COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Image Transforms and Image Enhancement in Frequency Domain

EE4830 Lecture 5
Feb 19th, 2007

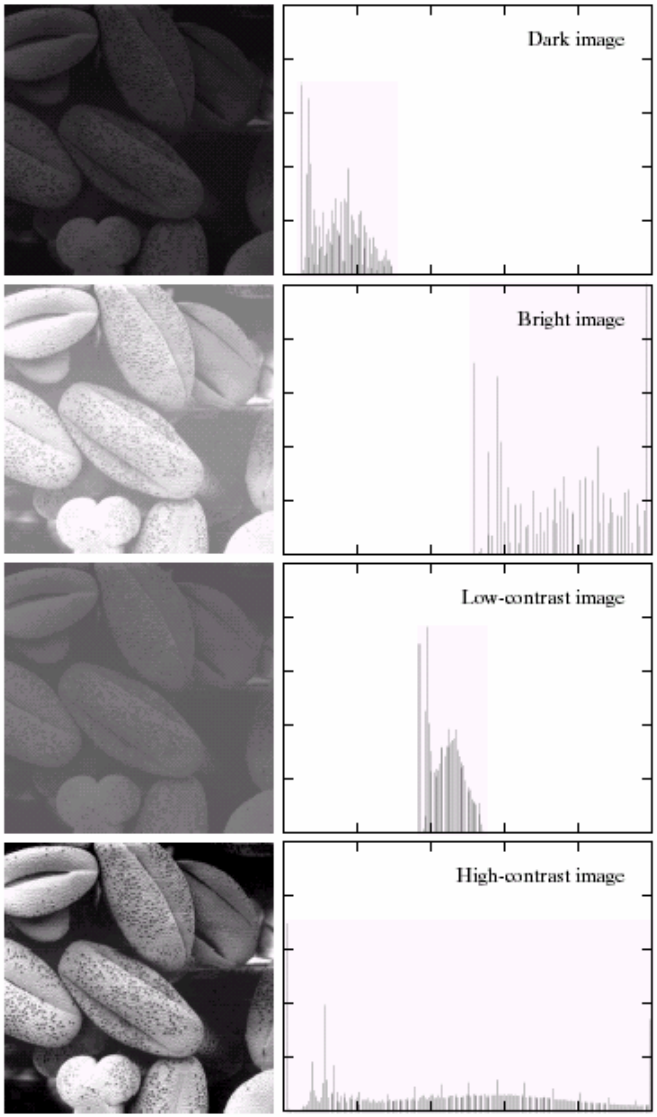
Lexing Xie

With thanks to G&W website, M. Thomas, M. Wu, W. Trappe, K. Lam, S. Sastry, Matlab demo, etc.

- Announcements
  - PS#2, EXP#1 due today
  - PS#3, EXP#2 will be posted in the next two days
  - Mid-term on March 5

- Recap of previous lecture
  - Image enhancements in spatial domain
  - Tri-color representation
- This lecture …
  - Image transforms and their uses
  - Readings for *today and next week*:
    G&W Chap 4, Jain 5.1-5.11

Review 1: Image Enhancement in Spatial Domain

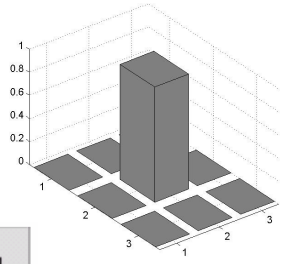


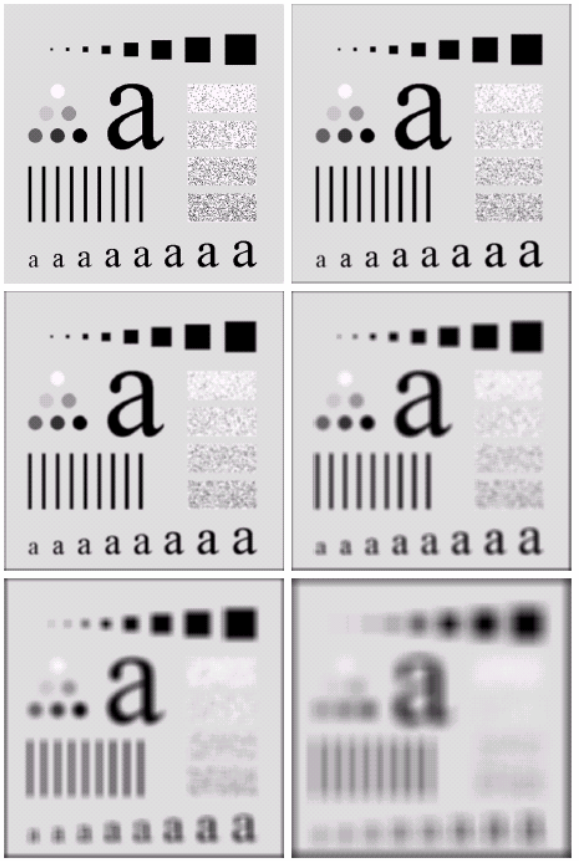
$\frac{1}{9}\times$

1	1	1
1	1	1
1	1	1

$\frac{1}{16}\times$

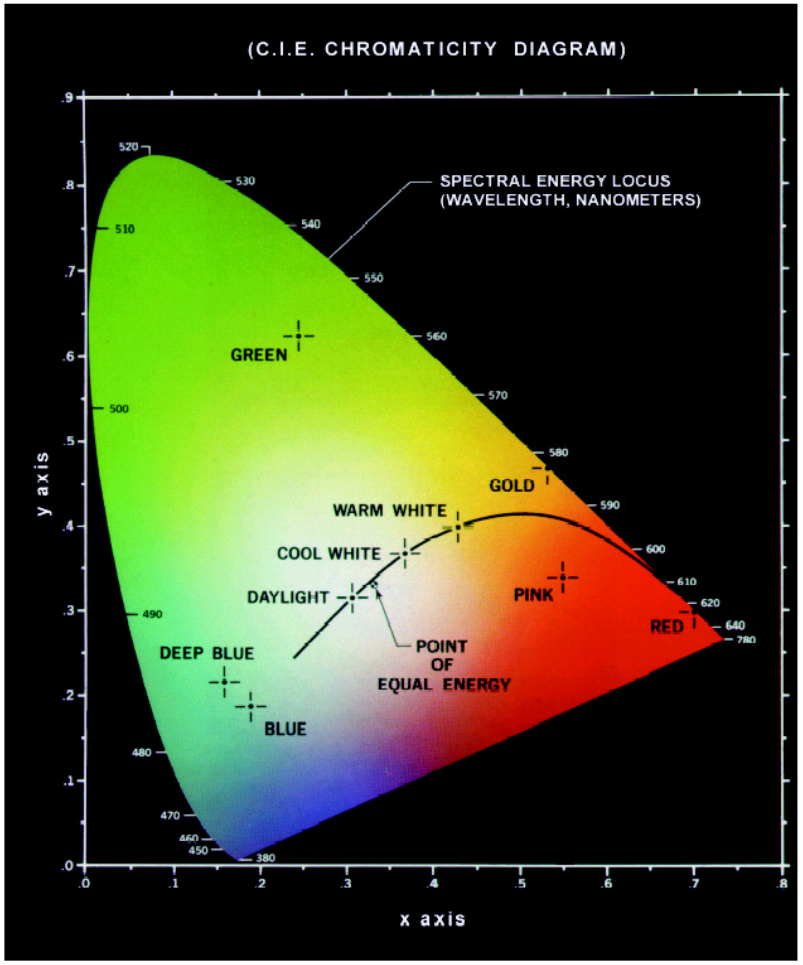
1	2	1
2	4	2
1	2	1

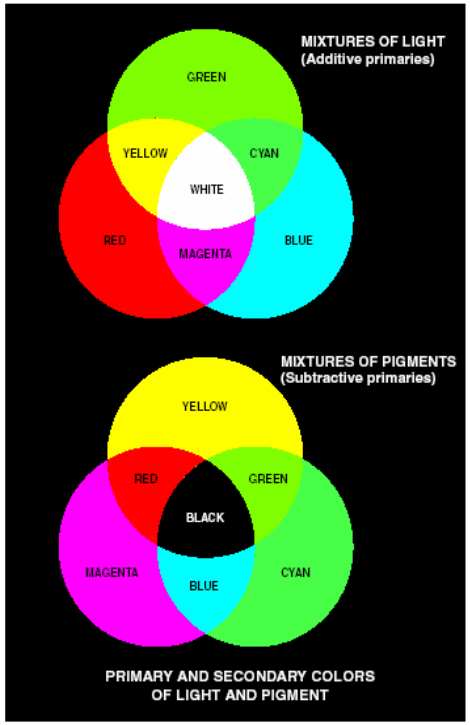




Review 2: Tri-color Representation

FIGURE 6.5 Chromaticity diagram. (Courtesy of the General Electric Co., Lamp Business Division.)





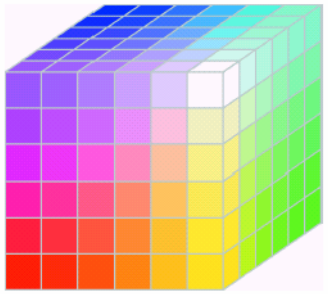


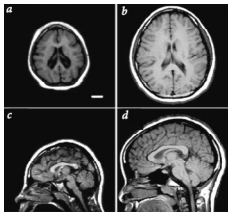
FIGURE 6.11 The RGB safe-color cube.

Lecture Outline

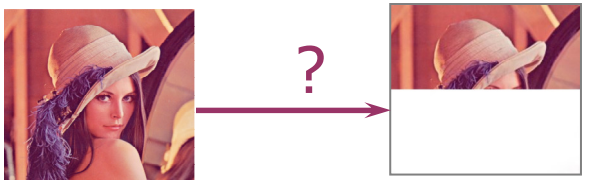
- Review of Previous lectures
- Image Transform
  - Why transform
  - 2D Fourier Transform
    - Definition, Properties, Implementation
    - DFT applications
  - Transform in other flavors
    - Unitary transforms
    - DCT, KLT

Why Do Transform?

