

Chapter 7: The Fourier Transform

CENGAGE
Learning

1D Discrete Fourier Transform

- Definition of the One-Dimensional DFT

$$\mathbf{f} = [f_0, f_1, f_2, \dots, f_{N-1}] \quad \mathbf{F} = [F_0, F_1, F_2, \dots, F_{N-1}],$$

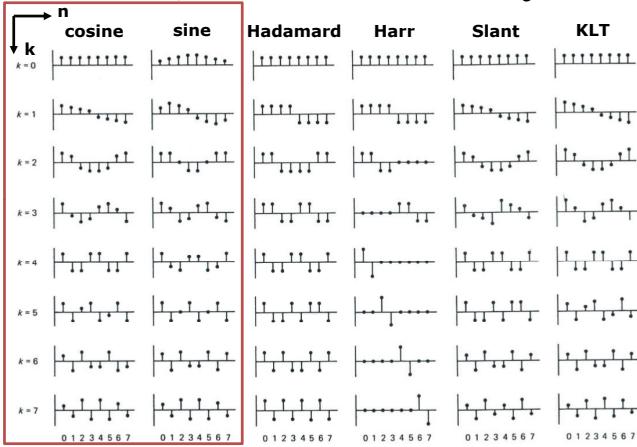
$$F_u = \frac{1}{N} \sum_{x=0}^{N-1} \exp \left[-2\pi i \frac{xu}{N} \right] f_x \quad (7.2)$$

- can be expressed as a matrix multiplication $\mathbf{F} = \mathcal{F}\mathbf{f}$,

$$\mathcal{F}_{m,n} = \frac{1}{N} \exp \left[-2\pi i \frac{mn}{N} \right] \quad \mathcal{F}_{m,n} = \frac{1}{N} \omega^{mn} \quad N \times N \text{ matrix}$$

$$\omega = \exp \left[\frac{-2\pi i}{N} \right] \quad \mathcal{F} = \frac{1}{N} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 & \omega^4 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \omega^8 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \omega^{12} & \cdots & \omega^{3(N-1)} \\ 1 & \omega^4 & \omega^8 & \omega^{12} & \omega^{16} & \cdots & \omega^{4(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \omega^{4(N-1)} & \cdots & \omega^{(N-1)^2} \end{bmatrix}$$

Examples of 1-D Kernel $k(n, k) = e^{-j \frac{2\pi nk}{N}}$



Introduction

- The Fourier transform allows us to perform tasks that would be impossible to perform any other way.
- It is **more efficient** to use the Fourier transform than a spatial filter for a large filter.
- The Fourier transform also allows us to **isolate** and process particular image **frequencies**.

CENGAGE
Learning

1D Discrete Fourier Transform

Example: Suppose $\mathbf{f} = [1, 2, 3, 4]$ so that $N = 4$.

Find DFT coefficients \mathbf{F} of \mathbf{f} .

$$\begin{aligned} \omega &= \exp \left[\frac{-2\pi i}{4} \right] \\ &= \exp \left[\frac{-\pi i}{2} \right] \quad \Rightarrow \quad \mathcal{F} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & (-i)^2 & (-i)^3 \\ 1 & (-i)^2 & (-i)^4 & (-i)^6 \\ 1 & (-i)^3 & (-i)^6 & (-i)^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \\ &= \cos \left(-\frac{\pi}{2} \right) + i \sin \left(-\frac{\pi}{2} \right) \\ &= -i. \end{aligned}$$

$$\mathbf{F} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 10 \\ -2+2i \\ -2 \\ -2-2i \end{bmatrix}$$

CENGAGE
Learning

THE INVERSE DFT

$$F_u = \frac{1}{N} \sum_{x=0}^{N-1} \exp \left[-2\pi i \frac{xu}{N} \right] f_x. \quad (7.2)$$

$$f_x = \sum_{u=0}^{N-1} \exp \left[2\pi i \frac{xu}{N} \right] F_u \quad (7.3)$$

✓ If you compare Equation (7.3) with Equation 7.2 you will see that there are really a few differences:

- There is **no scaling factor** $1/N$
- The **sign** inside the exponential function has been changed to positive.
- The **index** of the sum is u , instead of x

