

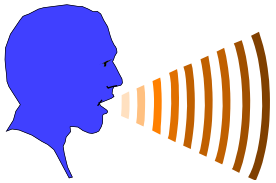
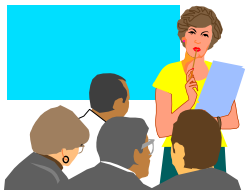



Image and Video Processing

Two-Dimensional Fourier Transform and Linear Filtering

Yao Wang
Tandon School of Engineering, New York University

Outline

- General concept of signals and transforms
  - Representation using basis functions
- Continuous Space Fourier Transform (CSFT)
  - 1D -> 2D
  - Concept of spatial frequency
- Discrete Space Fourier Transform (DSFT) and DFT
  - 1D -> 2D
- Continuous and discrete space convolution
- Convolution theorem
- Applications in image processing

Signal in 2D Space

- General 2D continuous space signal: f(x,y)
  - Can have infinite support: x,y= (-infty,…, infty)
  - f(x,y) can generally take on complex values
- General 2D discrete space signal: f(m,n)
  - Can have infinite support: m,n= -infty,…, 0, 1,.., infty
  - f(m,n) can generally take on complex values
- Each color component of an image is a 2D real signal with finite support
  - MxN image: m=0,1,…,M-1, n=0,1,…,N-1
  - We will use first index for row, second index for column
  - We will consider a single color component only
  - Same operations can be applied to each component
- Separability
  - f(m,n) is separable if $f(m,n)= f_v(m)\, f_h(n)$
  - Rank 1 matrix = product of 1D column vector and 1D row vevtor

Transform Representation of Signals

- Transforms are decompositions of a function f(x) into some basis functions Ø(x, u). u is typically the freq. index.

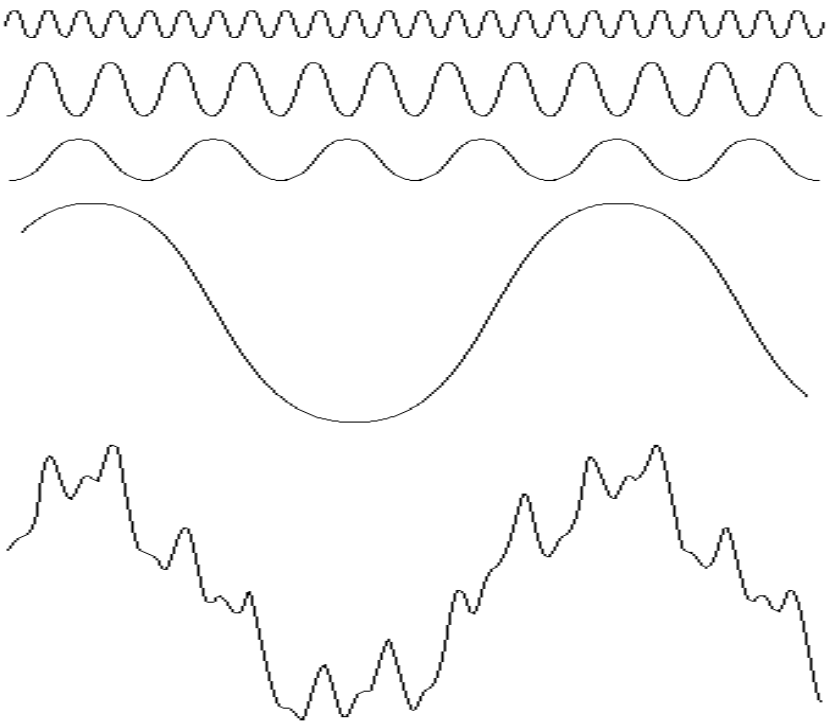
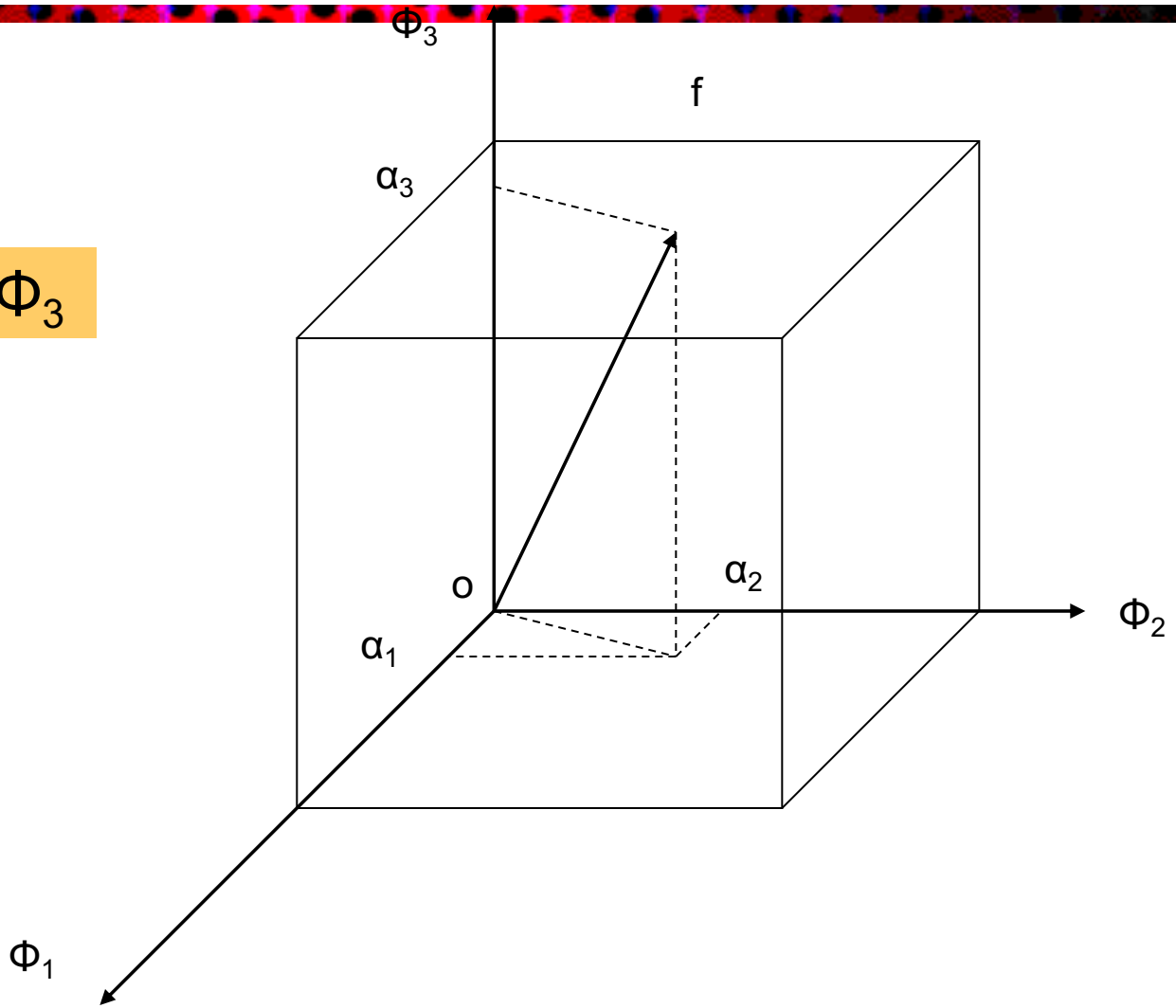


FIGURE 4.1 The function at the bottom is the sum of the four functions above it. Fourier's idea in 1807 that periodic functions could be represented as a weighted sum of sines and cosines was met with skepticism.

Illustration of Decomposition in Vector Space

$$f = \alpha_1\Phi_1+\alpha_2\Phi_2+\alpha_3\Phi_3$$



Decomposition of 1D Signal

- Orthonormal basis function

$$\int_{-\infty}^{\infty} \phi(x,u_1)\phi^*(x,u_2)dx = \begin{cases} 1, & u_1 = u_2 \\ 0, & u_1 \neq u_2 \end{cases}$$

- Forward transform

$$F(u) = < f(x), \phi(x,u) > = \int_{-\infty}^{\infty} f(x)\phi^*(x,u)dx$$

Projection of $f(x)$ onto $\phi(x,u)$

- Inverse transform

$$f(x) = \int_{-\infty}^{\infty} F(u)\phi(x,u)du$$

Representing $f(x)$ as sum of $\phi(x,u)$ for all u, with weight $F(u)$

1D Continuous Time Fourier Transform

- Basis function

$$\phi(x,u) = e^{j2\pi ux}, \quad u \in (-\infty, +\infty).$$

- Forward Transform

$$F(u) = F\{f(x)\} = \int_{-\infty}^{\infty} f(x) e^{-j2\pi ux} dx$$

- Inverse Transform

$$f(x) = F^{-1}\{F(u)\} = \int_{-\infty}^{\infty} F(u) e^{j2\pi ux} du$$

Important Transform Pairs

$$f(x)=1 \quad \Leftrightarrow \quad F(u)=\delta(u)$$

$$f(x)=e^{j2\pi f_0 x} \quad \Leftrightarrow \quad F(u)=\delta(u-f_0)$$

$$f(x)=\cos(2\pi f_0 x) \quad \Leftrightarrow \quad F(u)=\frac{1}{2}\left(\delta(u-f_0)+\delta(u+f_0)\right)$$

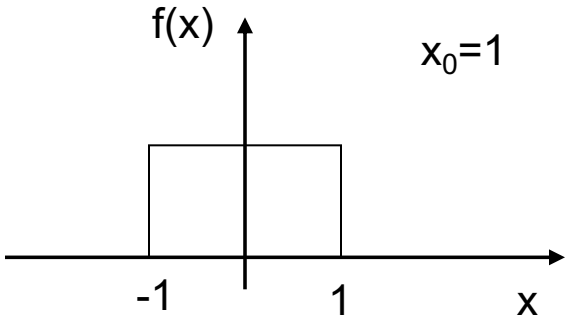
$$f(x)=\sin(2\pi f_0 x) \quad \Leftrightarrow \quad F(u)=\frac{1}{2j}\left(\delta(u-f_0)-\delta(u+f_0)\right)$$

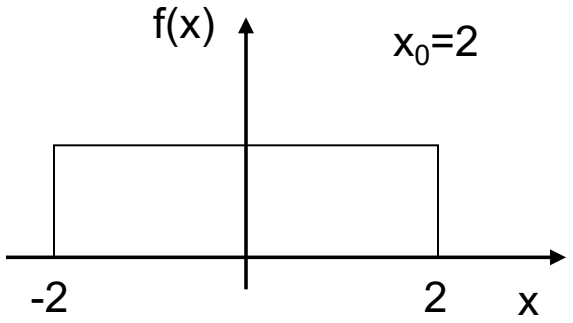
$$f(x)=\begin{cases} 1, & |x|<x_0 \\ 0, & otherwise \end{cases} \quad \Leftrightarrow \quad F(u)=\frac{\sin(2\pi x_0 u)}{\pi u}=2x_0\,\mathrm{sinc}(2x_0 u)$$

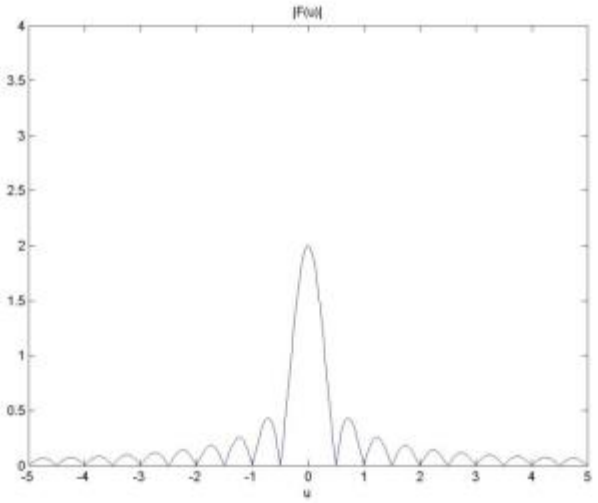
$$where,\ \mathrm{sinc}(t)=\frac{\sin(\pi t)}{\pi t}$$

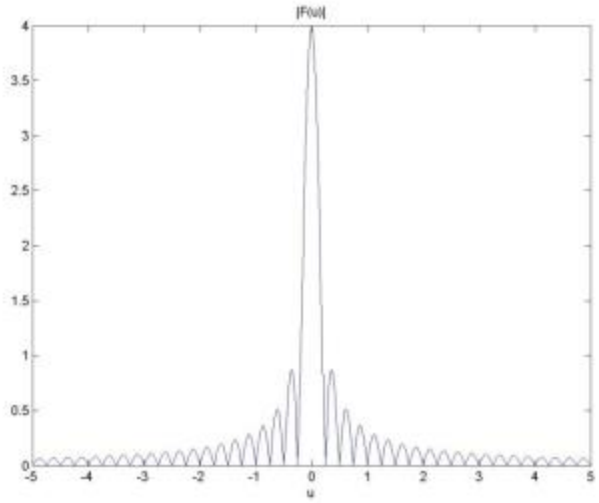
FT of the Rectangle Function

$$F(u) = \frac{\sin(2\pi x_0 u)}{\pi u} = 2x_0 \,\mathrm{sinc}(2x_0 u) \quad where, \mathrm{sinc}(t) = \frac{\sin(\pi t)}{\pi t}$$





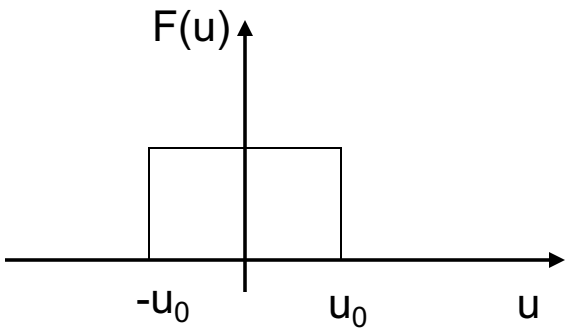




Note first zero occurs at $u_0=1/(2\,x_0)$=1/pulse-width, other zeros are multiples of this.

IFT of Ideal Low Pass Signal

- What is f(x)?



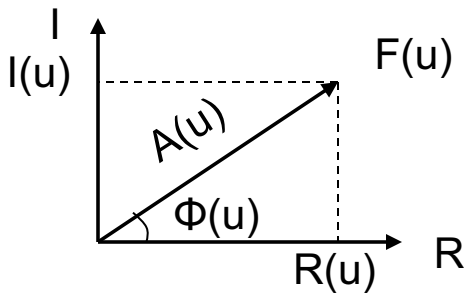
Representation of FT

- Generally, both f(x) and F(u) are complex
- Two representations
  - Real and Imaginary
  - Magnitude and Phase

$$F(u) = R(u) + jI(u)$$

$$F(u) = A(u)e^{j\phi(u)}, \quad where$$

$$A(u) = \sqrt{R(u)^2 + I(u)^2}, \phi(u) = \tan^{-1}\frac{I(u)}{R(u)}$$



- Relationship

$$R(u) = A(u)\cos\phi(u), \quad I(u) = A(u)\sin\phi(u)$$

- Power spectrum

$$P(u) = A(u)^2 = F(u) \times F(u)^* = |F(u)|^2$$

What if f(x) is real?

- Real world signals f(x) are usually real
- F(u) is still complex, but has special properties

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

$$R(u) = R(-u), A(u) = A(-u), P(u) = P(-u): \text{even function}$$

$$I(u) = -I(-u), \phi(u) = -\phi(-u): \text{odd function}$$

Property of Fourier Transform

- Duality

$$f(t) \Leftrightarrow F(u)$$
$$F(t) \Leftrightarrow f(-u)$$

- Linearity

$$F\{a_1 f_1(x) + a_2 f_2(x)\} = a_1 F\{f_1(x)\} + a_2 F\{f_2(x)\}$$

- Scaling

$$F\{af(x)\} = aF\{f(x)\}$$

- Translation

$$f(x - x_0) \Leftrightarrow F(u)e^{-j2\pi x_0 u}, \quad f(x)e^{j2\pi u_0 x} \Leftrightarrow F(u - u_0)$$

- Convolution

$$f(x) \otimes g(x) = \int f(x - \alpha) g(\alpha) d\alpha$$

$$f(x) \otimes g(x) \Leftrightarrow F(u)G(u)$$

We will review convolution later!