- Better image processing
  - Take into account long-range correlations in space
  - Conceptual insights in spatial-frequency information (smooth, moderate change, fast change, etc.)
- 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



DFT Recap

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

- 1D – FT

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

- 1D – DFT of length N

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

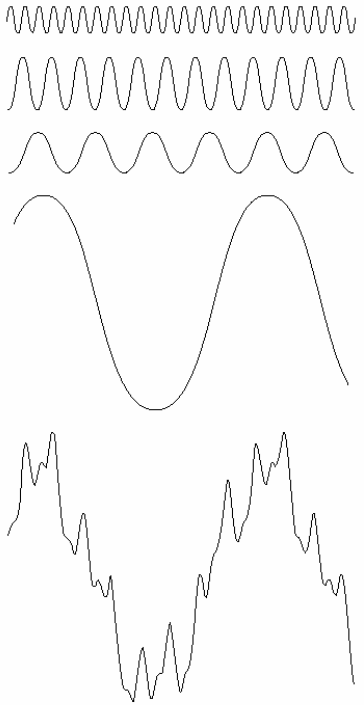
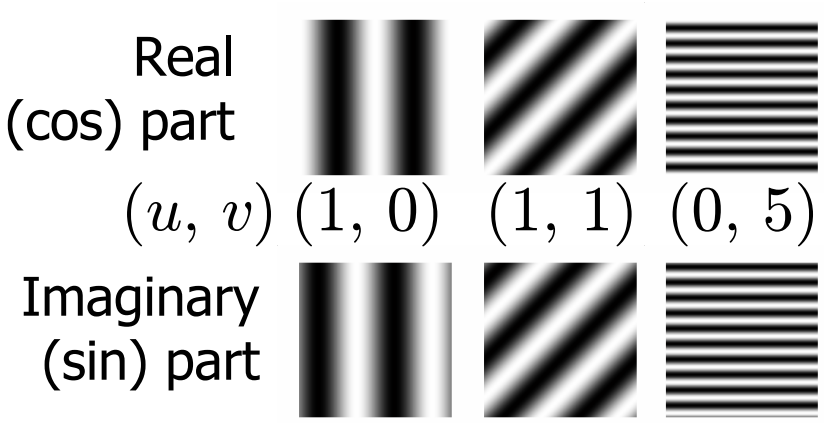


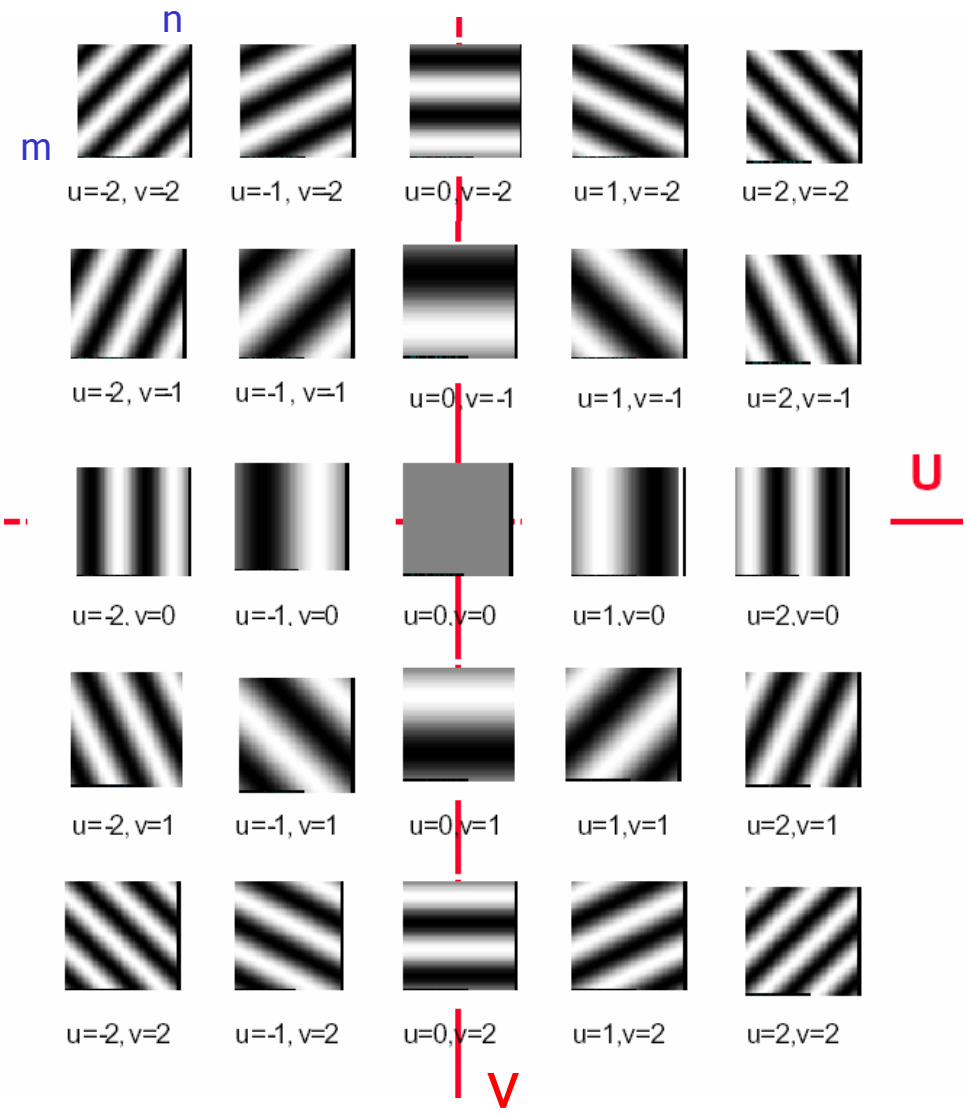
FIGURE 4.1 The function at the bottom is the sum of the four functions above it. Fourier's idea in 1807 that periodic functions could be represented as a weighted sum of sines and cosines was met with skepticism.

Fourier Basis

1D $e^{-j2\pi\frac{un}{N}}$

2D $e^{-j2\pi(\frac{um}{N}+\frac{vn}{N})}$





Parts of slides 10-12 adpated from Mani Thomas, Computer Vision lecture notes, CISC 489/689, UDel