CENGAGE
Learning

IDFT

$$f = \mathcal{F}^{-1}F$$

$$\mathcal{F}^{-1} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \bar{\omega}^1 & \bar{\omega}^2 & \bar{\omega}^3 & \bar{\omega}^4 & \dots & \bar{\omega}^{N-1} \\ 1 & \bar{\omega}^2 & \bar{\omega}^4 & \bar{\omega}^6 & \bar{\omega}^8 & \dots & \bar{\omega}^{2(N-1)} \\ 1 & \bar{\omega}^3 & \bar{\omega}^6 & \bar{\omega}^9 & \bar{\omega}^{12} & \dots & \bar{\omega}^{3(N-1)} \\ 1 & \bar{\omega}^4 & \bar{\omega}^8 & \bar{\omega}^{12} & \bar{\omega}^{16} & \dots & \bar{\omega}^{4(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 1 & \bar{\omega}^{N-1} & \bar{\omega}^{2(N-1)} & \bar{\omega}^{3(N-1)} & \bar{\omega}^{4(N-1)} & \dots & \bar{\omega}^{(N-1)^2} \end{bmatrix}$$

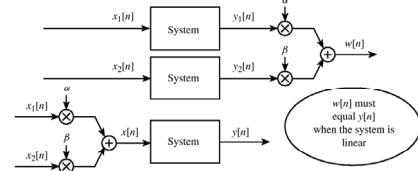
$$\bar{\omega} = \frac{1}{\omega} = \exp \left[\frac{2\pi i}{N} \right]$$

```
a =
1   2   3   4   5   6
>> fft(a')
ans =
21.0000
-3.0000 + 5.1962i
-3.0000 + 1.7321i
-3.0000 - 1.7321i
-3.0000 - 5.1962i
```

CENGAGE Learning

Properties of DFT: LINEARITY

- ✓ This is a direct consequence of the definition of the DFT as a matrix product.
- ✓ Suppose f and g are two vectors of equal length, and p and q are scalars, with $h = pf + qg$.
- ✓ If F , G , and H are the DFT's of f , g , and h , respectively, we have $H = pF + qG$



CENGAGE Learning

LINEAR SYSTEM EXAMPLES

Examples: Let $\alpha=1$, $\beta=2$

- $x_1[n] = \{2 3 5\}$ ($-1 \leq n \leq 1$), otherwise $x[n] = 0$
- $x_2[n] = \{3 2 1\}$ ($-1 \leq n \leq 1$), otherwise $x[n] = 0$
- $Q1: y[n] = x[n]^2$

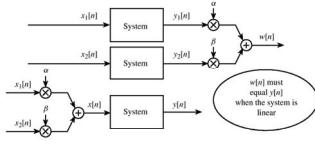


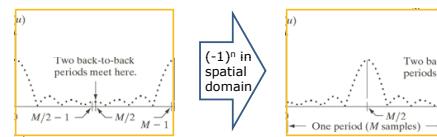
Figure 5.17 Testing linearity by checking the interchange of operations.

- $Q2: y[n] = x[-n]$

CENGAGE Learning

Properties of DFT: SHIFTING

- ✓ Suppose we multiply each element x_n of a vector x by $(-1)^n$. In other words, we change the sign of every second element.
- ✓ Let the resulting vector be denoted x' . The DFT X' of x' is equal to the DFT X of x with the swapping of the left and right halves



CENGAGE Learning

Properties of DFT: Shifting

```
>> x = [2 3 4 5 6 7 8 1];
>> x1=(-1).^(0:7).*x
x1 =
2   -3   4   -5   6   -7   8   -1
>> X=fft(x1')
X1 =
36.0000
-9.6569 + 4.0000i
-4.0000 + 4.0000i
-4.0000 - 4.0000i
1.6569 - 4.0000i
4.0000
1.6569 + 4.0000i
-4.0000 + 4.0000i
-9.6569 - 4.0000i
```

```
>> X1=fft(x1')
X1 =
4.0000
1.6569 + 4.0000i
-4.0000 + 4.0000i
-9.6569 - 4.0000i
36.0000
-9.6569 + 4.0000i
-4.0000 - 4.0000i
1.6569 - 4.0000i
```

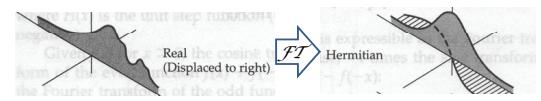
Notice that the first four elements of X are the last four elements of $X1$ and vice versa.