Two Dimension Continuous Space Fourier Transform (CSFT)

- Basis functions

$$\phi(x,y;u,v) = e^{j(2\pi ux+2\pi vy)} = e^{j2\pi ux}e^{j2\pi vy}, \quad u,v \in (-\infty,+\infty).$$

- Forward – Transform

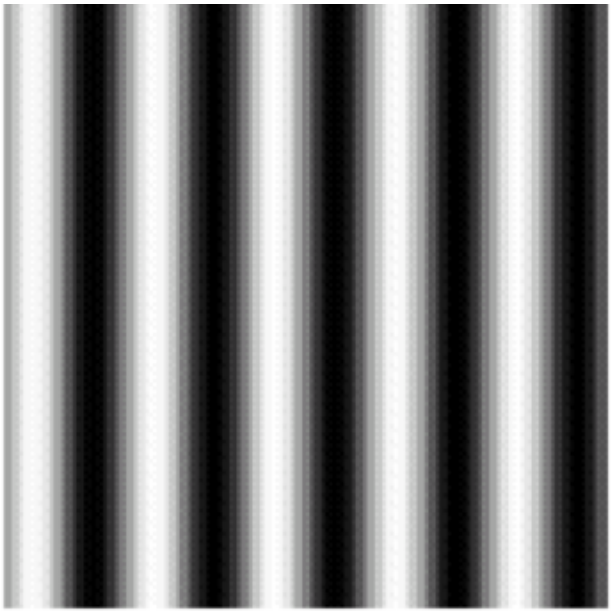
$$F(u,v) = F\{f(x,y)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}dxdy$$

- Inverse – Transform

$$f(x,y) = F^{-1}\{F(u,v)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{j2\pi(ux+vy)}dudv$$

  - Representing a 2D signal as sum of 2D complex exponential signals

Illustration of Spatial Frequency



$f(x,y)=\sin(10\pi x)$

$f_x=5, f_y=0, f_m=5, \phi=0$

f_x: unit= 1 cycle/image-width

f_y: unit=1 cycle/image-height

$f(x,y)=\sin(10\pi x-20\pi y)$

$f_x=5, f_y=-10, f_m=\sqrt{125}, \phi=\text{atan}(-2)$

Example 1

$$f(x,y)=\sin 4\pi x+\cos 6\pi y$$

$$F\{\sin 4\pi x\}=\iint \sin 4\pi x e^{-j2\pi(ux+vy)}dxdy$$

$$=\int \sin 4\pi x e^{-j2\pi ux}dx\int e^{-j2\pi vy}dy$$

$$=\int \sin 4\pi x e^{-j2\pi ux}dx\,\delta(v)$$

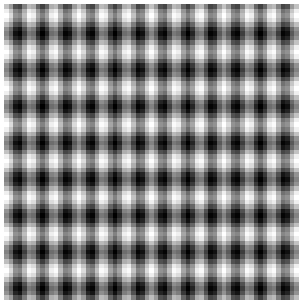
$$=\frac{1}{2j}(\delta(u-2)-\delta(u+2))\delta(v)$$

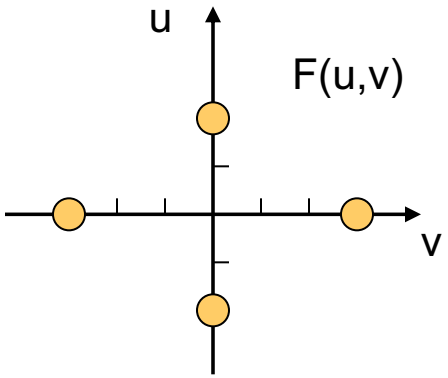
$$=\frac{1}{2j}(\delta(u-2,v)-\delta(u+2,v))$$

$$where\quad \delta(x,y)=\delta(x)\delta(y)=\begin{cases}\infty, & x=y=0\\ 0, & otherwise\end{cases}$$

$$Likewise,\quad F\{\cos 6\pi y\}=\frac{1}{2}(\delta(u,v-3)+\delta(u,v+3))$$

f(x,y)





Example 2

$$f(x,y)=\sin(2\pi x+3\pi y)=\frac{1}{2j}\left(e^{j(2\pi x+3\pi y)}-e^{-j(2\pi x+3\pi y)}\right)$$

$$F\left\{e^{j(2\pi x+3\pi y)}\right\}=\iint e^{j(2\pi x+3\pi y)}e^{-j2\pi(xu+yv)}dxdy$$

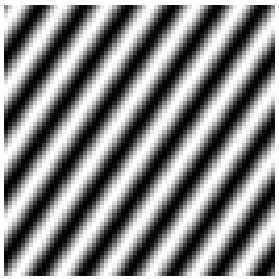
$$=\int e^{j2\pi x}e^{-j2\pi ux}dx\int e^{j3\pi y}e^{-j2\pi yv}dy$$

$$=\delta(u-1)\delta(v-\frac{3}{2})=\delta(u-1,v-\frac{3}{2})$$

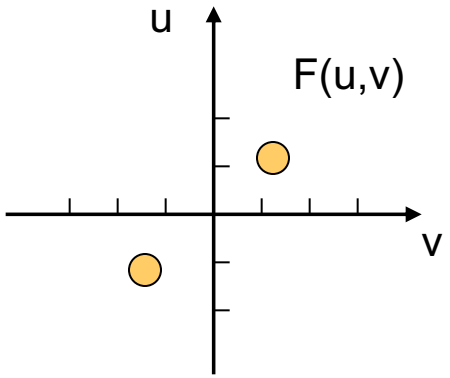
$$Likewise,\quad F\left\{e^{-j(2\pi x+3\pi y)}\right\}=\delta(u+1,v+\frac{3}{2})$$

Therefore,

$$F\left\{\sin(2\pi x+3\pi y)\right\}=\frac{1}{2j}\left(\delta(u-1,v-\frac{3}{2})-\delta(u+1,v+\frac{3}{2})\right)$$



```
[X,Y]=meshgrid(-2:1/16:2,-2:1/16:2);
f=sin(2*pi*X+3*pi*Y);
imagesc(f); colormap(gray)
Truesize, axis off;
```



A Little Teaser ☺

$f(x,y)=\sin(10\pi x)$

$f_x=5, f_y=0, f_m=5, \phi=0$

f_x : unit= 1 cycle/image-width

f_y : unit=1 cycle/image-height

What is its 2D FT?

Properties of 2D FT (1)

- Linearity

$$F\{a_1 f_1(x,y)+a_2 f_2(x,y)\}=a_1 F\{f_1(x,y)\}+a_2 F\{f_2(x,y)\}$$

- Translation

$$f(x-x_0,y-y_0) \quad \Leftrightarrow \quad F(u,v)e^{-j2\pi(x_0u+y_0v)},$$

$$f(x,y)e^{j2\pi(u_0x+v_0y)} \quad \Leftrightarrow \quad F(u-u_0,v-v_0)$$

- Conjugation

$$f^*(x,y) \quad \Leftrightarrow \quad F^*(-u,-v)$$

Properties of 2D FT (2)

- Symmetry

$$f(x,y)\ is\ real \quad \Leftrightarrow \quad |F(u,v)| = |F(-u,-v)|$$

- Convolution
  - Definition of convolution

$$f(x,y) \otimes g(x,y) = \iint f(x-\alpha, y-\beta) g(\alpha,\beta) d\alpha d\beta$$

  - Convolution theory

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

We will describe 2D convolution later!

Separability of 2D FT and Separable Signal

- Separability of 2D FT

$$F_2\{f(x,y)\} = F_y\{F_x\{f(x,y)\}\} = F_x\{F_y\{f(x,y)\}\}$$

  - where F_x, F_y are 1D FT along x and y.
  - one can do 1DFT for each row of original image, then 1D FT along each column of resulting image
- Separable Signal
  - $f(x,y) = f_x(x)f_y(y)$
  - $F(u,v) = F_x(u)F_y(v)$,
    - where $F_x(u) = F_x\{f_x(x)\}$, $F_y(u) = F_y\{f_y(y)\}$
  - For separable signal, one can simply compute two 1D transforms and take their product!

Example 1

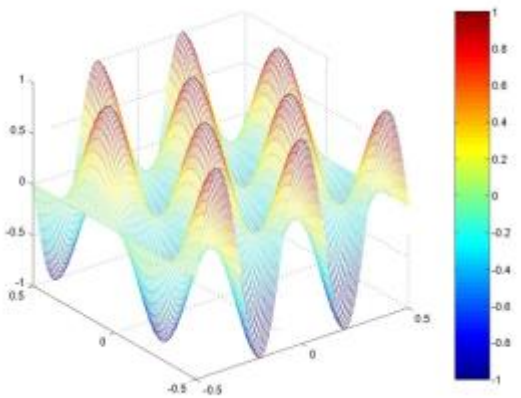
$$f(x,y) = \sin(3\pi x)\cos(5\pi y)$$

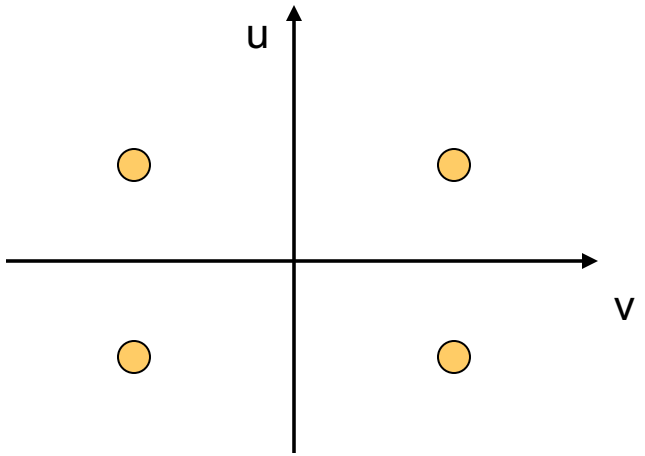
$$f_x(x) = \sin(3\pi x) \quad \Leftrightarrow \quad F_x(u) = \frac{1}{2j}\left(\delta(u-3/2) - \delta(u+3/2)\right)$$

$$f_y(y) = \cos(5\pi y) \quad \Leftrightarrow \quad F_y(v) = \frac{1}{2}\left(\delta(v-5/2) + \delta(v+5/2)\right)$$

$$F(u,v) = Fx(u)Fy(v)$$

$$= \frac{1}{4j}\left(\delta(u-\frac{3}{2},v-\frac{5}{2}) - \delta(u+\frac{3}{2},v-\frac{5}{2}) + \delta(u-\frac{3}{2},v+\frac{5}{2}) - \delta(u+\frac{3}{2},v+\frac{5}{2})\right)$$

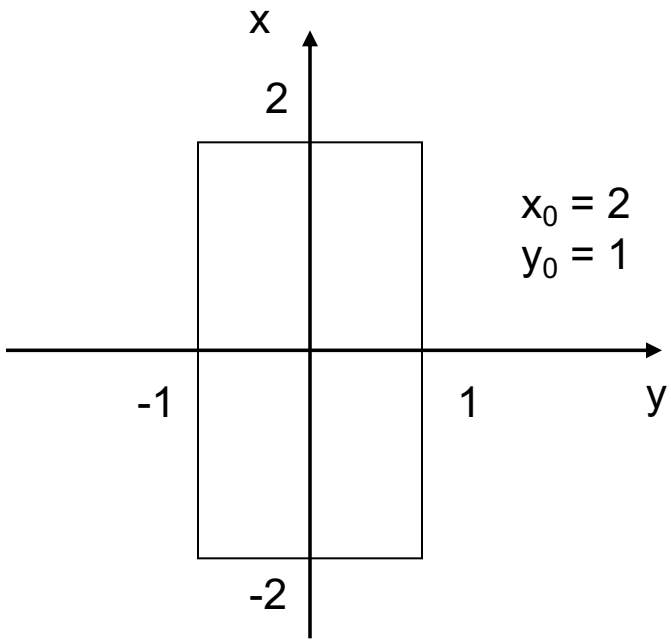


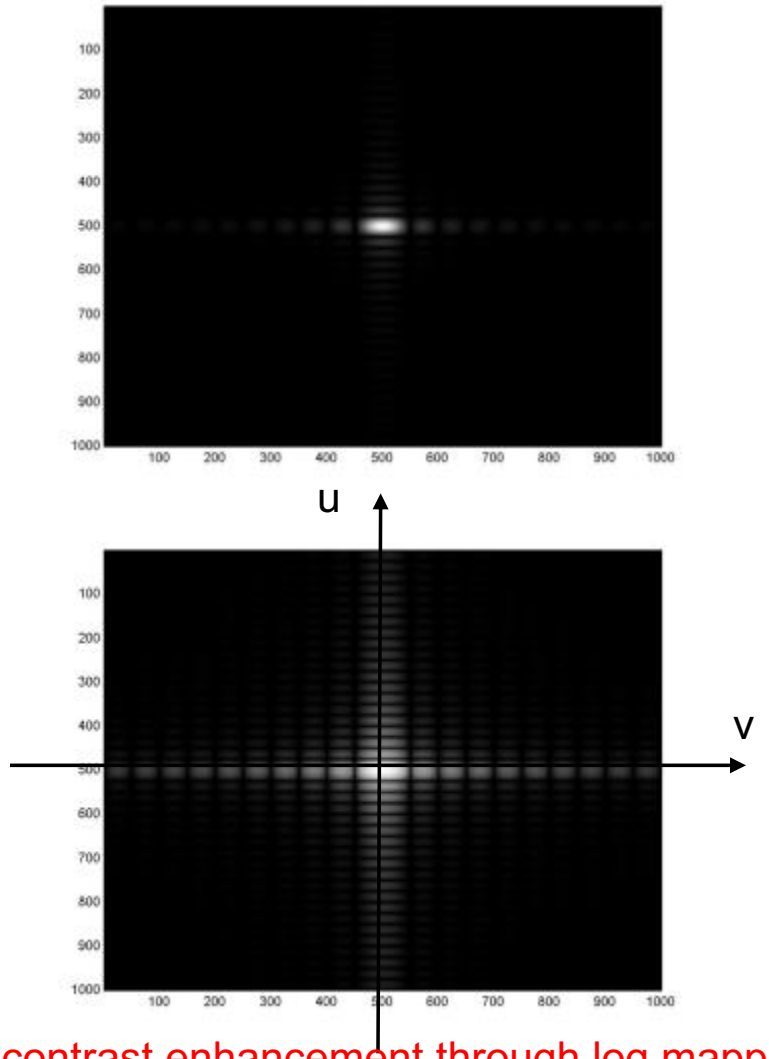


Example 2

$$f(x,y)=\begin{cases}1, & |x|\le x_0, |y|\le y_0\\ 0, & otherwise\end{cases} \Rightarrow$$

$$F(u,v)=4x_0y_0\,\mathrm{sinc}(2x_0u)\,\mathrm{sinc}(2y_0v)$$





w/ contrast enhancement through log mapping

Example 3: Gaussian Signal

- Gaussian Function's FT: Still a Gaussian Function!
- 1D Gaussian Signal

$$\exp\left\{-\frac{x^2}{2\sigma^2}\right\} \Leftrightarrow \exp\left\{-\frac{u^2}{2\beta^2}\right\}, \beta = \frac{1}{2\pi\sigma}$$

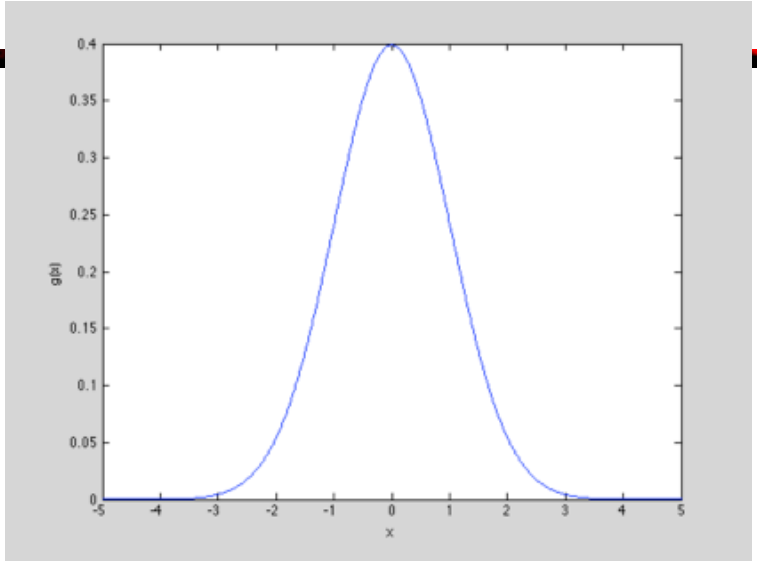
- 2D Gaussian Signal

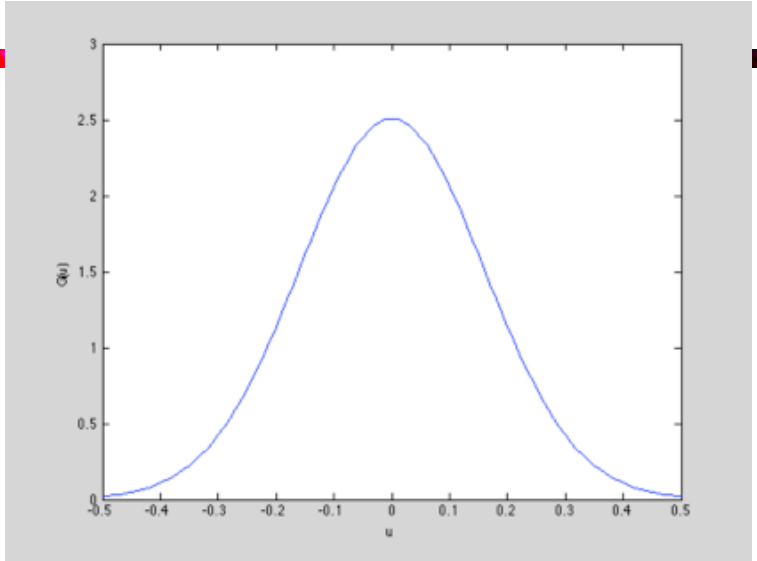
$$\exp\left\{-\frac{x^2+y^2}{2\sigma^2}\right\} = \exp\left\{-\frac{x^2}{2\sigma^2}\right\}\exp\left\{-\frac{y^2}{2\sigma^2}\right\}$$
$$\leftrightarrow \exp\left\{-\frac{u^2}{2\beta^2}\right\}\exp\left\{-\frac{v^2}{2\beta^2}\right\} = \exp\left\{-\frac{u^2+v^2}{2\beta^2}\right\}, \beta = \frac{1}{2\pi\sigma}$$

- Note that STD σ in space inversely related to STD β in freq.

