; = \frac{1}{N} \;\; \sum_{n=0}^{N-1} f(n) e^{\frac{-j2\pi un}{N}}$$



1-D DFT in as basis expansion

$$F(u) = \frac{1}{N} \sum_{n=0}^{N-1} f(n) e^{\frac{-j2\pi un}{N}}$$

Forward transform

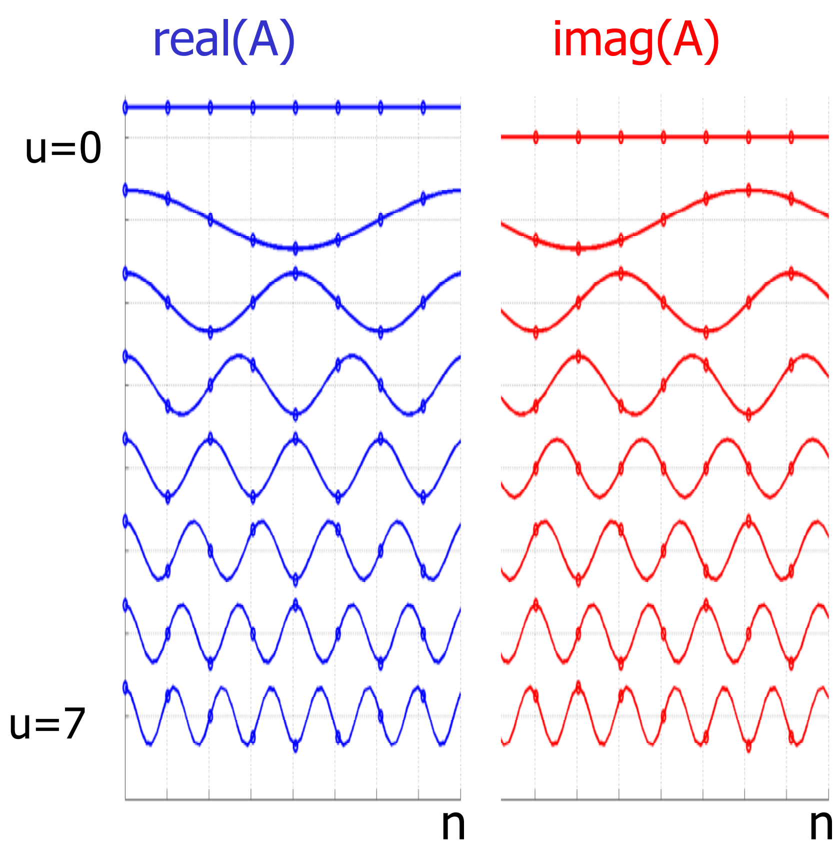
$$y(u) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) a(u,n)$$

Inverse transform

$$x(n) = \frac{1}{\sqrt{N}} \sum_{u=0}^{N-1} y(u) b(u,n)$$

basis

$$\begin{aligned} a(u,n) &= e^{-j2\pi\frac{un}{N}} \\ &= cos(2\pi\frac{un}{N}) - jsin(2\pi\frac{un}{N}) \end{aligned}$$



1-D DFT in matrix notations

$$y(u) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) a(u,n)$$

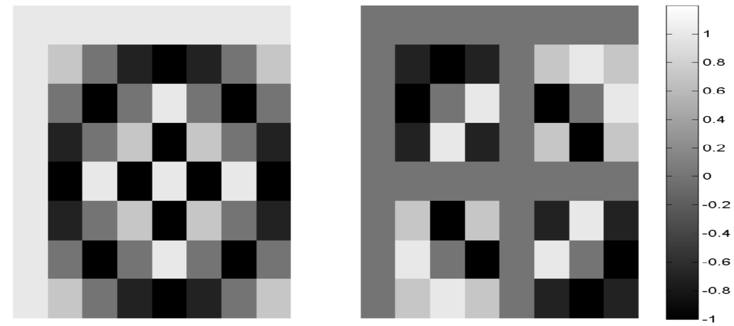
$$a(u,n) = e^{-j2\pi\frac{un}{N}}$$

$$= cos(2\pi\frac{un}{N}) - j sin(2\pi\frac{un}{N})$$

$$u = 0, 1, \ldots, N-1$$

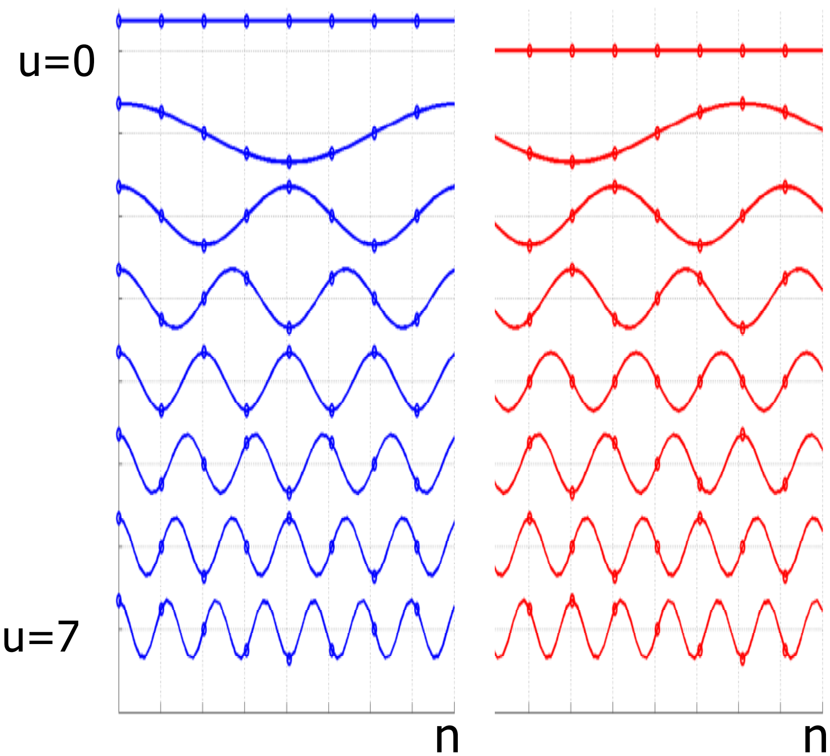
$$y = Ax$$

$$x = A^{-1}y$$



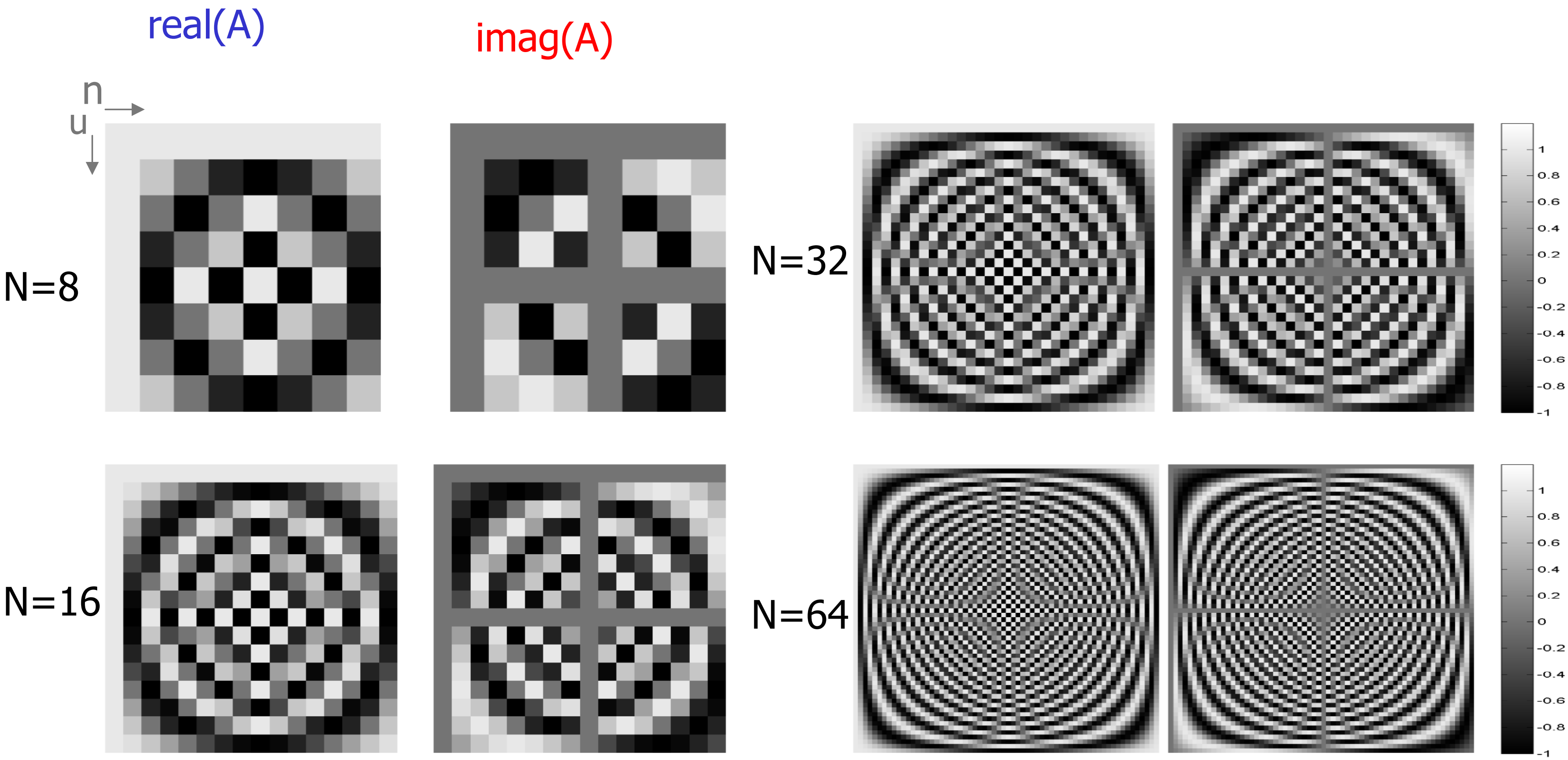
real(A) imag(A)

N=8



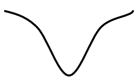
1-D DFT of different lengths

$$\begin{aligned} y &= Ax \\ x &= A^{-1}y \end{aligned} \qquad \begin{aligned} a(u,n) &= e^{-j2\pi\frac{un}{N}} \\ &= cos(2\pi\frac{un}{N}) - jsin(2\pi\frac{un}{N}) \end{aligned} \qquad u = 0, 1, \ldots, N-1$$



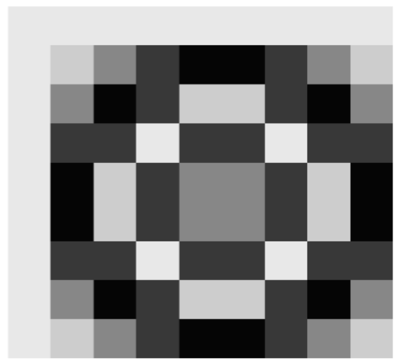
performing 1D DFT

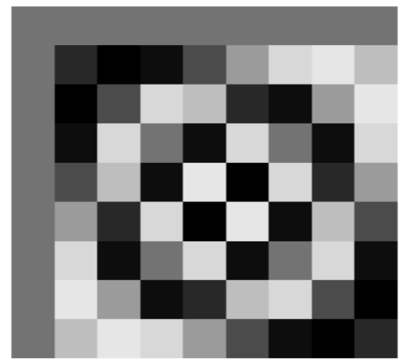
$x(n)$



real-valued input

$$a(u,n) = e^{-j2\pi\frac{un}{N}}$$





$y(u)$

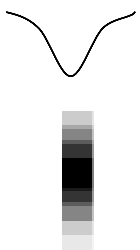
$$y \;=\; Ax$$

$$real(y) = real(A) * x \qquad imag(y) = imag(A) * x$$

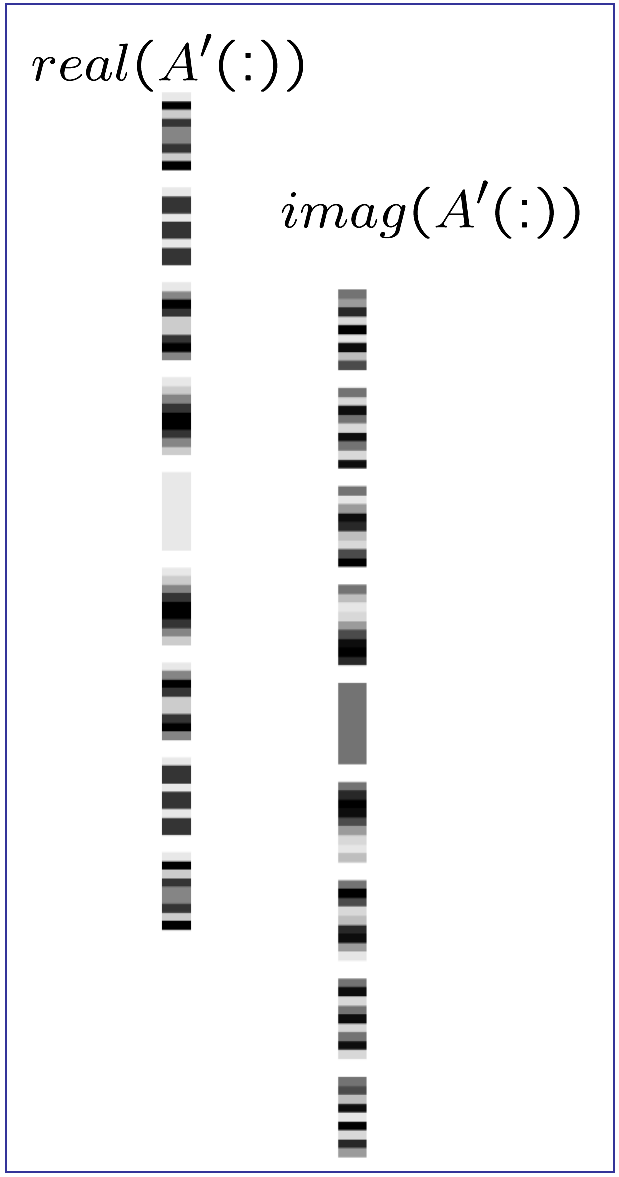
Note: the coefficients in x and y on this slide are only meant for illustration purposes, which are not numerically accurate.

another illustration of 1D-DFT

$x(n)$



real-valued input



$$y = Ax$$

$$y(u) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) a(u,n)$$

$real(y(u))$

$imag(y(u))$

Note: the coefficients in x and y are not numerically accurate

from 1D to 2D

1D

$$x(n)$$

$$a(u,n)$$

$$e^{-j2\pi \frac{un}{N}}$$

$$y(u) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) a(u,n)$$

$$y = Ax$$

2D

$$x(m,n)$$

$$a(u,v,\ m,n)$$

$$e^{-j2\pi (\frac{um}{N} + \frac{vn}{N})}$$

$$y(u,v) = \frac{1}{N} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(m,n) a(u,v,m,n)$$

?