CENGAGE Learning

Properties of DFT

• CONJUGATE SYMMETRY

- For real-valued signal $f(x,y)$, $F(u) = F^*(-u)$



• CONVOLUTION THEOREM

- convolution in the spatial domain is equivalent to multiplication in the frequency domain.

$$x * h \longleftrightarrow XH, \quad xh \longleftrightarrow X^*H$$

$$x(n) \xrightarrow{h(n)} x(n)*h(n)$$

CENGAGE Learning

7.4 Properties of the One-Dimensional DFT

• THE FAST FOURIER TRANSFORM Appendix B

TABLE 7.1 Comparison of FFT and direct arithmetic.

| 2^n | Direct Arithmetic | FFT | Increase in Speed |
|-------|-------------------|-------|-------------------|
| 4 | 16 | 8 | 2.0 |
| 8 | 84 | 24 | 2.67 |
| 16 | 256 | 64 | 4.0 |
| 32 | 1024 | 160 | 6.4 |
| 64 | 4096 | 384 | 10.67 |
| 128 | 16384 | 896 | 18.3 |
| 256 | 65536 | 2048 | 32.0 |
| 512 | 262144 | 4608 | 56.9 |
| 1024 | 1048576 | 10240 | 102.4 |

CENGAGE Learning

2D Discrete Fourier Transform

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi \left(\frac{ux}{M} + \frac{vy}{N} \right)} \Leftrightarrow f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{j2\pi \left(\frac{ux}{M} + \frac{vy}{N} \right)}$$

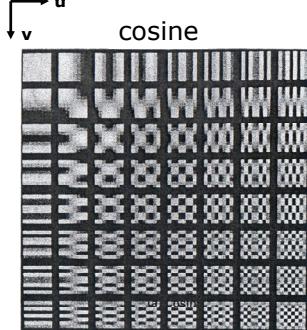
$f(x, y)$: pixel values of an image of size $M \times N$
 u, v : discrete variables

$$F(0, 0) = ?$$

$$\begin{aligned} F(0, 0) &= \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi \left(\frac{0x}{M} + \frac{0y}{N} \right)} \\ &= \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \\ &= \bar{f}(x, y) \end{aligned}$$

CENGAGE Learning

8x8 2-D FT Coefficients



$$e^{j2\pi(ux+vy)} = \cos[2\pi(ux + vy)] + j \sin[2\pi(ux + vy)]$$

CENGAGE Learning

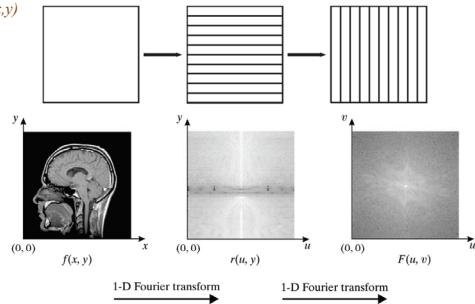
sine

Properties of the 2D DFT: Separability

- $F(u, v)$ of $f(x, y)$ can be calculated using two 1-D FT.

$$\int_{-\infty}^{\infty} r(u, y) e^{-j2\pi uy} dy = \left[\int_{-\infty}^{\infty} f(x, y) e^{-j2\pi ux} dx \right] e^{-j2\pi uy} dy = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j2\pi(ux+uy)} dxdy = F(u, v)$$

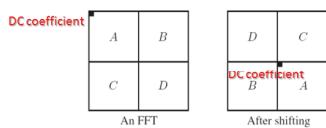
FT of $f(x, y)$
in x axis



CENGAGE Learning

Properties of the 2D DFT

- SHIFTING** $f_p(x, y)$ by $(-1)^{x+y}$ before transform



- CONJUGATE SYMMETRY**

$$F(u, v) = F^*(-u, -v)$$

| | | | |
|-------|-------|-------|-------|
| | a | | a^* |
| b^* | B^* | d^* | A^* |
| c | | | c^* |
| b | A | d | B |

CENGAGE Learning

How to Apply Convolution Theorem In Reality

- Suppose we wish to convolve an image M with a spatial filter S

- Pad S with zeroes so that it is the same size as M ; denote this padded result by S' : *make the length of both data to fit 2^n for best performance of fft*
- Form the DFTs of both M and S' to obtain $\mathcal{F}(M)$ and $\mathcal{F}(S')$.
- Form the element-by-element product of these two transforms: $\mathcal{F}(M)\mathcal{F}(S')$
- Take the inverse transform of the result: $\mathcal{F}^{-1}\{\mathcal{F}(M)\mathcal{F}(S')\}$

CENGAGE Learning

Fourier Transforms in MATLAB

- ✓ **fft**: takes the DFT of a vector
- ✓ **ifft**: takes the inverse DFT of a vector
- ✓ **fft2**: takes the DFT of a matrix
- ✓ **ifft2**: takes the inverse DFT of a matrix
- ✓ **fftshift**: shifts a transform



Fourier Transforms in MATLAB

e.g.

```
>> a=ones(8);
>> fft2(a)
```

```
ans =
64 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

Note that the DC coefficient is indeed the sum of all the matrix values



7.6 Fourier Transforms in MATLAB

e.g.