Illustration of Gaussian Signal

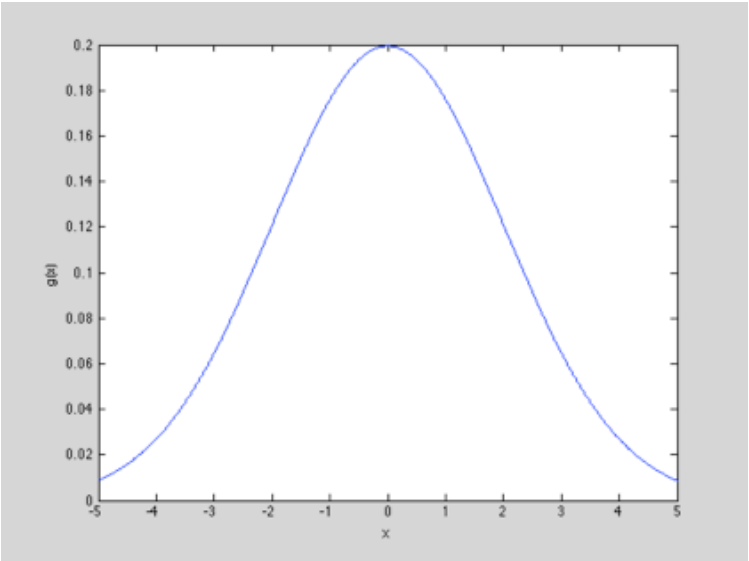
σ =1

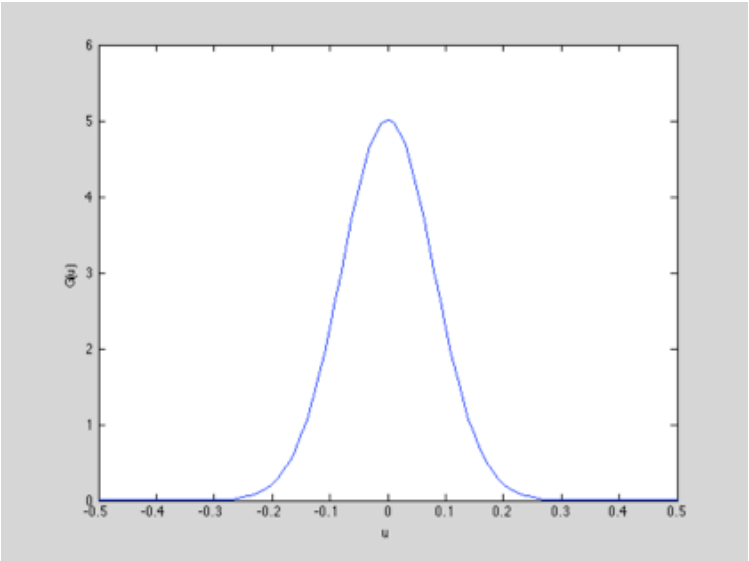




β =0.16

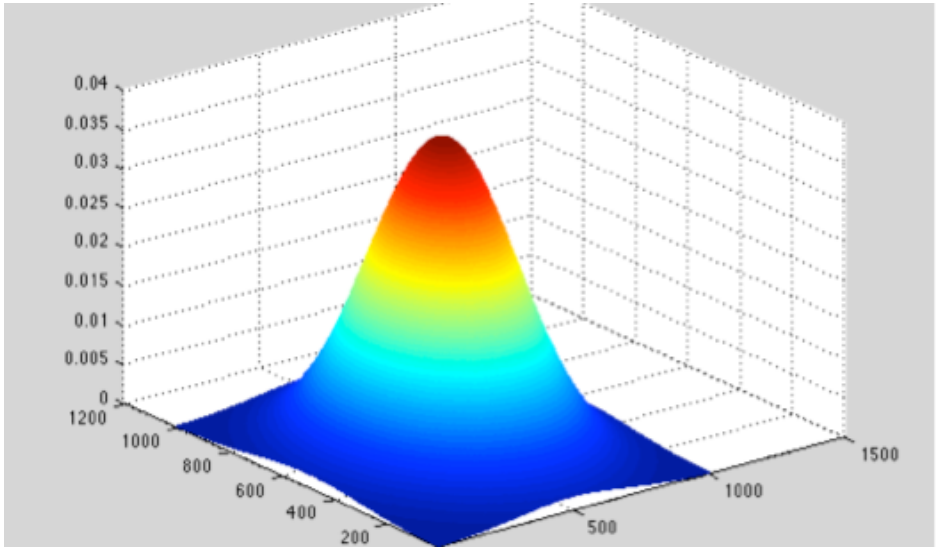
σ =2





β =0.08

2D Gaussian Function



Surface plot (surf( ))

Plot as image

Important Transform Pairs

- All following signals are separable and the relations can be proved by applying the separability of the CSFT

$$f(x,y)=1 \quad \Leftrightarrow \quad F(u,v)=\delta(u,v), \quad where \delta(u,v)=\delta(u)\delta(v)$$

$$f(x)=e^{j2\pi(f_1x+f_2y)} \quad \Leftrightarrow \quad F(u)=\delta(u-f_1,v-f_2)$$

$$f(x,y)=\cos(2\pi(f_1x+f_2y)) \quad \Leftrightarrow \quad F(u)=\frac{1}{2}\left(\delta(u-f_1,v-f_2)+\delta(u+f_1,v+f_2)\right)$$

$$f(x,y)=\sin(2\pi(f_1x+f_2y)) \quad \Leftrightarrow \quad F(u)=\frac{1}{2j}\left(\delta(u-f_1,v-f_2)-\delta(u+f_1,v+f_2)\right)$$

$$f(x,y)=\begin{cases}1, & |x|<x_0,|y|<y_0\\ 0, & otherwise\end{cases} \quad \Leftrightarrow$$

$$F(u,v)=\frac{\sin(2\pi x_0u)}{\pi u}\frac{\sin(2\pi y_0v)}{\pi v}=4x_0y_0\,\mathrm{sinc}(2x_0u)\mathrm{sinc}(2y_0v)$$

$$where\ \mathrm{sinc}(t)=\frac{\sin(\pi t)}{\pi t}$$

$$\exp\left\{-\frac{x^2+y^2}{2\sigma^2}\right\} \Leftrightarrow \exp\left\{-\frac{u^2+v^2}{2\beta^2}\right\}, \beta=\frac{1}{2\pi\sigma}$$

- Constant ⇔ impulse at (0,0) freq.
- Complex exponential ⇔ impulse at a particular 2D freq.
- 2D box ⇔ 2D sinc function
- Gaussian ⇔ Gaussian

Rotation

- Let $x = r\cos\theta, \quad y = r\sin\theta, \quad u = \rho\cos\omega, \quad v = \rho\sin\omega.$
- 2D FT in polar coordinate (r, θ) and (ρ, ϕ)

$$F(\rho,\phi) = \int_0^{\infty}\int_0^{2\pi} f(r,\theta)e^{-j2\pi(r\cos\theta\,\rho\cos\phi + r\sin\theta\,\rho\sin\phi)} r\,dr\,d\theta$$

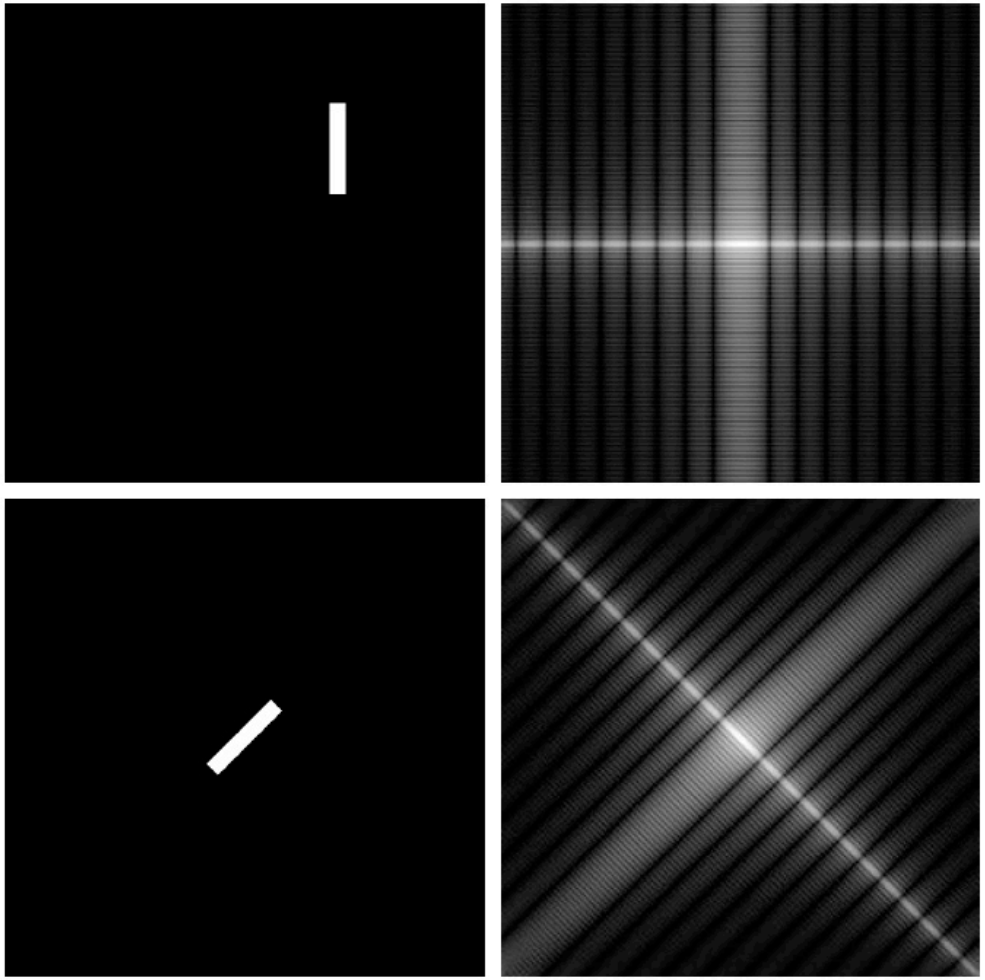
$$= \iint f(r,\theta)e^{-j2\pi r\rho\cos(\theta-\phi)} r\,dr\,d\theta$$

- Property

$$f(r,\theta+\theta_0) \quad \Leftrightarrow \quad F(\rho,\phi+\theta_0)$$

- Proof: Homework!

Example of Rotation



a b
c d

FIGURE 4.25
(a) The rectangle in Fig. 4.24(a) translated, and (b) the corresponding spectrum. (c) Rotated rectangle, and (d) the corresponding spectrum. The spectrum corresponding to the translated rectangle is identical to the spectrum corresponding to the original image in Fig. 4.24(a).

From Gonzalez

1D Fourier Transform For Discrete Time Sequence (DTFT) (Review)

- f(n) is a 1D discrete time sequence
- Forward Transform

$$F(u) = \sum_{n=-\infty}^{\infty} f(n) e^{-j2\pi un}$$

- Inverse Transform

$$f(n) = \int_{-1/2}^{1/2} F(u) e^{j2\pi un} du$$

- Representing f(n) as a weighted sum of many complex sinusoidal signals with frequency u, F(u) is the weight
- F(u) indicate the "amount" of sinusoidal component with freq. u in signal f(n)
- |u| = digital frequency = the number of cycles per integer sample. Period = 1/|u| (must be equal or greater than 2 samples-> |u|<=1/2)

Properties unique for DTFT

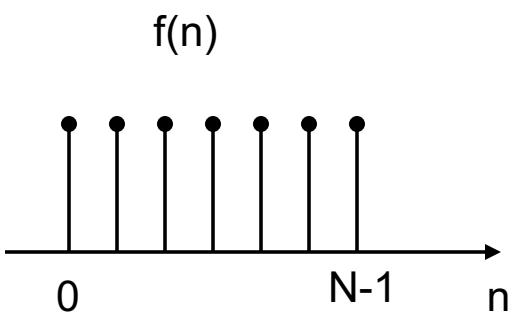
- Periodicity
  - F(u) = F(u+1)
  - The FT of a discrete time sequence is only considered for u ϵ (-½ , ½), and u = ±½ is the highest discrete frequency
- Symmetry for real sequences

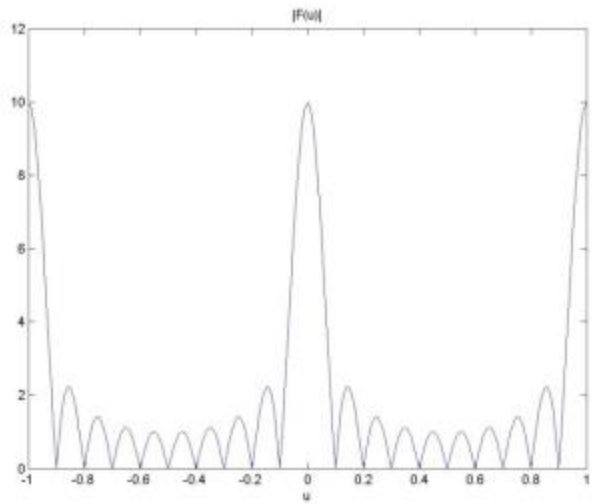
$$f(n) = f^*(n) \quad \Leftrightarrow \quad F(u) = F^*(-u)$$
$$\Rightarrow \quad |F(u)| = |F(-u)|$$
$$\Rightarrow \quad |F(u)| \text{ is symmetric}$$

Example

$$f(n)=\begin{cases}1, & n=0,1,\ldots,N-1;\\ 0, & others\end{cases}$$

$$F(u)=\sum_{n=0}^{N-1}e^{-j2\pi nu}=\frac{1-e^{-j2\pi uN}}{1-e^{-j2\pi u}}=e^{-j\pi(N-1)u}\,\frac{\sin 2\pi u(N/2)}{\sin 2\pi u(1/2)}$$





N=10

Digital sinc: There are N/2 zeros in (0, ½], 1/N apart

1D Discrete Fourier Transform (DFT)

- DTFT of N-pt sequence

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

- N-pt DFT of N-pt sequence

$$F_N(k)=\sum_{n=0}^{N-1} f(n)e^{-j2\pi\frac{kn}{N}}=F\left(u=\frac{k}{N}\right)$$

- DFT is the sampled version of DTFT with samples at 0, 1/N, …, (N-1)/N.
- FFT: Fast algorithm for computing DFT
  - Direct computation takes N^2 operations
  - FFT takes ~N log(N) operations!

Discrete Space Fourier Transform (DSFT) for Two Dimensional Signals

- Let f(m,n) represent a 2D sequence
- Forward Transform

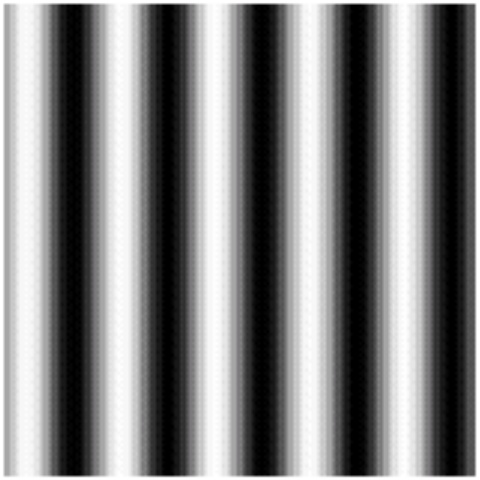
$$F(u,v)=\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}f(m,n)e^{-j2\pi(mu+nv)}$$

- Inverse Transform

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

- Representing an image as weighted sum of many 2D complex sinusoidal images
- u: number of cycles per vertical sample (vertical freq.)
- v: number of cycles per horizontal sample (horizontal freq)

Spatial Frequency for Digital Images



(a)

(b)

Figure 2.1 Two-dimensional sinusoidal signals: (a) $(f_x, f_y) = (5, 0)$; (b) $(f_x, f_y) = (5, 10)$. The horizontal and vertical units are the width and height of the image, respectively. Therefore, $f_x = 5$ means that there are five cycles along each row.

$$f_s = \sqrt{125}, \quad \varphi = \arctan(2)$$

If the image has 256x256 pixels, fx=5 cycles per width (analog frequency) -> u= 5 cycles/256 pixels =5/256 cycles/sample (digital frequency)
When both horizontal and vertical frequency are non-zero, we see directional patterns. fs is the frequency along the direction with the maximum change (orthogonal to the lines)
Note that 1/u may not correspond to integer.
If u=a/b, digital period= b. Ex u=3/8, Digital Period = 8

Periodicity

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

- *F(u,v)* is periodic in u, v with period 1, i.e., for all integers k, l:
  - F(u+k, v+l) = F(u, v)
- To see this consider

$$e^{-j2\pi(m(u+k)+n(v+l))}$$

$$=e^{-j2\pi(mu+nv)}e^{-j2\pi(mk+nl)}$$

$$=e^{-j2\pi(mu+nv)}$$

Example: Delta Function

- Fourier transform of a delta function

$$F(u,v)=\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\delta(m,n)e^{-j2\pi(mu+nv)}=1$$

$$\delta(m,n)\Leftrightarrow 1$$

  - Delta function contains all frequency components with equal weights!

