EL5123 --- Image Processing

DFT Domain Image Filtering

Yao Wang
Polytechnic Institute of NYU, Brooklyn, NY 11201

With contribution from Zhu Liu, Onur Guleryuz, and Gonzalez/Woods, Digital Image Processing, 2ed

Lecture Outline

- 1D discrete Fourier transform (DFT)
- 2D discrete Fourier transform (DFT)
- Fast Fourier transform (FFT)
- DFT domain filtering
- 1D unitary transform
- 2D unitary transform

Discrete Fourier Transform (DFT): DTFT for Finite Duration Signals

If the signal is only defined for $n = 0,1,\ldots, N-1$:

Fourier transform becomes:

$$F'(f) = \sum_{n=0}^{N-1} f(n)\exp(-j2\pi fn), \quad f \in (0,1)$$

Sampling $F'(f)$ at $f = k/N,\ k = 0,1,\ldots,N\text{-}1$, and rescaling yields:

Forward transform (DFT):

$$F(k) = F'\left(\frac{k}{N}\right) = \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} f(n)\exp\left(-j2\pi\frac{k}{N}n\right), \quad k = 0,1,\ldots, N-1$$

Inverse transform (IDFT):

$$f(n) = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} F(k)\exp\left(j2\pi\frac{k}{N}n\right), \quad n = 0,1,\ldots, N-1$$

Property of DFT (1)

- Periodicity

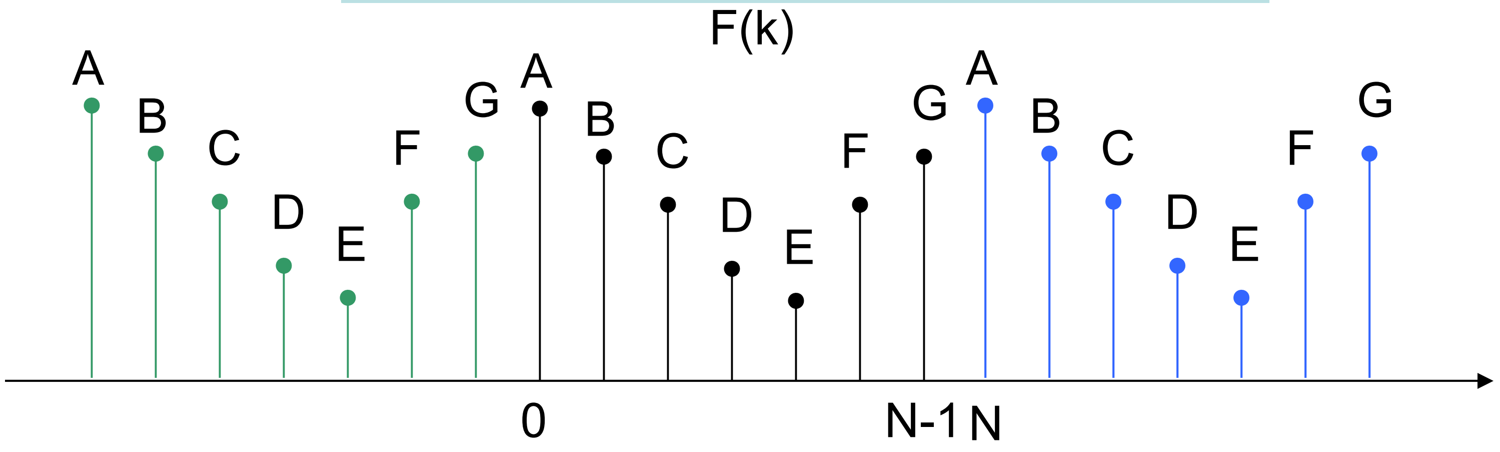
$$F(k) = F(((k))_N), \quad k < 0 \quad or \quad k \geq N.$$

$$\text{where} \quad ((k))_N \quad \text{represents modulo } N.$$

Proof

$$F(k+mN) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f(n) \exp(-j2\pi \frac{(k+mN)}{N} n)$$

$$= \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f(n) \exp(-j2\pi \frac{k}{N} n - j2\pi mn) = F(k)$$



low $\longrightarrow$ high $\longleftarrow$ low

Note: Highest frequency is at k=[N/2]. k=0,1, N-1 represent low frequency.

Property of DFT (2)

- Translation

$$f(((n-n_o))_N) \quad \Leftrightarrow \quad F(k)\exp\{-j2\pi(kn_o/N)\}$$
$$f(n)\exp\{j2\pi k_0 n/N\} \quad \Leftrightarrow \quad F(((k-k_0))_N).$$

– Special case

- N is even, k_0 = N/2.

$$f(n)\exp\{j\pi n\} = f(n)(-1)^n \quad \Leftrightarrow \quad F(((k-\frac{N}{2}))_N).$$

Shifting the frequency up by N/2

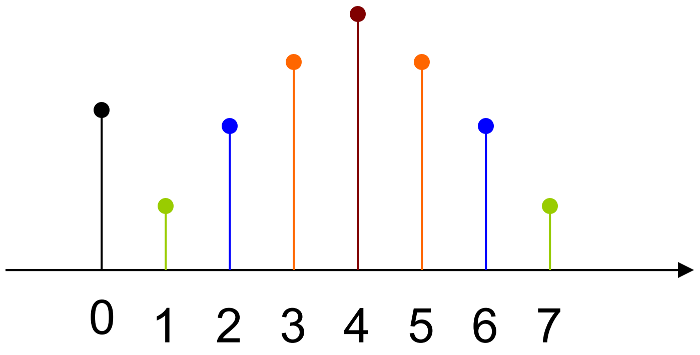
Property of DFT (3)

- Conjugate symmetry for real sequences

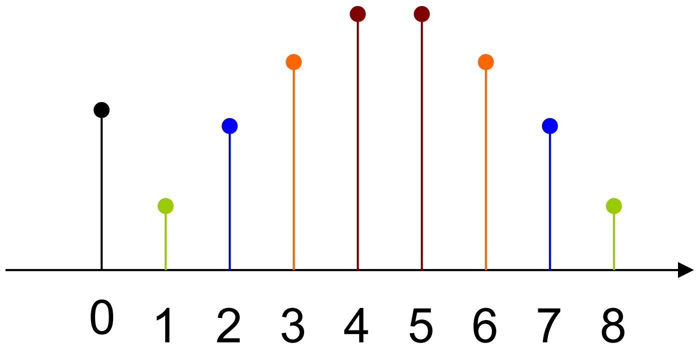
$$F(k) = F^*(-k) = F^*(N-k)$$

$$|F(k)| = |F(N-k)| \quad or$$

$$|F(\frac{N}{2}+k)| = |F(\frac{N}{2}-k)|, \quad when\ N\ is\ even.$$



N=8



N=9

2D Discrete Fourier Transform

- Definition
  - Assuming f(m, n), m = 0, 1, …, M-1, n = 0, 1, …, N-1, is a finite length 2D sequence

$$F(k,l)=\frac{1}{\sqrt{MN}}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)e^{-j2\pi\left(\frac{km}{M}+\frac{ln}{N}\right)},\quad k=0,1,\ldots,M-1,\, l=0,1,\ldots,N-1;$$

$$f(m,n)=\frac{1}{\sqrt{MN}}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} F(k,l)e^{j2\pi\left(\frac{km}{M}+\frac{ln}{N}\right)},\quad m=0,1,\ldots,M-1,\, n=0,1,\ldots,N-1.$$

  - Comparing to DTFT

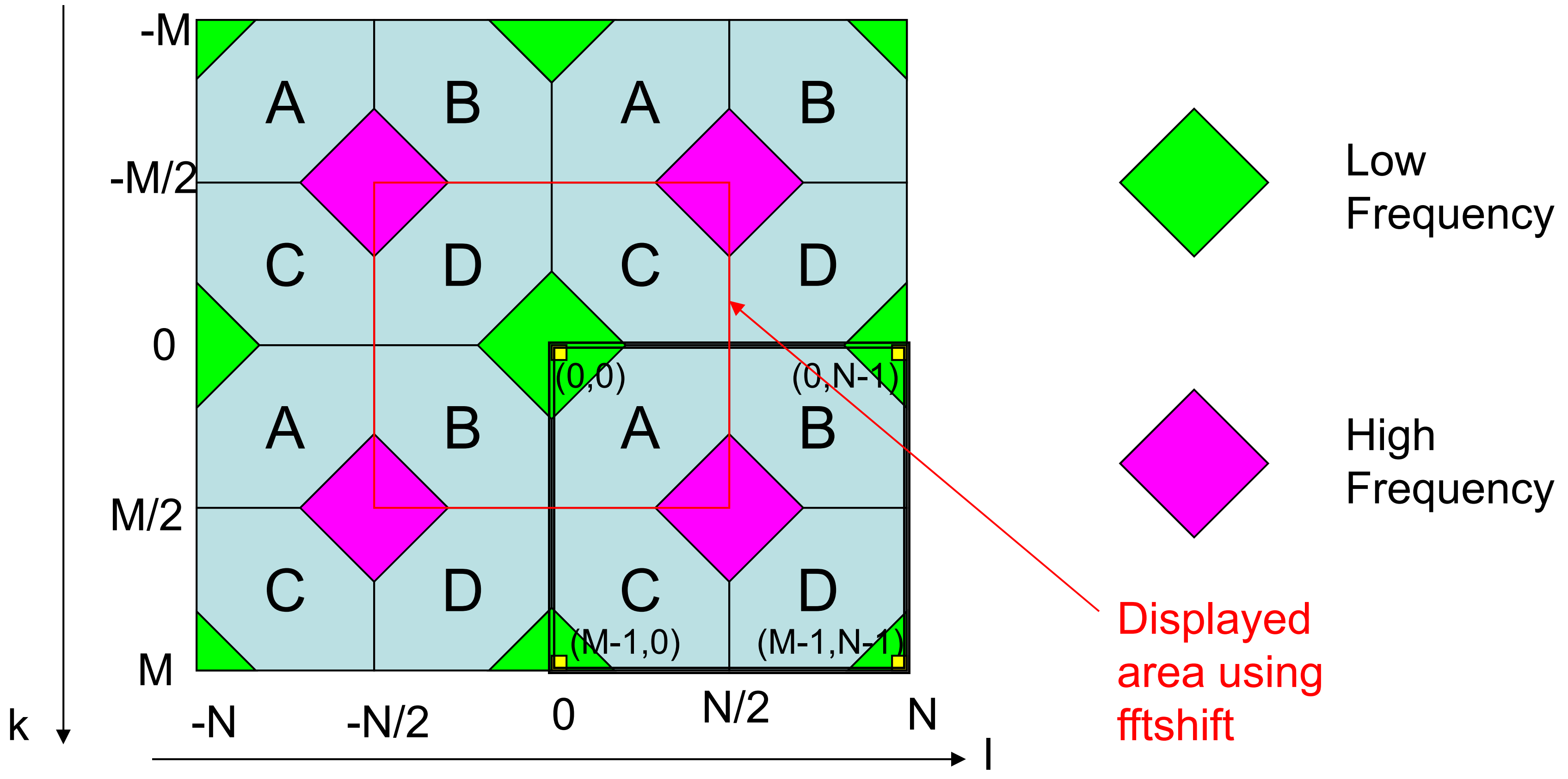
$$F(u,v)=\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} f(m,n)e^{-j2\pi(mu+nv)},\quad u=\frac{k}{M},v=\frac{l}{N}$$

$$f(m,n)=\int_{-1/2}^{1/2}\int_{-1/2}^{1/2} F(u,v)e^{j2\pi(mu+nv)}dudv$$

Property of 2D DFT (1)