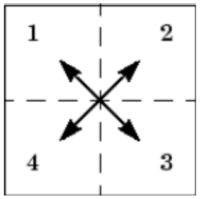
Computing 2D-DFT

DFT $$y(u,v) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(m,n) e^{\frac{-j2\pi um}{M}} e^{\frac{-j2\pi vn}{N}}$$

IDFT $$x(m,n) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} y(u,v) e^{\frac{j2\pi um}{M}} e^{\frac{j2\pi vn}{N}}$$

- Discrete, 2-D Fourier & inverse Fourier transforms are implemented in `fft2` and `ifft2`, respectively
- `fftshift`: Move origin (DC component) to image center for display
- Example:

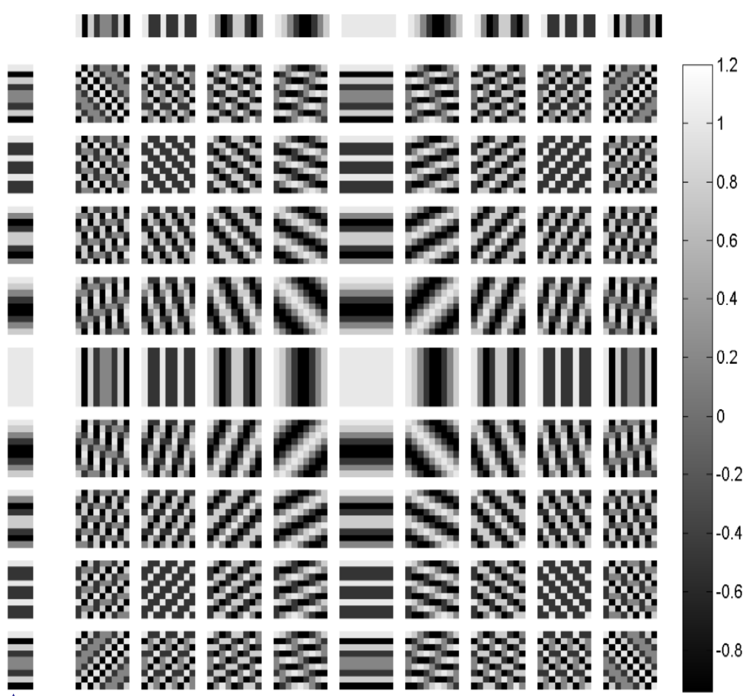
```
>> I = imread('test.png');    % Load grayscale image
>> F = fftshift(fft2(I));     % Shifted transform
>> imshow(log(abs(F)),[]);    % Show log magnitude
>> imshow(angle(F),[]);       % Show phase angle
```

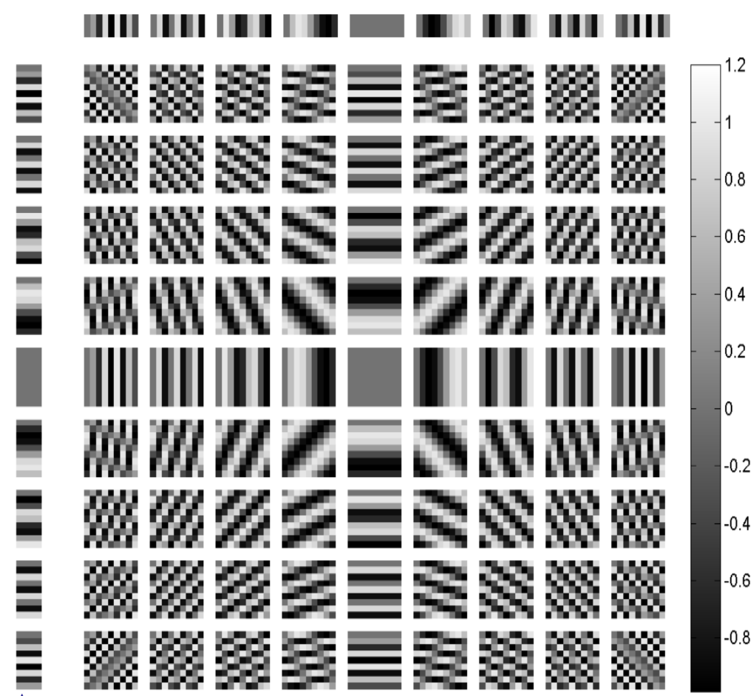


2-D Fourier basis

real $e^{-j2\pi(\frac{um}{N}+\frac{vn}{N})}$

imag $e^{-j2\pi(\frac{um}{N}+\frac{vn}{N})}$





real$(e^{-j2\pi\frac{um}{M}})$

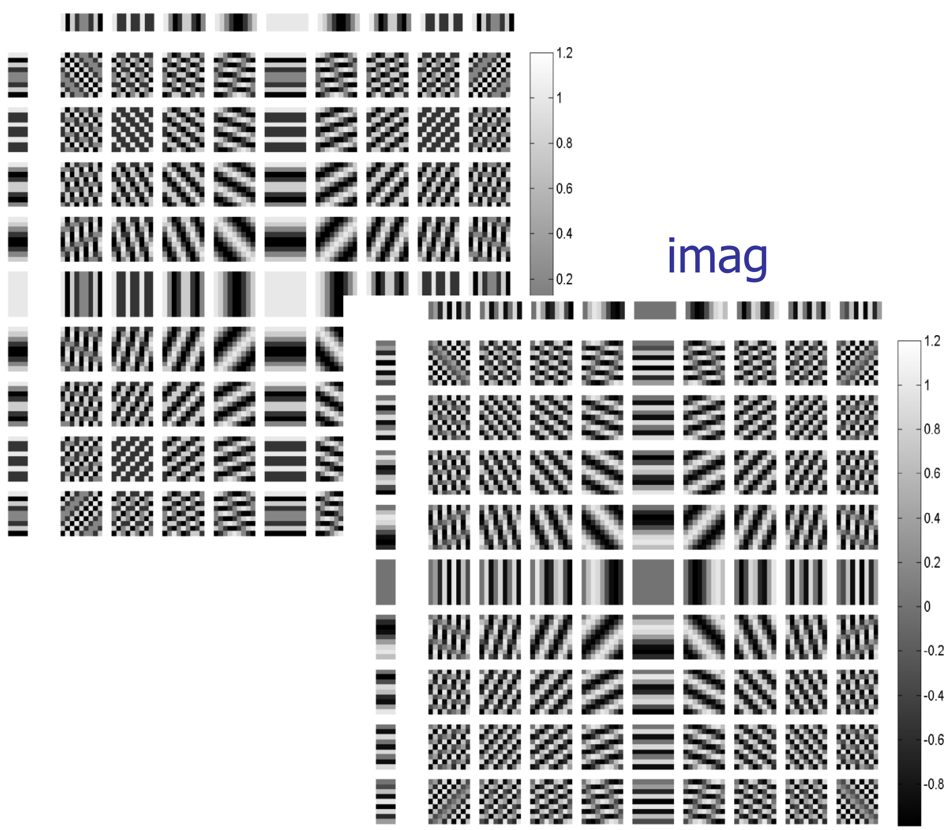
imag$(e^{-j2\pi\frac{um}{M}})$

2-D FT illustrated

real-valued

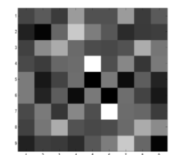
$x(m,n)$

$$a(u,v,m,n) = e^{-j2\pi\left(\frac{um}{N}+\frac{vn}{N}\right)}$$

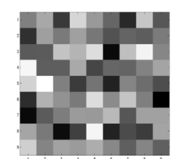


$y(u,v)$

$real(y(u,v))$



$imag(y(u,v))$



notes about 2D-DFT

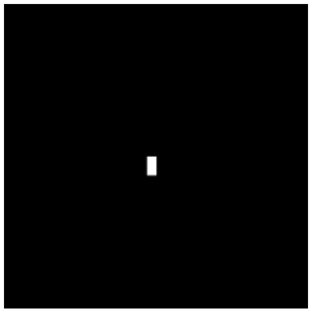
- Output of the Fourier transform is a complex number
  - Decompose the complex number as the magnitude and phase components
- In Matlab: `u = real(z), v = imag(z), r = abs(z),` and `theta = angle(z)`

Some useful FT pairs:

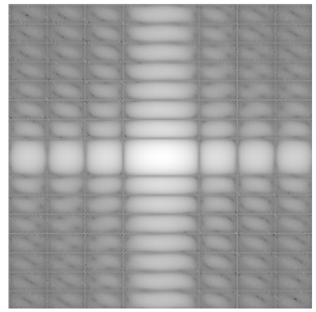
Impulse	$\delta(x, y) \Leftrightarrow 1$
Gaussian	$A\sqrt{2\pi}\sigma e^{-2\pi^2\sigma^2(x^2+y^2)} \Leftrightarrow Ae^{-(u^2+v^2)/2\sigma^2}$
Rectangle	$\text{rect}[a, b] \Leftrightarrow ab\dfrac{\sin(\pi ua)}{(\pi ua)}\dfrac{\sin(\pi vb)}{(\pi vb)}e^{-j\pi(ua+vb)}$
Cosine	$\cos(2\pi u_0 x + 2\pi v_0 y) \Leftrightarrow \frac{1}{2}\left[\delta(u + u_0, v + v_0) + \delta(u - u_0, v - v_0)\right]$
Sine	$\sin(2\pi u_0 x + 2\pi v_0 y) \Leftrightarrow j\frac{1}{2}\left[\delta(u + u_0, v + v_0) - \delta(u - u_0, v - v_0)\right]$

† Assumes that functions have been extended by zero padding.

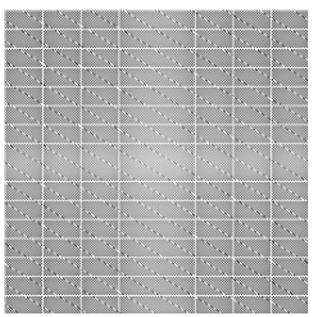
Explaining 2D-DFT

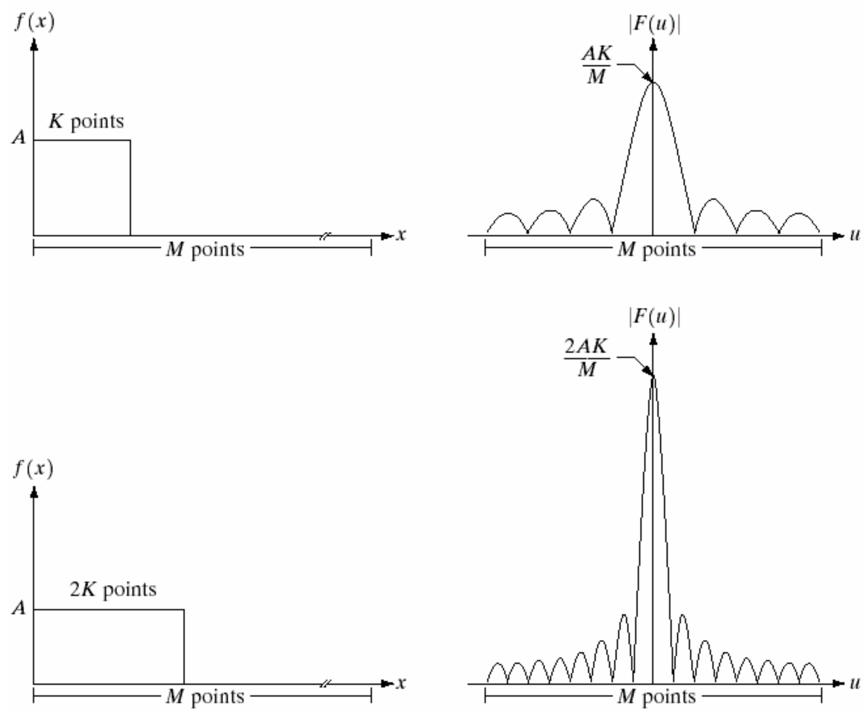


fft2



ifft2

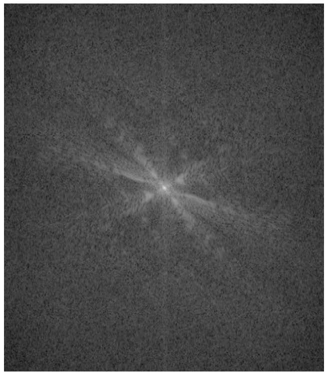
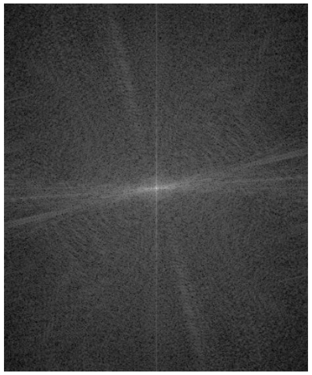




a b
c d

FIGURE 4.2 (a) A discrete function of M points, and (b) its Fourier spectrum. (c) A discrete function with twice the number of nonzero points, and (d) its Fourier spectrum.



circular convolution and zero padding

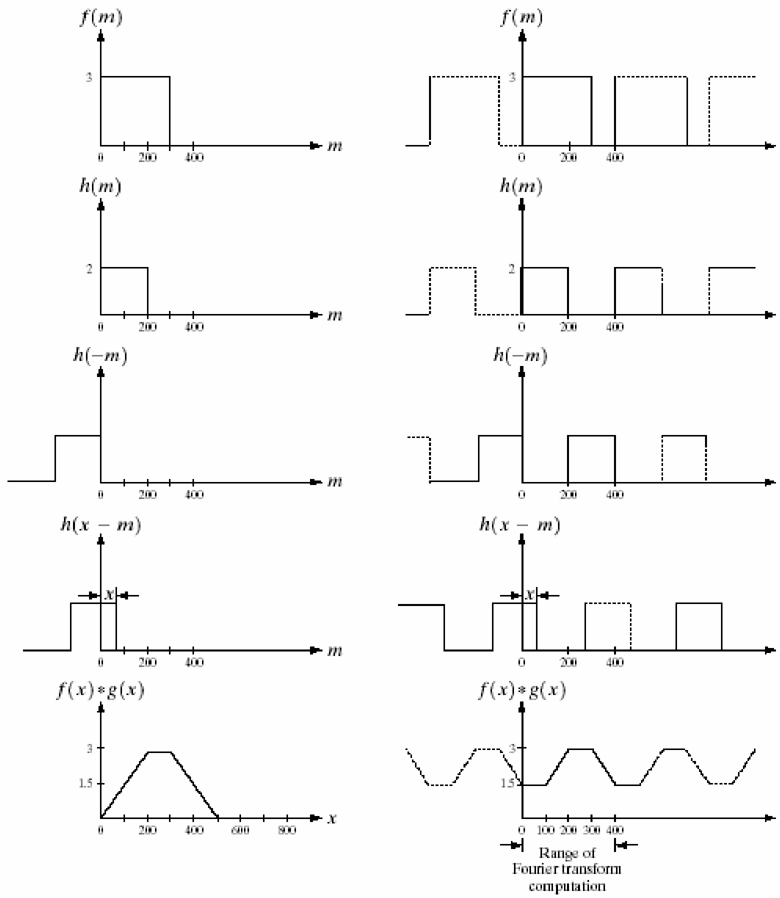


FIGURE 4.36 Left: convolution of two discrete functions. Right: convolution of the same functions, taking into account the implied periodicity of the DFT. Note in (j) how data from adjacent periods corrupt the result of convolution.