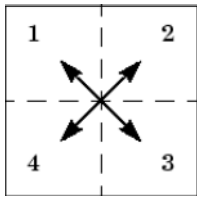
Computing 2D-DFT

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

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

- 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
```



Fourier transform in Matlab

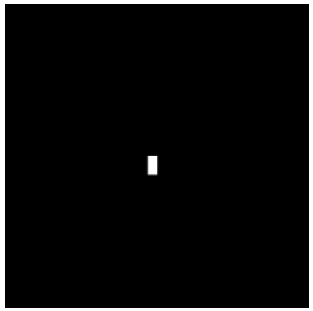
- Output of the Fourier transform is a complex number
  - Decompose the complex number as the magnitude and phase components
- In Matlab: `u = real(z), v = imag(z), r = abs(z),` and `theta = angle(z)`

Some useful FT pairs:

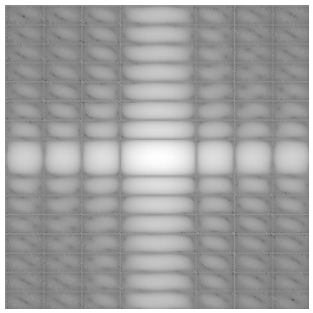
Impulse	$\delta(x, y) \Leftrightarrow 1$
Gaussian	$A\sqrt{2\pi}\sigma e^{-2\pi^2\sigma^2(x^2+y^2)} \Leftrightarrow Ae^{-(u^2+v^2)/2\sigma^2}$
Rectangle	$\text{rect}[a, b] \Leftrightarrow ab\frac{\sin(\pi ua)}{(\pi ua)}\frac{\sin(\pi vb)}{(\pi vb)}e^{-j\pi(ua+vb)}$
Cosine	$\cos(2\pi u_0 x + 2\pi v_0 y)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

† Assumes that functions have been extended by zero padding.

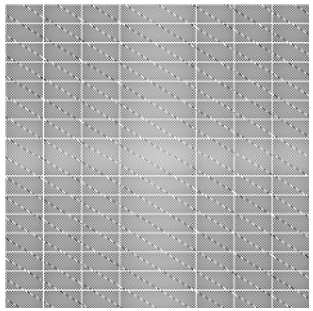
Explaining 2D-DFT

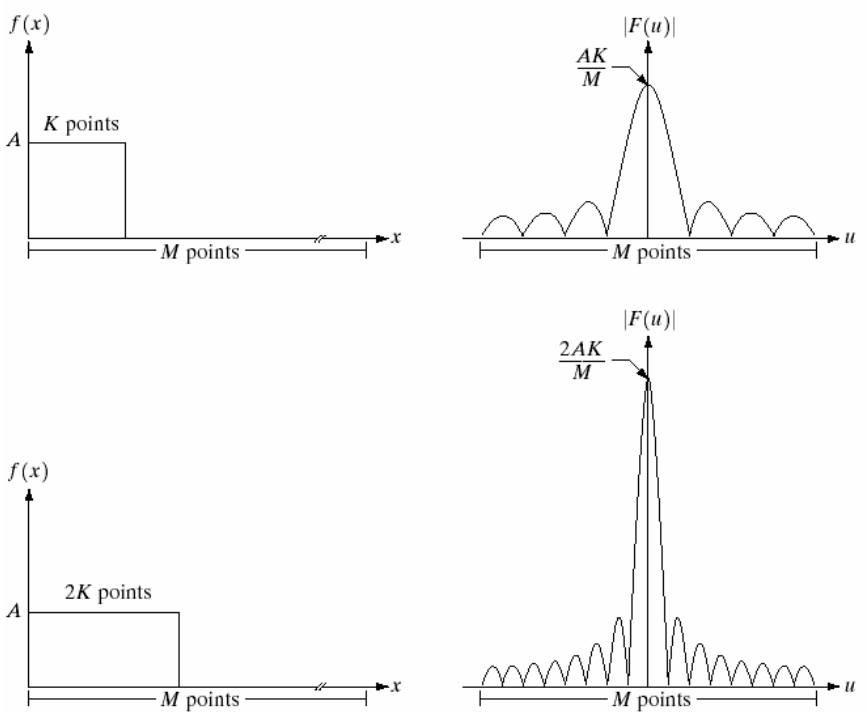


fft2



ifft2

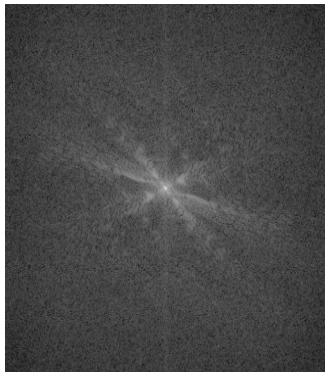


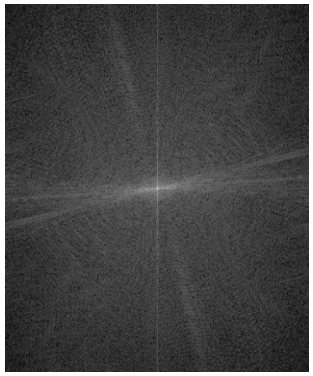


a b
c d

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.

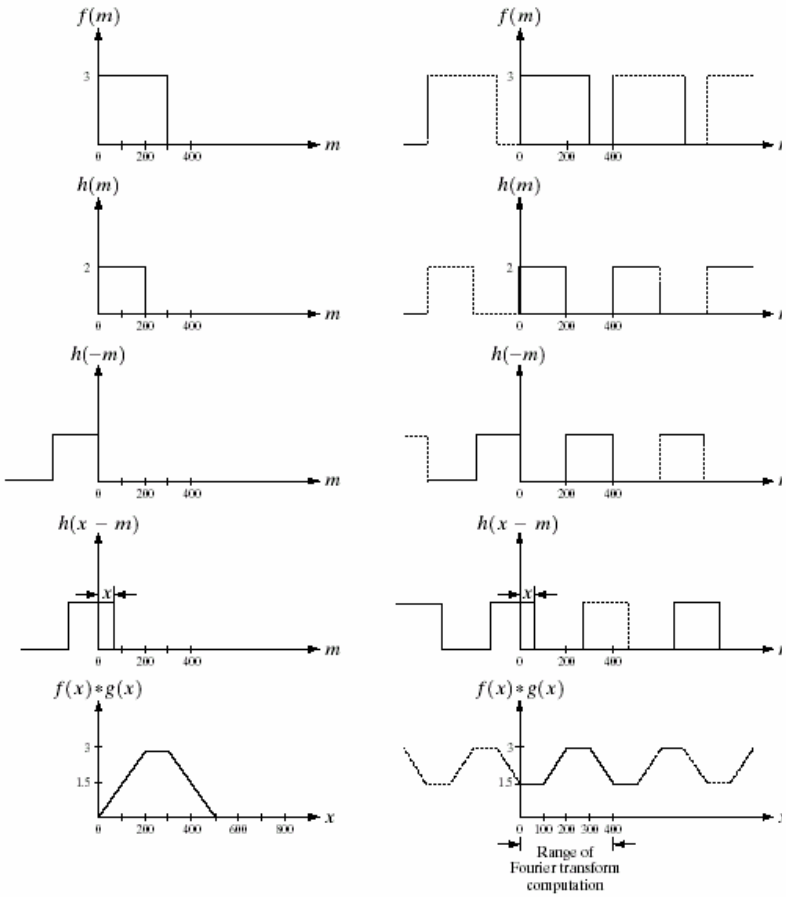
Explaining 2D-DFT (2)





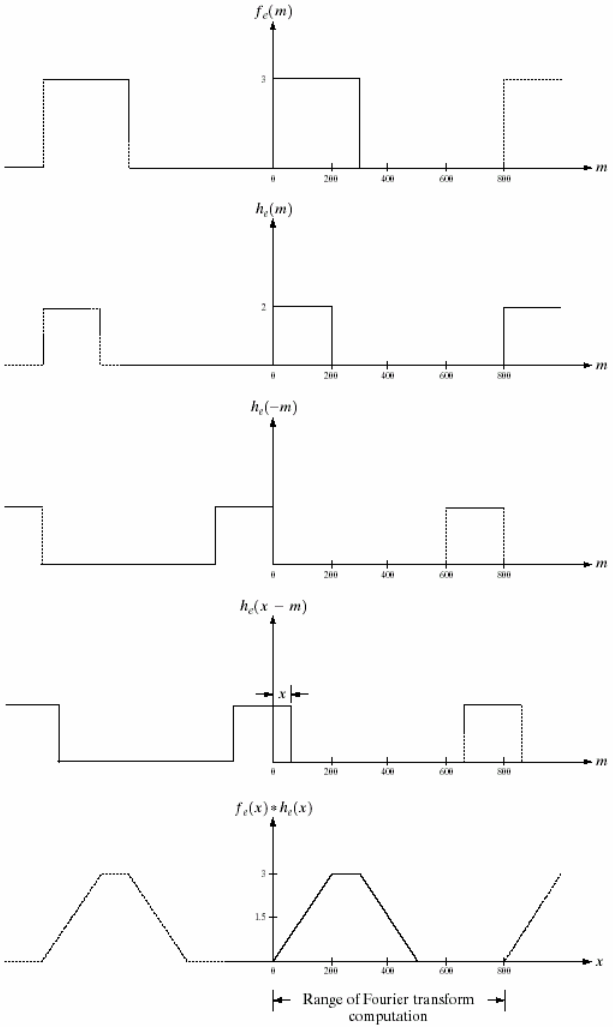
demo

Circular convolution and Zero Padding



a f
b g
c h
d i
e j

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.



a
b
c
d
e

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

Zero Padded Filter and Response

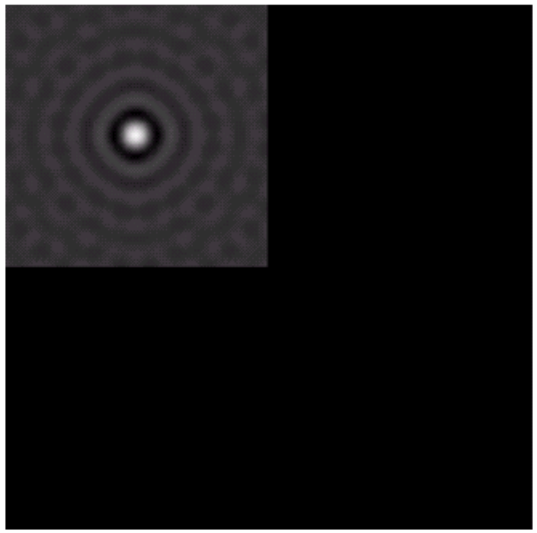
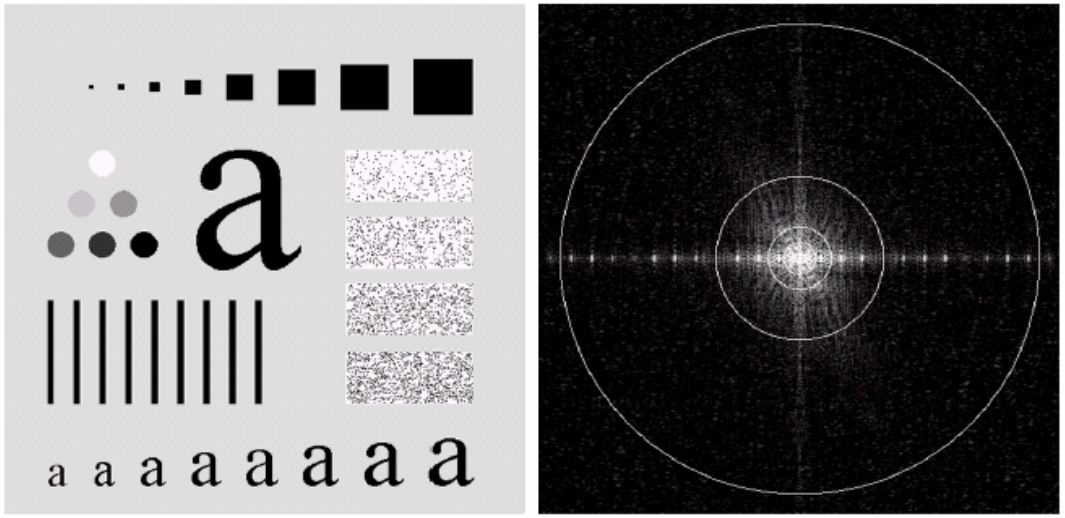


FIGURE 4.39 Padded lowpass filter is the spatial domain (only the real part is shown).

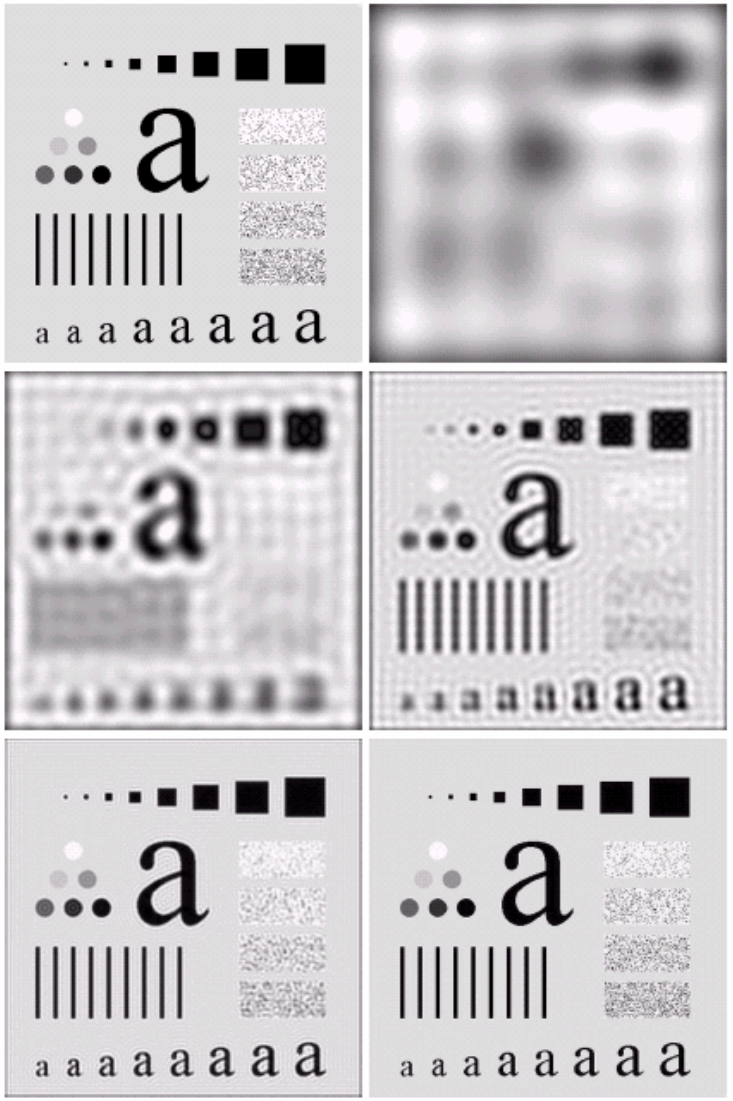
FIGURE 4.40 Result of filtering with padding. The image is usually cropped to its original size since there is little valuable information past the image boundaries.

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

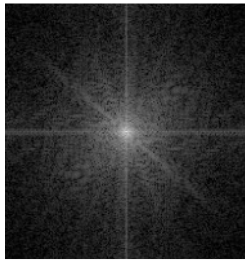
A = "Aron"

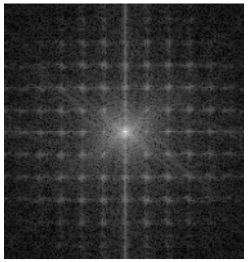
FA = fft2(A)

P = "Phyllis"

FP = fft2(P)

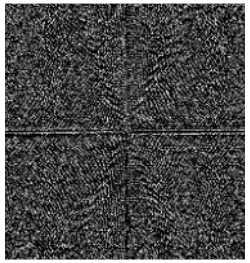
log(abs(FA))

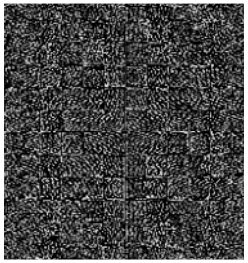




log(abs(FP))

angle(FA)





angle(FP)

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

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

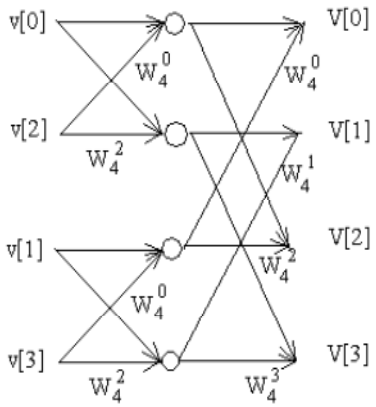
Adpated from http://robotics.eecs.berkeley.edu/~sastry/ee20/vision2/vision2.html

Fast Implementation of 2-D DFT

- 2 Dimensional DFT is separable

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

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

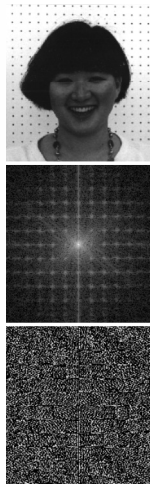