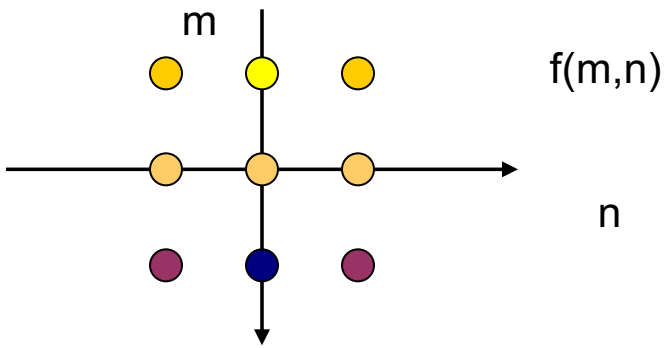
- Inverse Fourier transform of a delta function

$$f(m,n)=\int_{-0.5}^{0.5}\int_{-0.5}^{0.5}\delta(u,v)e^{j2\pi(mu+nv)}dudv=1$$

$$1\Leftrightarrow\delta(u,v)$$

Example

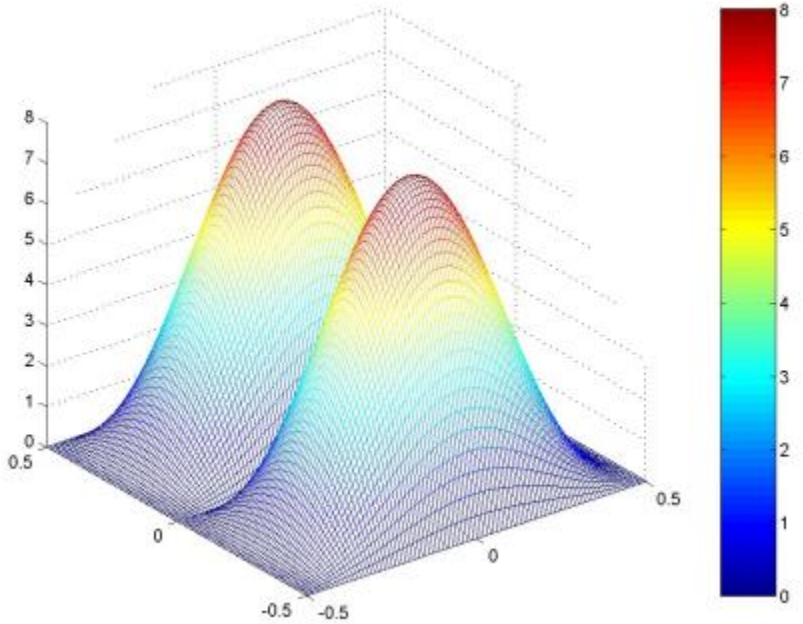
$$f(m,n)=\begin{bmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix}$$

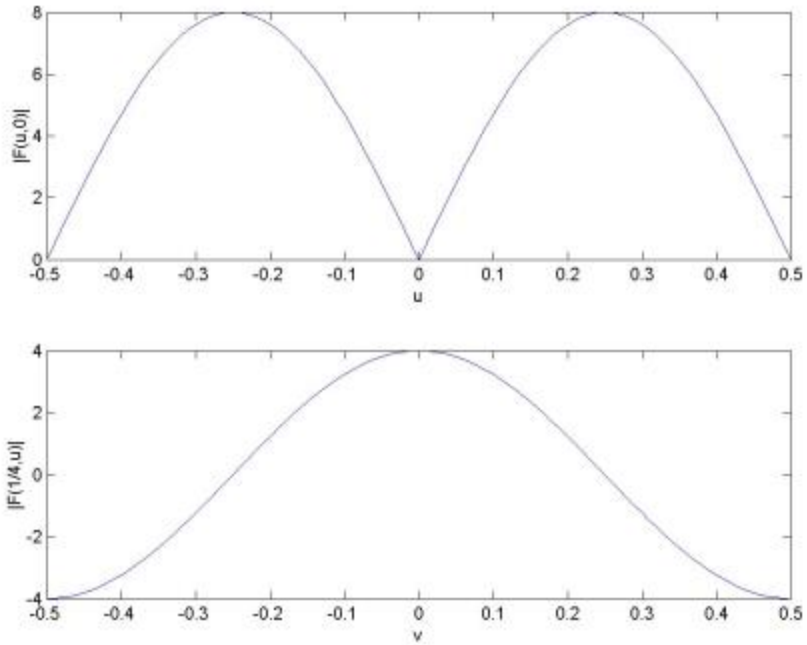


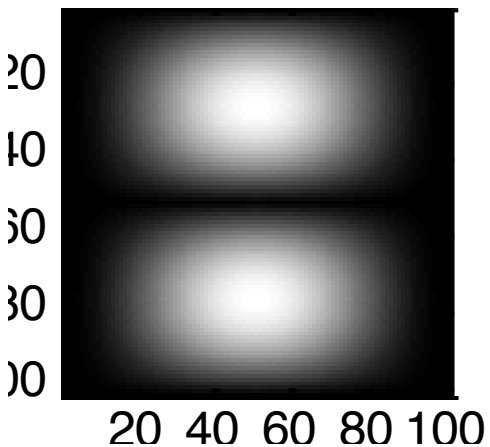
$$\begin{aligned}F(u,v) &= 1e^{-j2\pi(-1*u-1*v)}+2e^{-j2\pi(-1*u+0*v)}+1e^{-j2\pi(-1*u+1*v)}\\ &\quad -1e^{-j2\pi(1*u-1*v)}-2e^{-j2\pi(1*u+0*v)}-1e^{-j2\pi(1*u+1*v)}\\ &= j2\sin 2\pi u e^{j2\pi v}+j4\sin 2\pi u+j2\sin 2\pi u e^{-j2\pi v}\\ &= j4\sin 2\pi u(\cos 2\pi v+1)\end{aligned}$$

Note: This signal is low pass in the horizontal direction v, and band pass in the vertical direction u.

Graph of F(u,v)








```
du = [-0.5:0.01:0.5];
dv = [-0.5:0.01:0.5];
Fu = abs(sin(2 * pi * du));
Fv = cos(2 * pi * dv);
F = 4 * Fu' * (Fv + 1);
mesh(du, dv, F);
colorbar;
Imagesc(F);
colormap(gray); truesize;
```

Using MATLAB freqz2:

```
f=[1,2,1;0,0,0;-1,-2,-1];
freqz2(f)
```

DSFT vs. 2D DFT

- 2D DSFT

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

- 2D DFT

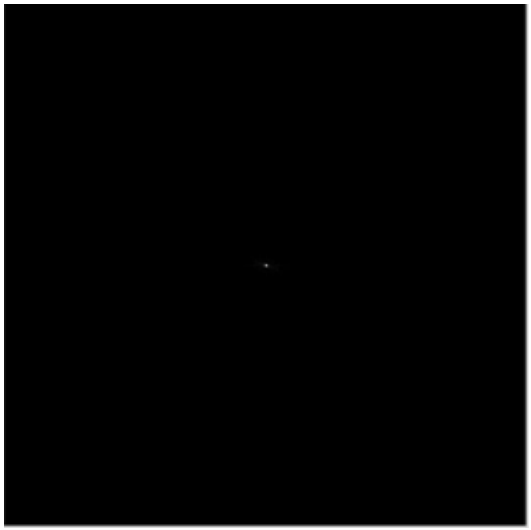
$$F(k,l)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}f(m,n)e^{-j2\pi(mk/M+nl/N)}$$

- (MxN) point 2D DFT are samples of DSFT for images of size MxN at u=m/M, v=n/N. Can be computed using FFT algorithms. 1D FFT along rows, then 1D FFT along columns

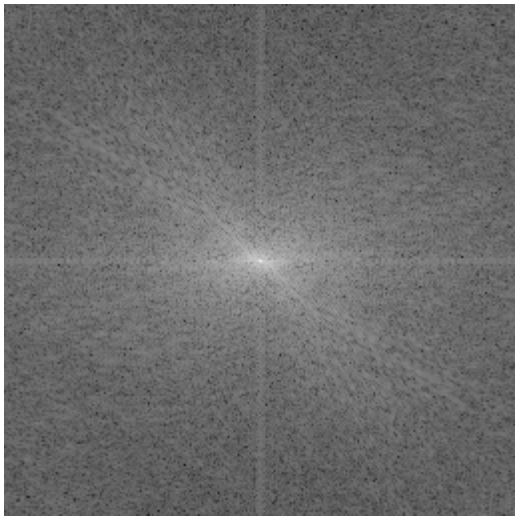
Display of DFT of Images

Imshow(img)

MFimg=abs(fft2(img)), imshow(MFimg,[ ])



SMFimg=fftshift(MFimg)
imshow(SMFimg,[ ])

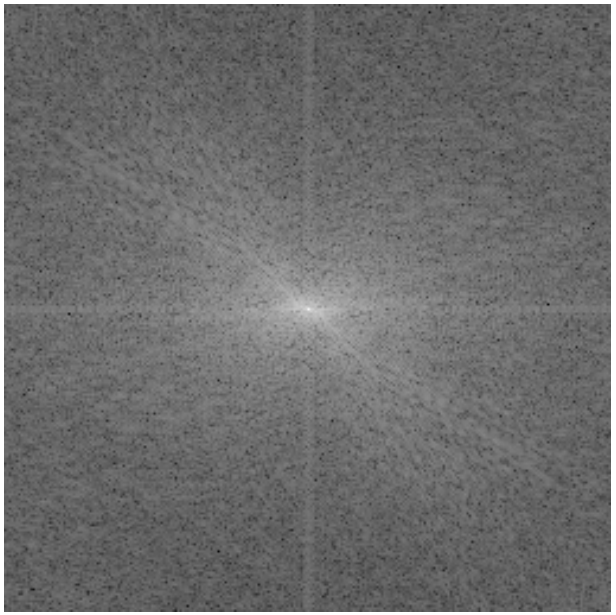


LSMFimg=log(SMFimg+1);
Imshow(LSMFimg,[ ])

fftshift to shift the (0,0) freq. to center

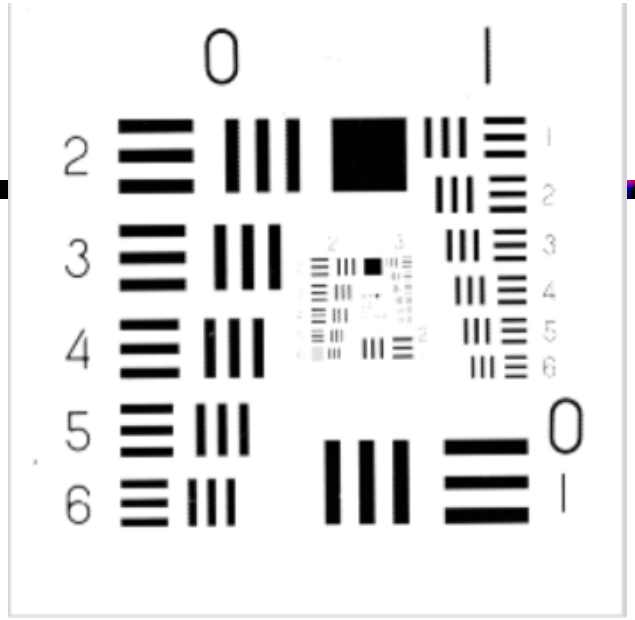
Log mapping to enhance contrast

DFT of Typical Images

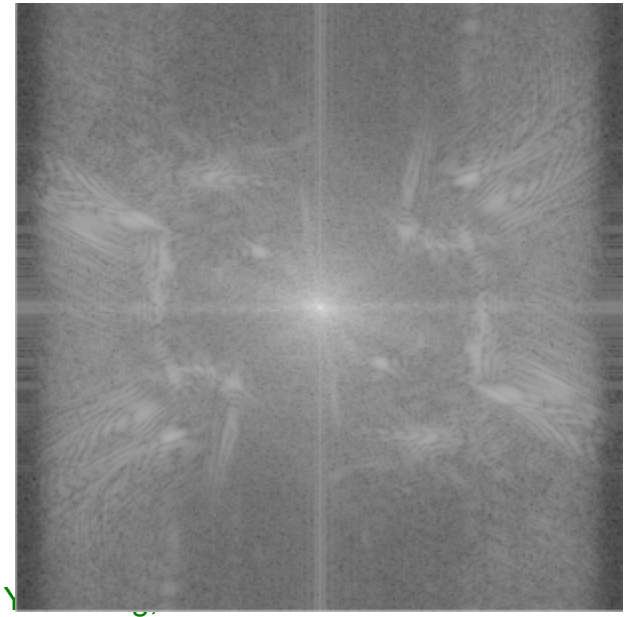


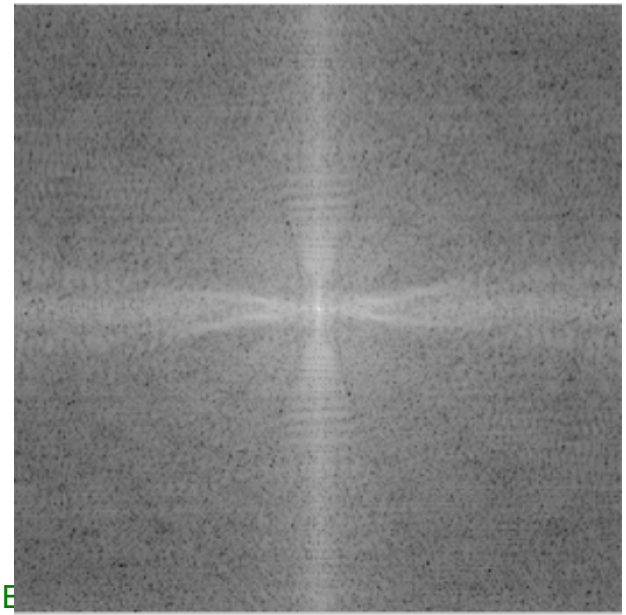
How to interpret the DFT image?
Why is it bright at center and has some line structures?

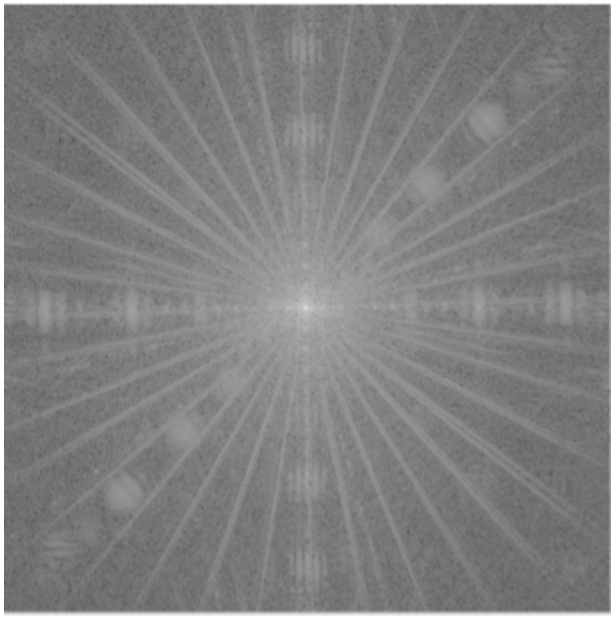
|F(u,v)| (obtained using 2D FFT)
ff=abs(fft2(f));
imagesc(fftshift(log(ff+1)));
Log mapping to enhance contrast
Fftshift to shift the (0,0) freq. to center

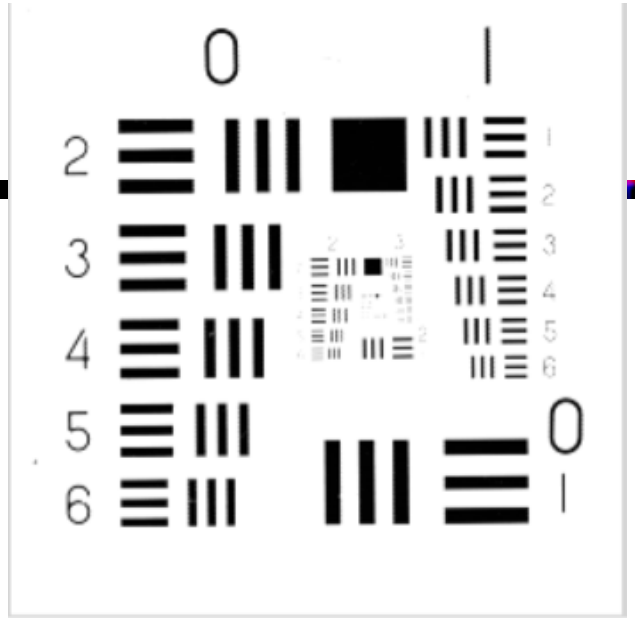


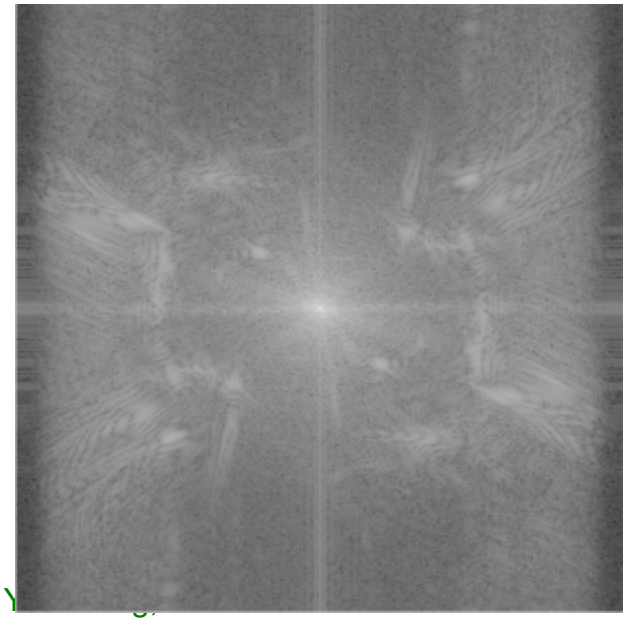
Which one below is the DFT of which one above?