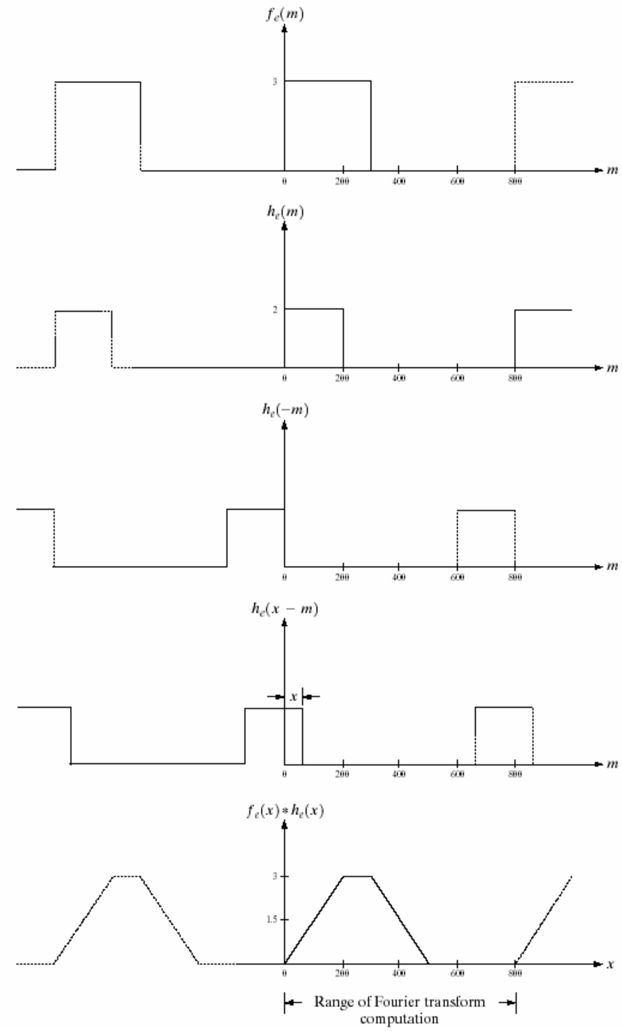


FIGURE 4.37 Result of performing convolution with extended functions. Compare Figs. 4.37(e) and 4.36(e).

zero padded filter and response

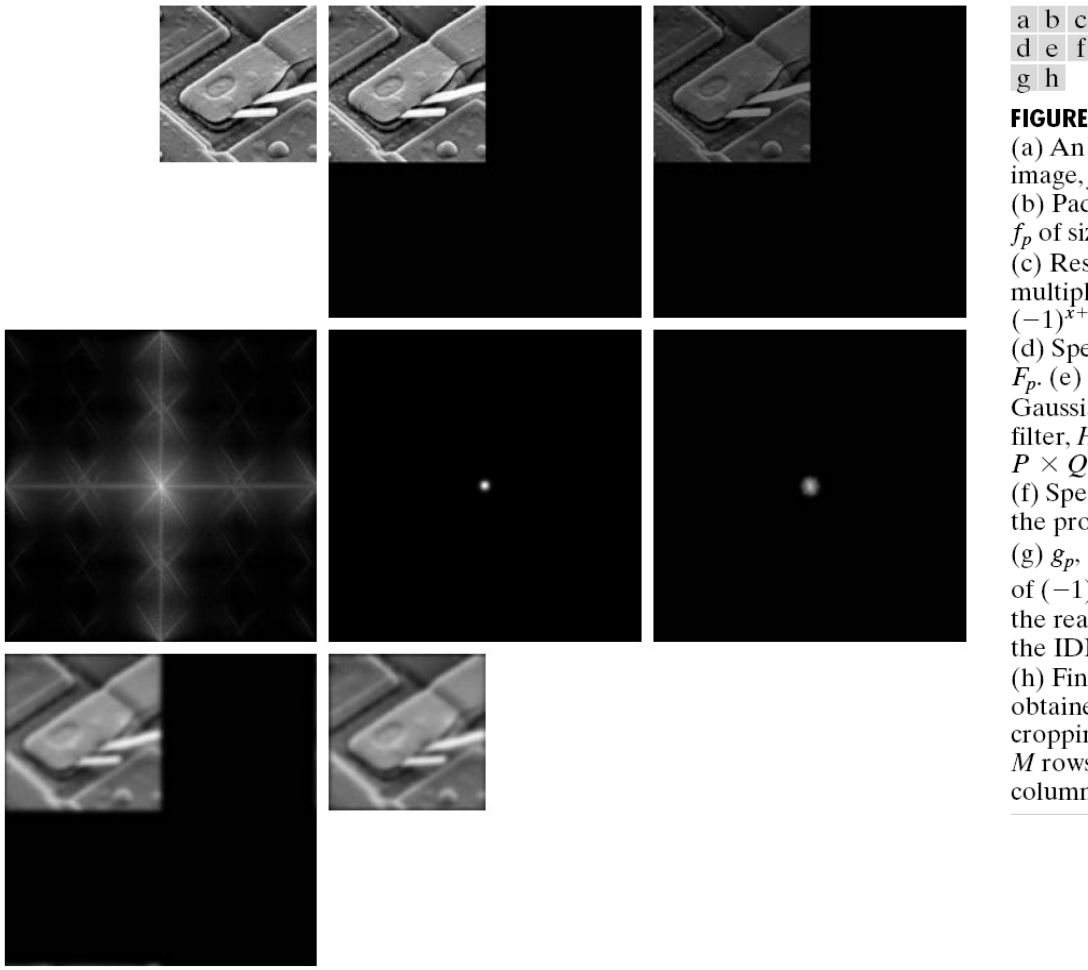


FIGURE 4.36 (a) An $M \times N$ image, f. (b) Padded image, f_p of size $P \times Q$. (c) Result of multiplying f_p by $(-1)^{x+y}$. (d) Spectrum of F_p. (e) Centered Gaussian lowpass filter, H, of size $P \times Q$. (f) Spectrum of the product HF_p. (g) g_p, the product of $(-1)^{x+y}$ and the real part of the IDFT of HF_p. (h) Final result, g, obtained by cropping the first M rows and N columns of g_p.

zero padded filter and response

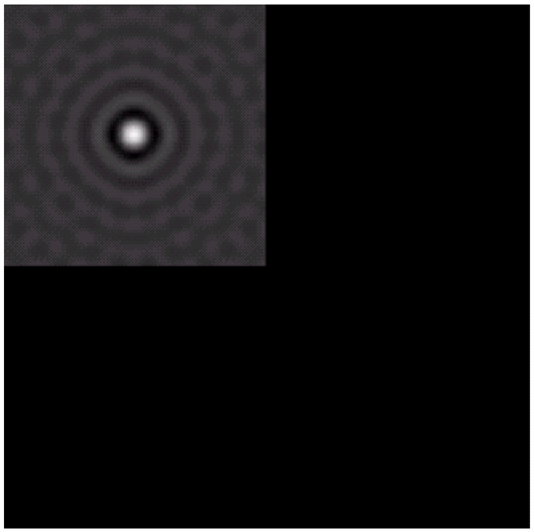
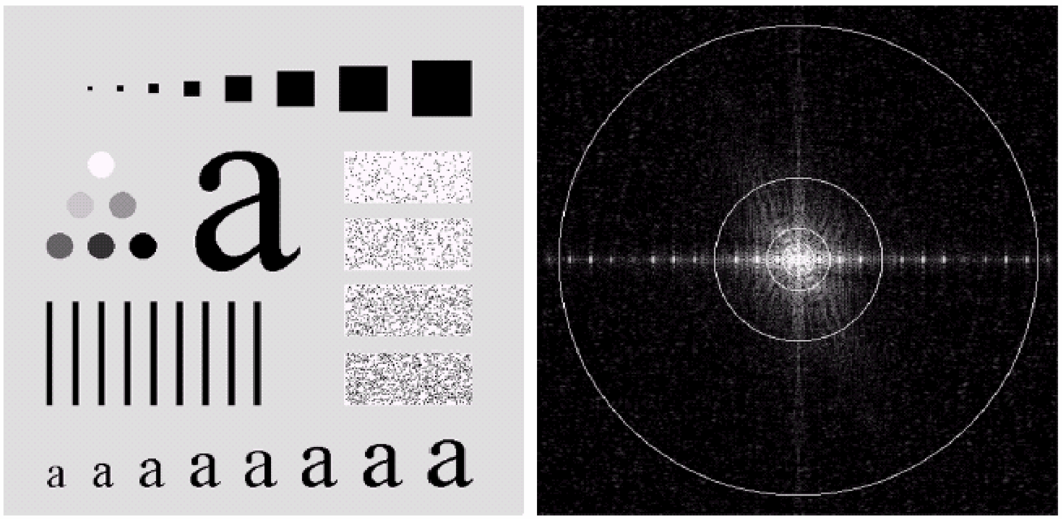


FIGURE 4.39 Padded lowpass filter is the spatial domain (only the real part is shown).

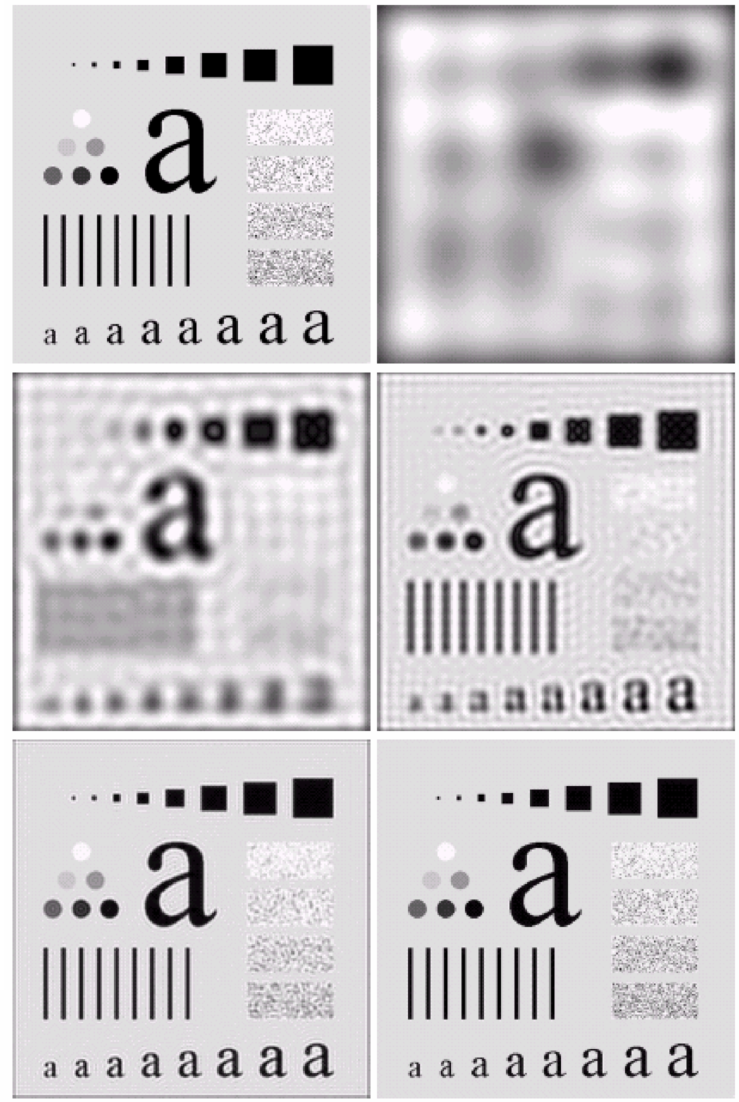
FIGURE 4.40 Result of filtering with padding. The image is usually cropped to its original size since there is little valuable information past the image boundaries.

observation 1: compacting energy



a b

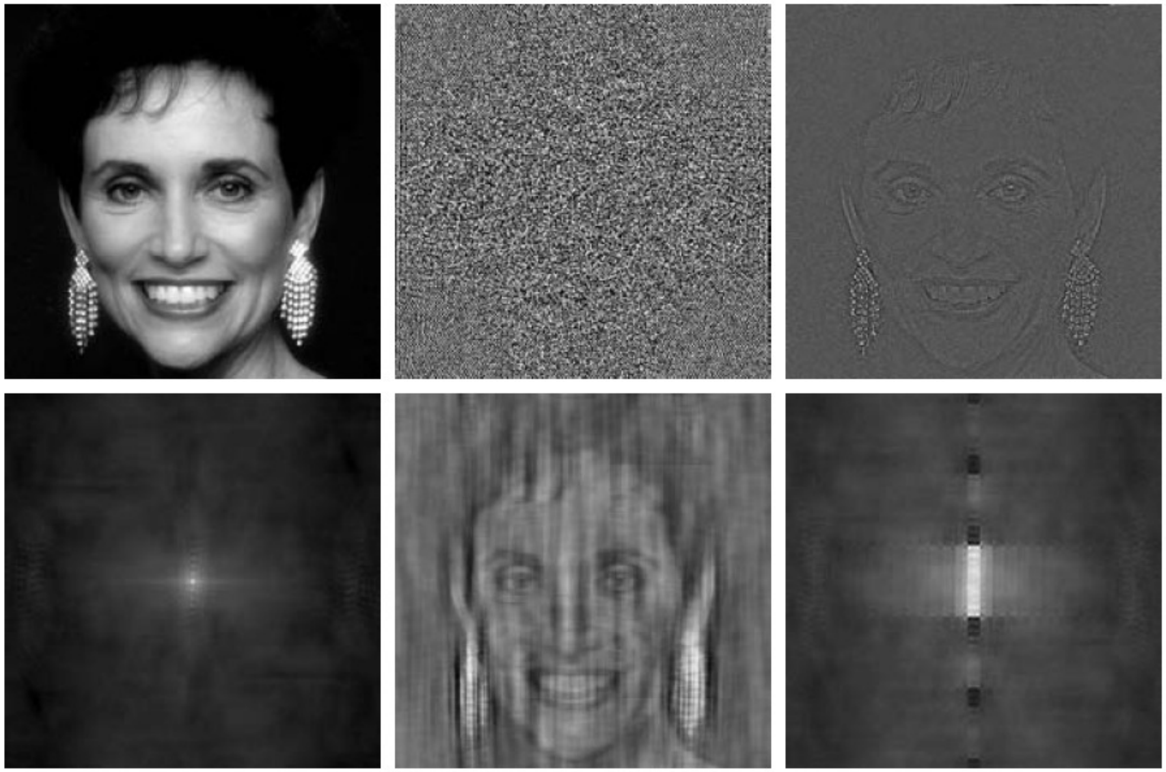
FIGURE 4.11 (a) An image of size 500 × 500 pixels and (b) its Fourier spectrum. The superimposed circles have radii values of 5, 15, 30, 80, and 230, which enclose 92.0, 94.6, 96.4, 98.0, and 99.5% of the image power, respectively.



a b
c d
e f

FIGURE 4.12 (a) Original image. (b)–(f) Results of ideal lowpass filtering with cutoff frequencies set at radii values of 5, 15, 30, 80, and 230, as shown in Fig. 4.11(b). The power removed by these filters was 8, 5.4, 3.6, 2, and 0.5% of the total, respectively.

observation 2: amplitude vs. phase



a b c
d e f

FIGURE 4.27 (a) Woman. (b) Phase angle. (c) Woman reconstructed using only the phase angle. (d) Woman reconstructed using only the spectrum. (e) Reconstruction using the phase angle corresponding to the woman and the spectrum corresponding to the rectangle in Fig. 4.24(a). (f) Reconstruction using the phase of the rectangle and the spectrum of the woman.