```
>> a = [100 200; 100 200];
>> a = repmat(a,4,4)
a: input
ans =
100 200 100 200 100 200 100 200
100 200 100 200 100 200 100 200
100 200 100 200 100 200 100 200
100 200 100 200 100 200 100 200
100 200 100 200 100 200 100 200
100 200 100 200 100 200 100 200
100 200 100 200 100 200 100 200
100 200 100 200 100 200 100 200
df:
DFT of a
ans =
9600 0 0 0 -3200 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```



7.6 Fourier Transforms in MATLAB

e.g.

```
>> a = [zeros(8,4) ones(8,4)]
```

```
a =
0 0 0 0 1 1 1 1
0 0 0 0 1 1 1 1
0 0 0 0 1 1 1 1
0 0 0 0 1 1 1 1
0 0 0 0 1 1 1 1
0 0 0 0 1 1 1 1
0 0 0 0 1 1 1 1
0 0 0 0 1 1 1 1
```

a: input

```
>> af=fftshift(fft2(a));
>> round(abs(af))
```

```
ans =
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 9 0 21 32 21 0 9
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

df:
DFT then
shifted



Showing the Fourier Spectrum in Matlab: fftshow

```
function fftshow(f,type)
% Usage: FFTSHOW(F,TYPE)
%
% Displays the fft matrix F using imshow, where TYPE must be one of
% 'abs' or 'log'. If TYPE='abs', then abs(f) is displayed; if
% TYPE='log' then log(1+abs(f)) is displayed. If TYPE is omitted, then
% 'log' is chosen as a default.
%
% Example:
% c=imread('cameraman.tif');
% cf=fftshift(fft2(c));
% fftshow(cf,'abs')
%
if nargin<2,
    type='log';
end
if (type=='log')
    f1 = log(1+abs(f));
    fm = max(f1(:));
    imshow(log2uint8(f1/fm))
else
    (type=='abs')
    fa=abs(f);
    fm=max(fa(:));
    imshow(fa/fm)
else
    error('TYPE must be abs or log.');
end
```

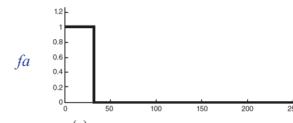
FIGURE 7.9 A function to display a Fourier transform.



DFT Example

```
>> fa = [ones(1, 32), zeros(1, 224)];
>> fb = [ones(1, 64), zeros(1, 192)];
>> figure, plot(fa)
```

```
>> af = fft(fa);
>> aft = fftshift(af);
>> figure, plot(abs(aft))
```



(a)

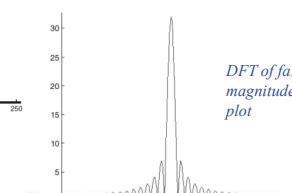


FIGURE 7.11 A 1-D single edge signal f_a and its FFT.



FIGURE 7.12

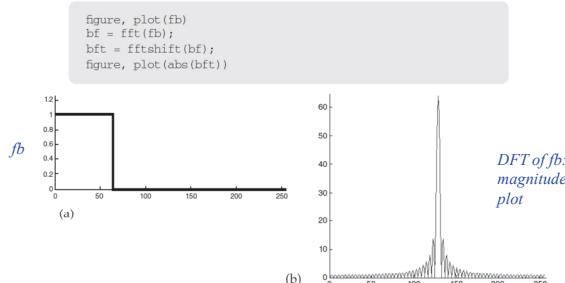


FIGURE 7.12 A 1-D single edge signal fb and its FFT.

CENGAGE
Learning

2D DFT Example: Vertical Edge



FIGURE 7.10 A single edge and its DFT.

CENGAGE
Learning

2D DFT Example: Square Edge

```
>> a=zeros(256,256);
>> a(78:178,78:178)=1;
>> imshow(a)
>> af=fftshift(fft2(a));
>> figure,fftshow(af,'abs')
```



FIGURE 7.13 A box and its DFT.

CENGAGE
Learning

2D DFT Example: Diagonal Edge

```
>> [x,y]=meshgrid(1:256,1:256);
>> b=(x+y<329)&(x+y>182)&(x-y>-67)&(x-y<73);
>> imshow(b)
>> bf=fftshift(fft2(b));
>> figure,fftshow(bf,'log')
```



FIGURE 7.14 A rotated box and its DFT.

CENGAGE
Learning

2D DFT Example: Circular Edge