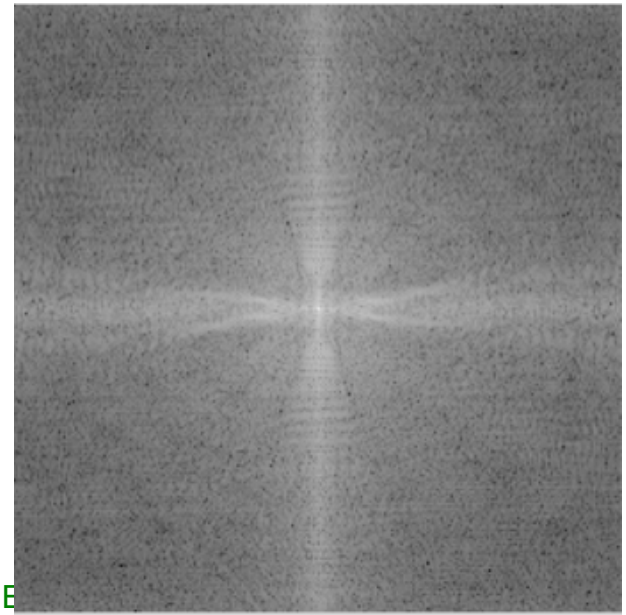


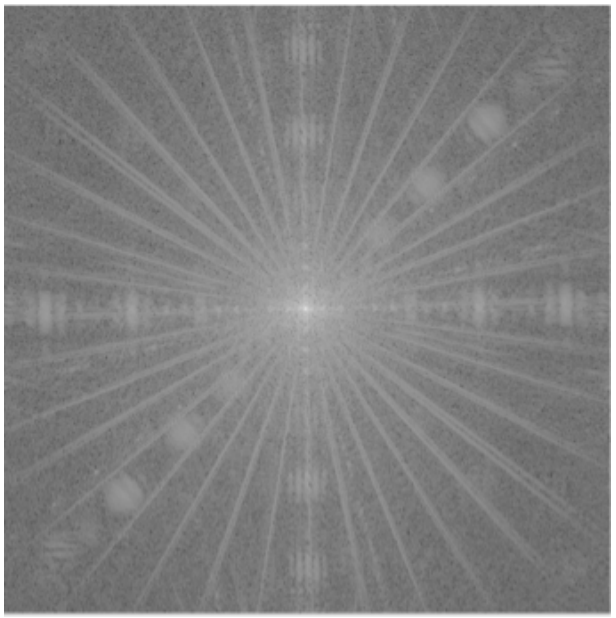












DSFT of Separable Signals

- Separable signal:
  - h(m,n)=hx(m) hy(n)
- 2D DSFT of separable signal = product of 1D DSFT of each 1D component
  - H(u,v)= Hx(u) Hy(v)
  - Hx(u): 1D FT of hx
  - Hy(v): 1D FT of hy

DSFT of Special Signals

- Constant <-> pulse at (0,0)
- Rectangle <-> 2D digital sinc
- Sinc <-> Rectangle (ideal low pass)
- Complex sinusoid with freq (u0,v0) <-> pulse at (u0,v0)
- …
- Can be shown easily by making use of the fact that the signal is separable

Using Separable Processing to Compute DSFT

- 3x3 averaging filter

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

- Recognizing that the filter is separable

$$H = 1/9\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = 1/3\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}1/3 = h_1 h_1^T$$

$$1/3\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} <-> H_1(v) = (1e^{j2\pi v} + 1 + 1e^{-j2\pi v})/3 = (1 + 2\cos 2\pi v)/3$$

$$H(u,v) = H_1(u)H_1(v) = (1+\cos 2\pi u)(1+\cos 2\pi v)/9$$

Using Separable Processing to Compute DSFT

- A separable signal (filter)
- $H = \frac{1}{9}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$
- Recognizing that the filter is separable

$$H = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} = h_x h_y^T$$

$$\begin{bmatrix} 1 & 0 & -1 \end{bmatrix} <-> Fy(v) = 1e^{j2\pi v} + 0 + (-1)e^{-j2\pi v} = 2j\sin 2\pi v$$

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} <-> Fx(u) = 1e^{j2\pi u} + 2 + e^{-j2\pi u} = 2 + 2\cos 2\pi u$$

$$F(u,v) = Fx(u)Fy(v) = 4j(1+\cos 2\pi u)\sin 2\pi v$$

What about rotation?

- No theoretical proof, however, roughly it is still true
- Rotation in space <-> Rotation in freq.
- Example:
- f(m,n)=δ(m) (horizontal line) <-> F(u,v) = δ(v) (vertical line)
- What about f(m,n)= δ(m-n) (diagonal line)?
  - <-> F(u,v)= δ(u+v) (antidiagonal line!)
  - Homework

Linear Convolution over Continuous Space

- 1D convolution

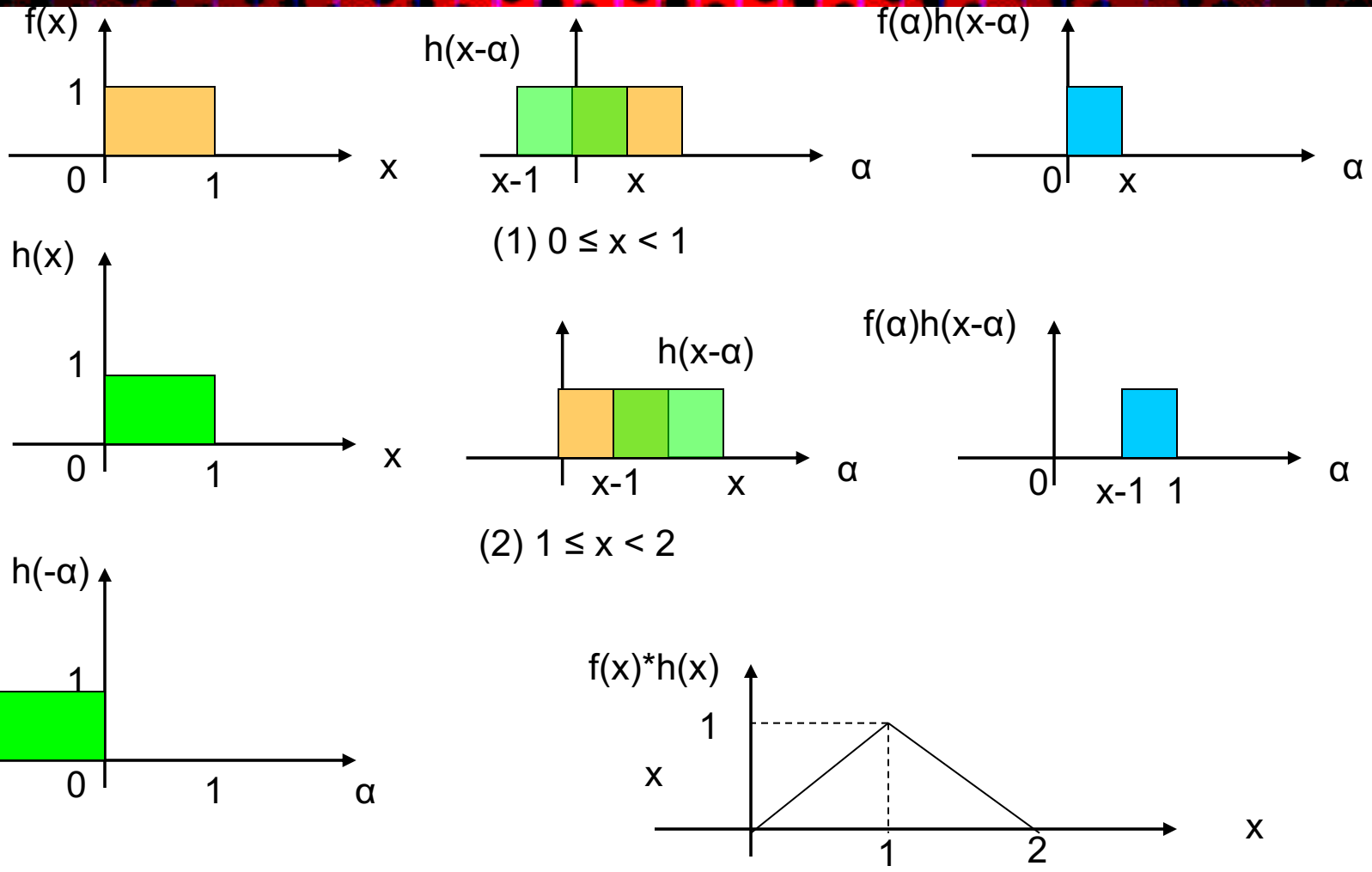
$$f(x)*h(x)=\int_{-\infty}^{\infty} f(x-\alpha)h(\alpha)d\alpha=\int_{-\infty}^{\infty} f(\alpha)h(x-\alpha)d\alpha$$

$$f(x)*\delta(x)=f(x),\quad f(x)*\delta(x-x_0)=f(x-x_0)$$

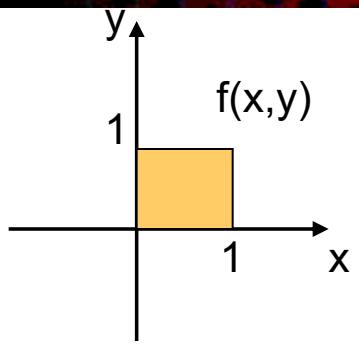
- 2D convolution

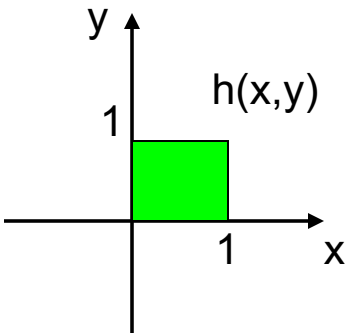
$$f(x,y)*h(x,y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-\alpha,y-\beta)h(\alpha,\beta)d\alpha d\beta$$
$$=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\alpha,\beta)h(x-\alpha,y-\beta)d\alpha d\beta$$

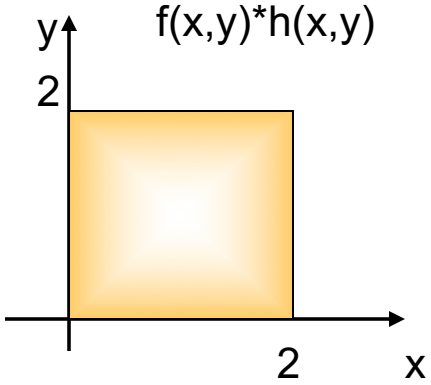
Examples of 1D Convolution

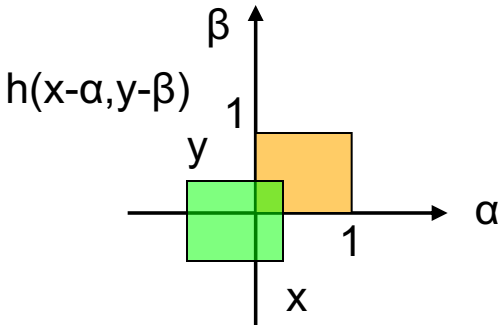


Example of 2D Convolution

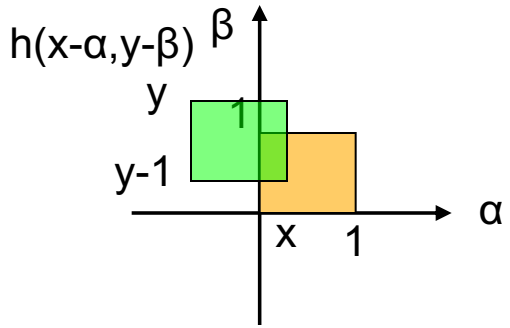




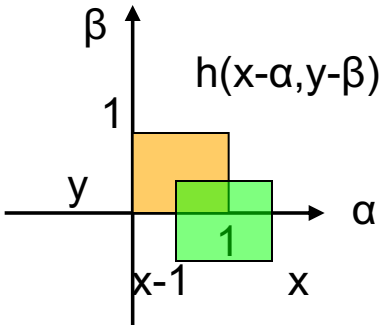




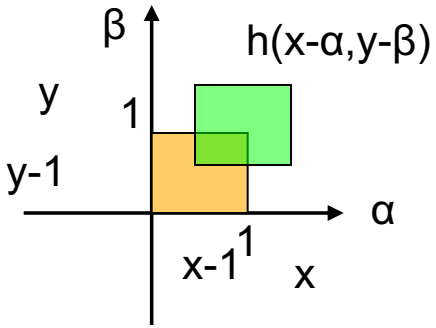
(1) 0<x≤1, 0<y≤1
g(x,y)=x*y



(2) 0<x≤1, 1<y≤2
g(x,y)=x*(2-y)



(3) 1<x≤2, 0<y≤1
g(x,y)=(2-x)*y



(4) 1<x≤2, 1<y≤2
g(x,y)=(2-x)*(2-y)

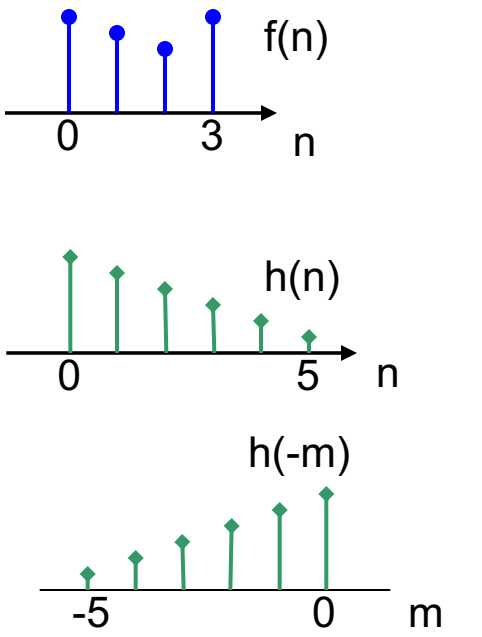
Convolution of 1D Discrete Signals

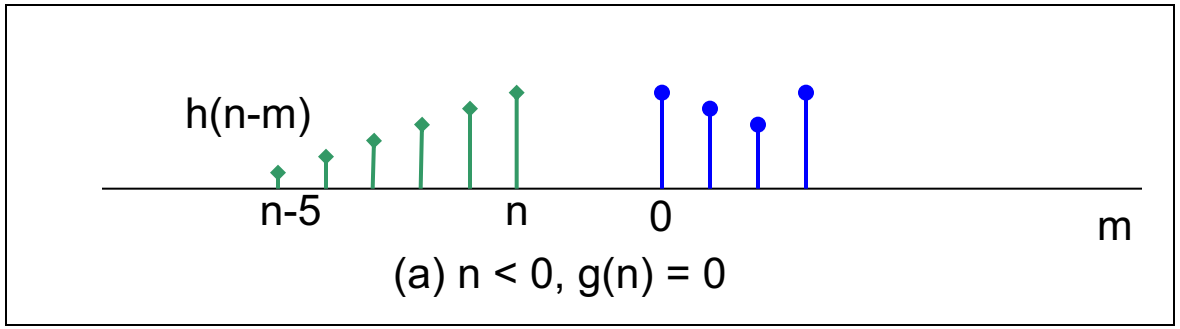
- Definition of convolution

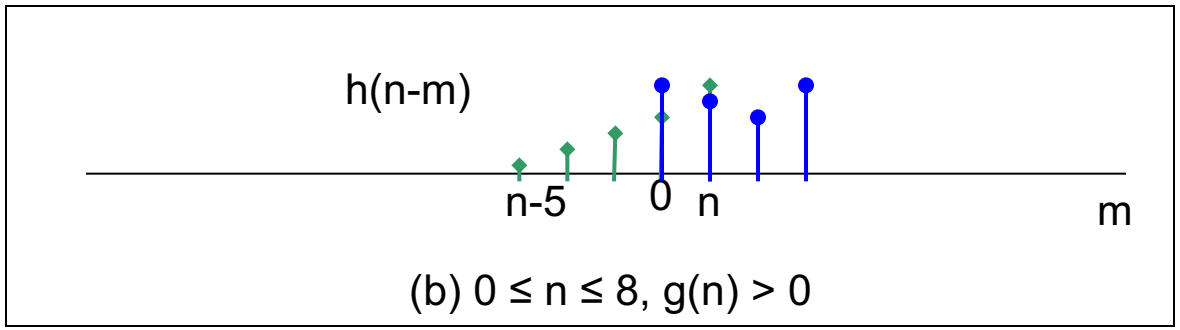
$$f(n) * h(n) = \sum_{m=-\infty}^{\infty} f(n-m)h(m) = \sum_{m=-\infty}^{\infty} f(m)h(n-m)$$

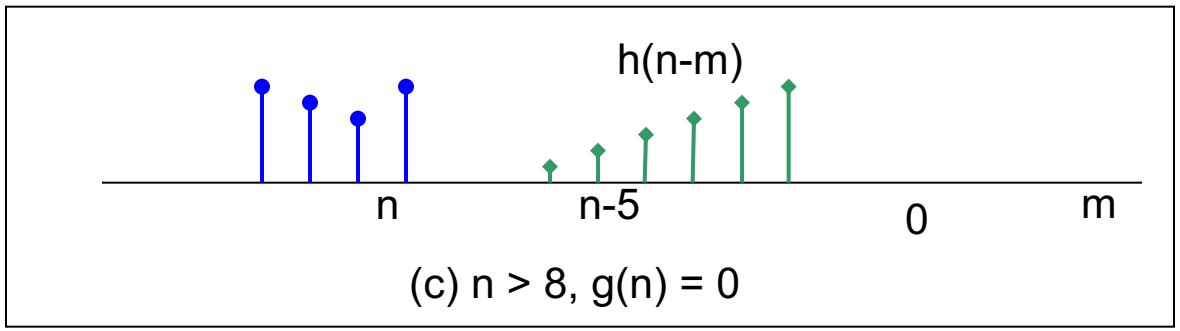
- The convolution with h(n) can be considered as the weighted average in the neighborhood of f(n), with the filter coefficients being the weights
  - sample f(n-m) is multiplied by h(m)
- The filter h(n) is the impulse response of the system (i.e. the output to an input that is an impulse)
- Signal length before and after filtering
  - Original signal length: N
  - Filter length: K
  - Filtered signal length: N+K-1