- Amplitude: relative prominence of sinusoids
- Phase: relative displacement of sinusoids

another example: amplitude vs. phase

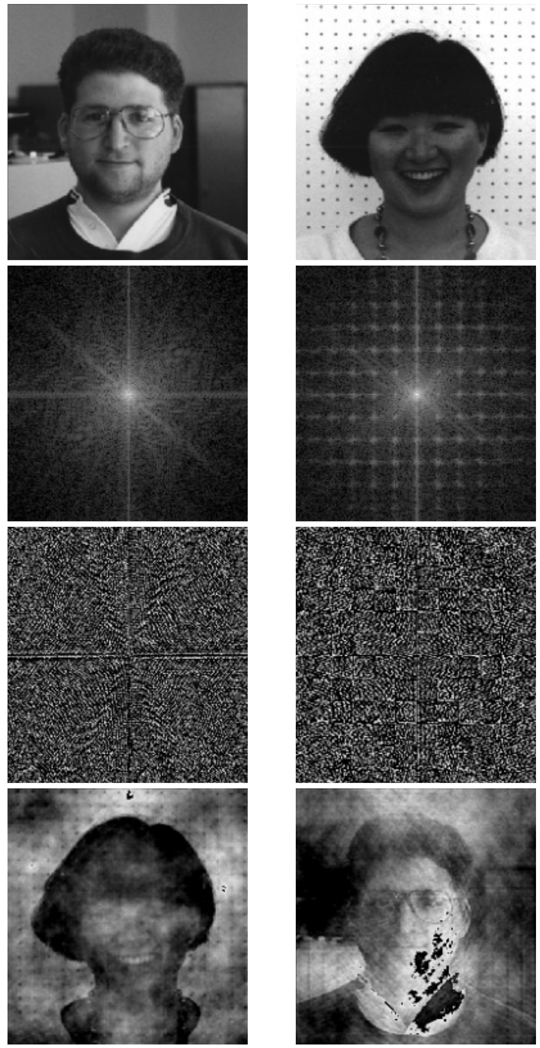
A = “Aron”

FA = fft2(A)

log(abs(FA))

angle(FA)

ifft2(abs(FA), angle(FP))



P = “Phyllis”

FP = fft2(P)

log(abs(FP))

angle(FP)

ifft2(abs(FP), angle(FA))

Adpated from http://robotics.eecs.berkeley.edu/~sastry/ee20/vision2/vision2.html

fast implementation of 2-D DFT

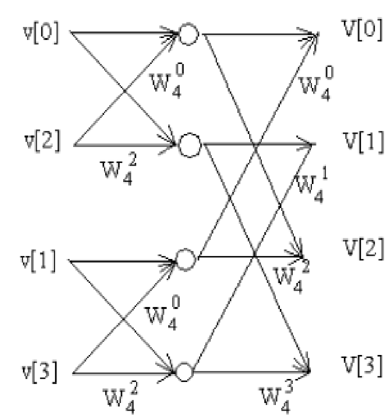
- 2 Dimensional DFT is separable

$$
\begin{aligned}
F(u,v) &= \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(m,n)e^{\frac{-2\pi jum}{M}}e^{\frac{-2\pi jvn}{N}} \\
&= \frac{1}{M}\sum_{x=0}^{M-1} e^{\frac{-2\pi jum}{M}} \cdot \frac{1}{N}\sum_{y=0}^{N-1} f(m,n)e^{\frac{-2\pi jvn}{N}} \\
&= \frac{1}{M}\sum_{x=0}^{M-1} e^{\frac{-2\pi jum}{M}} F(m,v)
\end{aligned}
$$

1-D DFT
of f(m,n)
w.r.t n

1-D DFT
of F(m,v)
w.r.t m

- 1D FFT: $O(N \cdot \log_2 N)$
- 2D DFT naïve implementation: $O(N^4)$
- 2D DFT as 1D FFT for each row and then for each column



Implement IDFT as DFT

DFT2 $$F(u,v) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) e^{-j2\pi(\frac{um}{M} + \frac{vn}{N})}$$

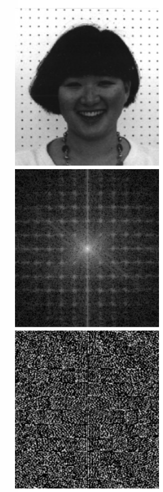
IDFT2 $$f(m,n) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{j2\pi(\frac{um}{M} + \frac{vn}{N})}$$

$$\Rightarrow \quad \begin{aligned} f^*(m,n) &= \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F^*(u,v) e^{-j2\pi(\frac{um}{M} + \frac{vn}{N})} \\ &= (MN) \cdot DFT2[F^*(u,v)] \end{aligned}$$

Properties of 2D-DFT

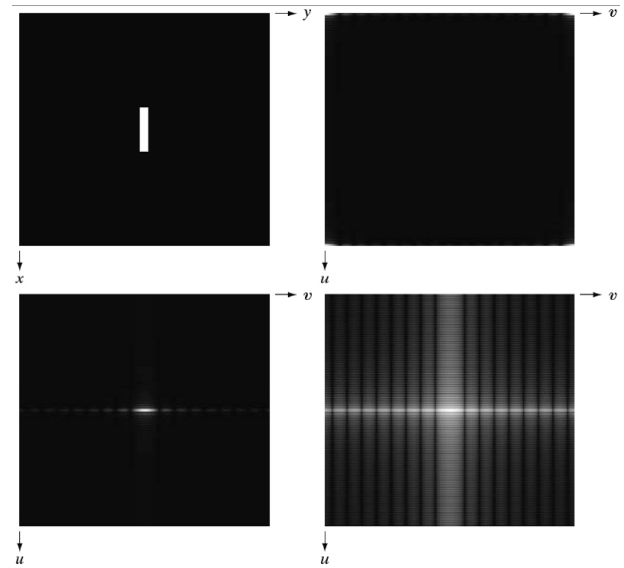
TABLE 4.1 Summary of some important properties of the 2-D Fourier transform.

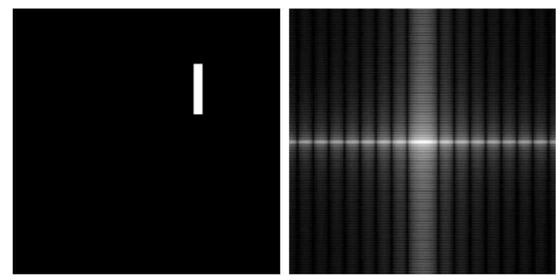
Property	Expression(s)
Fourier transform	$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)}$
Inverse Fourier transform	$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M+vy/N)}$
Polar representation	$F(u,v) = \lvert F(u,v)\rvert e^{-j\phi(u,v)}$
Spectrum	$\lvert F(u,v)\rvert = \left[R^2(u,v) + I^2(u,v)\right]^{1/2}$, $R = \text{Real}(F)$ and $I = \text{Imag}(F)$
Phase angle	$\phi(u,v) = \tan^{-1}\left[\frac{I(u,v)}{R(u,v)}\right]$
Power spectrum	$P(u,v) = \lvert F(u,v)\rvert^2$
Average value	$\bar{f}(x,y) = F(0,0) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)$
Translation	$f(x,y)e^{j2\pi(u_0x/M+v_0y/N)} \Leftrightarrow F(u-u_0, v-v_0)$ $f(x-x_0, y-y_0) \Leftrightarrow F(u,v)e^{-j2\pi(ux_0/M+vy_0/N)}$ When $x_0 = u_0 = M/2$ and $y_0 = v_0 = N/2$, then $f(x,y)(-1)^{x+y} \Leftrightarrow F(u-M/2, v-N/2)$ $f(x-M/2, y-N/2) \Leftrightarrow F(u,v)(-1)^{u+v}$

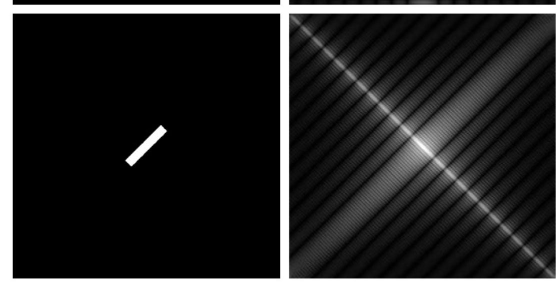


Name	DFT Pairs
1) Symmetry properties	See Table 4.1
2) Linearity	$af_1(x, y) + bf_2(x, y) \Leftrightarrow aF_1(u, v) + bF_2(u, v)$
3) Translation (general)	$f(x, y)e^{j2\pi(u_0x/M + v_0y/N)} \Leftrightarrow F(u - u_0, v - v_0)$ $f(x - x_0, y - y_0) \Leftrightarrow F(u, v)e^{-j2\pi(ux_0/M + vy_0/N)}$
4) Translation to center of the frequency rectangle, $(M/2, N/2)$	$f(x, y)(-1)^{x+y} \Leftrightarrow F(u - M/2, v - N/2)$ $f(x - M/2, y - N/2) \Leftrightarrow F(u, v)(-1)^{u+v}$
5) Rotation	$f(r, \theta + \theta_0) \Leftrightarrow F(\omega, \varphi + \theta_0)$ $x = r\cos\theta \quad y = r\sin\theta \quad u = \omega\cos\varphi \quad v = \omega\sin\varphi$
6) Convolution theorem†	$f(x, y) \star h(x, y) \Leftrightarrow F(u, v)H(u, v)$ $f(x, y)h(x, y) \Leftrightarrow F(u, v) \star H(u, v)$

(*Continued*)







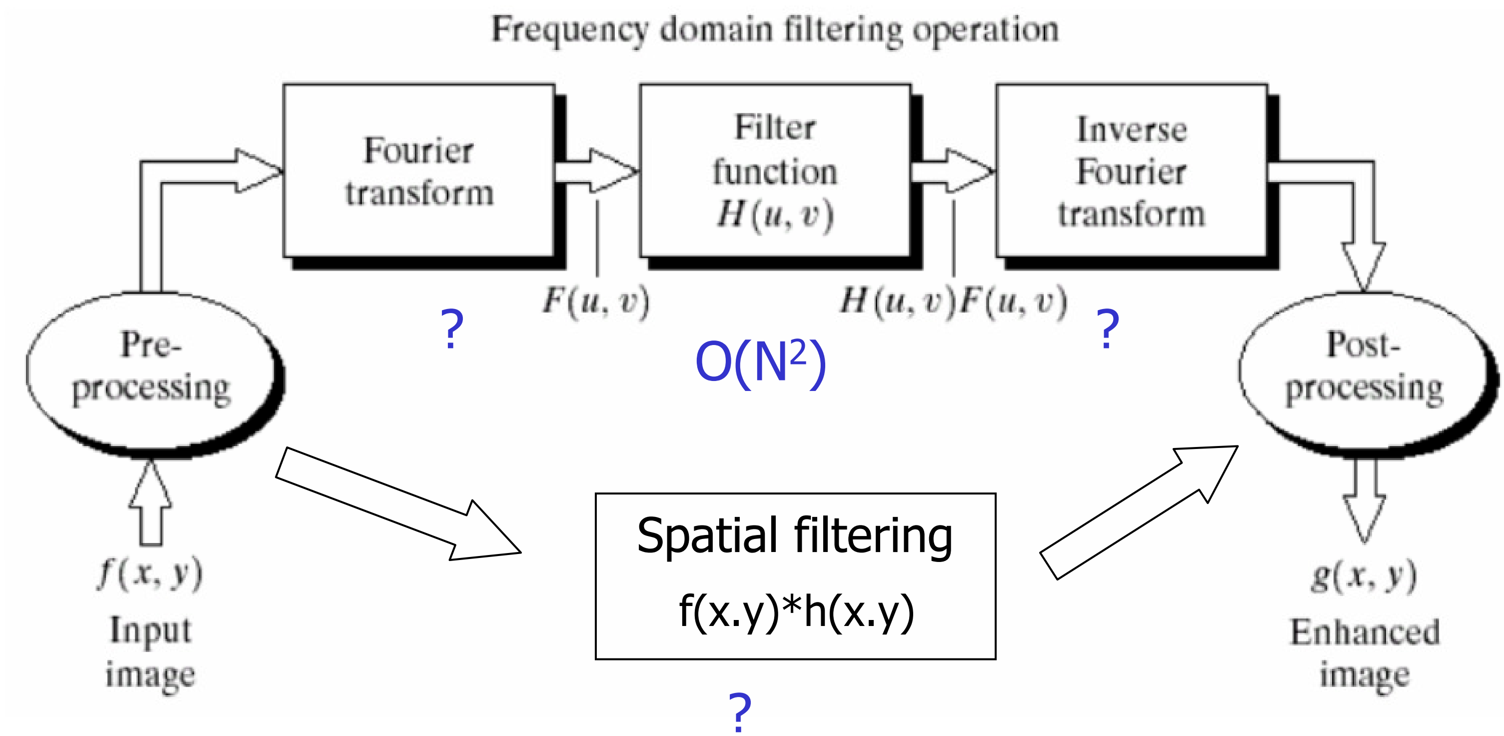
Property	Expression(s)
Computation of the inverse Fourier transform using a forward transform algorithm	$\frac{1}{MN} f^*(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F^*(u, v) e^{-j2\pi(ux/M + vy/N)}$ This equation indicates that inputting the function $F^*(u, v)$ into an algorithm designed to compute the forward transform (right side of the preceding equation) yields $f^*(x, y)/MN$. Taking the complex conjugate and multiplying this result by MN gives the desired inverse.
Convolution[†]	$f(x, y) * h(x, y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) h(x - m, y - n)$
Correlation[†]	$f(x, y) \circ h(x, y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f^*(m, n) h(x + m, y + n)$
Convolution theorem[†]	$f(x, y) * h(x, y) \Leftrightarrow F(u, v) H(u, v)$; $f(x, y) h(x, y) \Leftrightarrow F(u, v) * H(u, v)$
Correlation theorem[†]	$f(x, y) \circ h(x, y) \Leftrightarrow F^*(u, v) H(u, v)$; $f^*(x, y) h(x, y) \Leftrightarrow F(u, v) \circ H(u, v)$

duality result

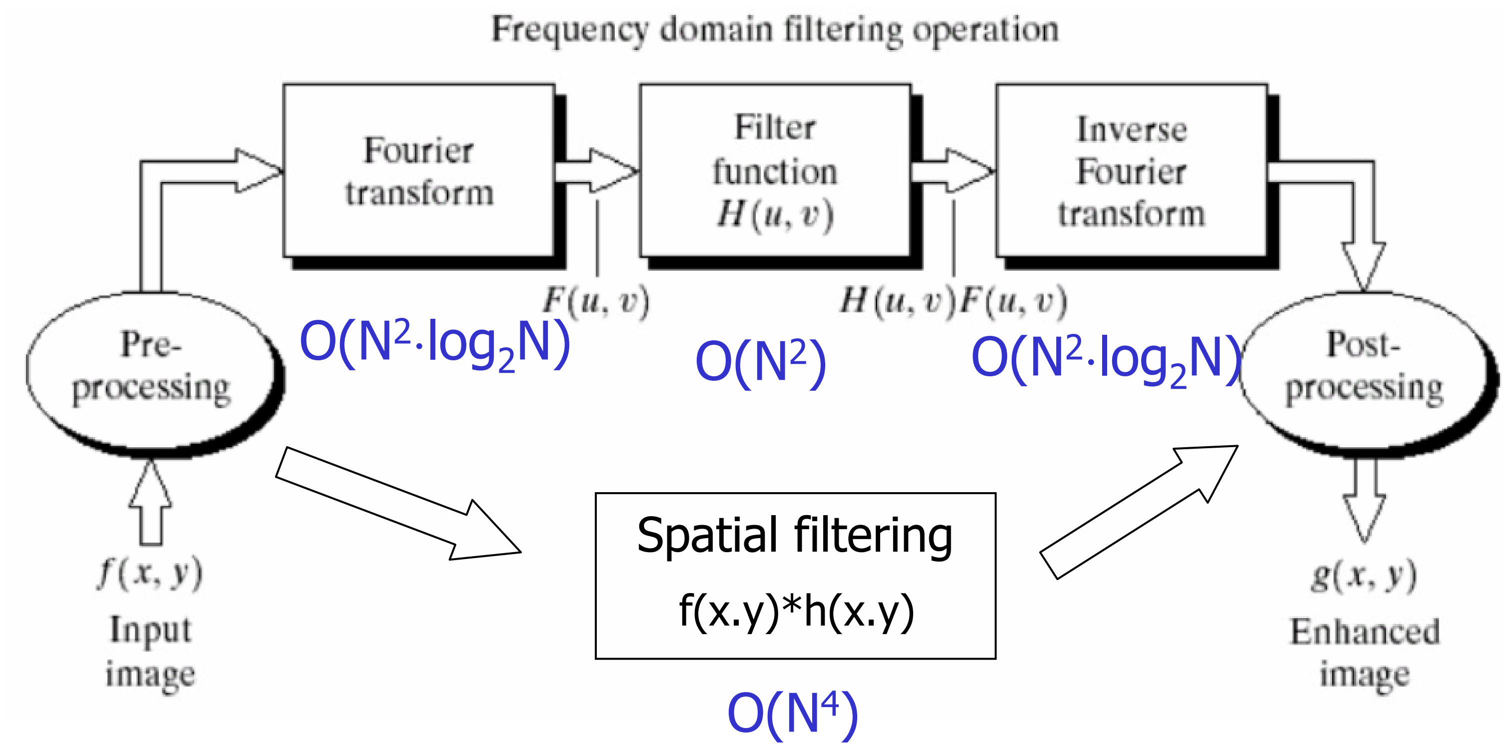
outline

- why transform
- 2D Fourier transform
  - a picture book for DFT and 2D-DFT
  - properties
  - implementation
  - applications
- discrete cosine transform (DCT)
  - definition & visualization
  - implementation