$$f^*(m,n) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F^*(u,v) e^{-j2\pi\frac{um}{M}} e^{\frac{vn}{N}}$$

$$= (MN) \cdot DFT2[F^*(u,v)]$$

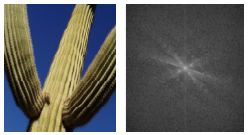
Properties of 2D-DFT

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	$f(x, y) e^{j2\pi(u_0 x/M + v_0 y/N)} \Leftrightarrow F(u - u_0, v - v_0)$ $f(x - x_0, y - y_0) \Leftrightarrow F(u, v) e^{-j2\pi(u x_0/M + v y_0/N)}$ When $x_0 = u_0 = M/2$ and $y_0 = v_0 = N/2$, then $f(x, y)(-1)^{x+y} \Leftrightarrow F(u - M/2, v - N/2)$ $f(x - M/2, y - N/2) \Leftrightarrow F(u, v)(-1)^{u+v}$



Conjugate symmetry	$F(u, v) = F^*(-u, -v)$ $\lvert F(u, v)\rvert = \lvert F(-u, -v)\rvert$
Differentiation	$\dfrac{\partial^n f(x, y)}{\partial x^n} \Leftrightarrow (ju)^n F(u, v)$ $(-jx)^n f(x, y) \Leftrightarrow \dfrac{\partial^n F(u, v)}{\partial u^n}$
Laplacian	$\nabla^2 f(x, y) \Leftrightarrow -(u^2 + v^2)F(u, v)$
Distributivity	$\Im[f_1(x, y) + f_2(x, y)] = \Im[f_1(x, y)] + \Im[f_2(x, y)]$ $\Im[f_1(x, y) \cdot f_2(x, y)] \neq \Im[f_1(x, y)] \cdot \Im[f_2(x, y)]$
Scaling	$af(x, y) \Leftrightarrow aF(u, v), f(ax, by) \Leftrightarrow \dfrac{1}{\lvert ab\rvert} F(u/a, v/b)$
Rotation	$x = r\cos\theta \quad y = r\sin\theta \quad u = \omega\cos\varphi \quad v = \omega\sin\varphi$ $f(r, \theta + \theta_0) \Leftrightarrow F(\omega, \varphi + \theta_0)$
Periodicity	$F(u, v) = F(u + M, v) = F(u, v + N) = F(u + M, v + N)$ $f(x, y) = f(x + M, y) = f(x, y + N) = f(x + M, y + N)$
Separability	See Eqs. (4.6-14) and (4.6-15). Separability implies that we can compute the 2-D transform of an image by first computing 1-D transforms along each row of the image, and then computing a 1-D transform along each column of this intermediate result. The reverse, columns and then rows, yields the same result.

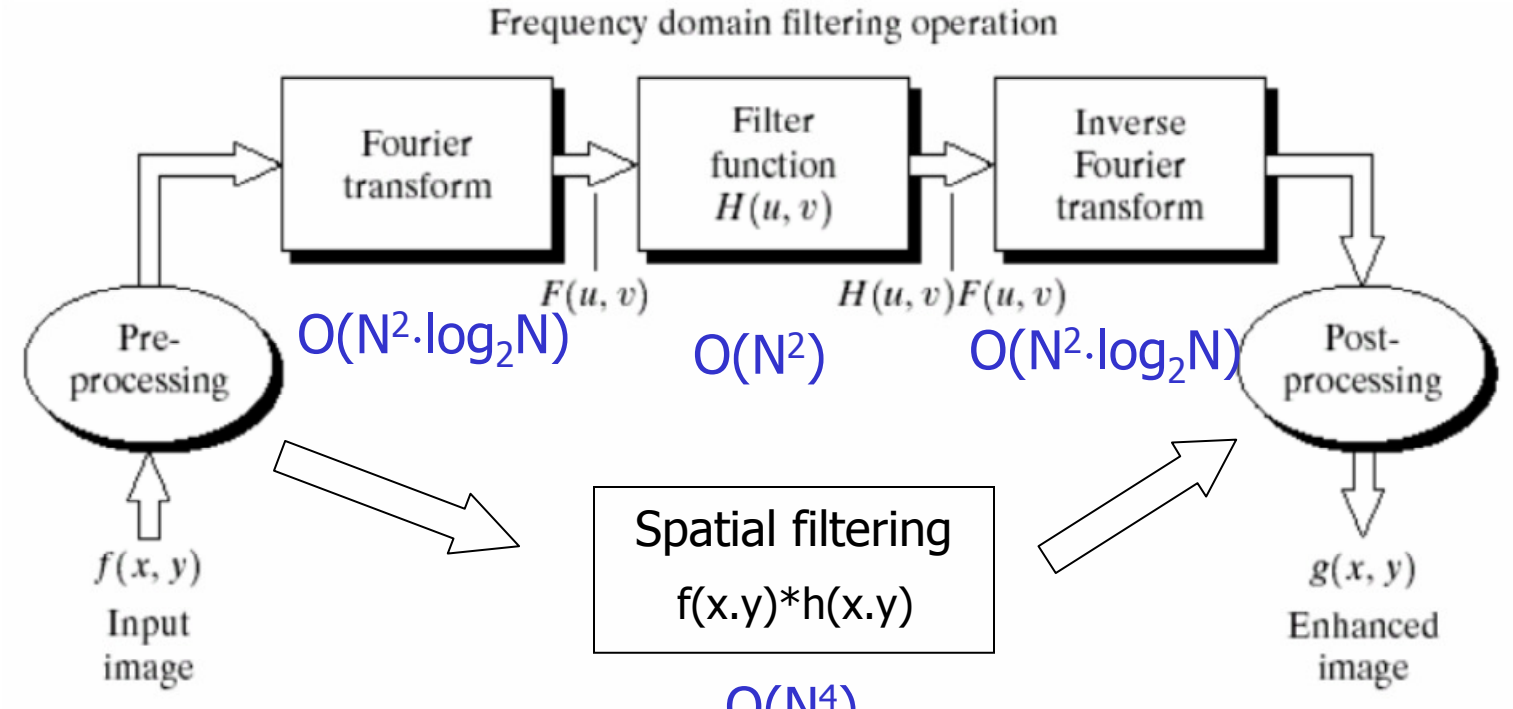


Property	Expression(s)
Computation of the inverse Fourier transform using a forward transform algorithm	$\frac{1}{MN} f^*(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F^*(u, v) e^{-j2\pi(ux/M + vy/N)}$ This equation indicates that inputting the function $F^*(u, v)$ into an algorithm designed 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†	$f(x, y) \circ h(x, y) \Leftrightarrow F^*(u, v) H(u, v)$; $f^*(x, y) h(x, y) \Leftrightarrow F(u, v) \circ H(u, v)$

Lecture Outline

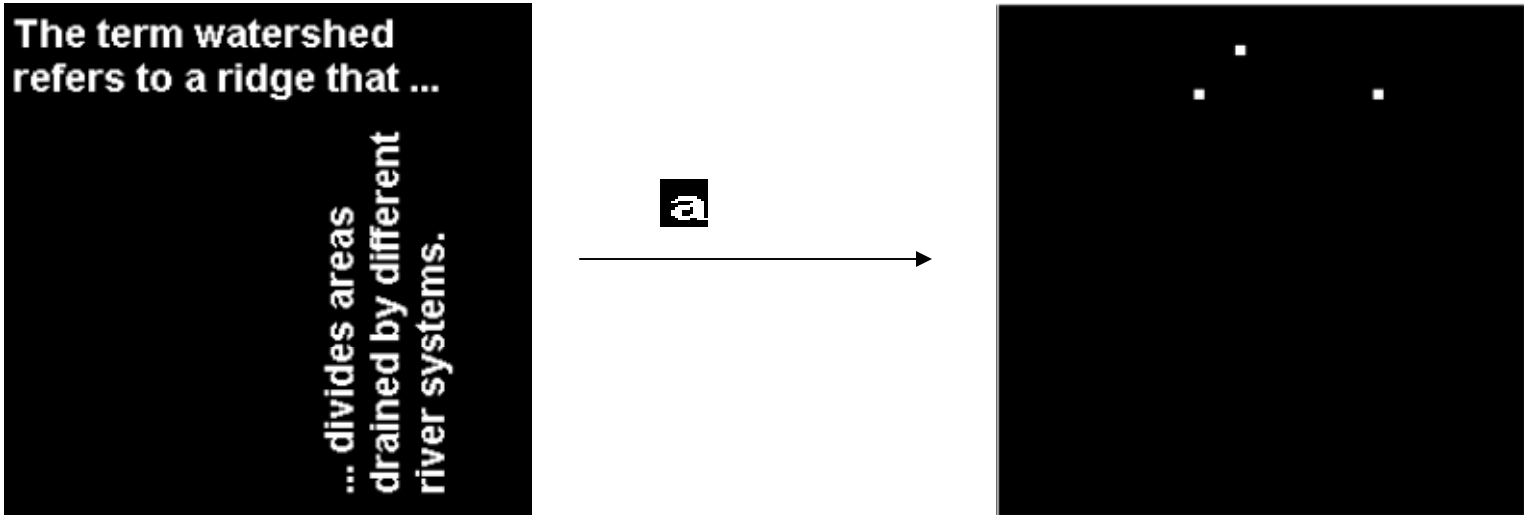
- Why transform
- 2D Fourier Transform
  - Definition
  - Properties
  - Implementation
  - Three DFT applications
    - Convolution, Filtering, Correlation
- Transform in other flavors
  - Unitary transforms
  - DCT and KLT

DFT Application #1: Fast Convolution



DFT Application #2: Feature Correlation

- Find letter “a” in the following image

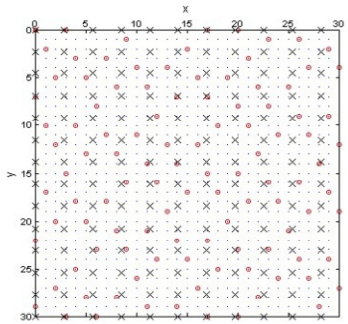


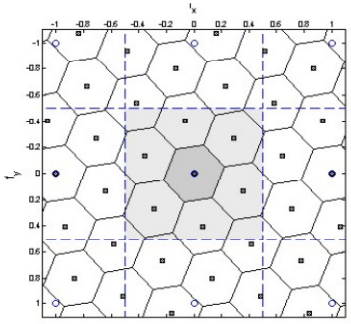