Example of 1D Discrete Convolution









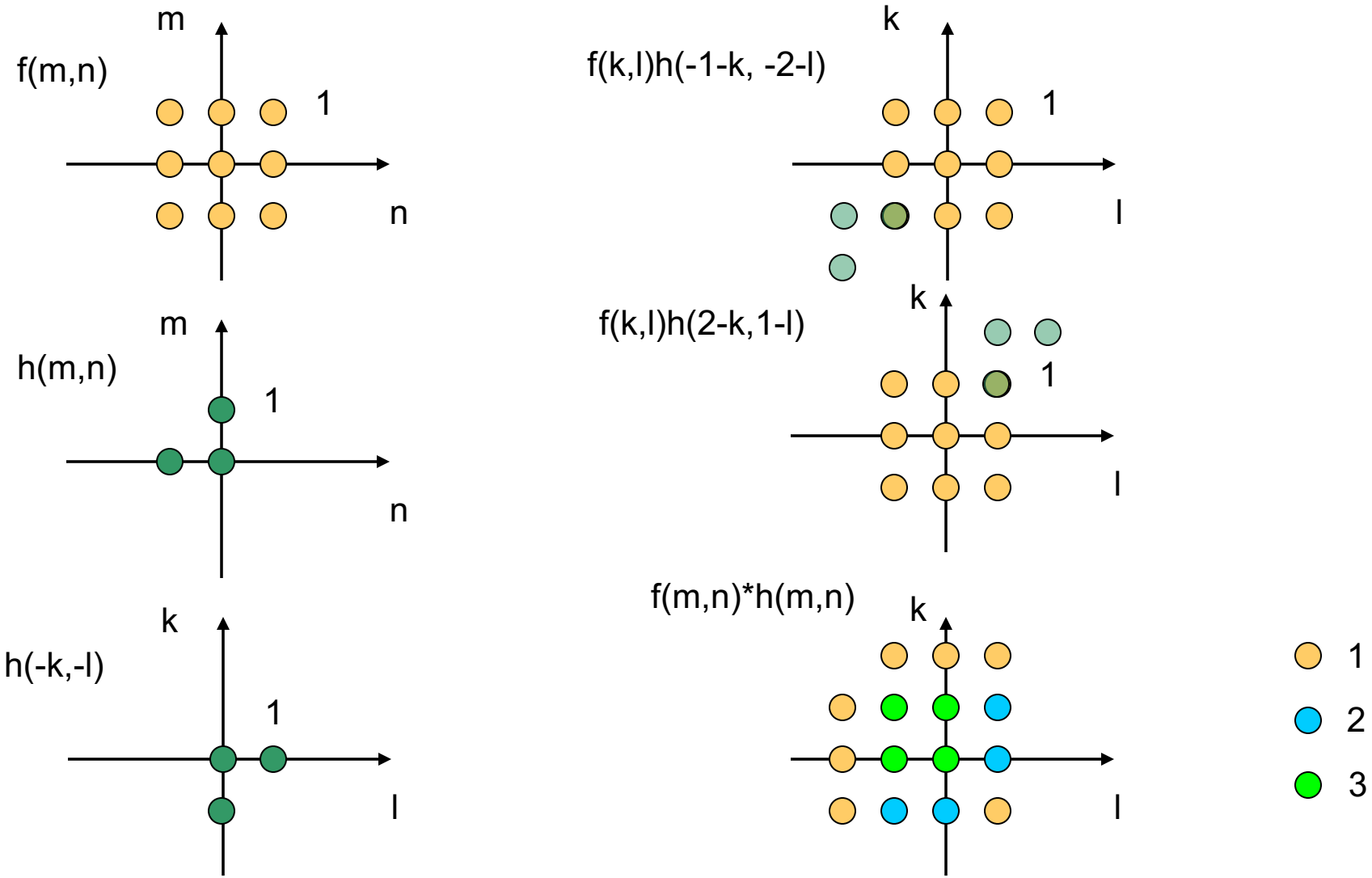
Convolution of 2D Discrete Signals

$$g(m,n) = f(m,n) * h(m,n) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f(m-k,n-l)h(k,l)$$

$$= \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f(k,l)h(m-k,n-l)$$

- Each new pixel g(m,n) is a weighted average of its neighboring pixels in the original image:
  - Pixel f(m-k,n-l) is weighted by h(k,l)
- We may use matrices to represent both signal (F) and filter (H) and use F*H to denote the convolution

Example of 2D Discrete Convolution



Example: Averaging and Weighted Averaging

100	100	100	100	100
100	200	205	203	100
100	195	200	200	100
100	200	205	195	100
100	100	100	100	100

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

1	1	1
1	1	1
1	1	1

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

1	2	1
2	4	2
1	2	1

100	100	100	100	100
100	144	167	145	100
100	167	200	168	100
100	144	166	144	100
100	100	100	100	100

100	100	100	100	100
100	156	176	158	100
100	174	201	175	100
100	156	175	156	100
100	100	100	100	100

Example

Original image

Average filtered image

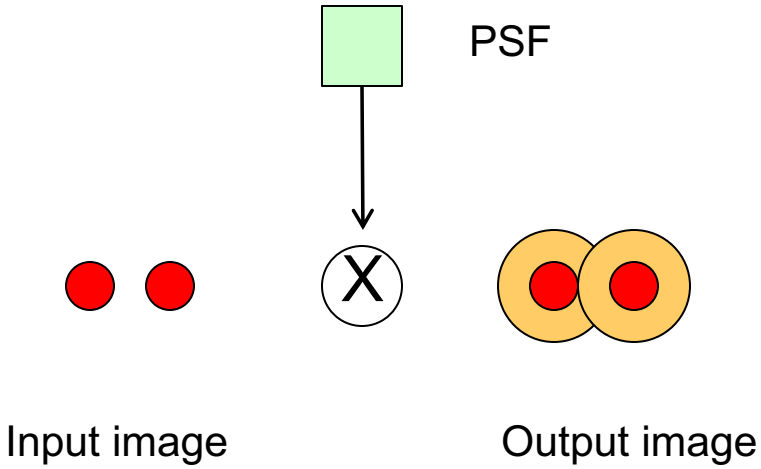
Weighted Average filtered image

What does h(m,n) mean?

- Any operation that is linear and shift invariant can be described by a convolution with a filter h(m,n)!
- h(m,n) is the impulse response of the system (i.e. output of the system to an impulse input)
- Better known as point spread function, indicating how a single point (i.e. an impulse) in the original image would be spread out in the output image

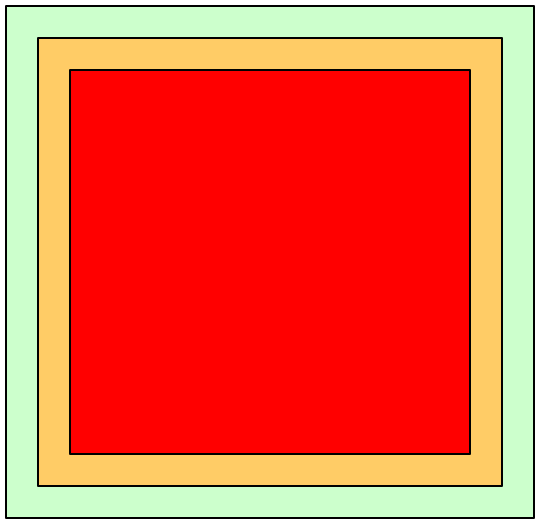
Point Spread Function

- The point spread function of an imaging system (e.g. a camera or a medical imaging system) describes the resolution of the system:
  - Two object points cannot be separated if they are closer than the support of the point spread function!



Boundary of Filtered Image

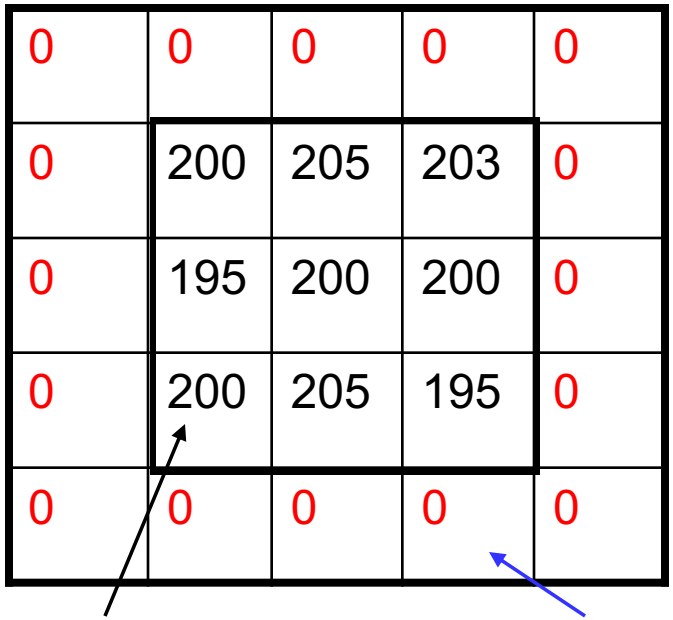
- An image of size MxN convolving with a filter of size KxL will yield an image of size (M+K-1,N+L-1)
- If the filter is symmetric with (2k+1)x(2k+1) samples, the convolved image should have an extra boundary of thickness k on each side outside the original image (outer boundary, green). The values along the outer boundary depend on the assumed pixel values outside the original image
- Filtered values in the inner boundary (orange) of k pixels inside the original image also depend on the assumed pixel values outside the original image
- Filtered values in the valid region (red) only depend on the available image values.



Orange+Red: original image size
Red: Valid part of the output image (does not depend on pixels outside the original image)
Orange: inner boundary
Green: outer boundary

Boundary Treatment: Zero Padding

- MxN image convolved with KxL filter -> (M+K-1)x(N+L-1) image
- Filtered values at the outer and inner boundary of the output image depend on the assumed value of the pixels outside the original image
- Zero padding: Assuming pixel values are 0 outside the original image



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

1	1	1
1	1	1
1	1	1

Outer boundary

200/9	(200+205)/9	...	…	…
(200+195)/9	(200+205+195+200)/9	…	…	…
…	---	…	…	…
…	…	…	…	…
…	…	…	…	…

Inner boundary

Actual image pixels

Extended pixels

Boundary Treatment: Symmetric Extension

- Assuming pixels values outside the image are the same as their mirroring pixels inside the image
  - Lead to less discontinuity in the filtered image along the outer and inner boundaries

200	200	205	203	203
200	200	205	203	203
195	195	200	200	200
200	200	205	195	195
200	200	205	195	195

Actual image pixels

Extended pixels

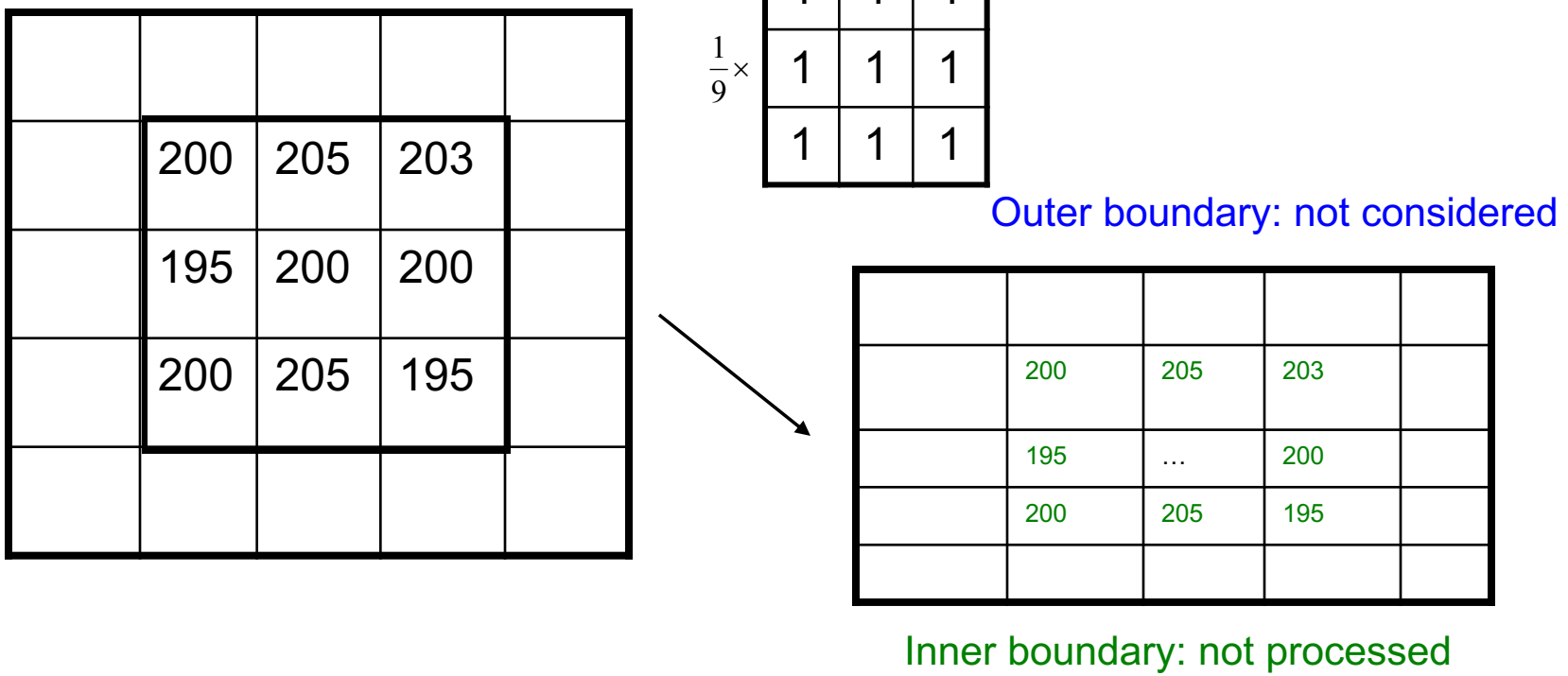
$\frac{1}{9}\times$

1	1	1
1	1	1
1	1	1

(200+200+200+200)/9	(200+200+205+200+200+205)/9	...	…	…
(200+200+200+200+195+195)/9	(200+200+205+200+200+205+195+195+200)/9	…	…	…
…	---	…	…	…
…	…	…	…	…
…	…	…	…	…

Simplified Boundary Treatment

- Filtered image size=original image size
- Only compute in the valid region.
  - Assign 0 (for high/band pass filters) or keep original value (for low pass) in the inner boundary
- Resulting image is correct only in the “valid” region



Example: Simplified Boundary Treatment

100	100	100	100	100
100	200	205	203	100
100	195	200	200	100
100	200	205	195	100
100	100	100	100	100

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

1	1	1
1	1	1
1	1	1

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

1	2	1
2	4	2
1	2	1

100	100	100	100	100
100	144	167	145	100
100	167	200	168	100
100	144	166	144	100
100	100	100	100	100

100	100	100	100	100
100	156	176	158	100
100	174	201	175	100
100	156	175	156	100
100	100	100	100	100

Sample Matlab Program (With Simplified Boundary Treatment)

```
% readin bmp file
x = imread('lena.bmp');
[xh xw] = size(x);
y = double(x);
```

```
% define 2D filter
h = ones(5,5)/25;
[hh hw] = size(h);
hhh = (hh - 1) / 2;
hhw = (hw- 1) / 2;
```

```
% linear convolution, assuming the filter is non-separable (although this example filter is separable)
z = y; %or z=zeros(xh,xw) if not low-pass filter
for m = hhh + 1:xh - hhh,
                %skip first and last hhh rows to avoid boundary problems
   for n = hhw + 1:xw - hhw,
                %skip first and last hhw columns to avoid boundary problems
      tmpy = 0;
      for k = -hhh:hhh,
         for l = -hhw:hhw,
            tmpv = tmpv + y(m - k,n – l)* h(k + hhh + 1, l + hhw + 1);
                        %h(0,0) is stored in h(hhh+1,hhw+1)
         end
      end
      z(m, n) = tmpv;
%for more efficient matlab coding, you can replace the above loop with
                z(m,n)=sum(sum(y(m-hhh:m+hhh,n-hhw:n+hhw).*h))
   end
end
```

Separable Filters

- A filter is separable if $h(m, n)=h_x(m)h_y(n)$.
- Matrix representation

$$H = h_x h_y^T$$

  - Where h_x and h_y are column vectors
- Example

$$H_x = \frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}; \quad H_y = \frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

Separable Filtering

- If h(m,n) is separable, the 2D convolution can be accomplished by first applying 1D filtering along each row using $h_y(n)$, and then applying 1D filtering to the intermediate result along each column using the filter $h_x(n)$ (or column filtering followed by row filtering)
- Proof

$$
\begin{aligned}
f(m,n)*h(m,n) &= \sum_k \sum_l f(m-k,n-l)h_x(k)h_y(l) \\
&= \sum_k \left(\sum_l f(m-k,n-l)h_y(l) \right) h_x(k) \\
&= \sum_k g_y(m-k,n)h_x(k) \\
&= (f(m,n)*h_y(n))*h_x(m)
\end{aligned}
$$

Results Using Sobel Filters

Original image

Filtered image by H_x

Filtered image by H_y

$$H_x = \frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}; \quad H_y = \frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

What do Hx and Hy do?