- Periodicity

$$F(k,l) = F(((k))_M, ((l))_N), \quad k < 0 \; or \; k \geq M, l < 0 \; or \; l \geq N.$$



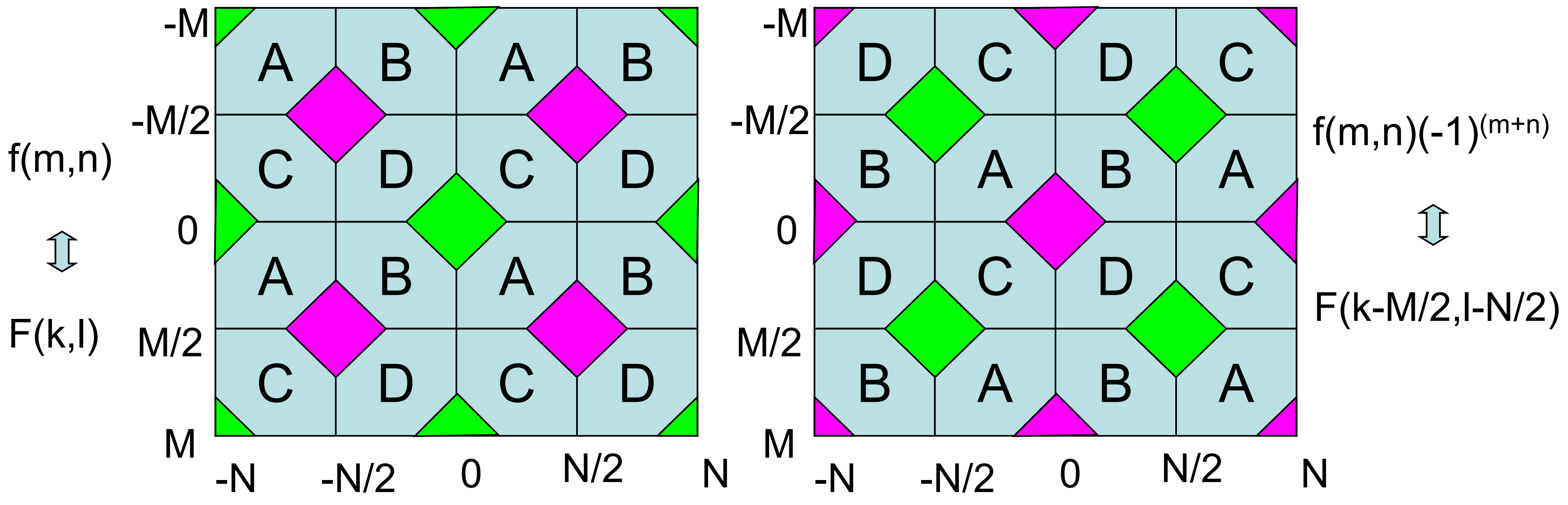
Property of 2D DFT (2)

- Translation

$$f(((m-m_0))_M,((n-n_0))_N) \Leftrightarrow F(k,l)\exp\{-j2\pi(km_0/M+ln_0/N)\},$$
$$f(m,n)\exp\{j2\pi(k_0m/M+l_0n/N)\} \Leftrightarrow F(((k-k_0))_M,((l-l_0))_N).$$

  – Special Case: M,N=even, k_0=M/2, l_0=N/2

$$f(m,n)\exp\{j\pi(m+n)\}=f(m,n)(-1)^{(m+n)} \Leftrightarrow F(((k-\frac{M}{2}))_M,((l-\frac{N}{2}))_N).$$



Property of 2D DFT (3)

- Conjugate symmetry for real sequences

$$F(k,l)=F^*(-k,-l)=F^*(M-k,N-l),$$
$$|F(k,l)|=|F(M-k,N-l)|,$$
$$|F(\frac{M}{2}+k,\frac{N}{2}+l)|=|F(\frac{M}{2}-k,\frac{N}{2}-l)|,\quad for\ M,N=even$$

0	1	2	3	4	3	2	1
5	6	7	8	9	33	32	31
10	11	12	13	14	30	29	28
15	16	17	18	19	27	26	25
20	21	22	23	24	23	22	21
15	25	26	27	19	18	17	16
10	28	29	30	14	13	12	11
5	31	32	33	9	8	7	6

M=N=8=even, 🟨 real 🟩 x ▨ x Conjugate pairs

0	1	2	3	4	4	3	2	1
5	6	7	8	9	40	39	38	37
10	11	12	13	14	36	35	34	33
15	16	17	18	19	32	31	30	29
20	21	22	23	24	28	27	26	25
20	25	26	27	28	24	23	22	21
15	29	30	31	32	19	18	17	16
10	33	34	35	36	14	13	12	11
5	37	38	39	40	9	8	7	6

M=N=9=odd

Property of 2D DFT (5)

- Separabability
  - 2D DFT can be accomplished by N-point 1D DFT of each row, followed by M-point 1D DFT of each column
- How many 1D DFT's?
  - M rows: M N-pt DFT's
  - N columns: N M-pt DFT's
  - M=N: 2N N-pt DFT's
  - Each N-pt DFT requires N^2 operations, total requiring $2N^3$ operations
- Direct calculation
  - Requiring M^2N^2 (N^4 if M=N) operations

Property of 2D DFT (6)

- 2D DFT of Separable Images

$$f(m,n) = f_x(m) f_y(n) \Leftrightarrow F(k,l) = F_x(k) F_y(l)$$

In matrix form:

$$\mathbf{f} = \mathbf{f}_\mathrm{x}\, \mathbf{f}_\mathrm{y}^\mathrm{T} \Leftrightarrow \mathbf{F} = \mathbf{F}_\mathrm{x}\, \mathbf{F}_\mathrm{y}^\mathrm{T}$$

$$\mathbf{f} = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \cdots & \cdots & \cdots & \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

$$\mathbf{f}_\mathrm{x} = \left[f_x(0) \quad f_x(1) \quad \cdots \quad f_x(M-1) \right]^T \xleftrightarrow{\text{M−pt 1D DFT}} \mathbf{F}_\mathrm{x}$$

$$\mathbf{f}_\mathrm{y} = \left[f_y(0) \quad f_y(1) \quad \cdots \quad f_y(N-1) \right]^T \xleftrightarrow{\text{N−pt 1D DFT}} \mathbf{F}_\mathrm{y}$$

Real images are seldom separable. But many 2D filters are separable.

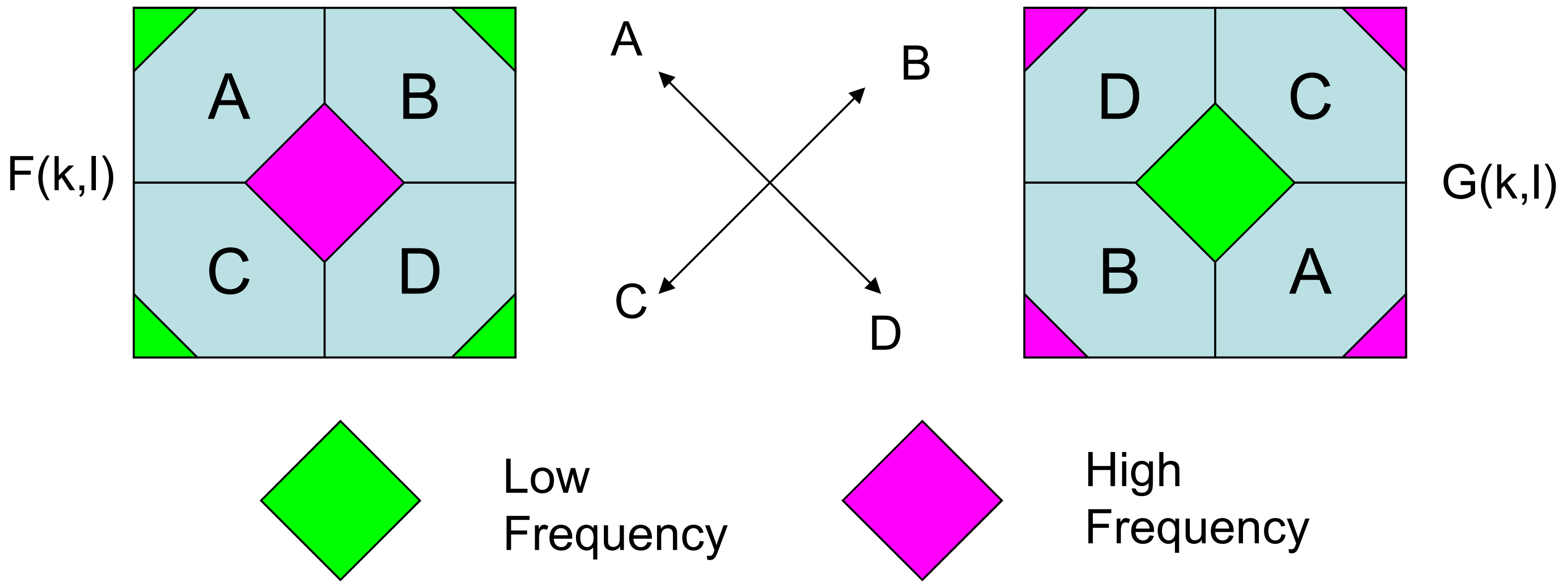
Computer Implementation of 2D DFT

- Complex array structure
- Implement complex multiplication and addition
- Pre-compute the constants: $e^{-j2\pi km/M}$ for all k,m=0, 1, …, M-1, and $e^{-j2\pi ln/N}$ for all l,n=0, 1, …, N-1.
- Use separable processing to speed up
- For real square image, only need to calculate half of the points