```
bw = imread('text.png');        a = imread('letter_a.png');
% Convolution is equivalent to correlation if you rotate the
convolution kernel by 180deg
C = real(ifft2(fft2(bw) .*fft2(rot90(a,2),256,256)));

% 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.

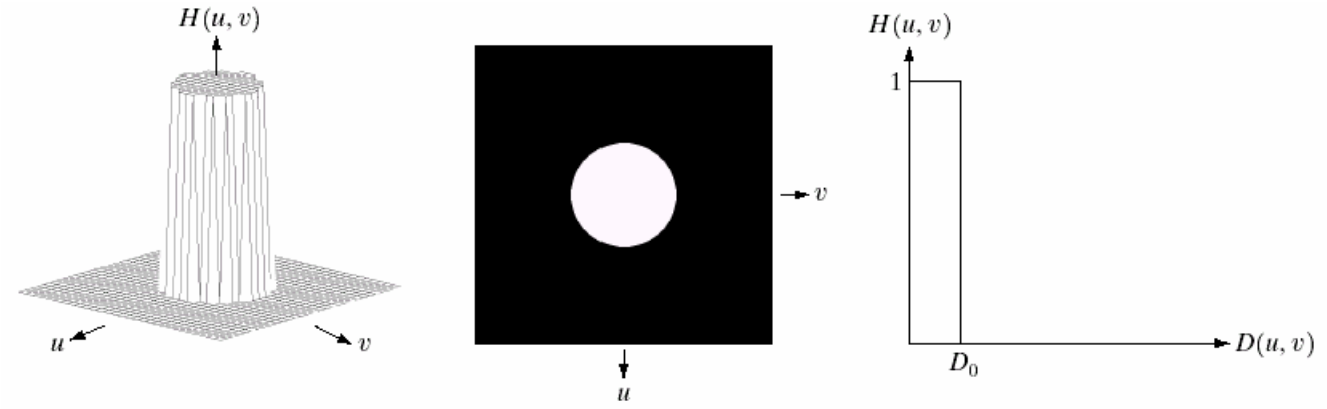
DFT Application #3: Image Filters

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





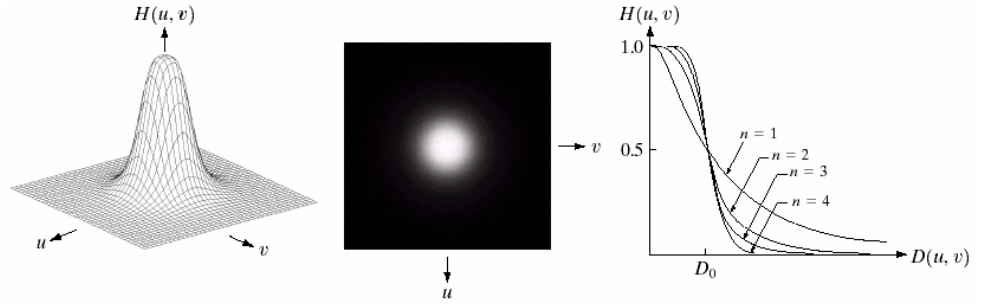
Smoothing Filters: Ideal Low-Pass



a b c

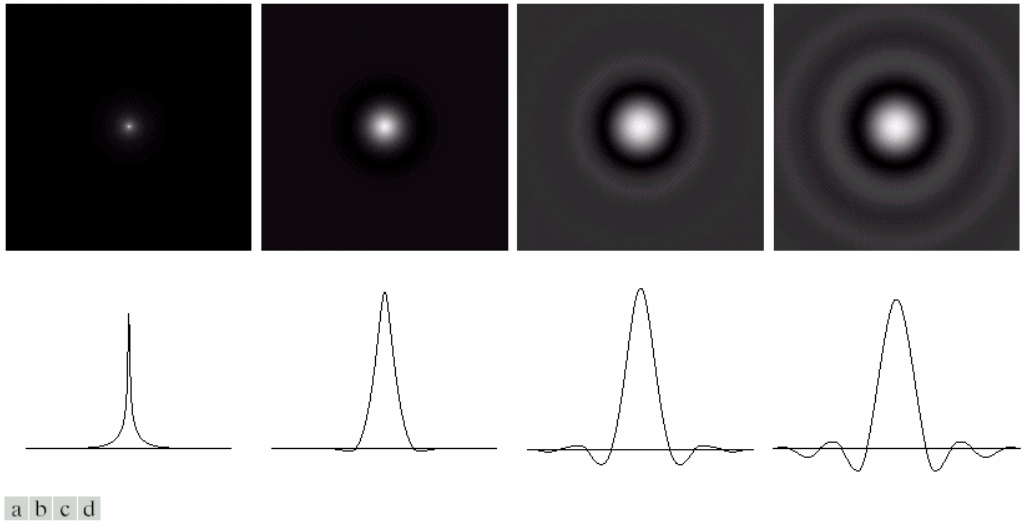
FIGURE 4.10 (a) Perspective plot of an ideal lowpass filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.

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.

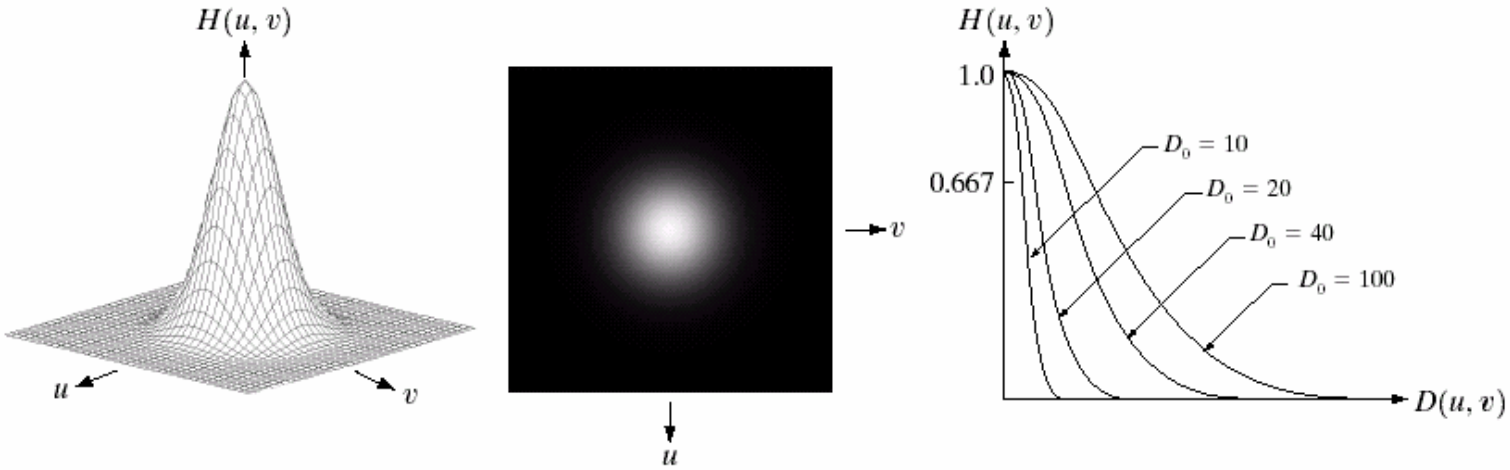


a b c d

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.

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

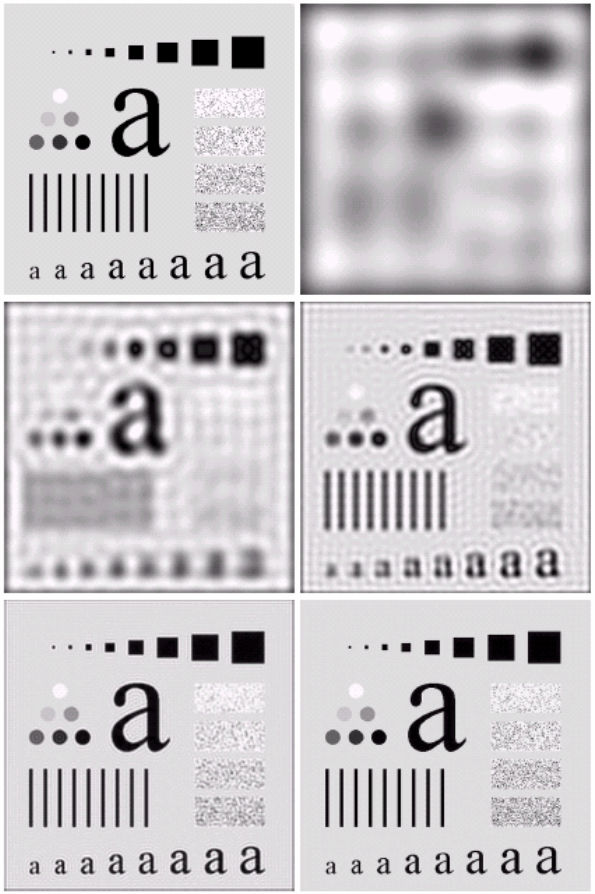
Gaussian Filters



a b c

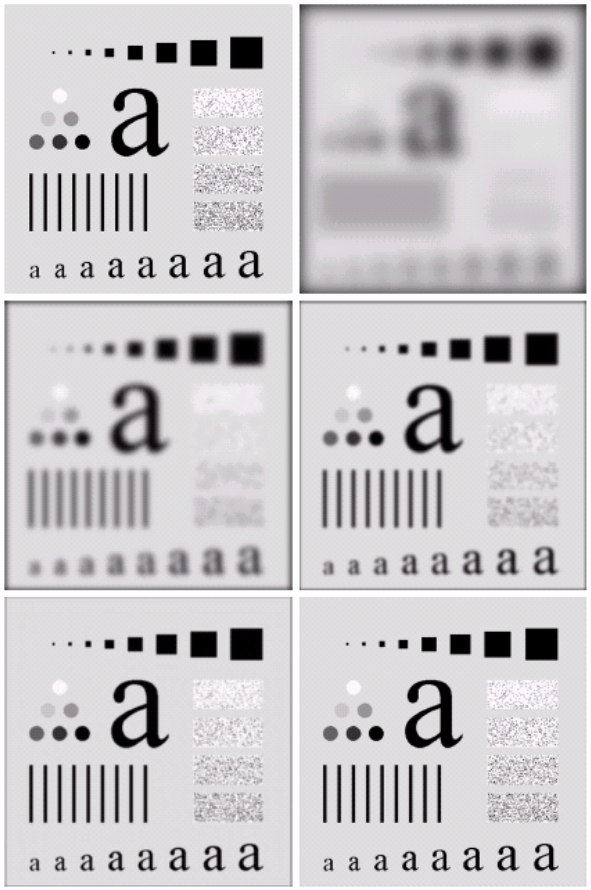
FIGURE 4.17 (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of D_0.