```
>> [x,y]=meshgrid(-128:127,-128:127);
>> z=sqrt(x.^2+y.^2);
>> c=(z<15);

>> cf=fftshift(fft2(c));
>> fftshow(cf,'log')
```

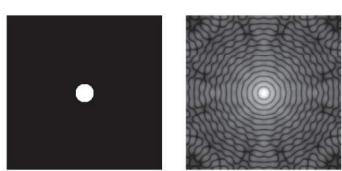
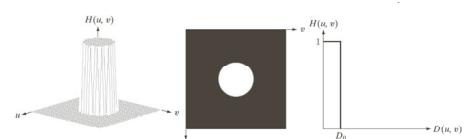


FIGURE 7.15 (a) A circle and (b) its DFT.

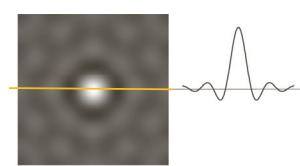
CENGAGE
Learning

Filtering in the Frequency Domain: Ideal LPF

$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \leq D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases}$$



ideal LPF in spatial
domain



CENGAGE
Learning

Ideal LPF Example (contd.)

```
>> cm=imread('cameraman.tif');
>> cf=fftshift(fft2(cm));
>> figure, fftshow(cf,'log')
>> cfl=cf.*cideal LPF
>> figure, fftshow(cfl,'log') % shown in the next page
```



FIGURE 7.16 The "cameraman" image and its DFT.

CENGAGE
Learning

Ideal LPF Example

```
>> cfli=ifft2(cfl);
>> figure, fftshow(cfli,'abs')
```

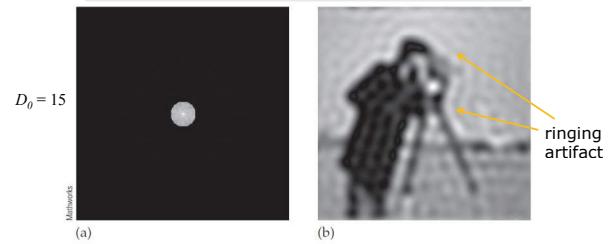
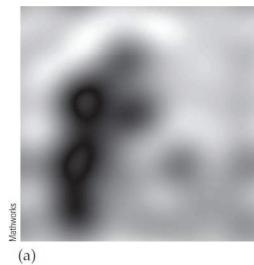


FIGURE 7.17 Applying ideal low-pass filtering. (a) Ideal filtering on the DFT.
(b) After inversion.

CENGAGE
Learning

Ideal LPF

$D_0 = 5$



(a)

$D_0 = 30$



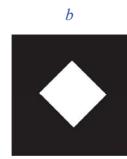
(b)

FIGURE 7.18 Ideal low-pass filtering with different cutoffs. (a) Cutoff of 5. (b) Cutoff of 30.

CENGAGE
Learning

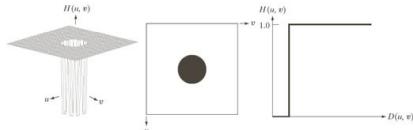
7.8 Filtering in the Frequency Domain

```
>> cfl = cf.*b;
>> cfli = ifft2(cfl);
>> figure, fftshow(cfli, 'abs')
```



CENGAGE
Learning

Filtering in the Frequency Domain: Ideal HPF

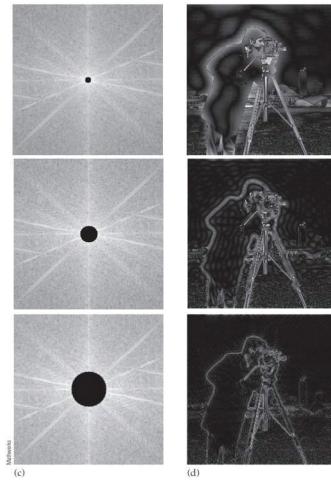


```
>> [x,y]=meshgrid(-128:127,-128:127);
>> z=sqrt(x.^2+y.^2);
>> c=(z>15);
```

```
>> cfh=cf.*c;
>> figure,fftshow(cfh,'log')
```

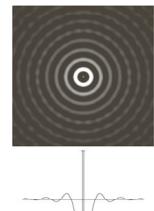
```
>> cfhi=ifft2(cfh);
>> figure,fftshow(cfhi,'abs')
```

CENGAGE
Learning



Ideal HPF

$$H(u,v) = \begin{cases} 0 & \text{if } D(u,v) \leq D_0 \\ 1 & \text{if } D(u,v) > D_0 \end{cases}$$



ringing artifact

CENGAGE
Learning

Butterworth Filtering

- ✓ Ideal filter simply **cuts off** the Fourier coefficients at the specified distance from the center.
- ✓ It has the disadvantage of introducing **unwanted artifacts (ringing)** into the result
- ✓ One way of avoiding ringing artifacts is to use as a filter matrix, a circle with a less sharp cutoff.
- ✓ Butterworth and Gaussian filters

 CENGAGE
Learning

Butterworth in Matlab