Display of the Magnitude of 2D DFT

- Shifting the low frequency components into the center

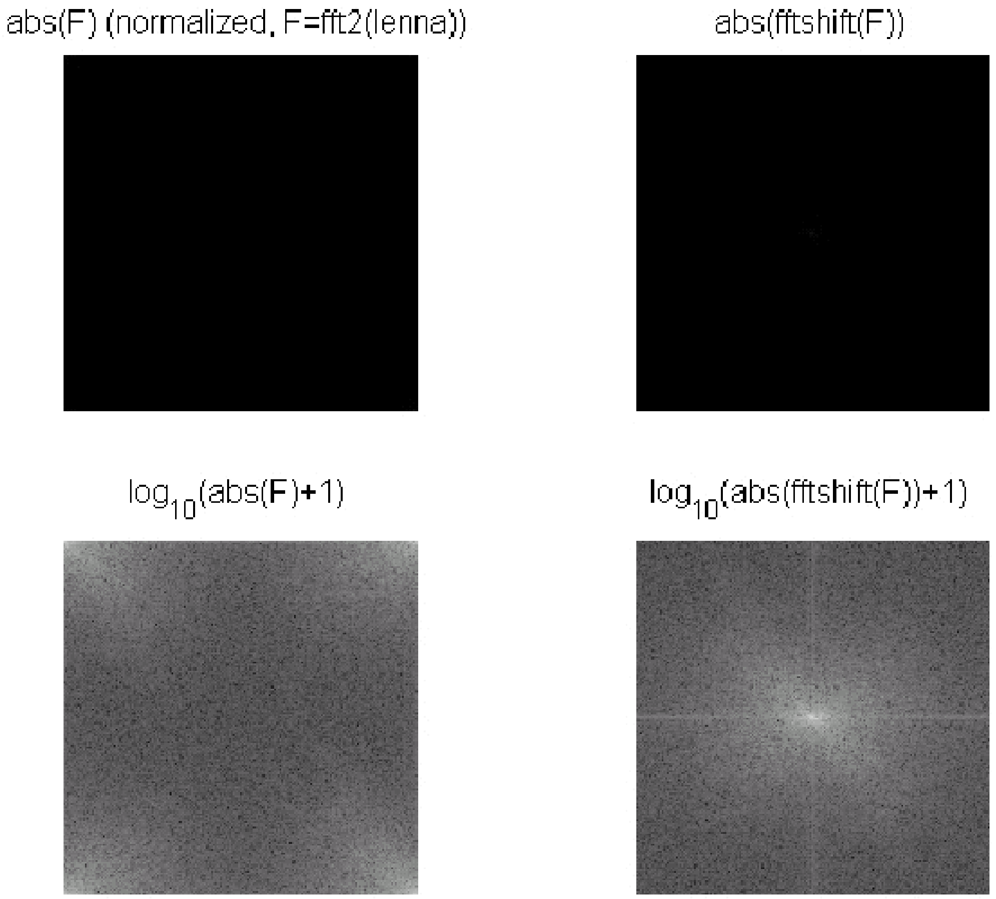
$$g(m,n) = f(m,n)(-1)^{(m+n)} \quad \Leftrightarrow \quad G(k,l) = F(((k-\frac{M}{2}))_M, ((l-\frac{N}{2}))_N).$$



Matlab command: fftshift

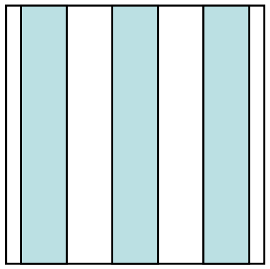
Display of the Magnitude of 2D DFT

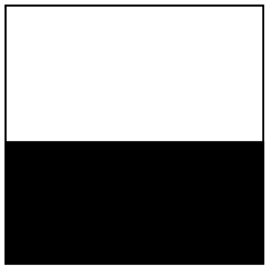
- Amplitude rescaling $G(k,l) = \log(1 + F(k,l))$

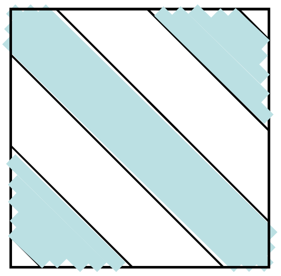

abs(F) (normalized, F=fft2(lenna))
abs(fftshift(F))
$\log_{10}$(abs(F)+1)
$\log_{10}$(abs(fftshift(F))+1)

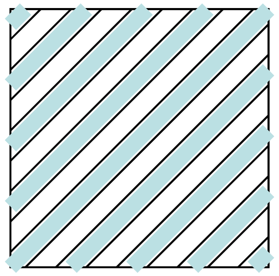
Sample Fourier pairs

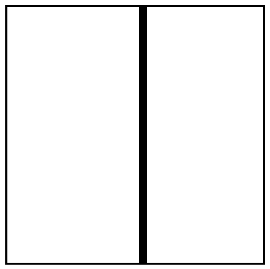
Which Fourier transform in the second row is for each image in the first row?

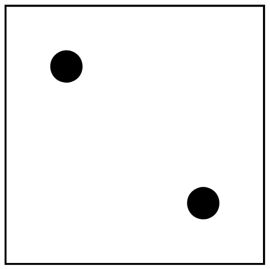

(a)

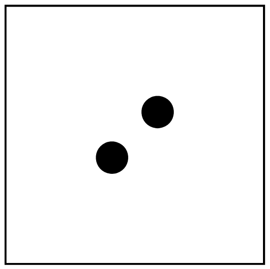

(b)

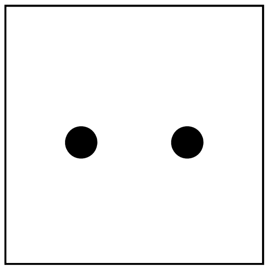

(c)


(d)


(e)


(f)


(g)


(h)

Fast Fourier Transform (FFT)

- Direct computation of N-point DFT takes N^2 operations
- FFT is a fast algorithm for computing DFT, reducing the computation from N^2 to $N \log_2(N)$
  - Complex conjugate symmetry of $e^{-j2\pi kn/N}$

$$e^{-j2\pi k(N-n)/N} = e^{j2\pi kn/N} = e^{-j2\pi k(-n)/N} = \left(e^{-j2\pi kn/N}\right)^*$$

  - Periodicity in n and k of $e^{-j2\pi kn/N}$

$$e^{-j2\pi kn/N} = e^{-j2\pi k(n+N)/N} = e^{-j2\pi (k+N)n/N}$$

- Matlab function for N-point DFT: *fft(A,N), fft2(A,M,N)*

Computation of 2D DFT

- 2D (*MxN*) point DFT can be computed in a separable manner:
  - First compute *N*-point FFT for each row ($M N \log_2(N)$)
  - Then compute *M*-point FFT for each column ($N M \log_2(M)$)
  - Total computation if $M=N$: $2N^2 \log_2(N)$

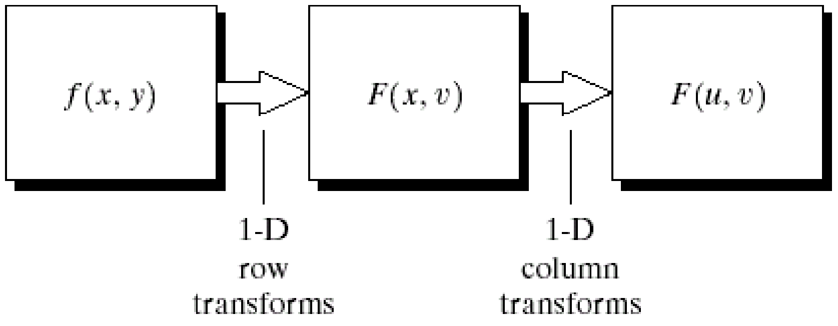
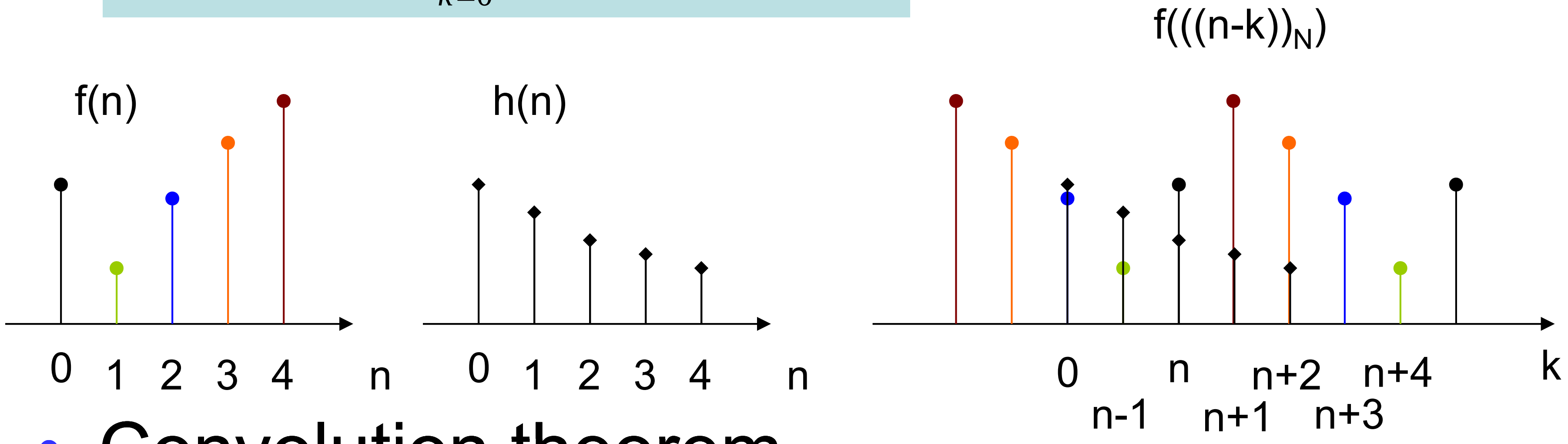


FIGURE 4.35 Computation of the 2-D Fourier transform as a series of 1-D transforms.