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



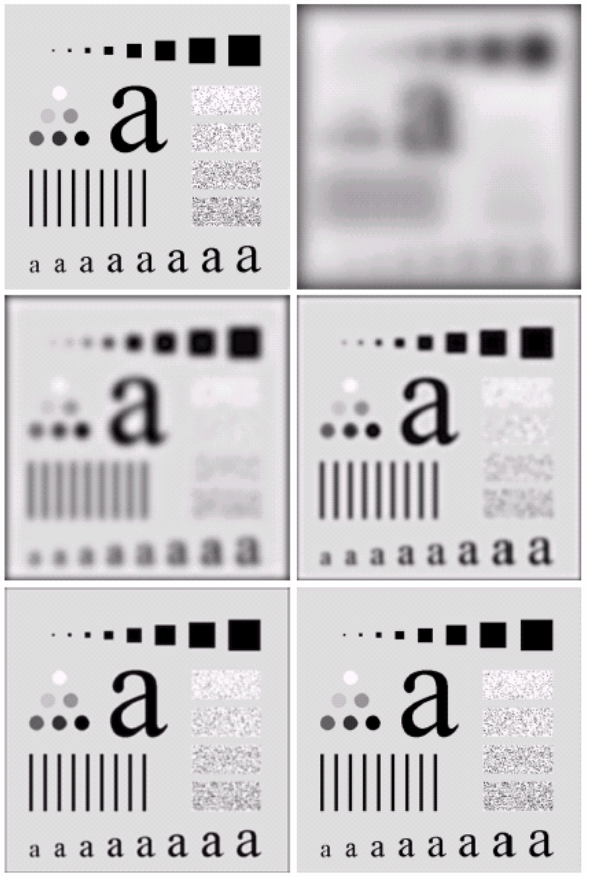
a b
c d
e f

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.



a b
c d
e f

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.



a b
c d
e f

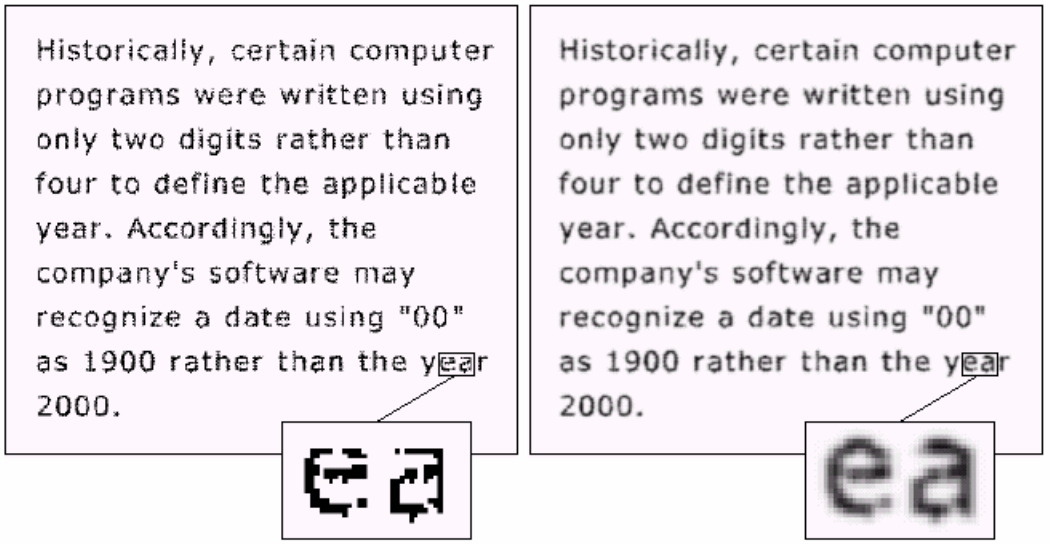
FIGURE 4.15 (a) Original image. (b)–(f) Results of filtering with BLPFs of order 2, with cutoff frequencies at radii of 5, 15, 30, 80, and 230, as shown in Fig. 4.11(b). Compare with Fig. 4.12.

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

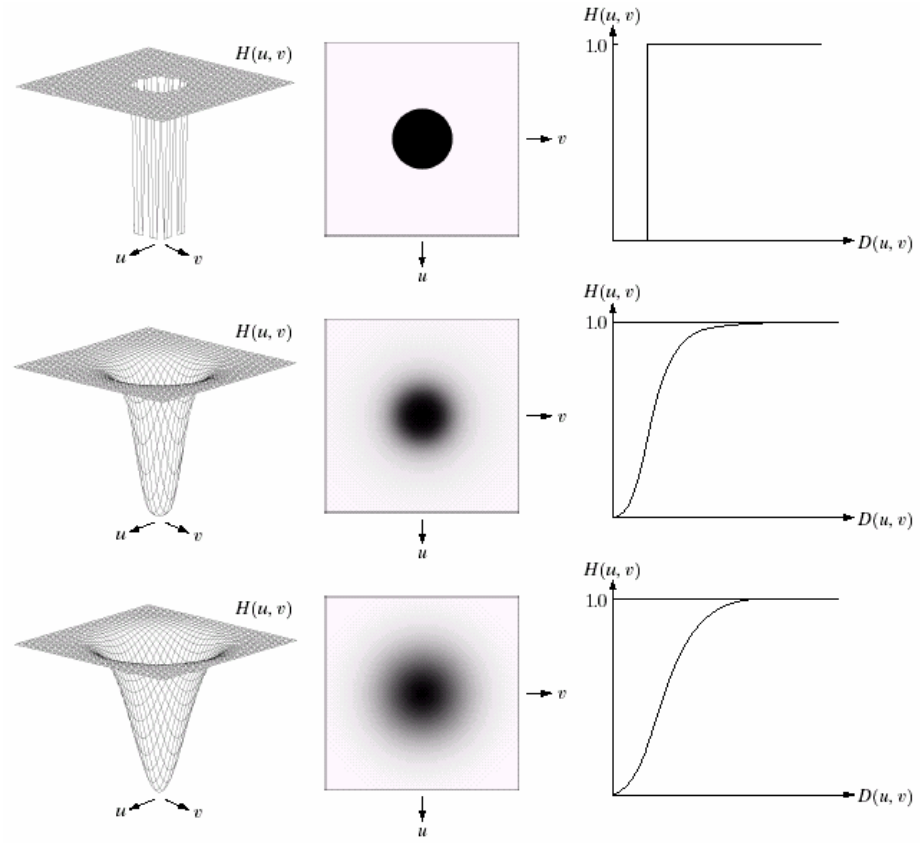
Beautify a photo ☺

a b c

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

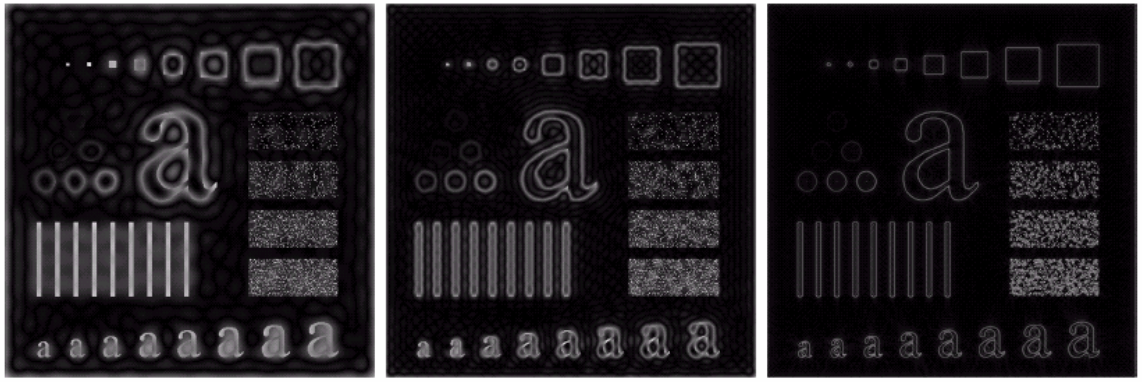
$$H_{HPF}(u,v) = 1 - H_{LPF}(u,v)$$



a b c
d e f
g h i

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.

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.

Lecture Outline

- Why transform
- 2D Fourier Transform
  - Definition, properties, implementation
  - Three DFT applications
    - Convolution, Filtering, Correlation
    - Readings G&W 4.4.4 and 4.4.5
- Transform in other flavors
  - Unitary transforms
  - DCT and KLT
  - …

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	✓	

The Quest for a *Better* Transform …

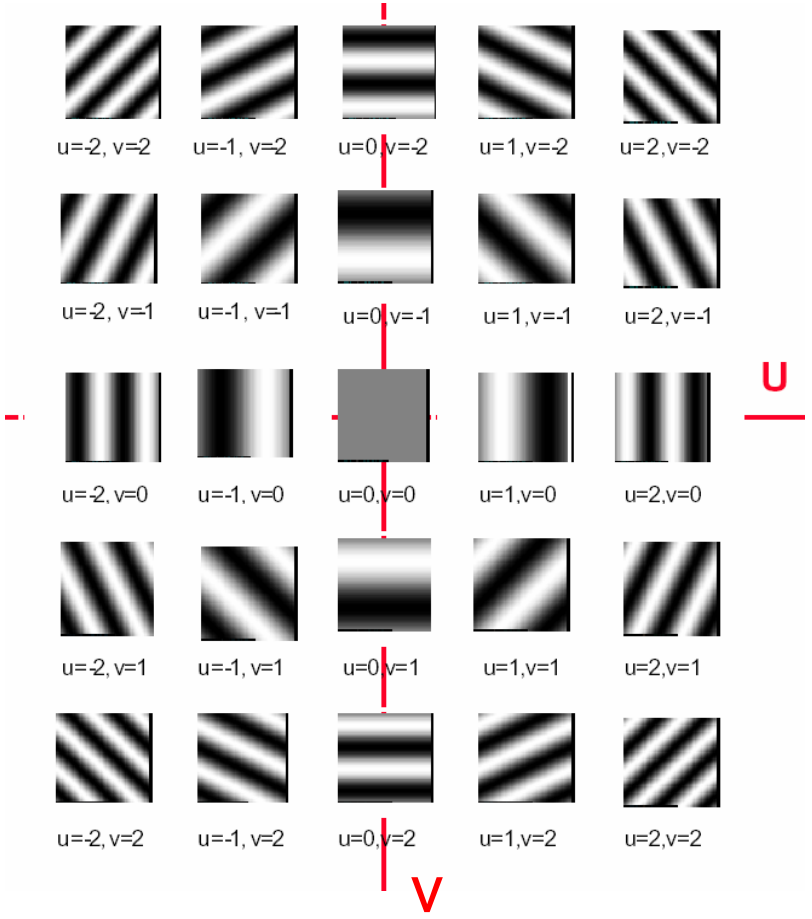
- Image transform (DFT2) as basis expansion:

$$g(u,v) \propto \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)a_{uv}(m,n)$$

$$f(m,n) \propto \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} g(u,v)a^{*}_{uv}(m,n)$$

where $a_{uv}(m,n) = e^{-j2\pi(\frac{um}{M}+\frac{vn}{N})}$

  - Orthonormal (Eq 5.5)
    : no two basis represent the same information in the image
  - Completeness (Eq 5.6)
    : all information in the image are represented in the set of basis functions
  - ? Real-valued → real basis images