Computation Cost: Non-Separable vs. Separable Filtering

- Suppose: Image size M*N, filter size K*L. Ignoring outer boundary pixels
- Non-separable filtering
  - Weighted average on each pixel: K*L mul; K*L – 1 add.
  - For all pixels: M*N*K*L mul; M*N*(K*L-1) add.
  - When M=N, K=L: M^2K^2 mul + $M^2(K^2-1)$ add.
- Separable filtering:
  - Each pixel in a row: L mul; L-1 add.
  - Each row: N*L mul; N*(L-1) add.
  - M rows: M*N*L mul; M*N*(L-1) add.
  - Each pixel in a column: K mul; K-1 add.
  - Each column: M*K mul; M*(K-1) add.
  - N columns: N*M*K mul; N*M*(K-1) add.
  - Total: M*N*(K+L) mul; M*N*(K+L-2) add.
  - When M=N, K=L: $2M^2K$ mul; $2M^2(K-1)$ add.
    - Significant savings if K (and L) is large!

MATLAB Function: conv2( )

- Matlab functions
  - C=conv2(H,F,shape)
    - Shape='full' (Default): C includes both outer and inner boundary, using zero padding
    - Shape="same": C includes the inner boundary, using zero padding
    - Shape="valid': C includes the convolved image without the inner boundary, computed without using pixels outside the original image
  - C=conv2(h1,h2,F,shape)
    - Separable filtering with h1 for column filtering and h2 for row filtering

Python Function: conv2( )

- Python functions
  - C=scipy.signal.convolve2d(*in1*, *in2*, *mode)*
    - *mode* ='full' (Default): C includes both outer and inner boundary, using zero padding
    - Shape="same": C includes the inner boundary, using zero padding
    - Shape="valid': C includes the convolved image without the inner boundary, computed without using pixels outside the original image

Notes about implementation

- Input image needs to be converted to float or double
  - MATLAB imread return unsigned characters
  - Never do numerical operations on unsigned character type!
  - Python cv2.imread return unsigned integers
- Output image value may not be in the range of (0,255) and may not be integers
- To display or save the output image properly
  - Renormalize to (0,255) using a two-pass operation
    - First pass: save directly filtered value in an intermediate floating-point array
    - Second pass: find minimum and maximum values of the intermediate image, renormalize to (0,255) and rounding to integers
      - F= round((F1-fmin)*255/(fmax-fmin))
  - To display the unnormalized image directly in MATLAB, use imagesc(img). Or imshow(img, [ ]).
  - In python: cv2.imshow('fig_name',img), cv2.waitKey() or plt.imshow(img) (check Lecture note 1: ContrastEnhancement page 24)

Convolution Theorem

- Convolution Theorem

$$f * h \quad \Leftrightarrow \quad F \times H, \qquad f \times h \quad \Leftrightarrow \quad F * H$$

- Proof

$$g(m,n) = f(m,n) * h(m,n) = \sum_k \sum_l f(m-k, n-l) h(k,l)$$

FT on both sides

$$G(u,v) = \sum_{m,n} \sum_{k,l} f(m-k, n-l) h(k,l) e^{-j2\pi(mu+nv)}$$

$$= \sum_{m,n} \sum_{k,l} f(m-k, n-l) e^{-j2\pi((m-k)u+(n-l)v)} h(k,l) e^{-j2\pi(ku+lv)}$$

$$= \sum_{m,n} f(m-k, n-l) e^{-j2\pi((m-k)u+(n-l)v)} \sum_{k,l} h(k,l) e^{-j2\pi(ku+lv)}$$

$$= \sum_{m',n'} f(m',n') e^{-j2\pi(m'u+n'v)} \sum_{k,l} h(k,l) e^{-j2\pi(ku+lv)}$$

$$= F(u,v) \times H(u,v)$$

Another view of convolution theorem

$$f(m,n) * h(m,n) \quad \Leftrightarrow \quad F(u,v)H(u,v)$$

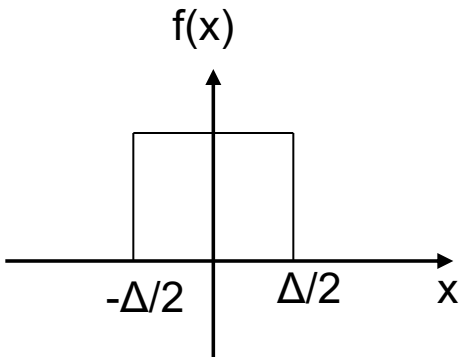
- F(u,v)H(u,v) = Modifying the signal's each frequency component' complex magnitude F(u,v) by H(u,v)

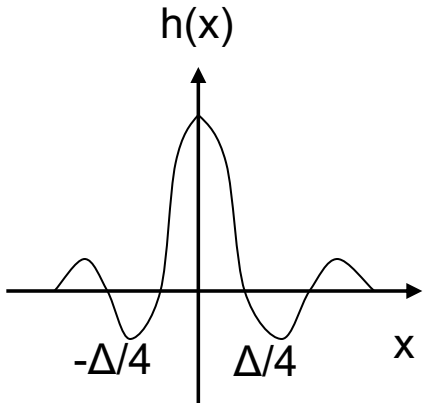
$$f(m,n) = \int_{-1/2}^{1/2}\int_{-1/2}^{1/2} F(u,v)e^{j2\pi(mu+nv)}dudv$$

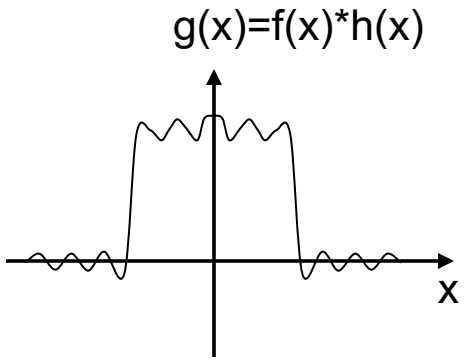
$$g(m,n) = \int_{-1/2}^{1/2}\int_{-1/2}^{1/2} F(u,v)H(u,v)e^{j2\pi(mu+nv)}dudv$$

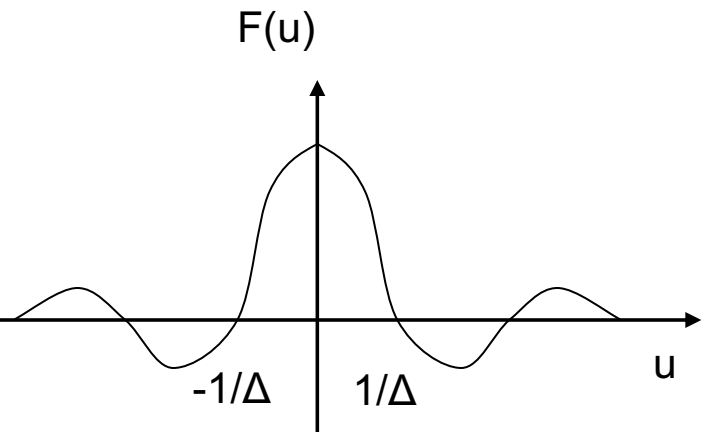
- H(u,v) is also called Frequency Response of the 2D LSI system
  - exp{2π(um+vn)} -> H(u,v) exp{2π(um+vn)} = |H(u,v)|exp{2π(um+vn)+φ(H(u,v))}
  - Sinusoid (or complex exponential) input -> sinusoid (complex exponential) output!
  - H(u,v) describes how the magnitude and phase of a sinusoid input with frequency (u,v) are changed!

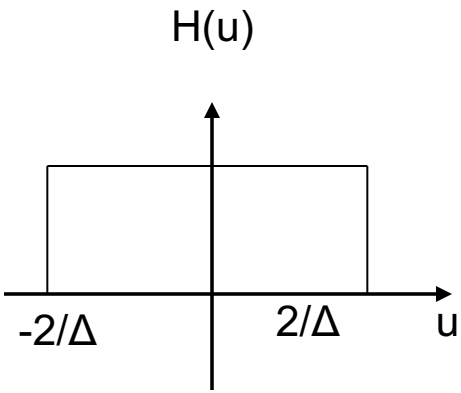
Explanation of Convolution in the Frequency Domain

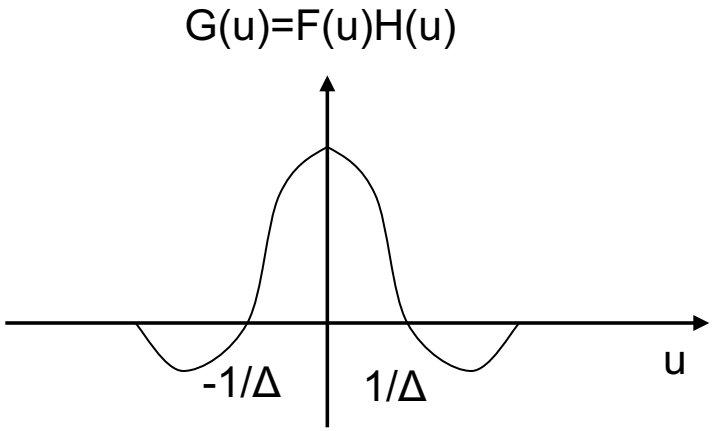








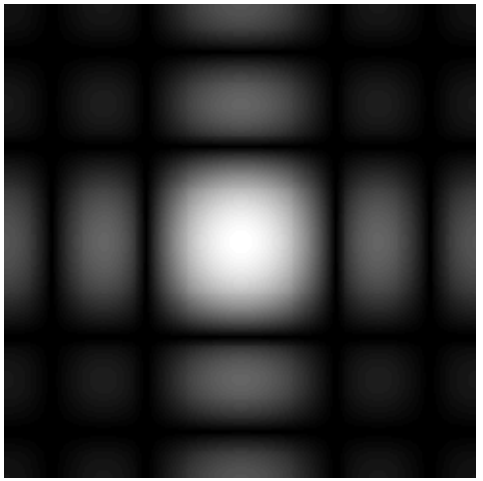




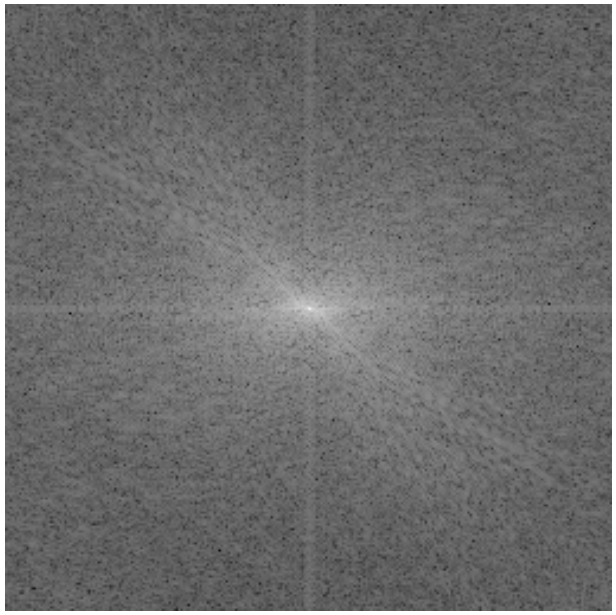
Example

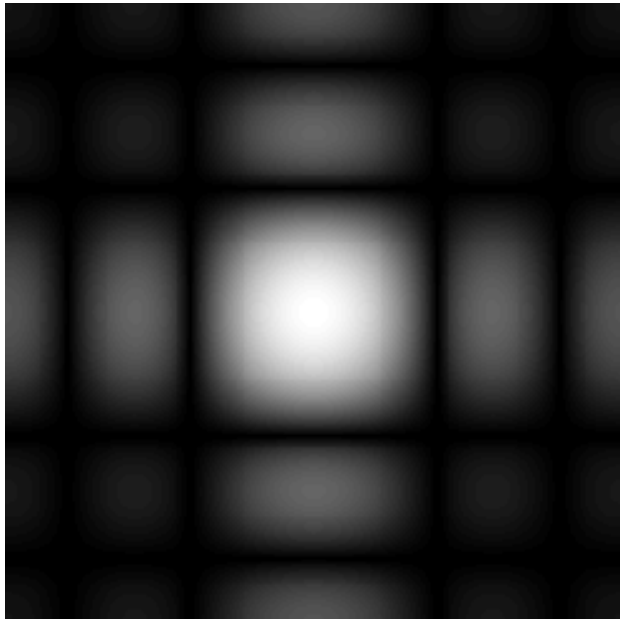
- Given a 2D filter, determine its frequency response. Apply to a given image, show original image and filtered image in pixel and freq. domain

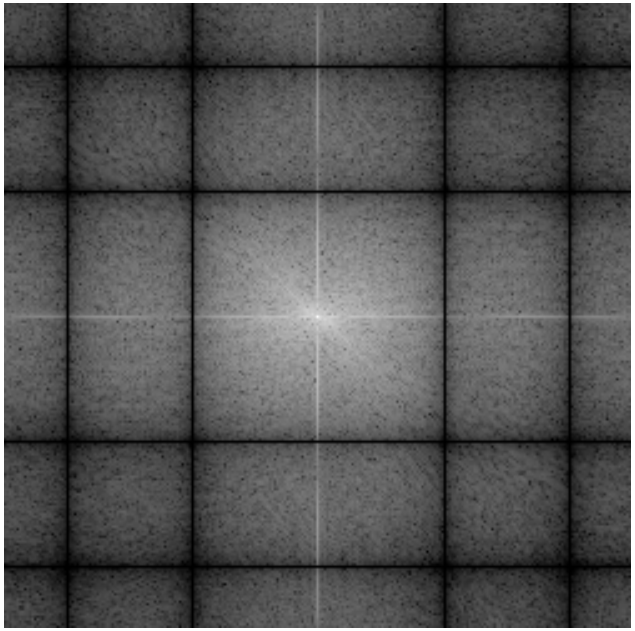
$$h = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$



$$h = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$



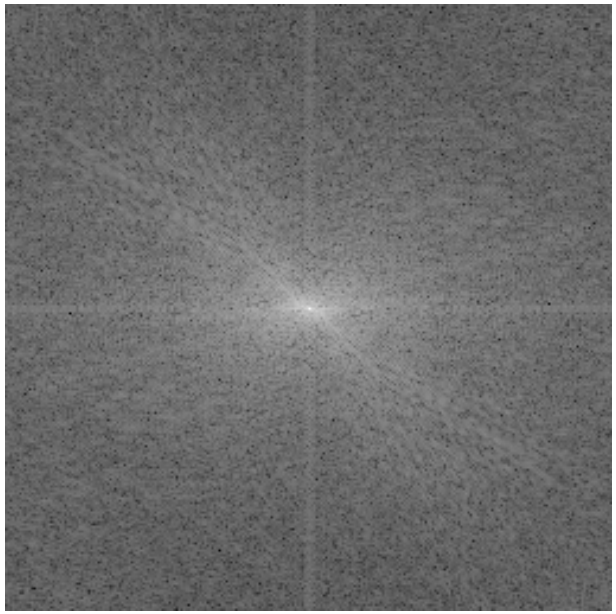


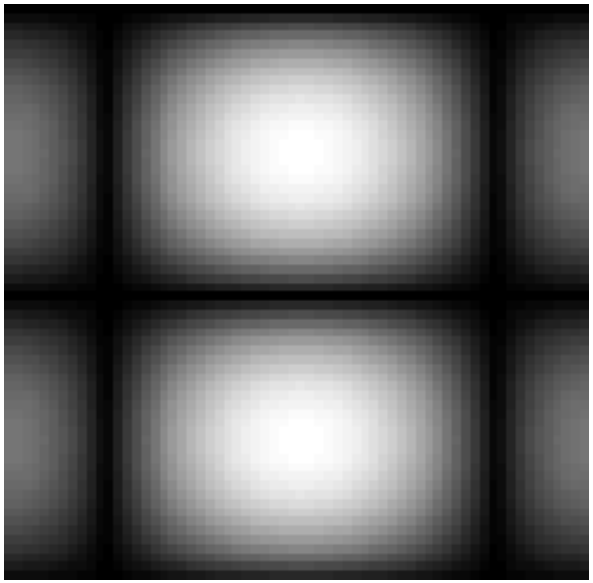


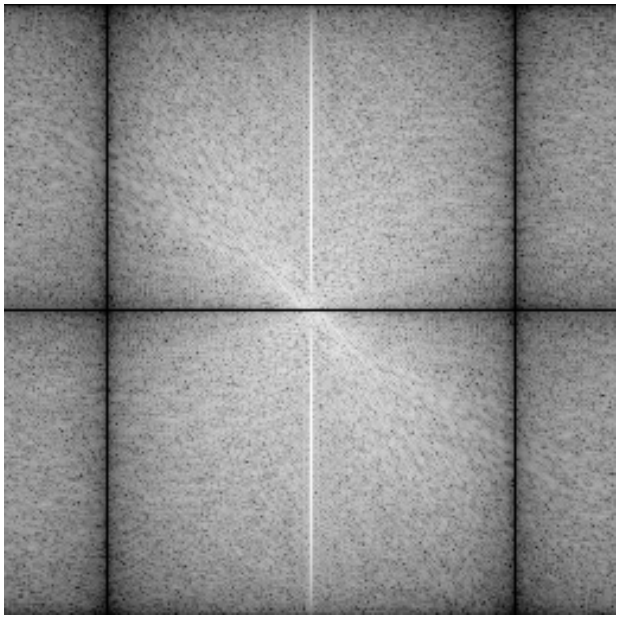
Matlab Program Used

```
x = imread('lena256.bmp');
figure(1); imshow(x);
f = double(x);
ff=abs(fft2(f));
figure(2); imagesc(fftshift(log(ff+1))); colormap(gray);truesize;axis off;
h = ones(5,5)/9;
hf=abs(freqz2(h));
figure(3);imagesc((log(hf+1)));colormap(gray);truesize;axis off;
y = conv2(f, h);
figure(4);imagesc(y);colormap(gray);truesize;axis off;
yf=abs(fft2(y));
figure(5);imagesc(fftshift(log(yf+1)));colormap(gray);truesize;axis off;
```