Convolution Theorem (1D)

- Circular convolution (N-pt)

$$f(n) \otimes h(n) = \sum_{k=0}^{N-1} f(((n-k))_N)h(k)$$



- Convolution theorem

$$f(n) \otimes h(n) \Leftrightarrow F(k)H(k)$$

Linear Convolution vs. Circular Convolution

- f(n): N1 pt, h(n): N2 pt, assume N1>=N2
- f(n)*h(n): N=N1+N2-1 pt
- Equivalent to circular convolution of M-pt, if M>=N

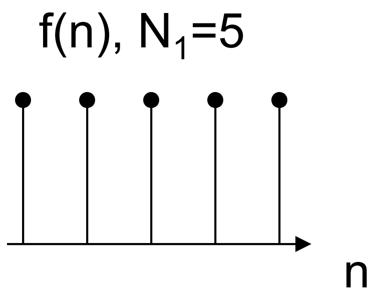
- If we do N1 pt circular convolution, which parts of the resulting output is equal to that of linear convolution (assume N2 is much smaller than N1)?
  - Illustrate on board
  - If f(n) is from 0 to N1-1, h(n) is from 0 to N2-1
  - Circular conv. = linear conv. Over N2-1 to N1-1

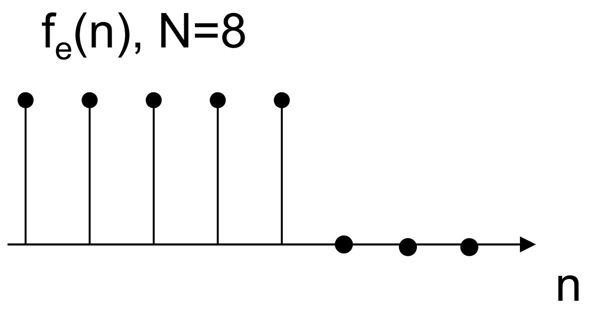
Calculate Linear Convolution Using DFT

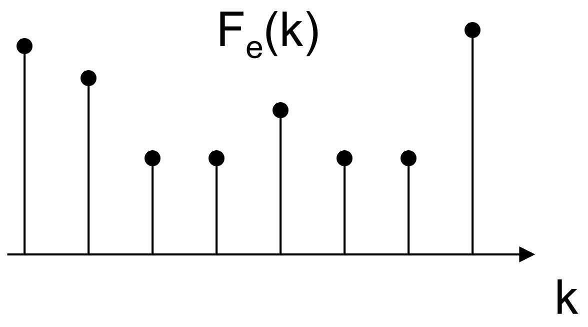
- 1D case
  - f(n) is length N_1, h(n) is length N_2
  - g(n) = f(n)*h(h) is length N = N_1+N_2-1.
  - To use DFT, need to **extend** f(n) and h(n) to length N by zero padding.

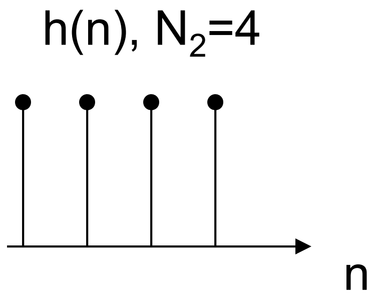
$$\begin{tikzcd}
f(n) \arrow[d, leftrightarrow, "N\text{-point DFT}"'] & * & h(n) \arrow[d, leftrightarrow, "\text{DFT}"'] \arrow[r, "\text{Convolution}"] & g(n) \arrow[d, leftrightarrow, "\text{DFT}"'] \\
F(k) & \text{x} & H(k) \arrow[r, "\text{Multiplication}"] & G(k)
\end{tikzcd}$$

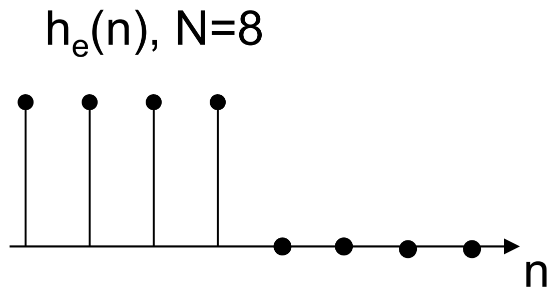
Calculate Linear Convolution Using DFT

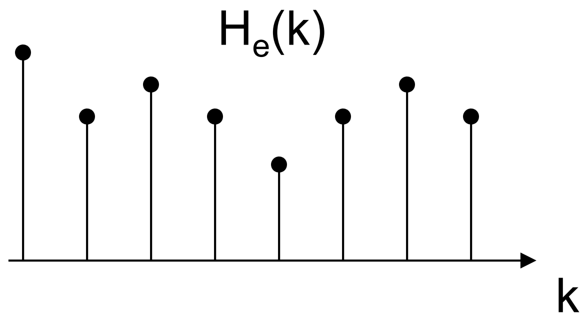




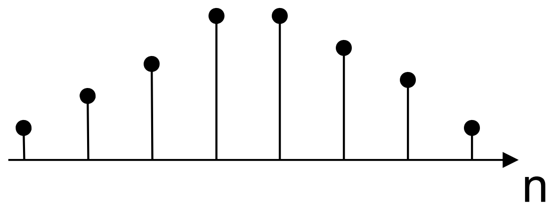


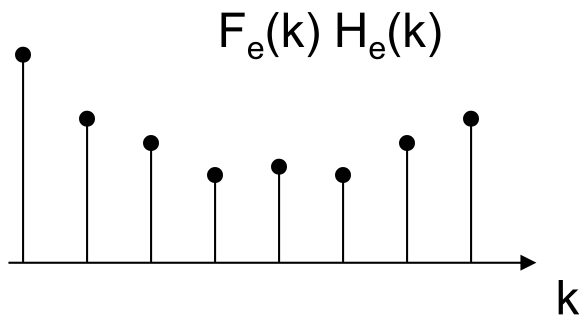






$$f(n) * h(n) = f_e(n) \otimes h_e(n)$$





Comparison of Complexity

- Direct calculation
  - Each point, N_2 multiplications
  - Overall, $N_2 * N \approx O(N^2)$
- Via FFT
  - N-point FFT for f(n) and h(n)
  - N multiplications in the DFT domain
  - N-point inverse FFT for F(k)*H(k)
  - Overall, $3N\log N + N \approx O(N\log N)$

Circular Convolution (2D)

- Circular Convolution

$$f(m,n) \otimes h(m,n) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f(((m-k))_M, ((n-l))_N) h(k,l)$$

- Convolution Theorem

$$f(m,n) \otimes h(m,n) \quad \Leftrightarrow \quad F(k,l) H(k,l)$$

Circular vs. Linear Convolution

– If f(m,n) is N1xM1 pt, h(m,n) is N2xm2 pt

– f(m,n)*h(m,n) is N'xM' pt, N'=N1+N2-1,M'=M1+M2-1

– Equivalent to NxM pt circular convolution if N>=N',M>-M'

– Can be implemented by using NxM pt 2D DFT!

Separable filter

- If the filter is separable,
  - h(m,n)=hx(m) hy(n)
  - Row-by-row circular convolution with hy(n), then column-by-column circulua convolution with hx(m)

Filtering in DFT Domain

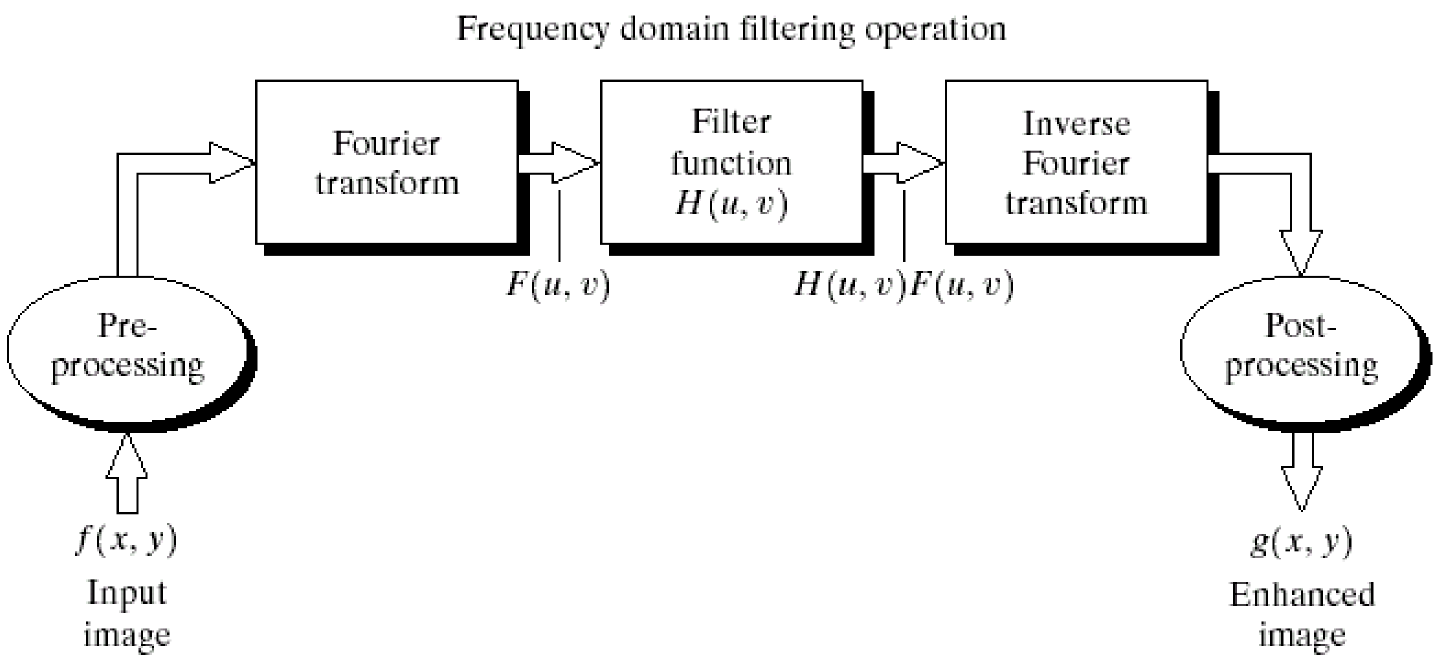


FIGURE 4.5 Basic steps for filtering in the frequency domain.

Relation between spatial and frequency domain operation:

$$g(x,y) = h(x,y) \otimes f(x,y) \quad \Leftrightarrow \quad G(u,v) = H(u,v)F(u,v)$$

$$h(x,y) = IDFT(H(u,v)), \quad H(u,v) = DFT(h(x,y)).$$

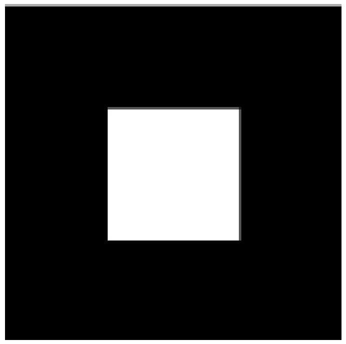
Typically DFT size=image size. This corresponds to circular convolution.