```
function out=lbutter(im,d,n)
% LBUTTER(IM,D,N) creates a low-pass Butterworth filter
% of the same size as image IM, with cutoff D, and order N
%
% Use:
%   x=imread('cameraman.tif');
%   l=lbutter(x,25,2);
%
height=size(im,1);
width=size(im,2);
[x,y]=meshgrid(-floor(width/2):floor((width-1)/2), -floor(height/2):...
floor((height-1)/2));
out=1./(1+(sqrt(2)-1)*((x.^2+y.^2)/d^2).^n);

function out=hbutter(im,d,n)
% HBUTTER(IM,D,N) creates a high-pass Butterworth filter
% of the same size as image IM, with cutoff D, and order N
%
% Use:
%   x=imread('cameraman.tif');
%   l=hbutter(x,25,2);
%
out=1-lbutter(im,d,n);
```

 CENGAGE
Learning

Butterworth HPF

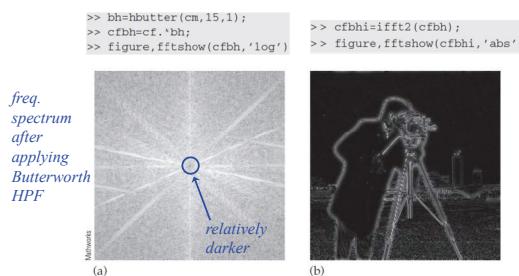


FIGURE 7.11 Butterworth high-pass filtering. (a) The DFT after Butterworth high-pass filtering. (b) The resulting image.

 CENGAGE
Learning

nth Order Butterworth LPF & HPF

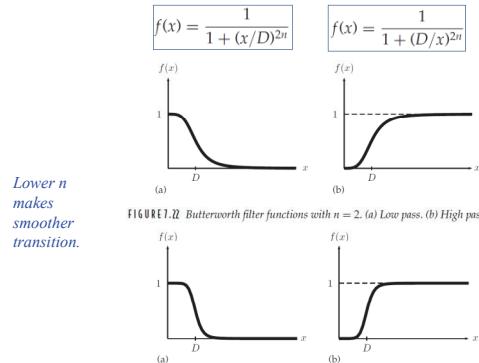


FIGURE 7.12 Butterworth filter functions with $n = 2$. (a) Low pass. (b) High pass.

 CENGAGE
Learning

Butterworth LPF

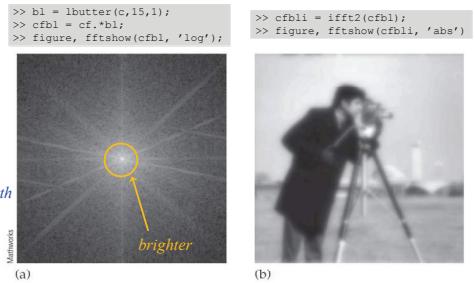


FIGURE 7.16 Butterworth low-pass filtering. (a) The DFT after Butterworth low-pass filtering. (b) The resulting image.

 CENGAGE
Learning

Gaussian Filtering

```
>> g1=mat2gray(fspecial('gaussian',256,10));
>> cgl=cf.*g1;
>> fftshow(cgl,'log')
>> g2=mat2gray(fspecial('gaussian',256,30));
>> cg2=cf.*g2;
>> figure,fftshow(cg2,'log')