DFT application #1: fast Convolution

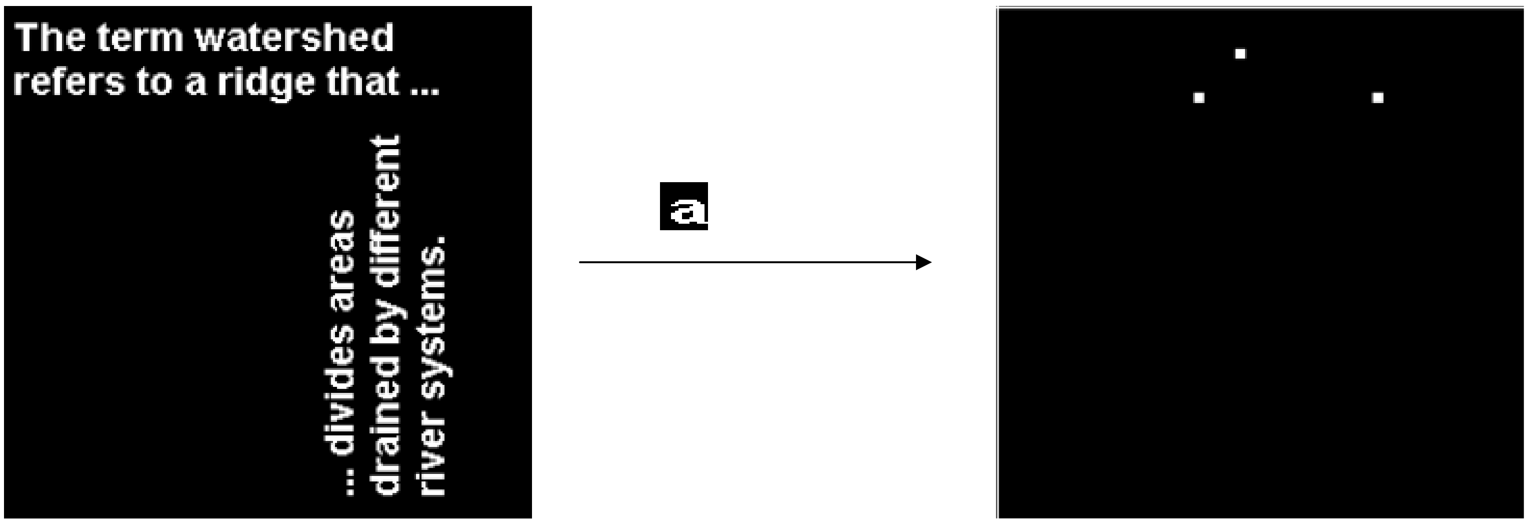


DFT application #1: fast convolution



DFT application #2: feature correlation

- Find letter "a" in the following image

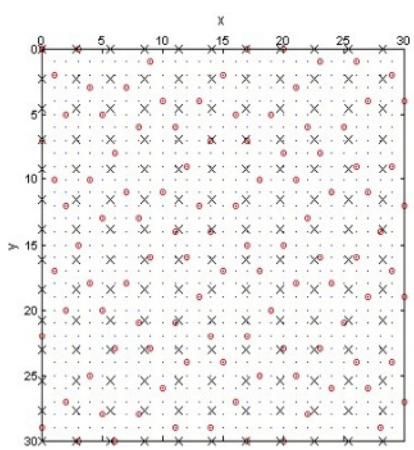


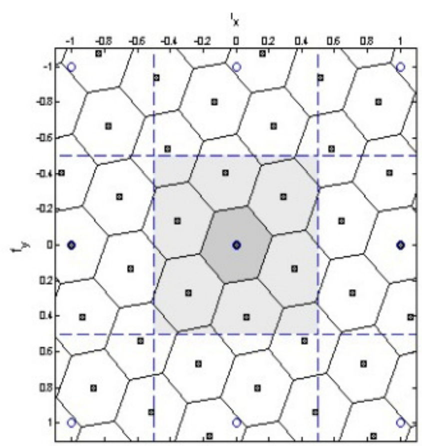

```
bw = imread('text.png');      a = imread('letter_a.png');
% Convolution is equivalent to correlation if you rotate the
convolution kernel by 180deg
C = real(ifft2(fft2(bw) .*fft2(rot90(a,2),256,256)));

% Use a threshold that's a little less than max.
% Display showing pixels over threshold.
thresh = .9*max(C(:));        figure, imshow(C > thresh)
```

from Matlab image processing demos.

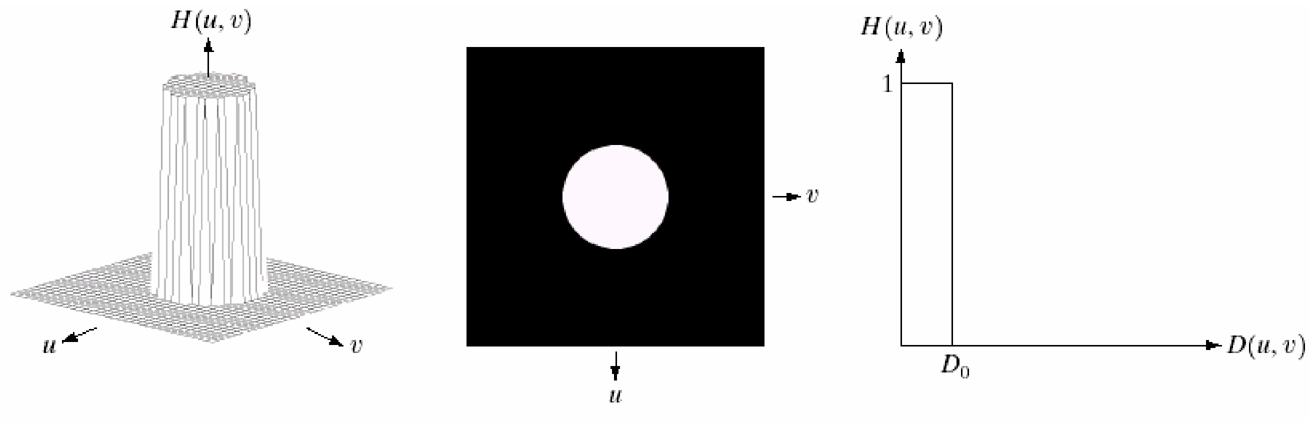
DFT application #3: image filters

- Zoology of image filters
  - Smoothing / Sharpening / Others
  - Support in time vs. support in frequency c.f. “FIR / IIR”
  - Definition: spatial domain/frequency domain
  - Separable / Non-separable





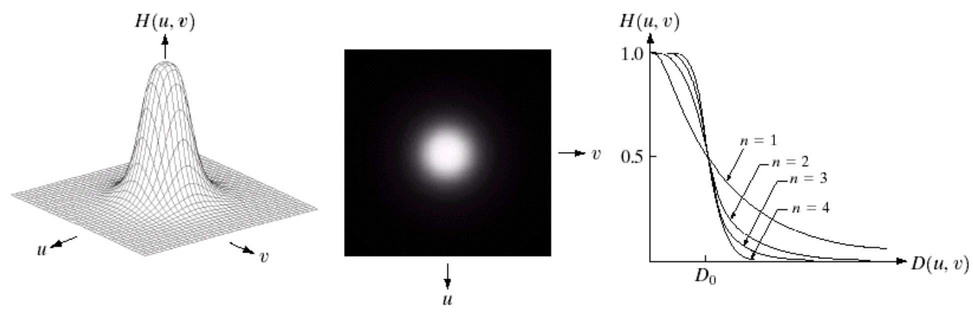
smoothing filters: ideal low-pass



a b c

FIGURE 4.10 (a) Perspective plot of an ideal lowpass filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.

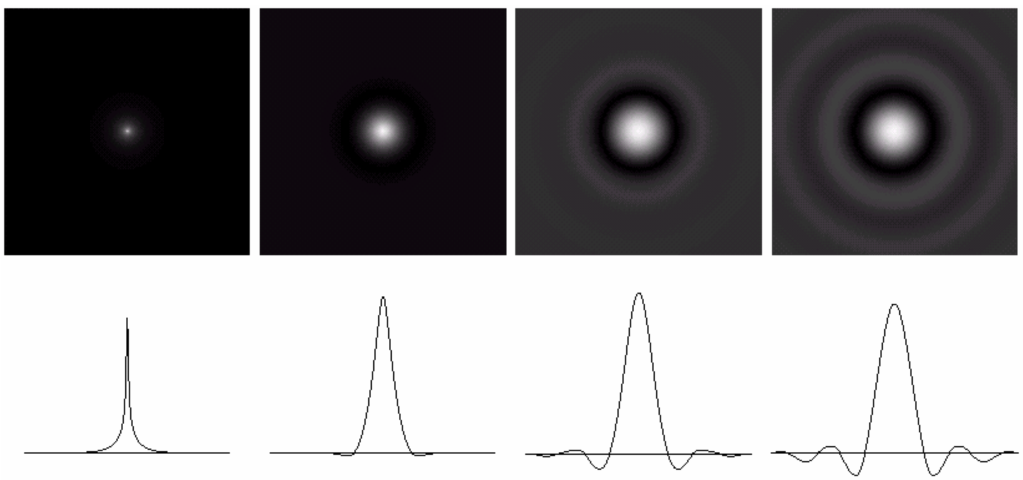
butterworth filters



a b c

FIGURE 4.14 (a) Perspective plot of a Butterworth lowpass filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections of orders 1 through 4.

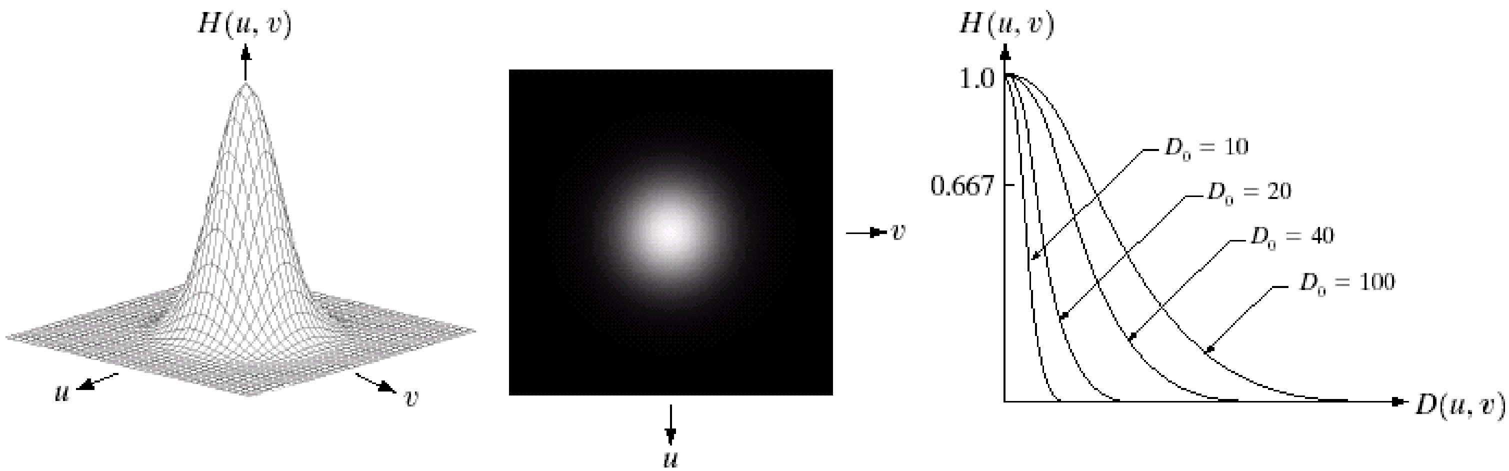
$$H(u,v) = \frac{1}{1 + [D(u,v)/D_0]^{2n}}$$



a b c d

FIGURE 4.16 (a)–(d) Spatial representation of BLPFs of order 1, 2, 5, and 20, and corresponding gray-level profiles through the center of the filters (all filters have a cutoff frequency of 5). Note that ringing increases as a function of filter order.

Gaussian filters



a b c

FIGURE 4.17 (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of D_0.

$$H(u,v) = e^{-D^2(u,v)/2\sigma^2}$$

low-pass filter examples

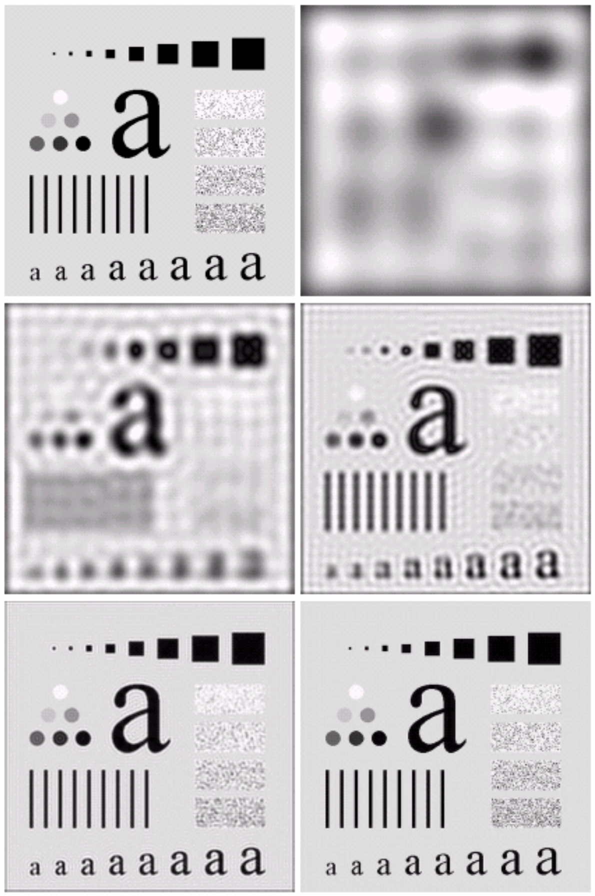


FIGURE 4.12 (a) Original image. (b)–(f) Results of ideal lowpass filtering with cutoff frequencies set at radii values of 5, 15, 30, 80, and 230, as shown in Fig. 4.11(b). The power removed by these filters was 8, 5.4, 3.6, 2, and 0.5% of the total, respectively.