Low-Pass Filter Using DFT Window

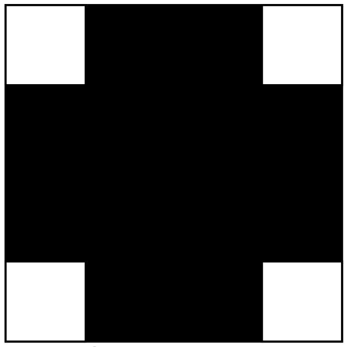
- Filtering in DFT domain:
  - G(k,l)=H(k,l) F(k,l)
- Ideal Low-Pass Filter
  - H(k,l)= 1 in low frequency range (the four corners!)

    = 0 in high frequency range
  - Can think of H(k,l) as a mask in the DFT domain
    - The (k,l)th coefficient is unchanged if H(k,l)=1, and is set to zero if H(k,l)=0.
- More generally, H(k,l) can take on higher values in low-frequency range, and lower values in high frequency range
- Note that equivalent spatial filter is truncated sinc. With Circular convolution, it leads to ringing artifacts.

Low-Pass Filter in DFT Domain



The window size where H(k,l)=1 is (2 W1+1)x(2 W2+1)

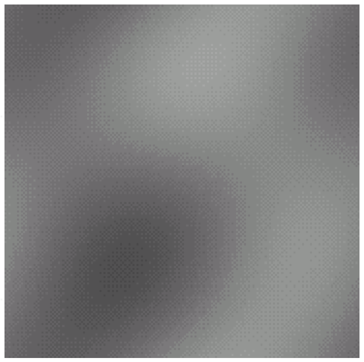


Before shifting: note low frequency are at four corners!

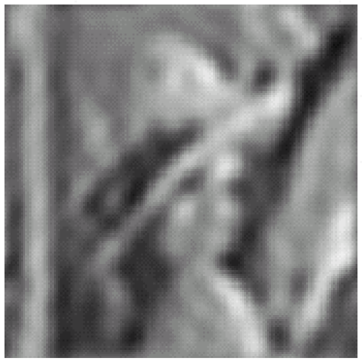
- The window can be implemented in matlab via
  - » R1=zeros(M,1);
  - » R1(1:W1+1,M-W1+1:M)=1;
  - » R2=zeros(N,1);
  - » R2(1:W2+1,N-W2+1:N)=1;
  - » H=R1*R2';
- Given G(k,l)=F(k,l)H(k,l), matlab may make small numerical errors in the inverse transform leading to complex valued images. g(m,n) is best reconstructed via:
  - » g=real(ifft2(F.*H));

Example

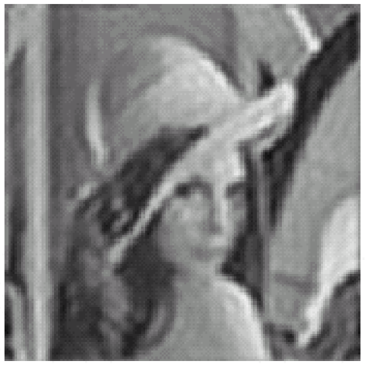
$W_1=W_2=1$



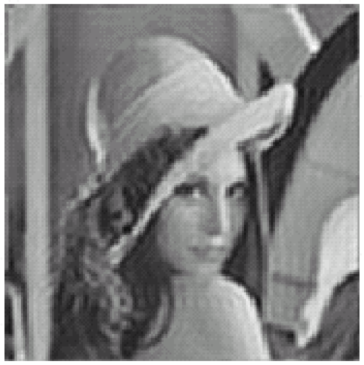
$W_1=W_2=10$



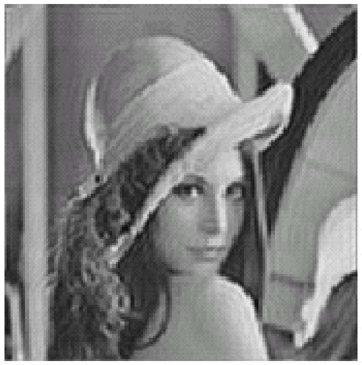
$W_1=W_2=20$



$W_1=W_2=30$

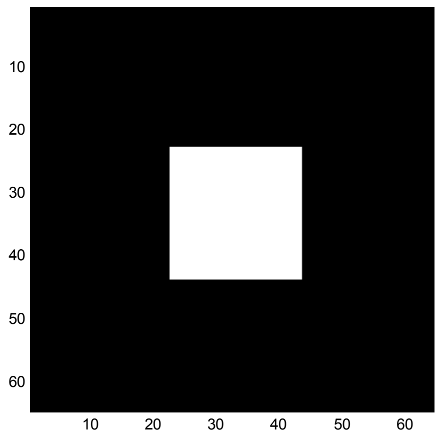


$W_1=W_2=40$



Courtesy of Onur Guleyuz

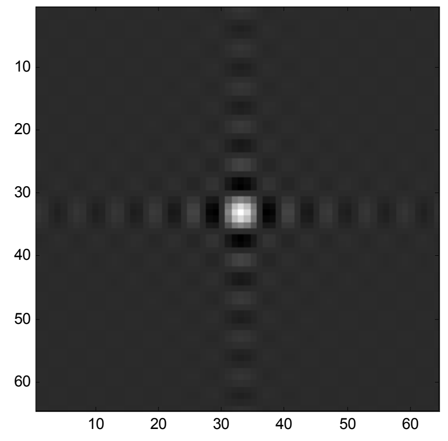
Spatial Filter Corresponding to Ideal LPF

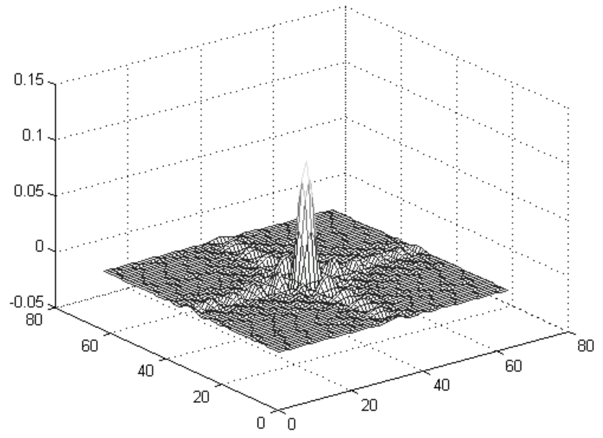


Ideal LPF with
W1=W2=10

$\xrightarrow{\hspace{2cm}}$

IDFT





To suppress the ringing artifacts, the window function should be smoothed by an appropriately designed smoothing function.

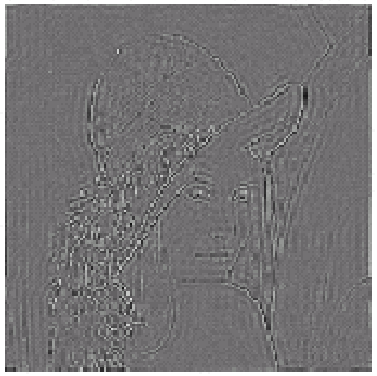
High Pass Filtering

- The high-pass filtered image can be thought of as the original image minus the low pass filtered image.
  - High-pass filtering by DFT windows:
    - If w(k,l) (*W*1 *W*2) is a low-pass DFT window, simply define a high-pass window *h*(*k, l*) by *h*(*k, l*) = 1 – w(k, l).
  - High-pass filtering in spatial domain:
    - If L is a low-pass filter of size W, simply define a high pass filter H via H(m,n) = δ(m,n) - L(m,n).

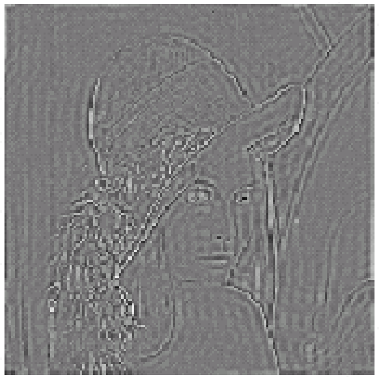
Courtesy of Onur Guleyuz

High Pass Filtering in DFT Domain

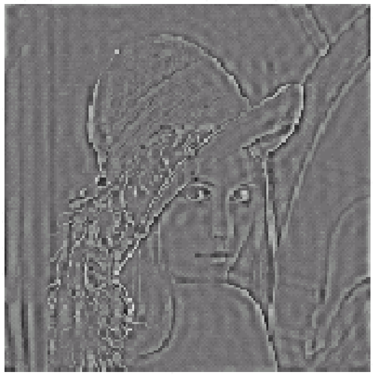
$W_1=W_2=40$ (nrml)



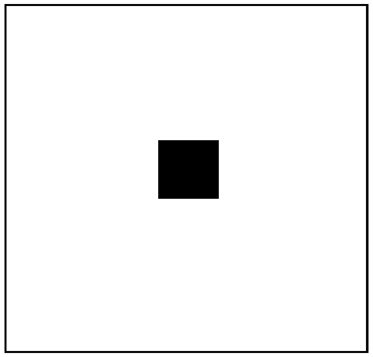
$W_1=W_2=30$ (nrml)



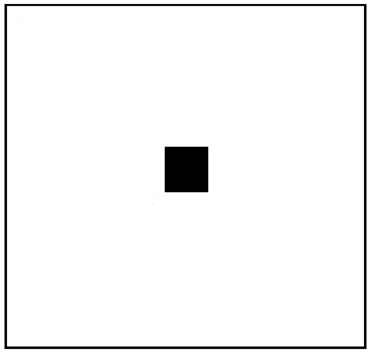
$W_1=W_2=20$ (nrml)



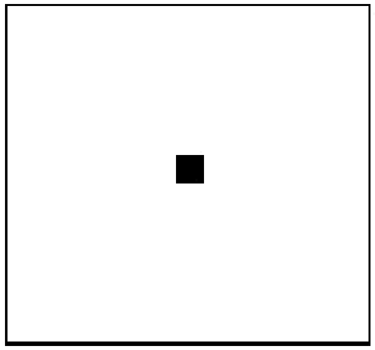
h (normalized)



h (normalized)



h (normalized)



DFT as Unitary Transform

- We have so far explained DFT as sampled version of DTFT for finite duration discrete time signals
- Finite duration discrete time signal = vector (1D) or image (2D)
- Transform = decomposition of vector or image into a weighted sum of some basis vectors or images.
- Unitary transform : basis functions are orthonormal to each other
- DFT uses a special type of basis functions