>> g=fspecial('gaussian',256,10);
>> format long, max(g()), format
ans =
0.00158757552679
```

- The most popular filter mask.
- A wider function, with a large standard deviation, will have a low maximum.

 CENGAGE
Learning

Gaussian LPF

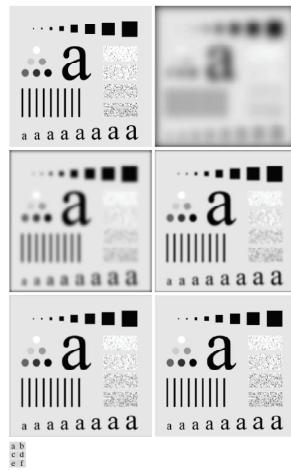
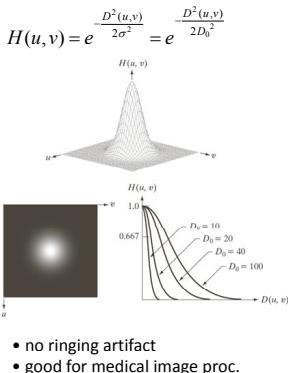


FIGURE 4.48 (a) Original image. (b)-(l) Results of filtering using GLPFs with cutoff frequencies at the radii shown in Fig. 4.41. Compare with Figs. 4.42 and 4.45.

Gaussian HPF

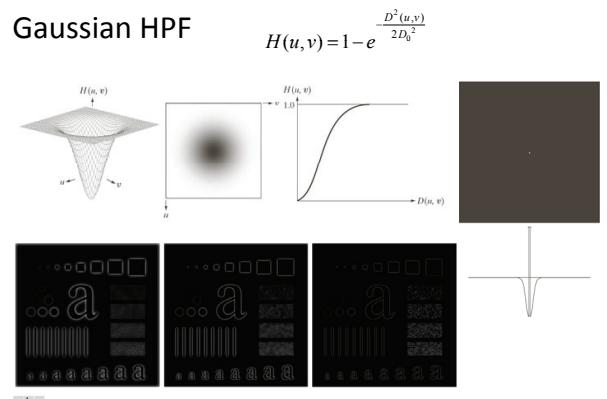


FIGURE 4.46 Results of highpass filtering the image in Fig. 4.41(a) using a GHPF with $D_0 = 30, 60$, and 160 , corresponding to the circles in Fig. 4.41(b). Compare with Figs. 4.54 and 4.55.

CENGAGE Learning

Gaussian LPF Examples

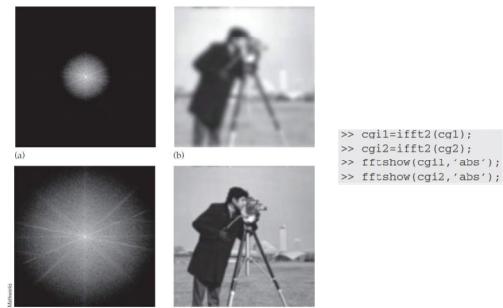


FIGURE 1.28 Applying a Gaussian low-pass filter in the frequency domain. (a) $\sigma = 10$. (b) Resulting image. (c) $\sigma = 30$. (d) Resulting image.

CENGAGE Learning

Gaussian HPF Examples