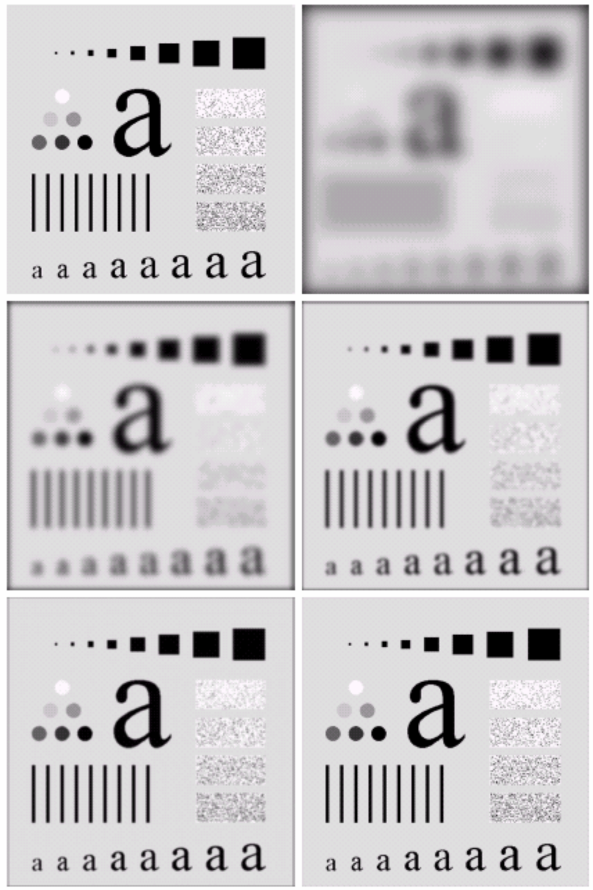


FIGURE 4.18 (a) Original image. (b)–(f) Results of filtering with Gaussian lowpass filters with cutoff frequencies set at radii values of 5, 15, 30, 80, and 230, as shown in Fig. 4.11(b). Compare with Figs. 4.12 and 4.15.

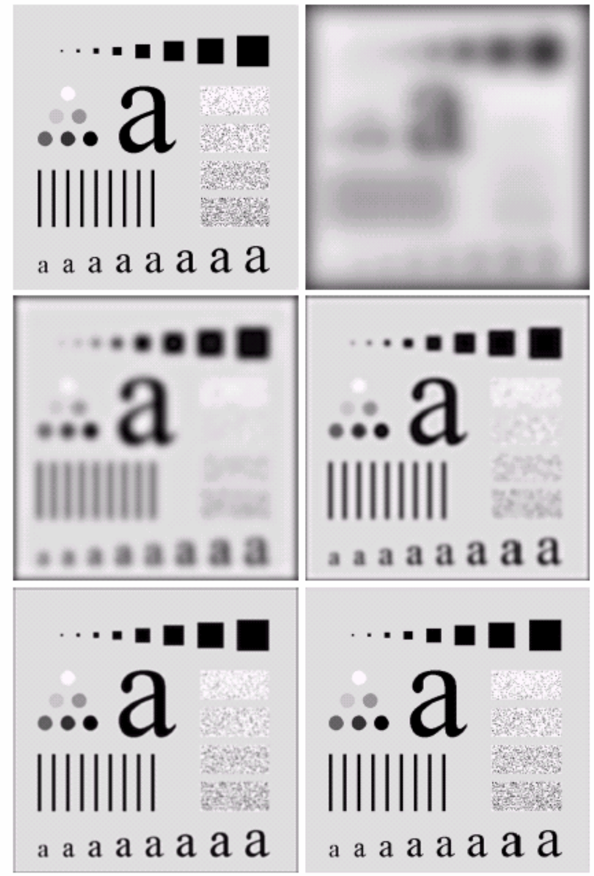


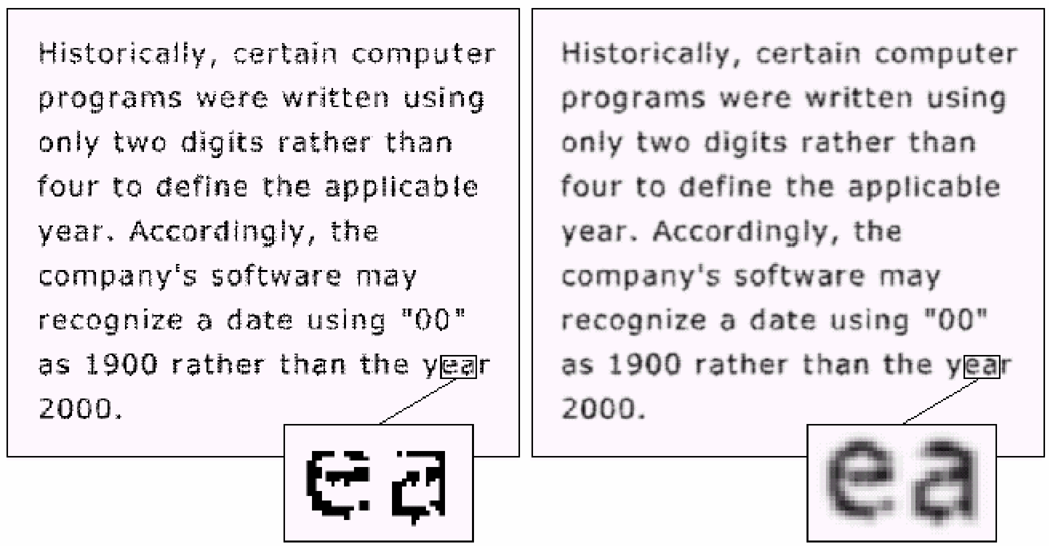
FIGURE 4.15 (a) Original image. (b)–(f) Results of filtering with BLPFs of order 2, with cutoff frequencies at radii of 5, 15, 30, 80, and 230, as shown in Fig. 4.11(b). Compare with Fig. 4.12.

smoothing filter application 1

text enhancement

a b

FIGURE 4.19
(a) Sample text of poor resolution (note broken characters in magnified view). (b) Result of filtering with a GLPF (broken character segments were joined).



smoothing filter application 2

beautify a photo

FIGURE 4.20 (a) Original image (1028×732 pixels). (b) Result of filtering with a GLPF with $D_0 = 100$. (c) Result of filtering with a GLPF with $D_0 = 80$. Note reduction in skin fine lines in the magnified sections of (b) and (c).

high-pass filters

$$H_{HPF}(u,v) = 1 - H_{LPF}(u,v)$$

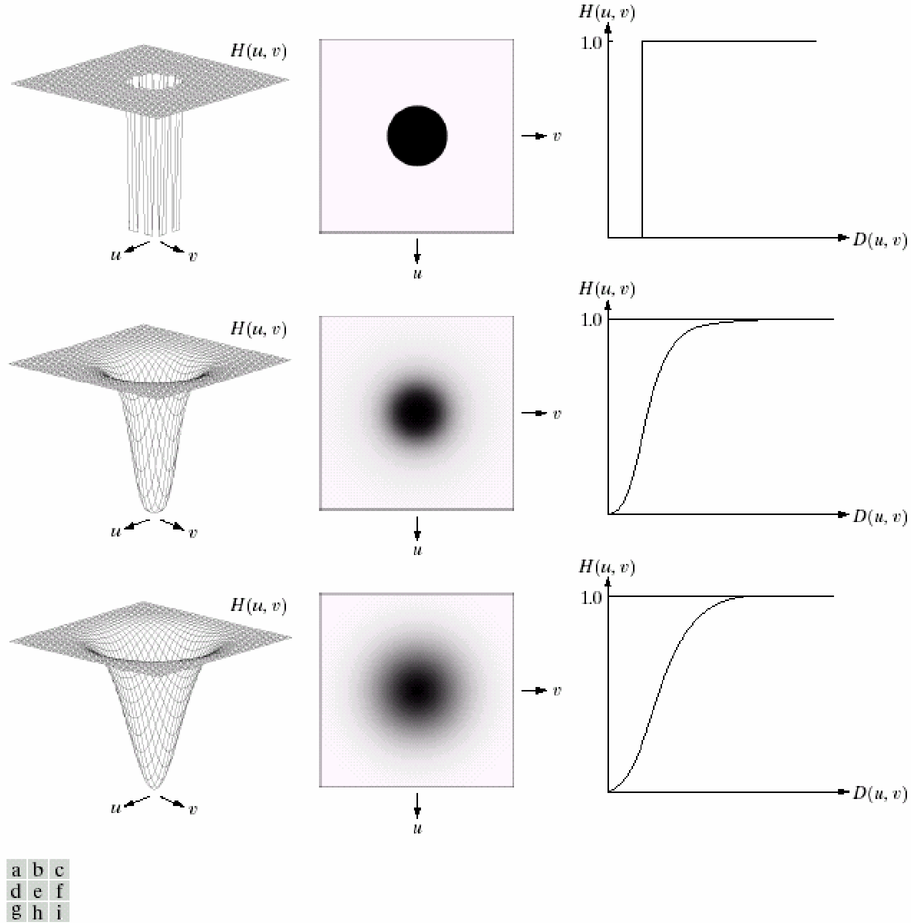
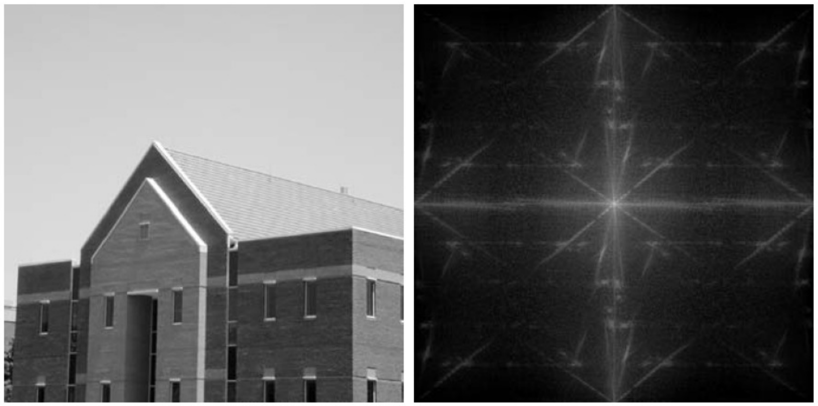
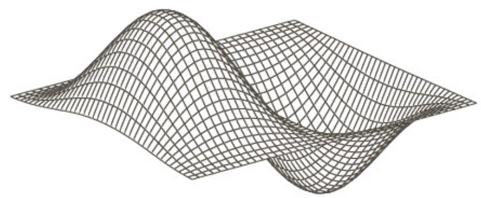


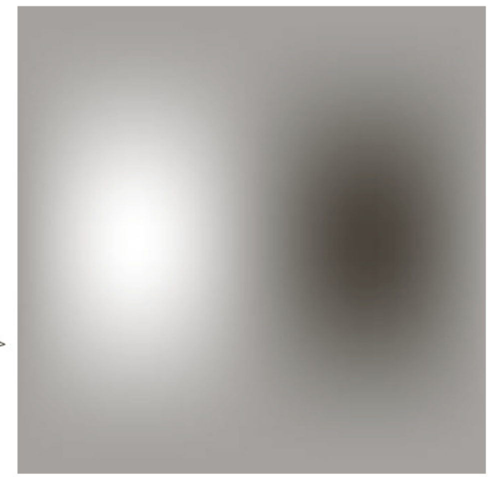
FIGURE 4.22 Top row: Perspective plot, image representation, and cross section of a typical ideal highpass filter. Middle and bottom rows: The same sequence for typical Butterworth and Gaussian highpass filters.

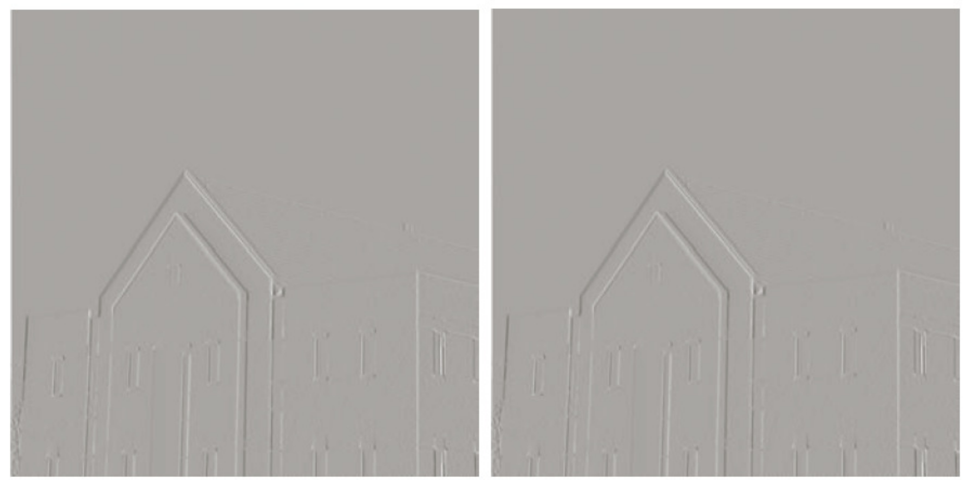
sobel operator in frequency domain



−1	0	1
−2	0	2
−1	0	1







a b
c d

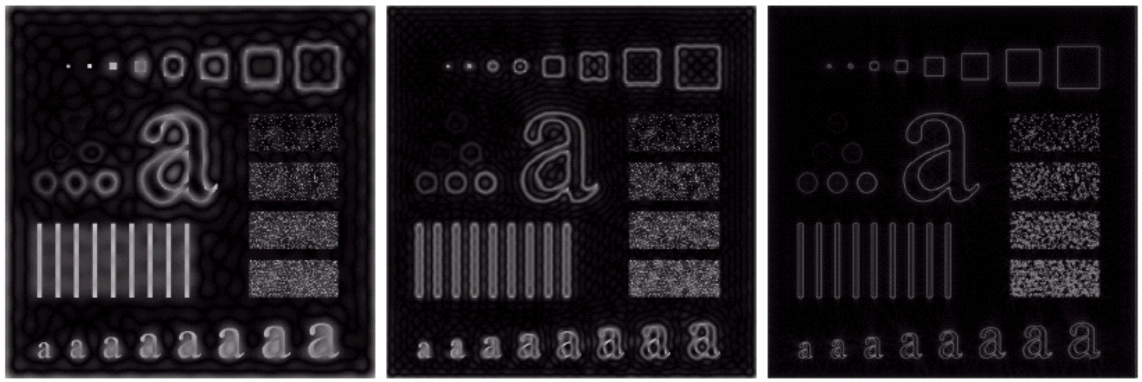
FIGURE 4.39 (a) A spatial mask and perspective plot of its corresponding frequency domain filter. (b) Filter shown as an image. (c) Result of filtering Fig. 4.38(a) in the frequency domain with the filter in (b). (d) Result of filtering the same image with the spatial filter in (a). The results are identical.

Question:

Sobel vs. other high-pass filters?