One Dimensional Linear Transform

- Let C^N represent the N dimensional complex space.
- Let $\mathbf{h}_0$, $\mathbf{h}_1$, …, $\mathbf{h}_{N-1}$ represent N linearly independent vectors in C^N.
- For any vector $\mathbf{f} \in C^N$,

$$\mathbf{f} = \sum_{k=0}^{N-1} t(k)\mathbf{h}_k = \mathbf{B}\mathbf{t},$$

$$where \quad \mathbf{B} = [\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{N-1}],\ \mathbf{t} = \begin{bmatrix} t(0) \\ t(1) \\ \vdots \\ t(N-1) \end{bmatrix}.$$

$$\Rightarrow \quad \mathbf{t} = \mathbf{B}^{-1}\mathbf{f} = \mathbf{A}\mathbf{f}$$

f and t form a transform pair

Inner Product

- Definition of inner product

$$<\mathbf{f}_1, \mathbf{f}_2> = \mathbf{f}_1^H \mathbf{f}_2 = \sum_{n=0}^{N-1} f_1^*(n) f_2(n)$$

- Orthogonal

$$<\mathbf{f}_1, \mathbf{f}_2> = 0$$

- Norm of a vector

$$\|\mathbf{f}\|^2 = <\mathbf{f}, \mathbf{f}> = \mathbf{f}^H \mathbf{f} = \sum_{n=0}^{N-1} | f(n) |^2$$

- Normalized: unit norm $\|\mathbf{f}\|^2 = 1$
- Orthonomal = orthogonal + normalized

Orthonormal Basis Vectors (OBV)

- $\{\mathbf{h}_k, k=0,\ldots N-1\}$ are OBV if

$$<\mathbf{h}_k,\mathbf{h}_l>=\delta_{k,l}=\begin{cases}1 & k=l\\ 0 & k\neq l\end{cases}$$

$$<\mathbf{h}_l,\mathbf{f}>=<\mathbf{h}_l,\sum_{k=0}^{N-1}t(k)\mathbf{h}_k>=\sum_{k=0}^{N-1}t(k)<\mathbf{h}_l,\mathbf{h}_k>=t(l)=\mathbf{h}_l{}^H\mathbf{f}$$

$$\mathbf{t}=\begin{bmatrix}\mathbf{h}_0{}^H\\ \mathbf{h}_1{}^H\\ \vdots\\ \mathbf{h}_{N-1}{}^H\end{bmatrix}\mathbf{f}=\mathbf{B}^H\mathbf{f}=\mathbf{A}\mathbf{f}$$

$$\mathbf{B}^{-1}=\mathbf{B}^H,\ or\ \mathbf{B}^H\mathbf{B}=\mathbf{B}\mathbf{B}^H=\mathbf{I}.$$

B is unitary

Definition of Unitary Transform

- Basis vectors are orthonormal
- $$t(k) = \langle \mathbf{h}_k, \mathbf{f} \rangle = \sum_{n=0}^{N-1} h_k(n)^* f(n),$$

$$\mathbf{t} = \begin{bmatrix} \mathbf{h}_0^H \\ \mathbf{h}_1^H \\ \vdots \\ \mathbf{h}_{N-1}^H \end{bmatrix} \mathbf{f} = \mathbf{B}^H \mathbf{f} = \mathbf{A}\mathbf{f}$$

- $$f(n) = \sum_{k=0}^{N-1} t(k) h_k(n),$$

$$\mathbf{f} = \sum_{k=0}^{N-1} t(k) \mathbf{h}_k = \begin{bmatrix} \mathbf{h}_0 & \mathbf{h}_1 & \cdots & \mathbf{h}_{N-1} \end{bmatrix} \mathbf{t} = \mathbf{B}\mathbf{t} = \mathbf{A}^H \mathbf{t}$$

Example of 1D Unitary Transform

$$\mathbf{h}_0 = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}, \mathbf{h}_1 = \begin{bmatrix} 1/2 \\ 1/2 \\ -1/2 \\ -1/2 \end{bmatrix}, \mathbf{h}_2 = \begin{bmatrix} 1/2 \\ -1/2 \\ -1/2 \\ 1/2 \end{bmatrix}, \mathbf{h}_3 = \begin{bmatrix} 1/2 \\ -1/2 \\ 1/2 \\ -1/2 \end{bmatrix},$$

$$\mathbf{f} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \Rightarrow \begin{cases} t_0 = 5 \\ t_1 = -2 \\ t_2 = 0 \\ t_3 = -1 \end{cases}$$

Property of Unitary Transform

- Energy preservation: ||**f**||=||**t**||.

Proof: $\|\mathbf{f}\| = \mathbf{f}^H \mathbf{f} = \mathbf{t}^H \mathbf{A}\mathbf{A}^H \mathbf{t} = \mathbf{t}^H \mathbf{t} = \|\mathbf{t}\|$

- Mean vector relation:

$$\boldsymbol{\mu}_t = \mathbf{A}\boldsymbol{\mu}_f, \quad \boldsymbol{\mu}_f = \mathbf{A}^H \boldsymbol{\mu}_t, \quad where$$
$$\boldsymbol{\mu}_f = E\{\mathbf{f}\}, \, and \; \boldsymbol{\mu}_t = E\{\mathbf{t}\}$$

- Covariance relation:

$$\mathbf{C}_t = \mathbf{A}\mathbf{C}_f\mathbf{A}^H, \mathbf{C}_f = \mathbf{A}^H\mathbf{C}_t\mathbf{A}, \quad where$$
$$\mathbf{C}_f = E\{(\mathbf{f}-\boldsymbol{\mu}_f)(\mathbf{f}-\boldsymbol{\mu}_f)^H\}, \quad \mathbf{C}_t = E\{(\mathbf{t}-\boldsymbol{\mu}_t)(\mathbf{t}-\boldsymbol{\mu}_t)^H\}$$

Proof:

$$\mathbf{t}-\boldsymbol{\mu}_t = \mathbf{A}(\mathbf{f}-\boldsymbol{\mu}_f) \quad \Rightarrow \quad \mathbf{C}_t = E\{\mathbf{A}(\mathbf{f}-\boldsymbol{\mu}_f)(\mathbf{f}-\boldsymbol{\mu}_f)^H\mathbf{A}^H\} = \mathbf{A}\mathbf{C}_f\mathbf{A}^H.$$

1D DFT as a Unitary Transform

$$F(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f(n) e^{-j2\pi \frac{kn}{N}}, \quad k = 0, 1, \ldots, N-1;$$

$$f(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} F(k) e^{j2\pi \frac{kn}{N}}, \quad n = 0, 1, \ldots, N-1.$$

$$h_k(n) = \frac{1}{\sqrt{N}} e^{j2\pi \frac{kn}{N}}, \quad or$$

$$\mathbf{h}_k = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 \\ e^{j2\pi \frac{k}{N}} \\ \vdots \\ e^{j2\pi \frac{(N-1)k}{N}} \end{bmatrix}, k = 0,1,\ldots, N-1.$$

Example: 1D DFT

$N = 2$ case : there are only two basis vectors :

$$\mathbf{h}_k = \frac{1}{\sqrt{2}}\begin{bmatrix} \exp(j2\pi \frac{k}{2}0) \\ \exp(j2\pi \frac{k}{2}1) \end{bmatrix}: \quad \mathbf{h}_0 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{h}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

if $\mathbf{f} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, determine t_0, t_1

Using $t_k = < \mathbf{h}_k, \mathbf{f} >$, we obtain

$$t_0 = \frac{1}{\sqrt{2}}\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) = \frac{1}{\sqrt{2}}(1*1+1*2) = \frac{3}{\sqrt{2}}, t_1 = \frac{1}{\sqrt{2}}\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) = \frac{1}{\sqrt{2}}(1*1-1*2) = \frac{-1}{\sqrt{2}}$$

$$\text{Verify}: t_0\mathbf{h}_0 + t_1\mathbf{h}_1 = \frac{3}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix} - \frac{1}{2}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \mathbf{f}$$

Another Example: 1D DFT

$$N = 4 \text{ case: using } \mathbf{h}_k = \frac{1}{2}\begin{bmatrix} \exp(j2\pi \frac{k}{4}0) \\ \exp(j2\pi \frac{k}{4}1) \\ \exp(j2\pi \frac{k}{4}2) \\ \exp(j2\pi \frac{k}{4}3) \end{bmatrix} \text{ yields: } \quad \mathbf{h}_0 = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}; \mathbf{h}_1 = \frac{1}{2}\begin{bmatrix} 1 \\ j \\ -1 \\ -j \end{bmatrix}; \mathbf{h}_2 = \frac{1}{2}\begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}; \mathbf{h}_3 = \frac{1}{2}\begin{bmatrix} 1 \\ -j \\ -1 \\ j \end{bmatrix}$$

$$\mathbf{f} = \begin{bmatrix} 2 \\ 4 \\ 5 \\ 3 \end{bmatrix} \Rightarrow \begin{array}{l} t_0 = \frac{1}{2}(2+4+5+3) = 7; \quad t_1 = \frac{1}{2}(2-4j-5+3j) = -\frac{1}{2}(3+j); \\ t_2 = \frac{1}{2}(2-4+5-3) = 0; \quad t_3 = \frac{1}{2}(2+4j-5-3j) = -\frac{1}{2}(3-j). \end{array}$$

$$\text{Verify: } t_0\mathbf{h}_0 + t_1\mathbf{h}_1 + t_2\mathbf{h}_2 + t_3\mathbf{h}_3 = \frac{1}{4}\begin{bmatrix} 14-(3+j)-(3-j) \\ 14-(3+j)j+(3-j)j \\ 14+(3+j)+(3-j) \\ 14+(3+j)j-(3-j)j \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 8 \\ 16 \\ 20 \\ 12 \end{bmatrix} = \mathbf{f}$$