```
>> h1=l-g1;
>> h2=l-g2;
>> ch1=cft.*h1;
>> ch2=cft.*h2;
>> ch1=ifft2(ch1);
>> ch2=ifft2(ch2);
>> fftshow(ch1,'abs')
>> figure,fftshow(ch2,'abs')
```

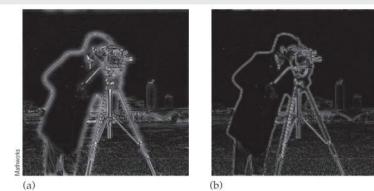


FIGURE 1.29 Applying a Gaussian high-pass filter in the frequency domain. (a) Using $\sigma = 10$. (b) Using $\sigma = 30$.

CENGAGE Learning

Procedure of Filtering in Frequency Domain

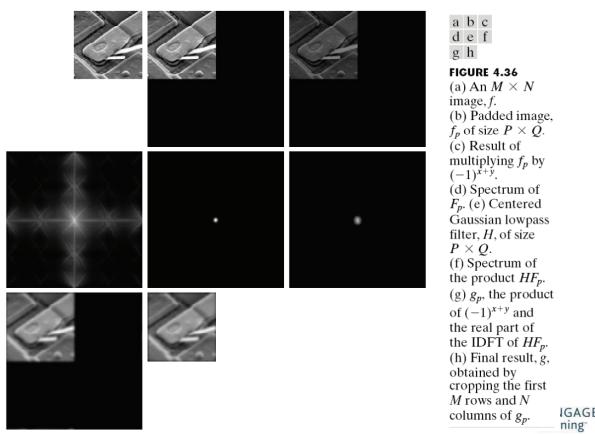
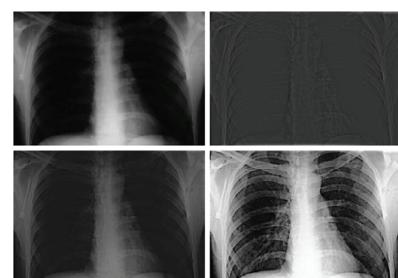


IMAGE
ning

High-Freq.-Emphasis Filter

$$g(x,y) = F^{-1} \{ [k_1 + k_2 H_{HP}(u,v)] F(u,v) \}$$

original
image



HFE
filtered
image

HP
filtered
image

HE of HFE
filtered
image

CENGAGE
Learning

Homomorphic Filtering

- An image $x(n_1, n_2)$ is created by "reflecting" the light from an object that has been "illuminated" by some light source.
- If the illuminating intensity $i(n_1, n_2)$ is a slowly varying component (controlling overall dynamic range = low freq. component), the reflecting source $r(n_1, n_2)$ is a fast varying component (affecting the local contrast = high freq. component),

$$x(n_1, n_2) = i(n_1, n_2) \cdot r(n_1, n_2)$$

CENGAGE Learning

Homomorphic Filtering

- How about separating these two components and emphasizing/deemphasizing each one?

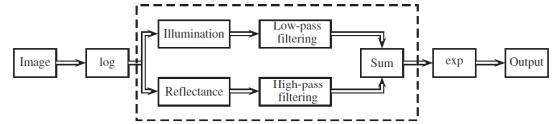
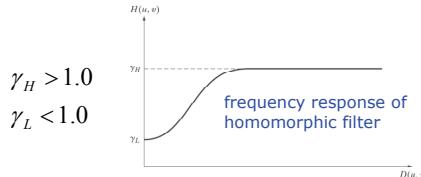
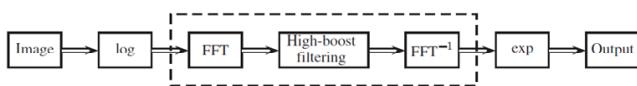


FIGURE 7.30 A schema for homomorphic filtering.

$$y(n_1, n_2) = \exp \left\{ \gamma_L \cdot LPF [\log x(n_1, n_2)] + \gamma_H \cdot HPF [\log x(n_1, n_2)] \right\}$$

CENGAGE Learning

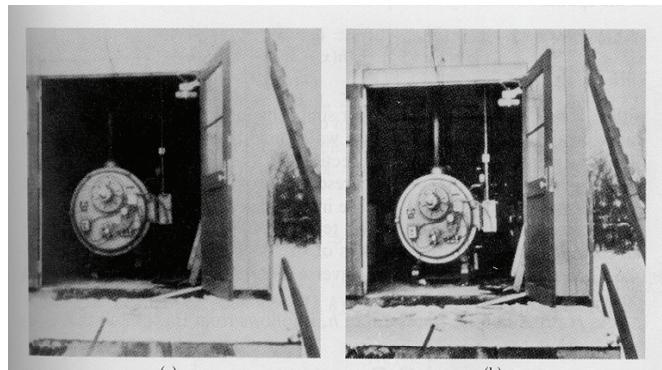
Homomorphic Filtering



can be approximated using $H(u, v) = (\gamma_H - \gamma_L) \left[1 - e^{-c \frac{D^2(u, v)}{D_0^2}} \right] + \gamma_L$

CENGAGE Learning

Homomorphic Filtering



CENGAGE Learning

Homomorphic Filtering

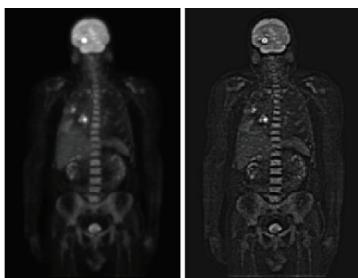


FIGURE 4.4.2
(a) Original PET scan. (b) Image enhanced using homomorphic filtering. (Original image courtesy of Dr. Michael E. Casey, CTI PET Systems.)

- Reduced the effects of the dominant illumination components (hot spots, low freq. component).
- The reflectance components (edge, high freq. component) were sharpened considerably.
- Sharper in hot spots, brain, and skeleton
- More detailed information is shown.

CENGAGE Learning

Homomorphic Filtering in Matlab

```
function res=homfilt(im,cutoff,order,lowgain,highgain)
% HOMFILT(IMAGE,FILTER) applies homomorphic filtering to the image IMAGE
% with the given parameters

u=im2uint8(im);

u(find(u==0))=1;
l=log(double(u));
ft=fftshift(fft2(l));
f=hb_butter(im,cutoff,order,lowgain,highgain);
b=f.*ft;
ib=abs(ifft2(b));
res=exp(ib);
```

FIGURE 7.32 A function to apply homomorphic filtering.

CENGAGE Learning