Spatial vs frequency domain implementation?

high-pass filter examples



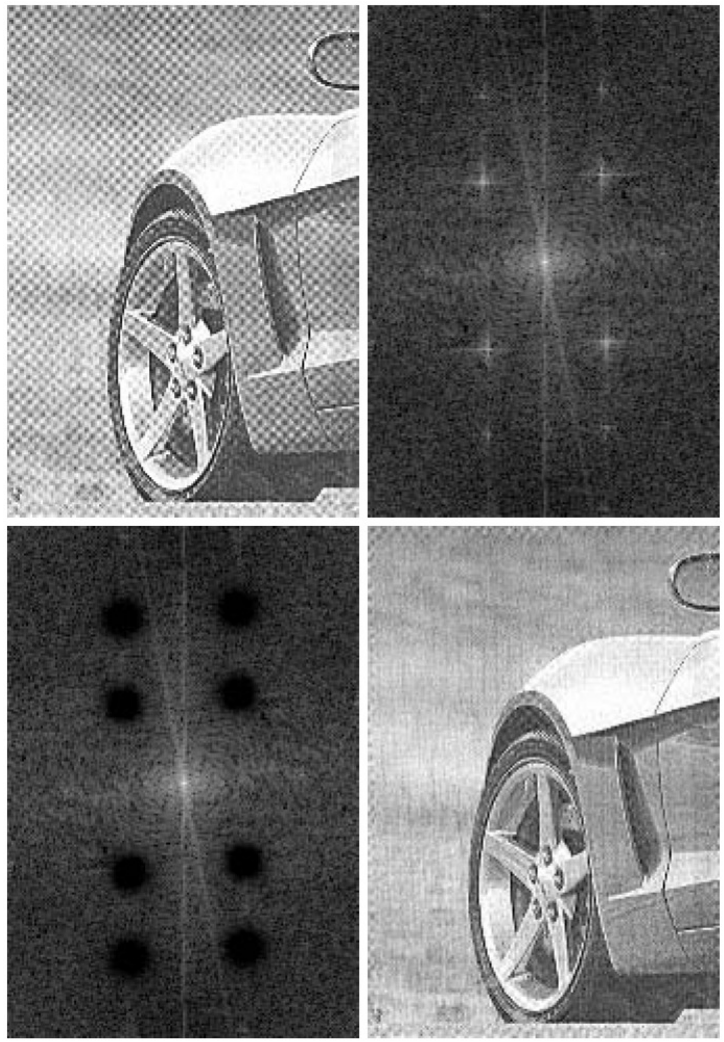
a b c

FIGURE 4.24 Results of ideal highpass filtering the image in Fig. 4.11(a) with $D_0 = 15$, 30, and 80, respectively. Problems with ringing are quite evident in (a) and (b).

a b c

FIGURE 4.26 Results of highpass filtering the image of Fig. 4.11(a) using a GHPF of order 2 with $D_0 = 15$, 30, and 80, respectively. Compare with Figs. 4.24 and 4.25.

band-pass, band-reject filters



a b
c d

FIGURE 4.64 (a) Sampled newspaper image showing a moiré pattern. (b) Spectrum. (c) Butterworth notch reject filter multiplied by the Fourier transform. (d) Filtered image.

outline

- why transform
- 2D Fourier transform
  - a picture book for DFT and 2D-DFT
  - properties
  - implementation
  - applications in enhancement, correlation
- discrete cosine transform (DCT)
  - definition & visualization
  - implementation

Is DFT a Good (enough) Transform?

- Theory

- Implementation

- Application

The Desirables for Image Transforms

	DFT	???
Theory		
▪ Inverse transform available	✓	
▪ Energy conservation (Parsevell)	✓	
▪ Good for compacting energy	?	
▪ Orthonormal, complete basis	✓	
▪ (sort of) shift- and rotation invariant	✓	
Implementation		
▪ Real-valued	x	
▪ Separable	✓	
▪ Fast to compute w. butterfly-like structure	✓	
▪ Same implementation for forward and inverse transform	✓	
Application		
▪ Useful for image enhancement	✓	
▪ Capture perceptually meaningful structures in images	✓	

DFT vs. DCT

$$y = Ax$$

1D-DCT

$$a(0,n) = \sqrt{\frac{1}{N}} \qquad u = 0$$

$$a(u,n) = \sqrt{\frac{2}{N}} cos \frac{\pi(2n+1)u}{2N}$$

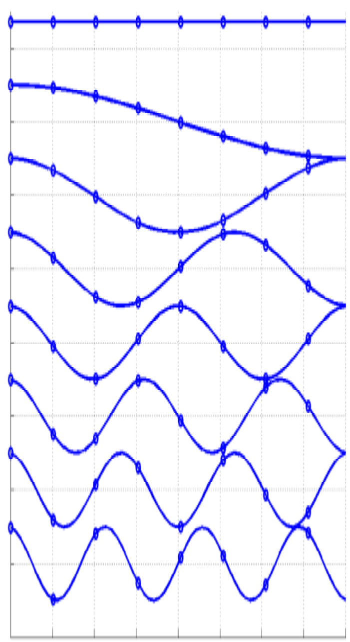
1D-DFT

$$\begin{aligned} a(u,n) &= e^{-j2\pi\frac{un}{N}} \\ &= cos(2\pi\frac{un}{N}) + jsin(2\pi\frac{un}{N}) \end{aligned}$$

$$u = 1, 2, \ldots, N-1$$

a

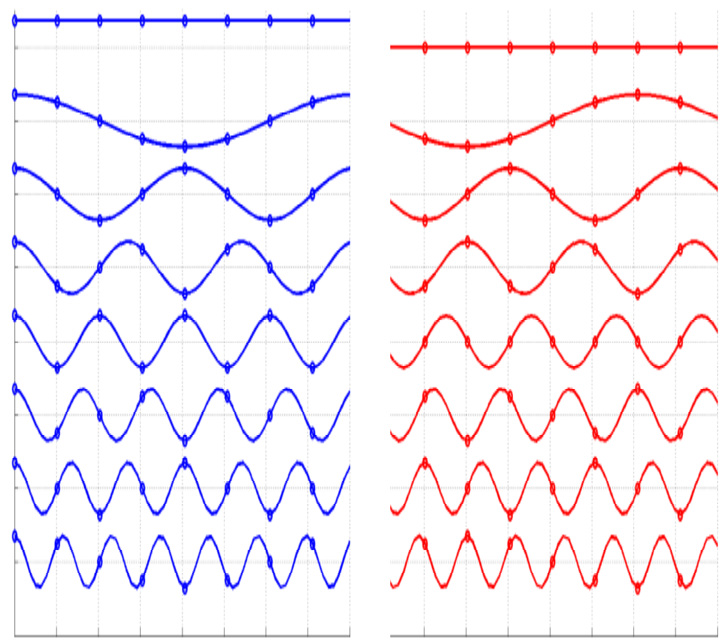
u=0



u=7

real(a) imag(a)

u=0



u=7

n=7

1-D Discrete Cosine Transform (DCT)

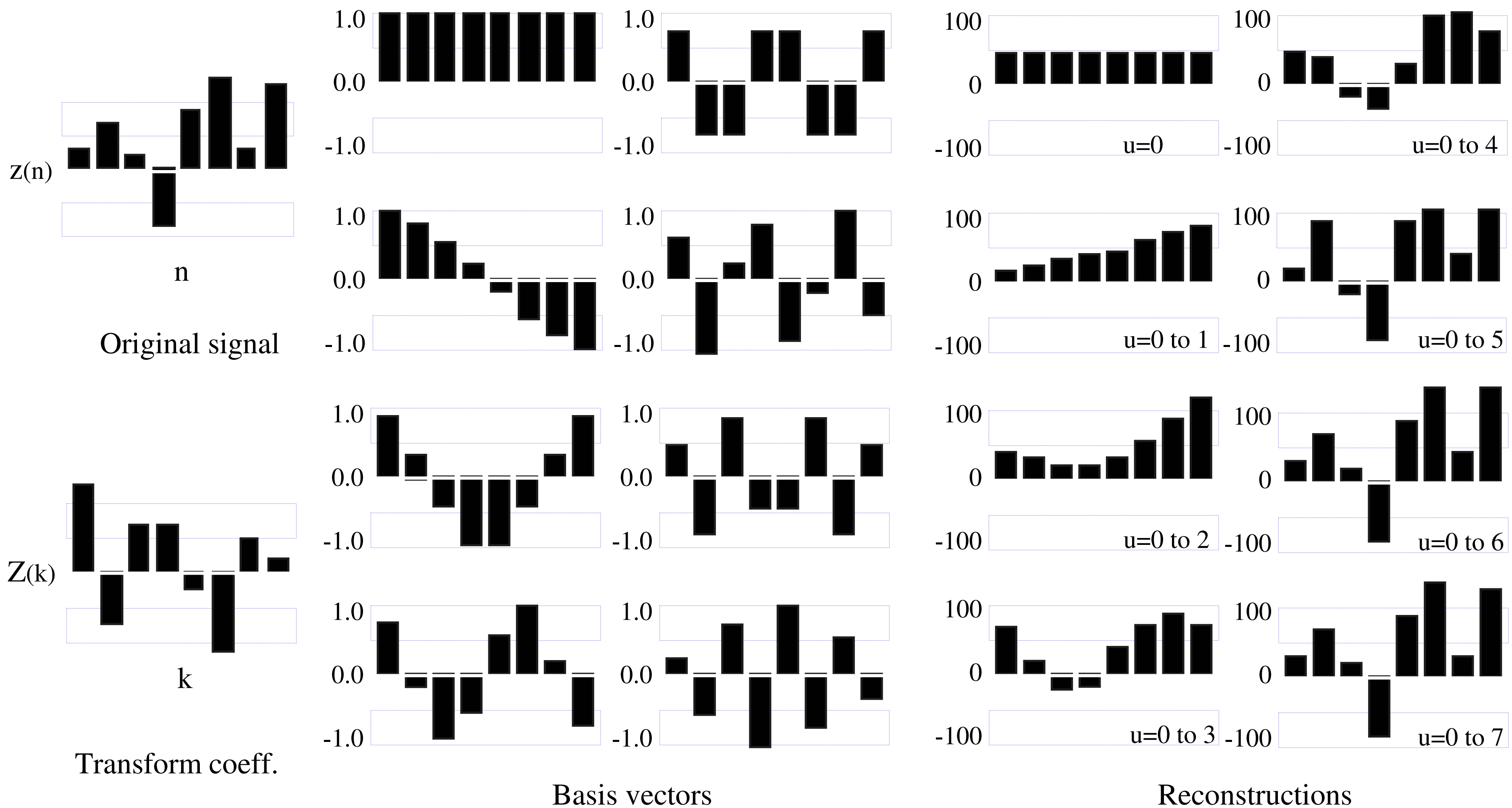
$$\begin{cases} Z(k) = \sum_{n=0}^{N-1} z(n) \cdot \alpha(k) \cos\left[\frac{\pi(2n+1)k}{2N}\right] \\ z(n) = \sum_{k=0}^{N-1} Z(k) \cdot \alpha(k) \cos\left[\frac{\pi(2n+1)k}{2N}\right] \end{cases}$$

$$\alpha(0) = \frac{1}{\sqrt{N}}, \alpha(k) = \sqrt{\frac{2}{N}}$$

- Transform matrix *A*
  - a(k,n) = α(0) for k=0
  - a(k,n) = α(k) cos[π(2n+1)/2N] for k>0

- A is real and orthogonal
  - rows of A form orthonormal basis
  - A is not symmetric!
  - DCT is <u>not</u> the real part of unitary DFT!

1-D DCT



z(n)
n
Original signal
Z(k)
k
Transform coeff.
1.0
0.0
-1.0
Basis vectors
100
0
-100
u=0
u=0 to 1
u=0 to 2
u=0 to 3
u=0 to 4
u=0 to 5
u=0 to 6
u=0 to 7
Reconstructions

DFT and DCT in Matrix Notations

Matrix notation for 1D transform

$$y = Ax, \;\; x = A^{-1}y$$

1D-DCT

$$a(0,n) = \sqrt{\frac{1}{N}} \qquad u = 0$$

$$a(u,n) = \sqrt{\frac{2}{N}} cos \frac{\pi(2n+1)u}{2N}$$

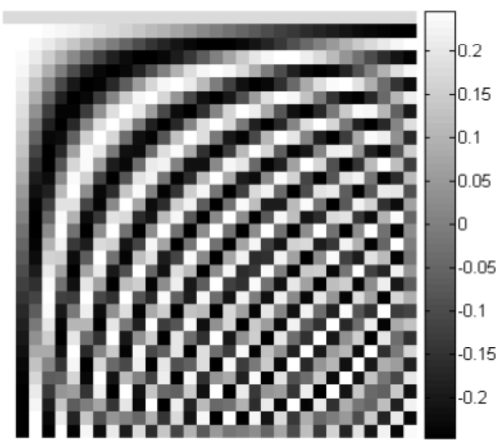
$$u = 1, 2, \ldots, N-1$$

1D-DFT

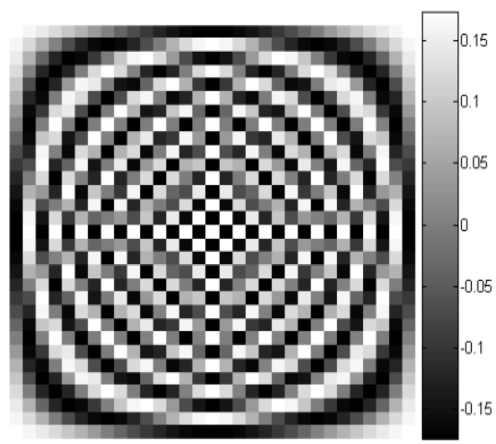
$$\begin{aligned} a(u,n) &= e^{-j2\pi\frac{un}{N}} \\ &= cos(2\pi\frac{un}{N}) - jsin(2\pi\frac{un}{N}) \end{aligned}$$

N=32

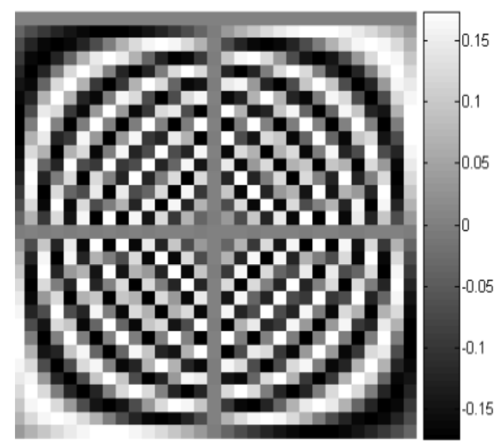
A



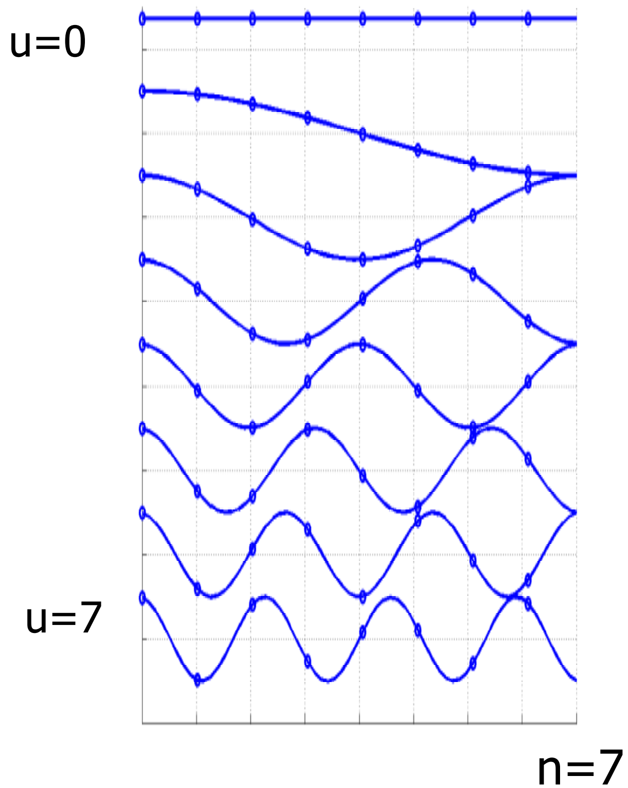
real(A)