The Quest for a *Better* Transform …

- Separable basis expansion:

$$g(u,v) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)a_{uv}(m,n)$$

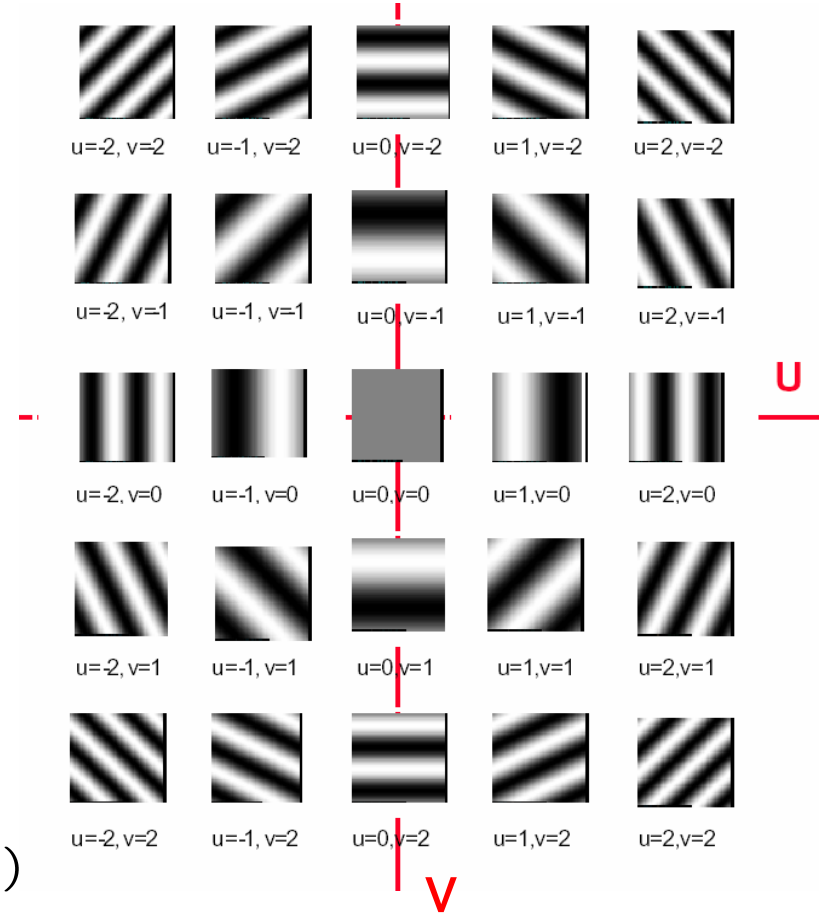
$$f(m,n) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} g(u,v)a^*_{uv}(m,n)$$

where $a_{uv}(m,n) = a_u(m)b_v(n)$

- Denote matrix $A_{ij}=a_i(j)$
- Choose $AA^{*T}=A^*A^T=I$, and b=a, then:

$$g(u,v) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} a_u(m)f(m,n)a_v(n)$$

$$f(m,n) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} a^*_u(m)g(u,v)a^*_v(n)$$



Separable, Real, Unitary Transform

when $a_{uv}(m,n) = a_u(m)b_v(n)$ → We only need to discuss 1D transforms

- Denote matrix $A_{ij}=a_i(j)$
- Hermitian of matrix A: $A^H=A^{*T}$
- Choose $AA^H=A^*A^T=I$ → Orthonormality obtained: row vectors of A form a set of basis vectors

- Also choose b=a, then:

$$g(u,v) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} a_u(m)f(m,n)a_v(n)$$

$$f(m,n) \propto \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} a_u^*(m)g(u,v)a_v^*(n)$$

$\Leftrightarrow$

$$G = AFA^T$$

$$F = A^H G A^*$$

- Choose A among orthogonal matrixes: $A^{-1} = A^T$, $AA^T=I$
  - Real-valued unitary matrix is also an orthogonal matrix
  - Row vectors of real orthogonal matrix A form orthonormal basis vectors

Properties of 1-D Unitary Transform

- Energy Conservation
  - $|| \underline{f}||^2 = || \underline{g}||^2$
    - $|| \underline{g}||^2 = ||A\underline{f}||^2 = (A\underline{f})^{*T}(A\underline{f}) = f^{*T} A^{*T} A \underline{f} = \underline{f}^{*T}\underline{f} = ||\underline{f}||^2$
- Rotation
  - The angles between vectors are preserved
  - A unitary transformation is a rotation of a vector in an N-dimension space, i.e., a rotation of basis coordinates

Slides 39-45, 47-49 are adapted from UMCP ENEE631 Slides created by Min Wu.

Properties of 1-D Unitary Transform

- Energy Compaction
  - Many common unitary transforms tend to pack a large fraction of signal energy into just a few transform coefficients
- Decorrelation
  - Highly correlated input elements → quite uncorrelated output coefficients
  - Covariance matrix $E[(\underline{g} - E(\underline{g})) (\underline{g} - E(\underline{g}))^{*T}]$
    - *small correlation implies small off-diagonal terms*

Question: What unitary transform gives the best compaction and decorrelation?

Coming at the end of this lecture …

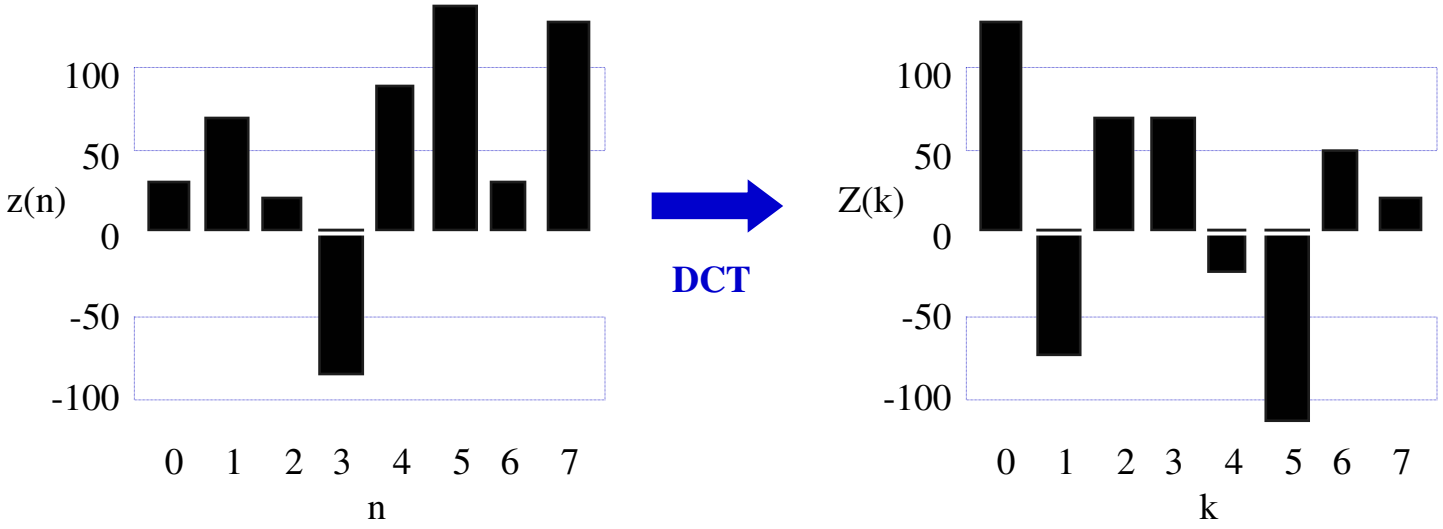
1-D Discrete Cosine Transform (DCT)

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

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

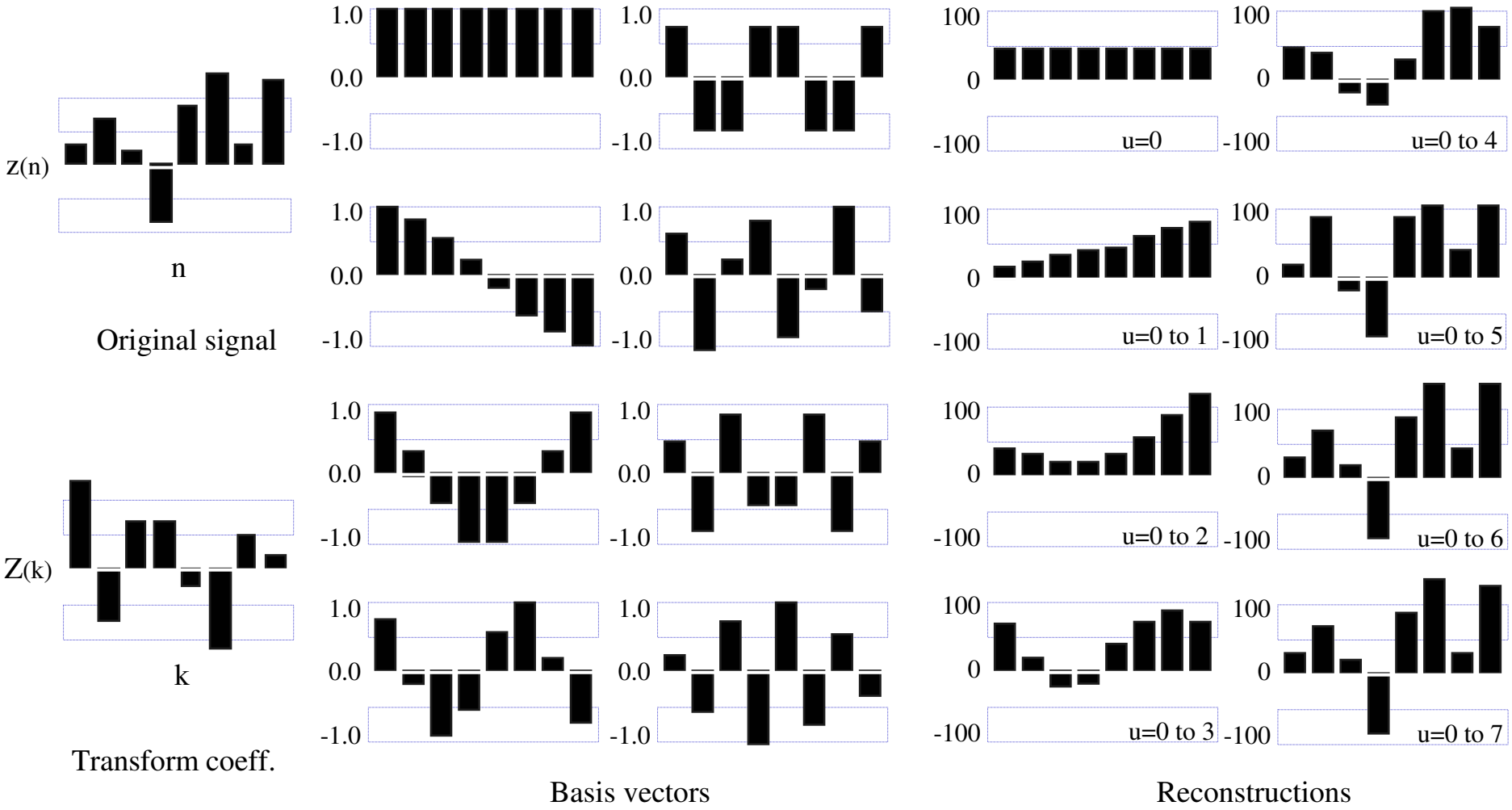
- Transform matrix *C*
  - c(k,n) = $\alpha(0)$ for k=0
  - c(k,n) = $\alpha(k)$ cos[$\pi(2n+1)/2N$] for k>0
- C is real and orthogonal
  - rows of C form orthonormal basis
  - C is not symmetric!
  - DCT is not the real part of unitary DFT!