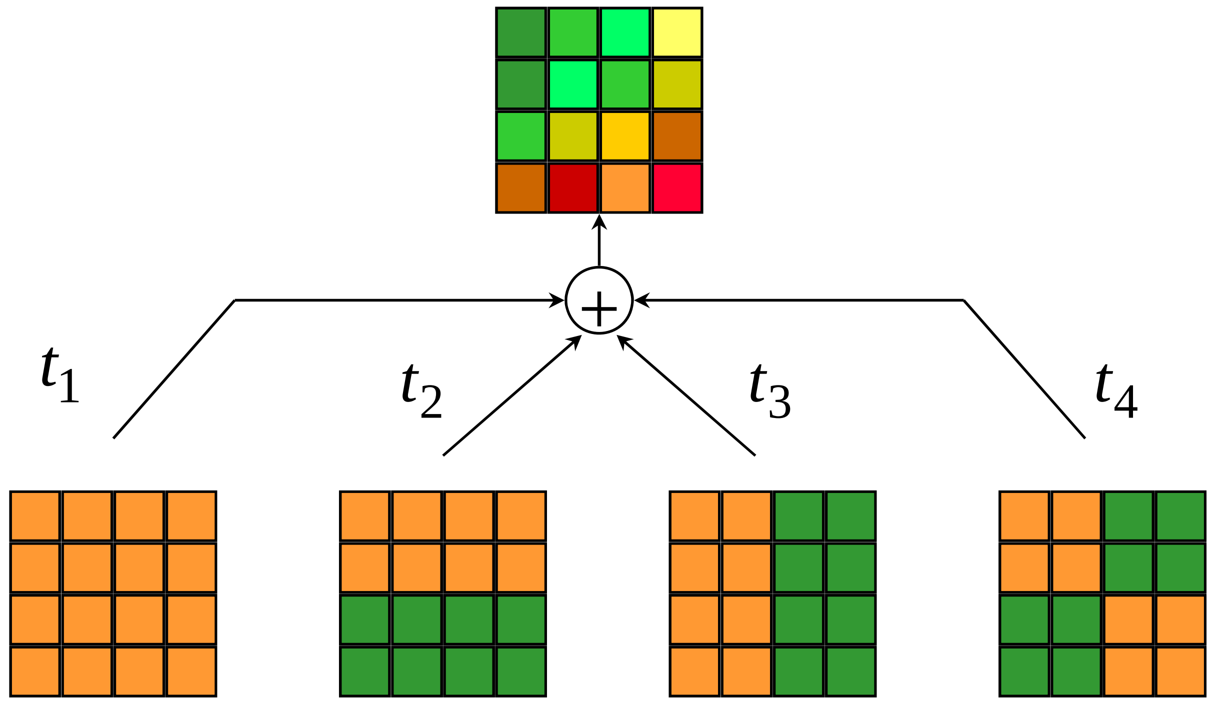
Two Dimensional Decomposition

- Decompose an MxN 2D matrix F=[F(m,n)] into a linear combination of some basic images, $H_{k,l}$=[$H_{k,l}$(m,n)], so that:

$$\mathbf{F} = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} T(k,l)\mathbf{H}_{k,l},$$

$$F(m,n) = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} T(k,l)H_{k,l}(m,n)$$

Graphical Interpretation



Inverse transform: Represent a vector (e.g. a block of image samples) as the superposition of some basis vectors (block patterns)
Forward transform: Determine the coefficients associated with each basis vector

Two Dimensional Inner Product

- Inner Product

$$<\mathbf{F}_1,\mathbf{F}_2>=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}F_1^*(m,n)F_2(m,n)$$

- Norm of a Matrix

$$\|\mathbf{F}\|=<\mathbf{F},\mathbf{F}>=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}|F(m,n)|^2$$

- A set of basis images {$\mathbf{H}_{k,l}$, k=0,1,…,M-1, l=0,1,…,N-1} is orthonormal if

$$<\mathbf{H}_{k,l},\mathbf{H}_{i,j}>=\delta_{k,i}\delta_{l,j}=\begin{cases}1, & if\ k=i,l=j\\ 0, & otherwise\end{cases}$$

Two Dimensional Unitary Transform

- $\{H_{k,l}\}$ is an orthonormal set of basis images
- Forward transform

$$T(k,l) = <\mathbf{H}_{k,l}, \mathbf{F}> = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} H_{k,l}^{*}(m,n)F(m,n)$$

- Inverse transform

$$F(m,n) = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} T(k,l)H_{k,l}(m,n), \quad or$$

$$\mathbf{F} = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} T(k,l)\mathbf{H}_{k,l}$$

Example of 2D Unitary Transform

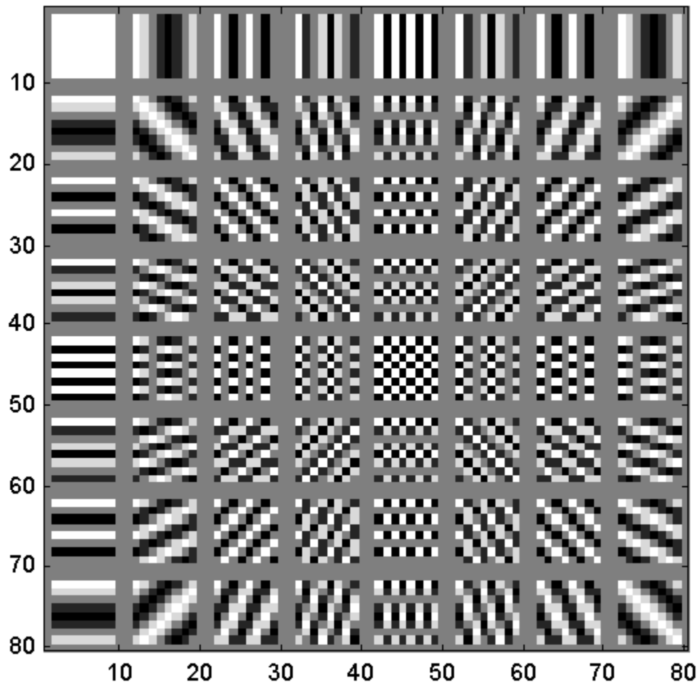
$$\mathbf{H}_{00}=\begin{bmatrix}1/2 & 1/2\\ 1/2 & 1/2\end{bmatrix}, \mathbf{H}_{01}=\begin{bmatrix}1/2 & 1/2\\ -1/2 & -1/2\end{bmatrix}, \mathbf{H}_{10}=\begin{bmatrix}1/2 & -1/2\\ 1/2 & -1/2\end{bmatrix}, \mathbf{H}_{11}=\begin{bmatrix}1/2 & -1/2\\ -1/2 & 1/2\end{bmatrix},$$

$$\mathbf{F}=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} \quad \Rightarrow \quad \begin{cases} T(0,0)=5 \\ T(0,1)=-2 \\ T(1,0)=-1 \\ T(1,1)=0 \end{cases}$$

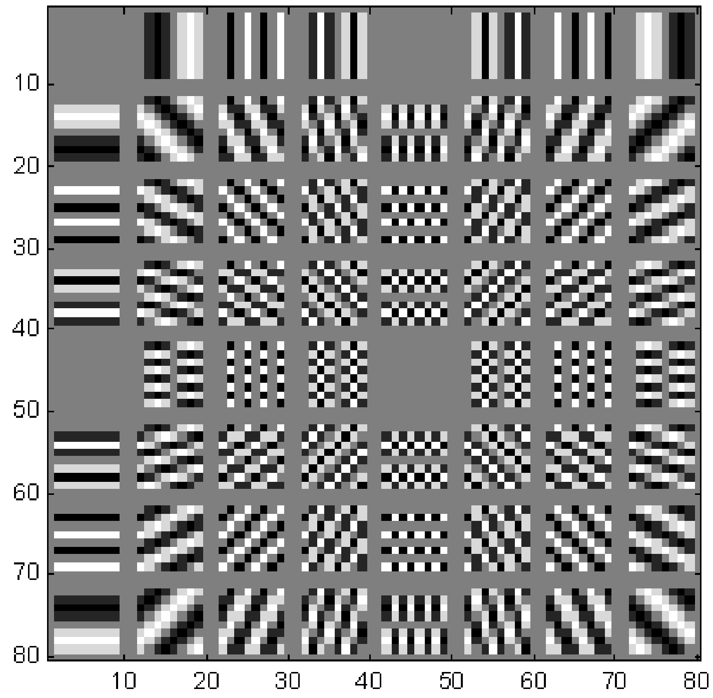
2D DFT as a Unitary Transform

$$f(m,n) = \frac{1}{\sqrt{MN}} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} F(k,l) e^{j2\pi(\frac{km}{M} + \frac{ln}{N})}, \quad m = 0,1,\ldots,M-1,\ n = 0,1,\ldots,N-1.$$

$$H_{k,l}(m,n) = \frac{1}{\sqrt{MN}} e^{j2\pi\left(\frac{km}{M} + \frac{ln}{N}\right)}$$



Basis Images for 8x8 DFT (Real Part)



Basis Images for 8x8 DFT (Imaginary Part)

Notice how the frequency content of the basis images change !

Separable Unitary Transform

- Let $\mathbf{h}_k$, k=0, 1, …, M-1 represent orthonormal basis vectors in C^M,
- Let $\mathbf{g}_l$, l=0, 1, …, N-1 represent orthonormal basis vectors in C^N,
- $\mathbf{H}_{k,l}=\mathbf{h}_k\mathbf{g}_l^T$, or $H_{k,l}(m,n)=h_k(m)g_l(n)$.
- Then $\mathbf{H}_{k,l}$ will form an orthonormal basis set in C^{MxN}.

Example of Separable Unitary Transform

- Example 1

$$\mathbf{h}_0 = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}, \mathbf{h}_1 = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}.$$

$$\mathbf{H}_{00} = \mathbf{h}_0\mathbf{h}_0^T = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix} \qquad \mathbf{H}_{01} = \mathbf{h}_0\mathbf{h}_1^T = \begin{bmatrix} 1/2 & -1/2 \\ 1/2 & -1/2 \end{bmatrix}$$

$$\mathbf{H}_{10} = \mathbf{h}_1\mathbf{h}_0^T = \begin{bmatrix} 1/2 & 1/2 \\ -1/2 & -1/2 \end{bmatrix} \quad \mathbf{H}_{11} = \mathbf{h}_1\mathbf{h}_1^T = \begin{bmatrix} 1/2 & -1/2 \\ -1/2 & 1/2 \end{bmatrix}$$

- 2D DFT

$$H_{k,l}(m,n) = \frac{1}{\sqrt{MN}} e^{j2\pi\left(\frac{km}{M}+\frac{ln}{N}\right)},$$

$$h_k(m) = \frac{1}{\sqrt{M}} e^{j2\pi\frac{km}{M}}, \quad g_l(n) = \frac{1}{\sqrt{N}} e^{j2\pi\frac{ln}{N}}$$