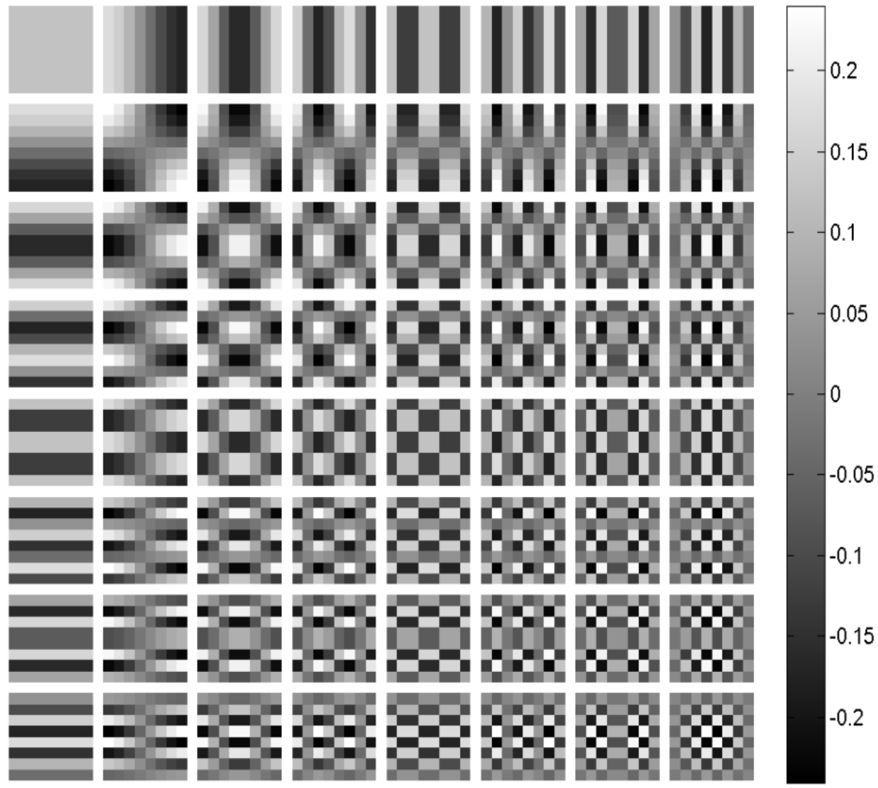


imag(A)



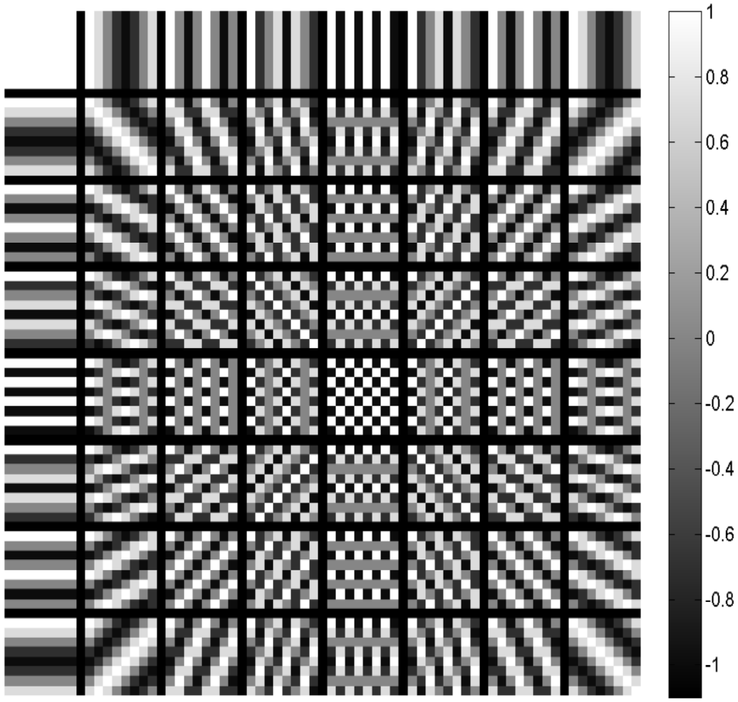
From 1D-DCT to 2D-DCT

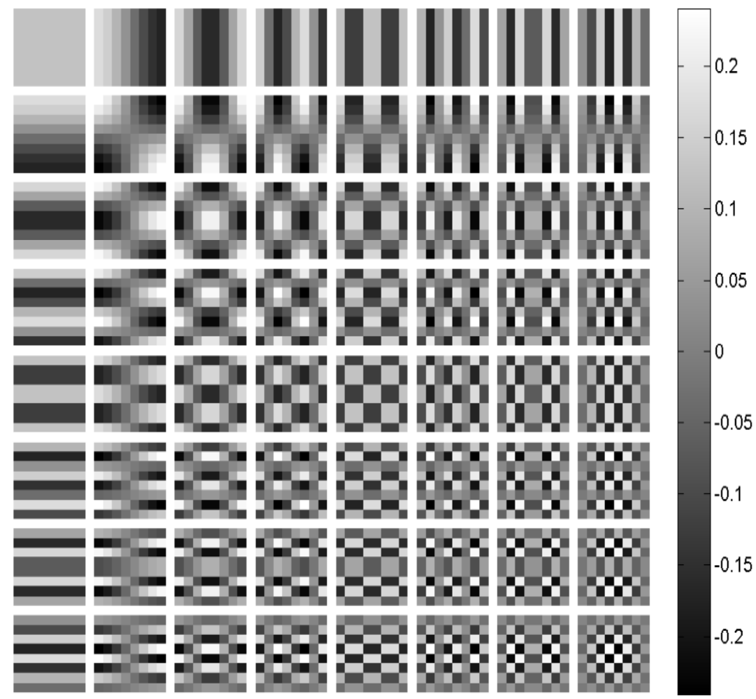




- Rows of A form a set of orthonormal basis
- A is not symmetric!
- DCT is <u>not</u> the real part of unitary DFT!

basis images: DFT (real) vs DCT





Periodicity Implied by DFT and DCT

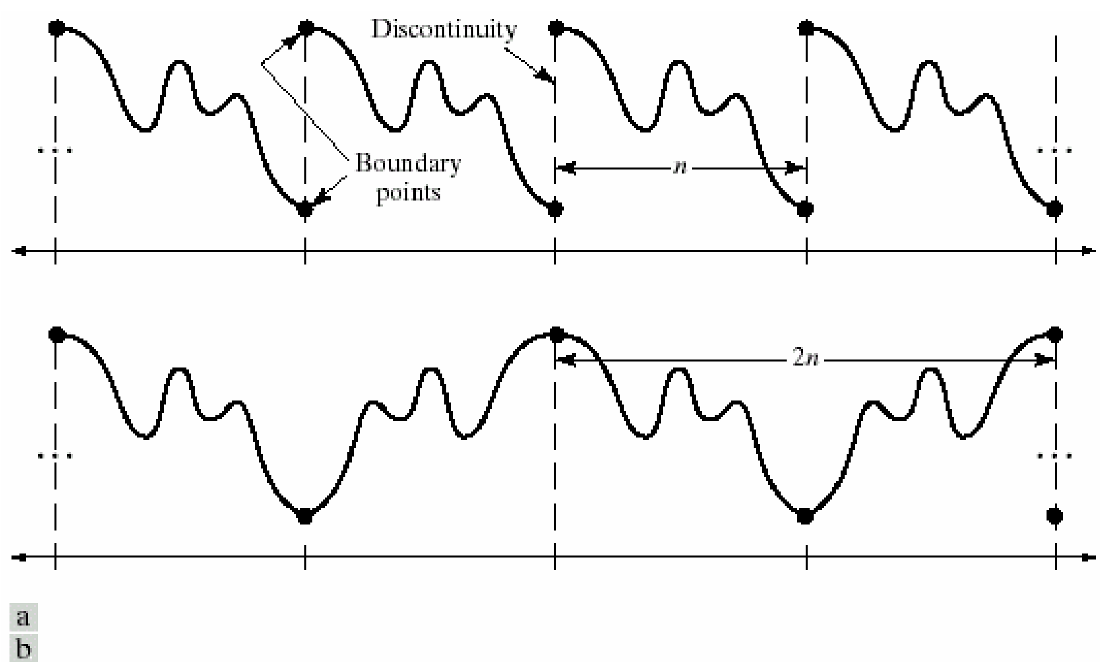
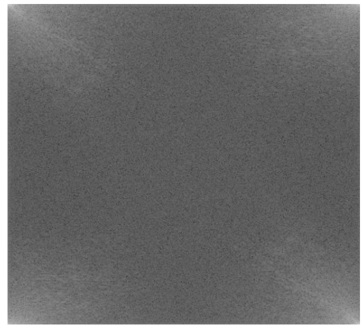


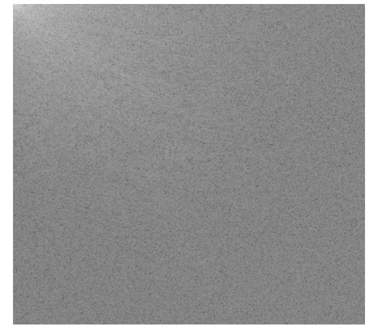
FIGURE 8.32 The periodicity implicit in the 1-D (a) DFT and (b) DCT.

DFT and DCT on Lena

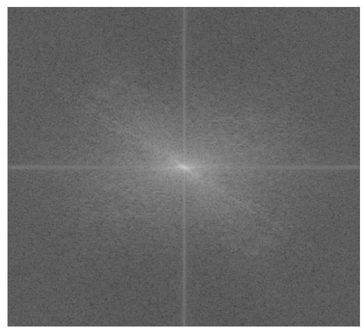
DFT2

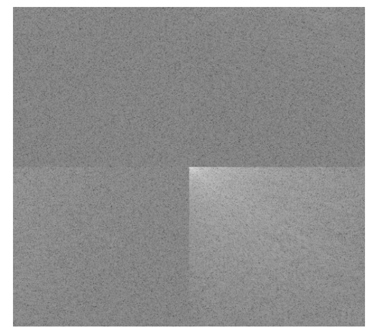
DCT2





Shift low-freq
to the center





Assume periodic and zero-padded …

Assume reflection …

Using FFT to implement fast DCT

- Reorder odd and even elements

$$\begin{cases} \tilde{z}(n) = z(2n) \\ \tilde{z}(N-n-1) = z(2n+1) \end{cases} \quad \text{for } 0 \le n \le \frac{N}{2} - 1$$

- Split the DCT sum into odd and even terms

$$\begin{aligned}
Z(k) &= \alpha(k)\left\{\sum_{n=0}^{N/2-1} z(2n)\cdot\cos\left[\frac{\pi(4n+1)k}{2N}\right] + \sum_{n=0}^{N/2-1} z(2n+1)\cdot\cos\left[\frac{\pi(4n+3)k}{2N}\right]\right\} \\
&= \alpha(k)\left\{\sum_{n=0}^{N/2-1} \tilde{z}(n)\cdot\cos\left[\frac{\pi(4n+1)k}{2N}\right] + \sum_{n=0}^{N/2-1} \tilde{z}(N-n-1)\cdot\cos\left[\frac{\pi(4n+3)k}{2N}\right]\right\} \\
&= \alpha(k)\left\{\sum_{n=0}^{N/2-1} \tilde{z}(n)\cdot\cos\left[\frac{\pi(4n+1)k}{2N}\right] + \sum_{n'=N/2}^{N-1} \tilde{z}(n')\cdot\cos\left[\frac{\pi(4N-4n'-1)k}{2N}\right]\right\} \\
&= \alpha(k)\sum_{n=0}^{N-1} \tilde{z}(n)\cdot\cos\left[\frac{\pi(4n+1)k}{2N}\right] = \mathrm{Re}\left[\alpha(k)e^{-j\pi k/2N}\sum_{n=0}^{N-1}\tilde{z}(n)\cdot e^{-j2\pi nk/N}\right] \\
&= \mathrm{Re}\left[\alpha(k)e^{-j\pi k/2N}\, DFT\ \{\tilde{z}(n)\}_N\right]
\end{aligned}$$

The Desirables for Image Transforms

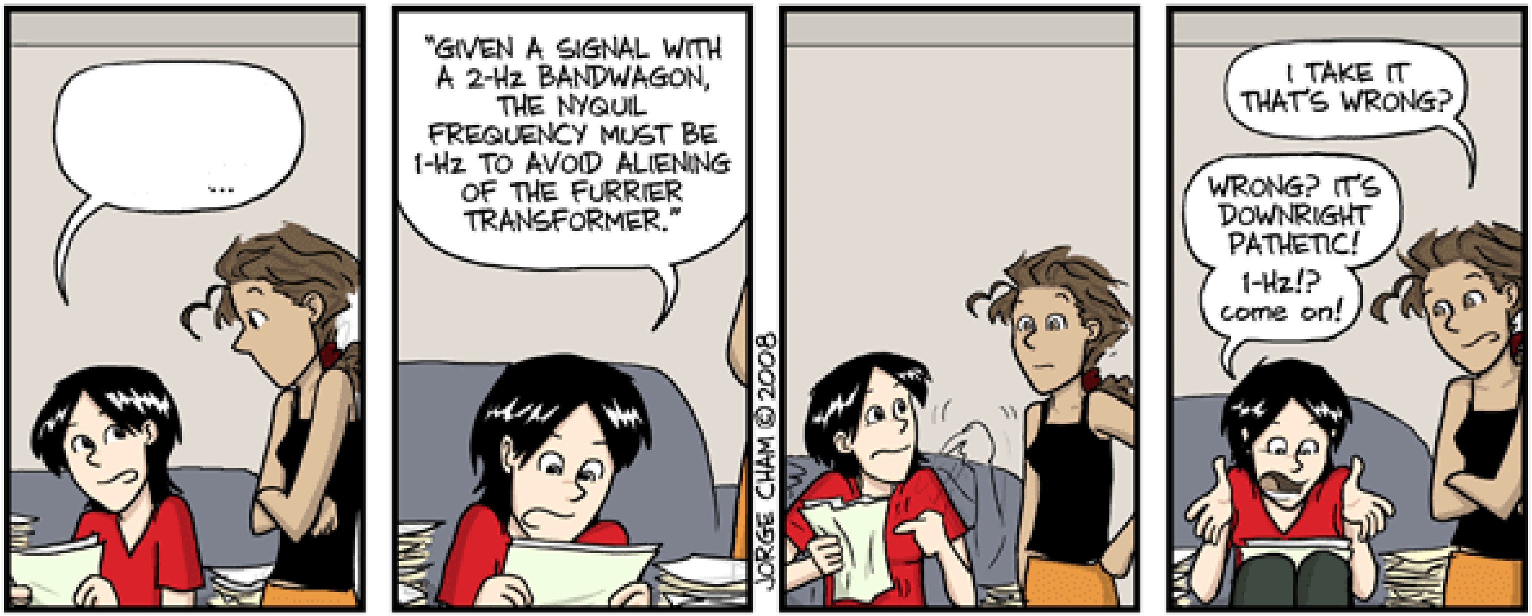
	DFT	DCT	???
Theory			
Inverse transform available	✓	✓	
Energy conservation (Parsevell)	✓	✓	
Good for compacting energy	?	?	
Orthonormal, complete basis	✓	✓	
(sort of) shift- and rotation invariant	✓	✓	
Implementation			
Real-valued	x	✓	
Separable	✓	✓	
Fast to compute w. butterfly-like structure	✓	✓	
Same implementation for forward and inverse transform	✓	✓	
Application			
Useful for image enhancement	✓		
Capture perceptually meaningful structures in images	✓		

Summary of Lecture 5

- Why we need image transform
- DFT revisited
  - Definitions, properties, observations, implementations, applications
- What do we need for a transform
- DCT

- Coming in Lecture 6:
  - Unitary transforms, KL transform, DCT
  - examples and optimality for DCT and KLT, other transform flavors, Wavelets, Applications
- Readings: G&W chapter 4, chapter 5 of Jain has been posted on Courseworks

- “Transforms” that do not belong to lectures 5-6:
  Rodon transform, Hough transform, …



WWW.PHDCOMICS.COM