Example of 1-D DCT



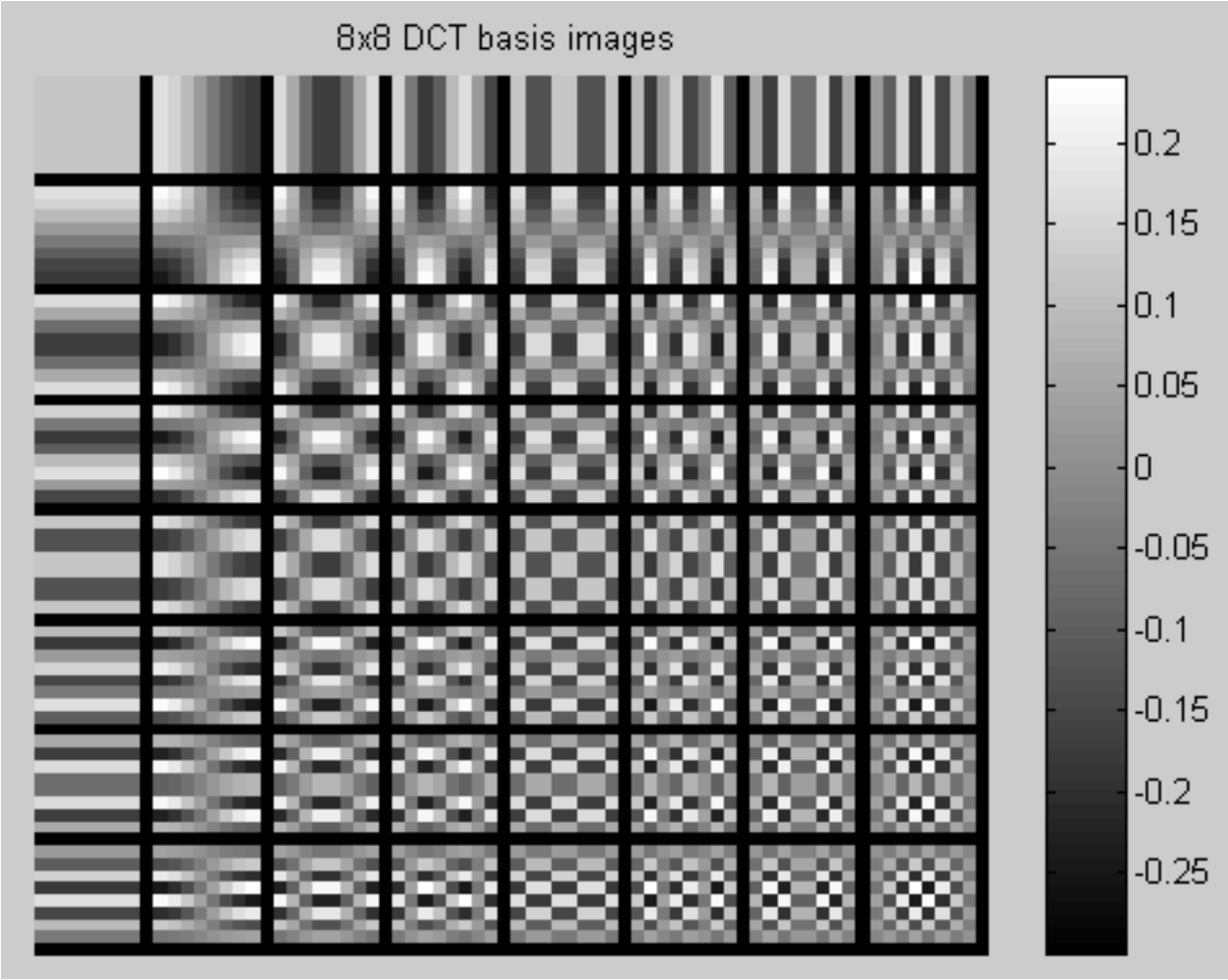
From Ken Lam's DCT talk 2001 (HK Polytech)

1-D DCT

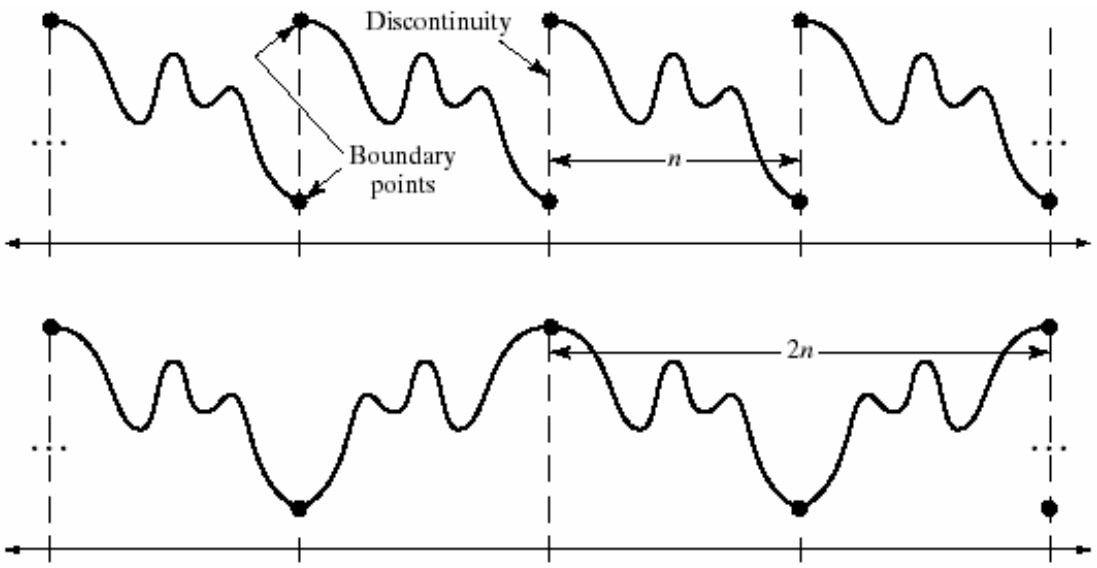


From Ken Lam's DCT talk 2001 (HK Polytech)

DCT Basis Images



Periodicity Implied by DFT and DCT



a
b

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

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

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

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	✓		

Two Observations

- Unitary transforms we've dealt so far are data-independent
  - Transform basis/filters are not depending on the signals we are processing
- We have not proof/found a unitary transform that gives the best energy compaction and de-correlation.
  - "Optimal" in a statistical sense to allow the transform to work well with many images
  - Signal statistics would play an important role

Review: Correlation After a Linear Transform

- Consider an Nx1 zero-mean random vector $\underline{x}$
  - Covariance (autocorrelation) matrix $R_x = E[\, \underline{x}\, \underline{x}^H \,]$
    - give ideas of correlation between elements
    - R_x is a diagonal matrix iff. all N r.v.'s are uncorrelated

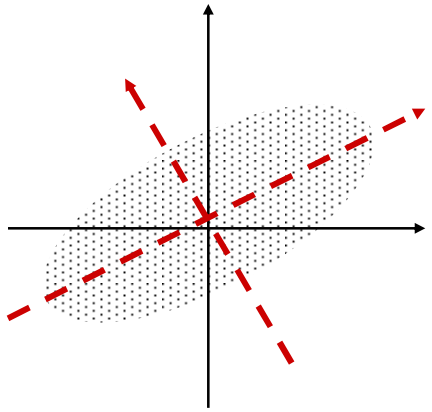
- Apply a linear transform to $\underline{x}$: $\underline{y} = A\, \underline{x}$
- What is the correlation matrix for $\underline{y}$?

$$R_y = E[\, \underline{y}\, \underline{y}^H \,] = E[\, (A\underline{x})\, (A\underline{x})^H \,] = E[\, A\, \underline{x}\, \underline{x}^H\, A^H \,]$$
$$= A\, E[\, \underline{x}\, \underline{x}^H \,]\, A^H = A\, R_x\, A^H$$

- Decorrelation: try to search for A that can produce a decorrelated $\underline{y}$ (i.e. a diagonal correlation matrix R_y)

Karhunen-Loeve Transform (KLT)

- a.k.a the Hotelling transform or the Principle Component Analysis (PCA)
- Eigen decomposition of R_x: $R_x \underline{u}_k = \lambda_k \underline{u}_k$
  - Recall the properties of R_x
    - Hermitian (conjugate symmetric $R^H = R$);
    - Nonnegative definite (real non-negative eigen values)



- Karhunen-Loeve Transform (KLT)

  $y = U^H x \Leftrightarrow x = U y$ with $U = [\, \underline{u}_1, \ldots \underline{u}_N \,]$

  - KLT is a unitary transform with basis vectors in U being the orthonormalized eigenvectors of R_x

  - $U^H R_x U = \mathrm{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_N\}$ i.e. KLT performs decorrelation
  - Often order $\{\underline{u}_i\}$ so that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$

Summary of Lecture 5

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

- Coming in Lecture 6: 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, …