Example: 4x4 DFT

$$\text{Recall the 1D DFT basis are: } \mathbf{h}_0 = \frac{1}{2}\begin{bmatrix}1\\1\\1\\1\end{bmatrix}; \mathbf{h}_1 = \frac{1}{2}\begin{bmatrix}1\\j\\-1\\-j\end{bmatrix}; \mathbf{h}_2 = \frac{1}{2}\begin{bmatrix}1\\-1\\1\\-1\end{bmatrix}; \mathbf{h}_3 = \frac{1}{2}\begin{bmatrix}1\\-j\\-1\\j\end{bmatrix}$$

$$\text{using } \mathbf{H}_{k,l} = \mathbf{h}_k (\mathbf{h}_l)^T \text{ yields:}$$

$$\mathbf{H}_{0,0} = \frac{1}{4}\begin{bmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{bmatrix}, \quad \mathbf{H}_{0,1} = \frac{1}{4}\begin{bmatrix}1&j&-1&-j\\1&j&-1&-j\\1&j&-1&-j\\1&j&-1&-j\end{bmatrix}, \quad \mathbf{H}_{0,2} = \frac{1}{4}\begin{bmatrix}1&-1&1&-1\\1&-1&1&-1\\1&-1&1&-1\\1&-1&1&-1\end{bmatrix}, \quad \mathbf{H}_{0,3} = \frac{1}{4}\begin{bmatrix}1&-j&-1&j\\1&-j&-1&j\\1&-j&-1&j\\1&-j&-1&j\end{bmatrix}$$

$$\mathbf{H}_{1,0} = \frac{1}{4}\begin{bmatrix}1&1&1&1\\j&j&j&j\\-1&-1&-1&-1\\-j&-j&-j&j\end{bmatrix}, \mathbf{H}_{1,1} = \frac{1}{4}\begin{bmatrix}1&j&-1&-j\\j&-1&-j&1\\-1&-j&1&j\\-j&1&j&-1\end{bmatrix}, \mathbf{H}_{1,2} = \frac{1}{4}\begin{bmatrix}1&-1&1&-1\\j&-j&j&-j\\-1&1&-1&1\\-j&j&-j&j\end{bmatrix}, \mathbf{H}_{1,3} = \frac{1}{4}\begin{bmatrix}1&-j&-1&j\\j&1&-j&-1\\-1&j&1&-j\\-j&-1&j&1\end{bmatrix}$$

$$\mathrm{H}_{2,0} = \frac{1}{4}\begin{bmatrix}1&1&1&1\\-1&-1&-1&-1\\1&1&1&1\\-1&-1&-1&-1\end{bmatrix}, \mathbf{H}_{2,1} = \frac{1}{4}\begin{bmatrix}1&j&-1&-j\\-1&-j&1&j\\1&j&-1&-j\\-1&-j&1&j\end{bmatrix}, \mathbf{H}_{2,2} = \frac{1}{4}\begin{bmatrix}1&-1&1&-1\\-1&1&-1&1\\1&-1&1&-1\\-1&1&-1&1\end{bmatrix}, \mathbf{H}_{2,3} = \frac{1}{4}\begin{bmatrix}1&-j&-1&j\\-1&j&1&-j\\1&-j&-1&j\\-1&j&1&-j\end{bmatrix}$$

$$\mathbf{H}_{3,0} = \frac{1}{4}\begin{bmatrix}1&1&1&1\\-j&-j&-j&-j\\-1&-1&-1&-1\\j&j&j&j\end{bmatrix}, \mathbf{H}_{3,1} = \frac{1}{4}\begin{bmatrix}1&j&-1&-j\\-j&1&j&-1\\-1&-j&1&j\\j&-1&-j&1\end{bmatrix}, \mathbf{H}_{3,2} = \frac{1}{4}\begin{bmatrix}1&-1&1&-1\\-j&j&-j&j\\-1&1&-1&1\\j&-j&j&-j\end{bmatrix}, \mathbf{H}_{3,3} = \frac{1}{4}\begin{bmatrix}1&-j&-1&j\\-j&-1&j&1\\-1&j&1&-j\\j&1&-j&-1\end{bmatrix}$$

Example: 4x4 DFT

For $\mathbf{F} = \begin{bmatrix} 1 & 2 & 2 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 1 & 2 & 1 \\ 1 & 2 & 2 & -1 \end{bmatrix}$, compute $T_{k,l}$

Using $T_{k,l} = <\mathbf{H}_{k,l}, \mathbf{F}>$ yields, e.g,

$$T_{0,0} = <\mathbf{H}_{0,0}, \mathbf{F}> = \left(\frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 2 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 1 & 2 & 1 \\ 1 & 2 & 2 & -1 \end{bmatrix} \right) = \frac{1}{4}(1+2+2+0+0+1+3+1+0+1+2+1+1+2+2-1) = \frac{18}{4}$$

$$T_{2,3} = <\mathbf{H}_{2,3}, \mathbf{F}> = \left(\frac{1}{4} \begin{bmatrix} 1 & -j & -1 & j \\ -1 & j & 1 & -j \\ 1 & -j & -1 & j \\ -1 & j & 1 & -j \end{bmatrix}^{*}, \begin{bmatrix} 1 & 2 & 2 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 1 & 2 & 1 \\ 1 & 2 & 2 & -1 \end{bmatrix} \right) = \frac{1}{4}(1+2j-2-j+3+j+j-2-j-1-2j+2-j) = \frac{1}{4}(1-j)$$

Property of Separable Transform

- When the transform is separable, we can perform the 2D transform separately.
  - First, do 1D transform for each row using basis vectors g_l,
  - Second, do 1D transform for each column of the intermediate image using basis vectors h_k.
  - Proof:

$$T(k,l) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} H_{k,l}^{*}(m,n)F(m,n) = \sum_{m=0}^{M-1} h_k^{*}(m)\sum_{n=0}^{N-1} g_l^{*}(n)F(m,n) = \sum_{m=0}^{M-1} h_k^{*}(m)U(m,l)$$

Computational Savings of Separable Transform

- A MxN transform takes about MxN calculations for each transform value, and the total number of calculation is M^2N^2 (N^4 if M=N)
- If the transform is separable
  1. Calculate M N-point 1D transform, each requiring N^2 calculations
  2. Calculate N M-point 1D transform, each requiring M^2 calculations
  3. Overall: MN^2+NM^2 ($2N^3$, if M=N)

Other Transforms

- Discrete cosine transform (DCT)
  - Basis functions are real
  - Can be thought of as “real version” of DFT
  - We will discuss DCT and its application in image compression (JPEG) later
- Discrete sine transform (DST)
- Hadamard transform
- Walsh transform
- Haar transform
- Slant transform
- Wavelet transform
  - Multi-resolution transform
  - We will discuss wavelet transform and its application in image compression (JPEG2000) later

Homework

1. Find the NxN point DFT of the following 2D image f(m, n), $0 \le m, n \le N$:

   a) f(m, n) = 1, n = 0; f(m, n) = 0; n ≠ 0.

   b) f(m, n) = 1, m = 0; f(m, n) = 0, m ≠ 0.

   c) f(m, n) = 1, m = n; f(m, n) = 0, otherwise.

   d) f(m, n) = 1, m = N – 1- n; f(m, n) = 0, otherwise.

   From the result, what can you say about the relation between the directionality of an image with that of its DFT?

2. One can use the DFT algorithms to compute the linear convolution of an image F(m, n) with a filter H(m, n). Let the convolved image be denoted by Y(m, n).

   i) Suppose the image size is 256x256 and the filter size is 11x11: What is the required size of the DFT to obtain the convolution of these two? Explain the exact steps to obtain the convolution result.

   ii) Suppose we use a 256x256 point DFT algorithm for F(m, n) and H(m, n), and obtain Z(m, n) as follows: Z = IDFT (DFT(X).*DFT(H)). The DFT and IDFT in the above equation are both 256x256 points. For what values of (m, n) does Z(m, n) equal Y(m, n) ?

Homework (cnt'd)

3. Show that the following vectors form an orthonormal basis set for any value of θ:

$$\mathbf{h}_0 = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}, \mathbf{h}_1 = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$$

Determine the corresponding matrix **A** and **B** for the forward and inverse transform described by

$$\text{Forward transform: } \mathbf{t} = \mathbf{A}\mathbf{f}; \ \text{Inverse transform: } \mathbf{f} = \mathbf{B}\mathbf{t}.$$

4. Consider a zero mean random vector u with covariance matrix

$$C_u = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$$

From the class of unitary transforms given in Prob. 3, determine the value of θ for which the components of transformed vector v will be uncorrelated (i.e., the covariance matrix of the transformed vector is a diagonal matrix).

5. The basis images of the 2D-DCT are:

$$H_{00} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}; \quad H_{01} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}; \quad H_{10} = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}; \quad H_{11} = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix};$$

Calculate the transform of the image: $F = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$

Find the reconstructed image $\hat{F}$ obtained with the 2 largest coefficients (in magnitude). Calculate the error between the original and reconstructed images defined by $E = \sum_{m,n} [F(m,n) - \hat{F}(m,n)]^2$ You should see that $E = \sum_{k,l:t(k,l) \text{ are omitted}} |t(k,l)|^2$ where $t(k, l)$ represents the transform coefficient associated with the basis image H_{kl}.

Computer Assignment

1. Perform 2D DFT (using the fft2 function in Matlab) on several test images of your choice (sample images are available at the Matlab image toolbox directory). Display the magnitude of the DFT image with and without shifting (using fftshift) and with and without logarithmic mapping, to see the effect of shifting and logarithmic mapping. Include the printouts in your submission. Also, examine in which frequency range the DFT coefficients have large magnitudes and explain why. Note that you may want to work on a small portion of your image (say 256x256 or less), to save computation time.

   Note that if your image is an RGB image, you should convert it to a grayscale image using "rgb2gray" function and apply the above operations on the grayscale image only. If the original image is an index image, you should not apply DFT to the index image directly, rather you should derive the grayscale image from it, using the "ind2gray" function.

2. Write a program (in Matlab or C) which filters an image by zeroing out certain DFT coefficients. The program consists of 3 steps:

   1) performing 2D DFT (you can use the "fft2" function in Matlab);

   2) zeroing out the coefficients at certain frequencies (see below);

   3) performing inverse DFT to get back a filtered image. Truncate or scale the image properly such that its range is between 0 and 255.

   For part 2, try the following two types of filters:

   (a) let $F(k, l) = 0$ for $TN < \{k, l\} < (1 - T)N$, $T = 1/4, 1/8$ (i.e., low-pass filtering);

   (b) let $F(k, l) = 0$ for the following regions i) $0 \le \{k \text{ and } l\} \le TN$; ii) $0 <= k <= TN$, and $(1 - T)N \le l \le N - 1$; iii) $(1 - T)N \le k \le N - 1$ and $0 \le \{l\} \le TN$; iv) $(1 - T)N \le k$ and $l \le N - 1$; $T = 1/4, 1/8$

   Compare the original and processed images. Comment on the function of the two types of filters.

Reading

- R. Gonzalez, “Digital Image Processing,” Chap.4, except Sec. 4.5.
- A. Jain, “Fundamentals of Digital Image Processing,” Chapter 5.