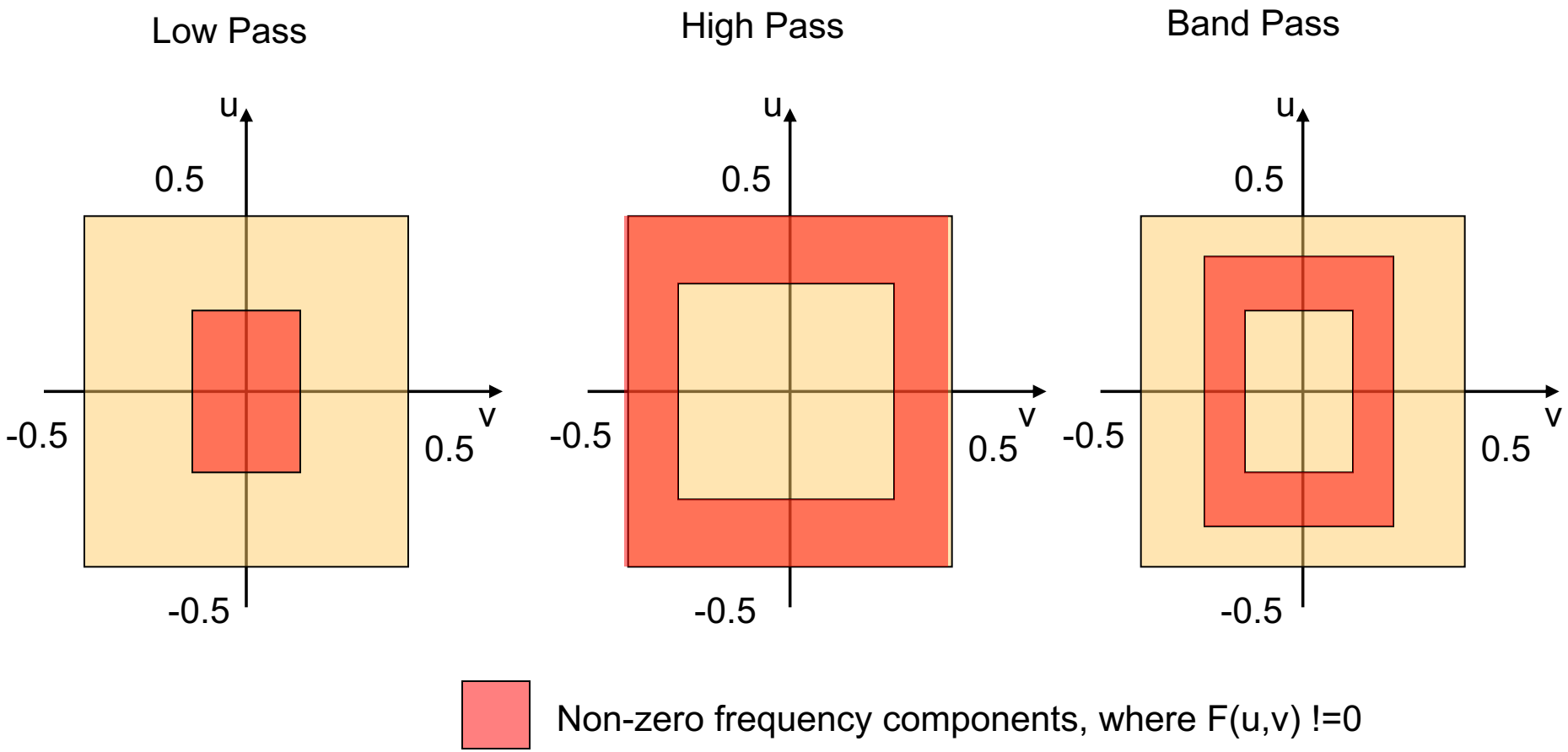
$$H_1 = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$







Typical Filter Types

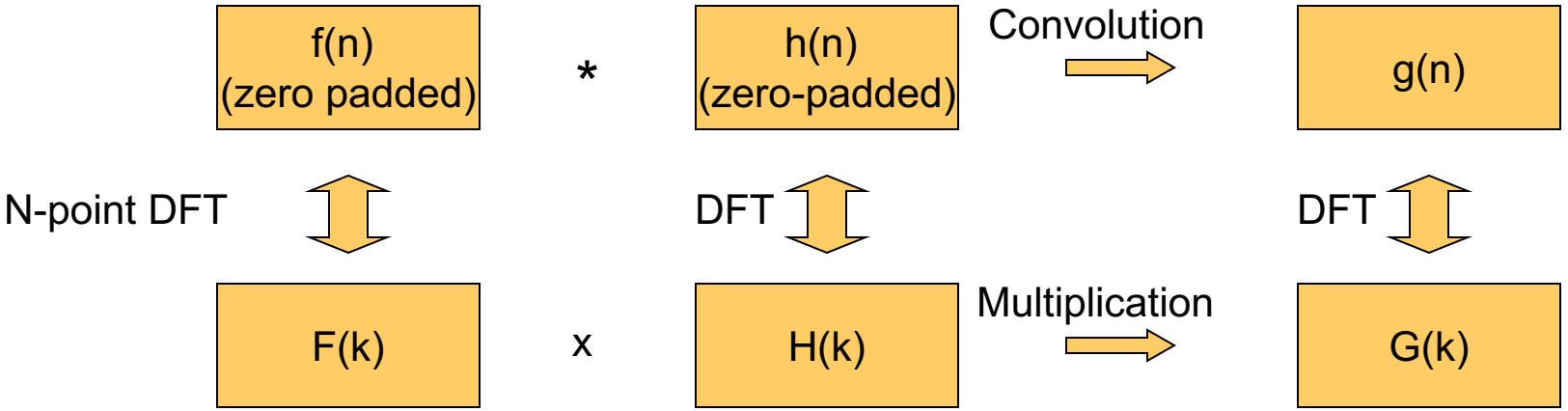


Filter Design

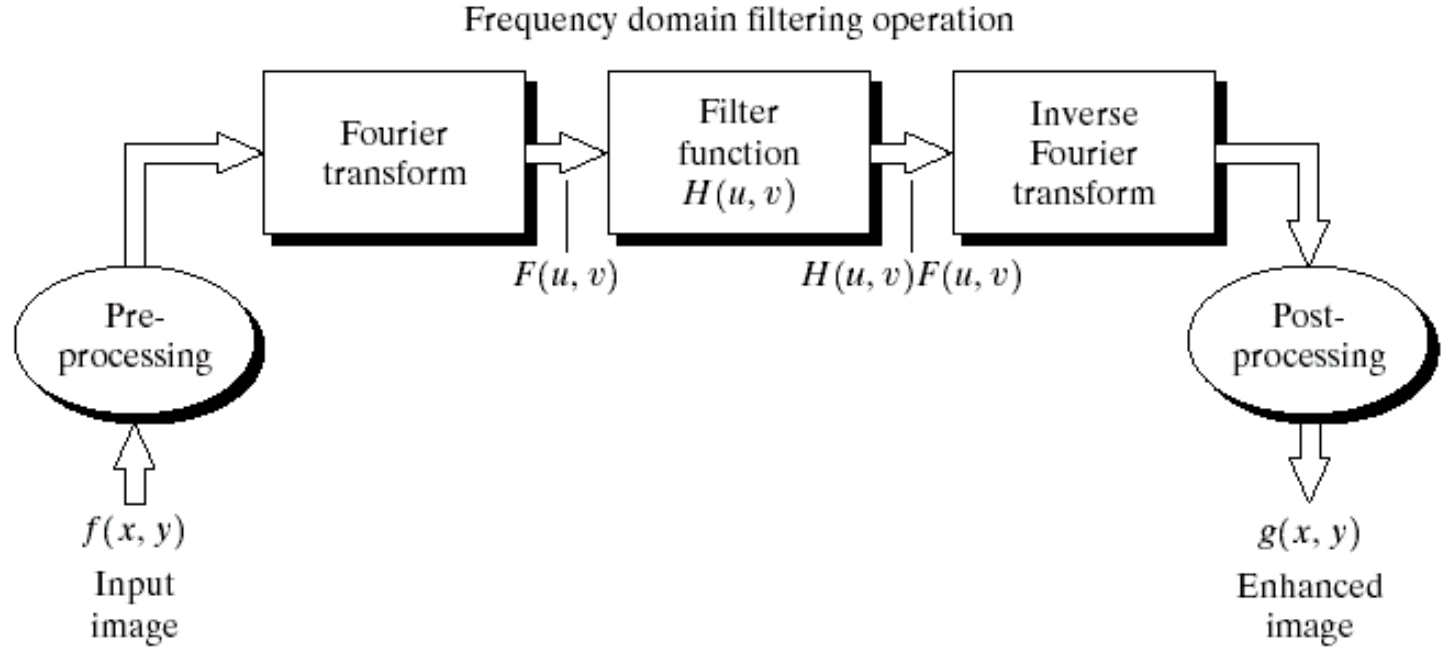
- The ideal low-pass, high-pass, band-pass filters have infinite length in the spatial domain
  - Very sharp transition in freq -> very long filter in space
- Filter design
  - Start with ideal frequency response
  - Apply a window to smooth the transition
  - Apply optimization technique to find the filter to match the windowed frequency response
  - FIR vs. IIR filters
  - Linear phase vs. non-linear phase
  - IIR filters can realize same transition band/attenuation with much shorter filter order than FIR!
  - Non-linear phase of IIR can be overcome through filtering forward and backward (filtfilt( ))!
  - 1D filter design method can be used to design separable 2D filters
  - Not a focus of this class

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.
  - H(k) can be precalculated



Computing 2D Convolution Using 2D DFT



Zero padding

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, which differs from linear convolution at the inner boundaries. Only correct in the valid region.

Image Filtering Using DFT

- Typically DFT size=image size. This corresponds to circular convolution, which differs from linear convolution at the inner boundaries. Only correct in the valid region.
- Circular convolution

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

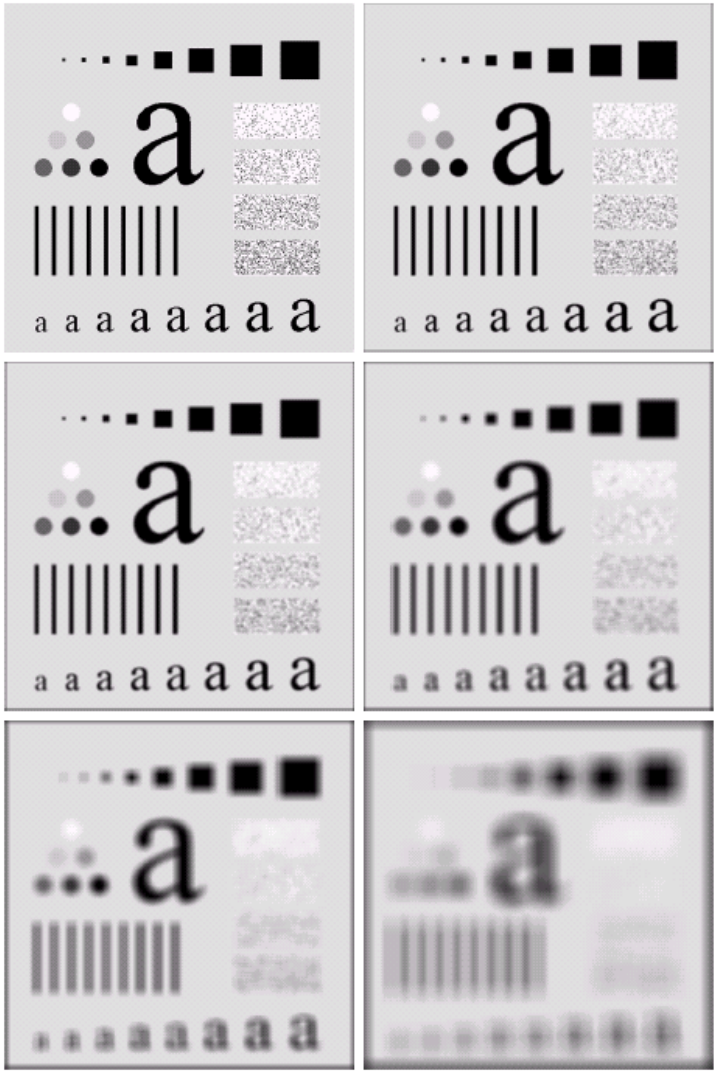
$$f(n) \otimes h(n) \Leftrightarrow F_N(k) H_N(k)$$

- Image filters typically have short length to avoid boundary problems and ringing effect.
- For image filtering with short filters, it is more efficient to do convolution in the spatial domain.

Typical Image Processing Tasks

- Noise removal (image smoothing): low pass filter
- Edge detection: high pass filter followed by non-linear processing (e.g. peak detection)
- Image sharpening: high emphasis filter
- …
- In image processing, we rarely use very long filters
- We compute convolution directly, instead of using 2D FFT
- Filter design: For simplicity we often use separable filters, and design 1D filter based on the desired frequency response in 1D
- We do not focus on filter design in this class

Noise Removal Using Averaging Filters

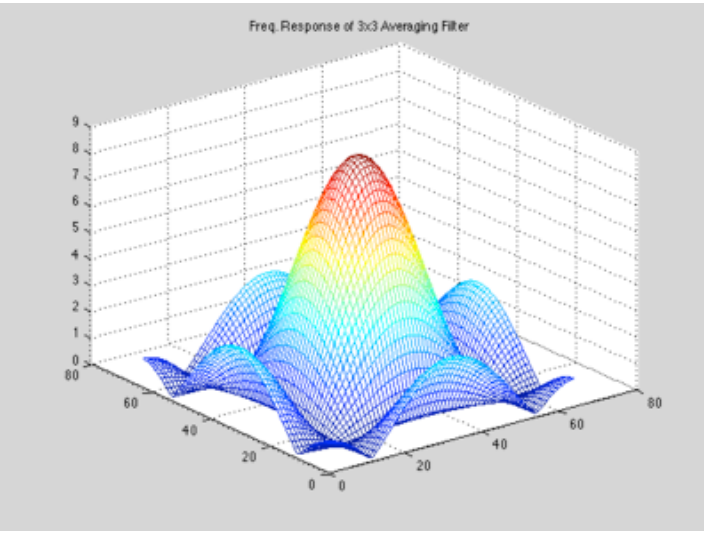


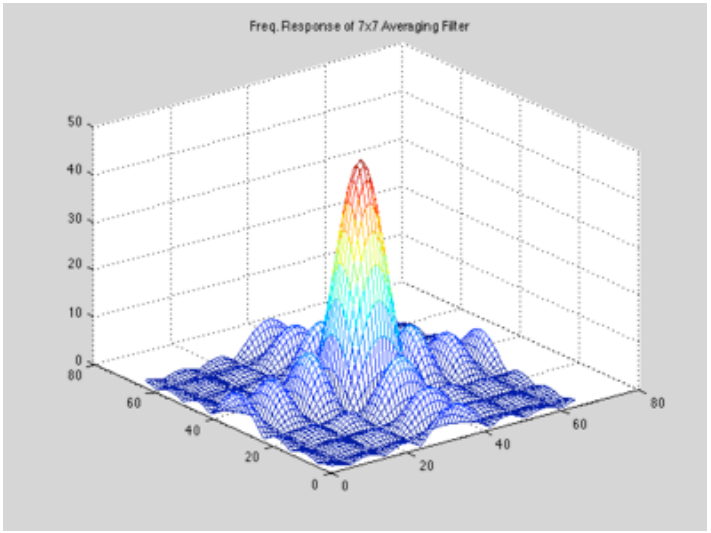
a b
c d
e f

FIGURE 3.35 (a) Original image, of size 500 × 500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $n = 3, 5, 9, 15$, and 35, respectively. The black squares at the top are of sizes 3, 5, 9, 15, 25, 35, 45, and 55 pixels, respectively; their borders are 25 pixels apart. The letters at the bottom range in size from 10 to 24 points, in increments of 2 points; the large letter at the top is 60 points. The vertical bars are 5 pixels wide and 100 pixels high; their separation is 20 pixels. The diameter of the circles is 25 pixels, and their borders are 15 pixels apart; their gray levels range from 0% to 100% black in increments of 20%. The background of the image is 10% black. The noisy rectangles are of size 50 × 120 pixels.

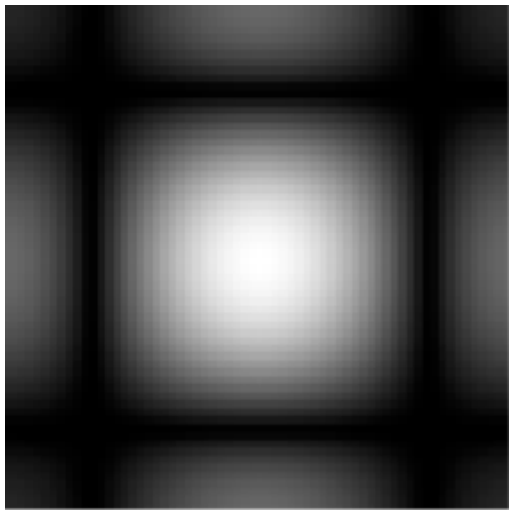
Window size controls tradeoff between noise removal power and blurring

Freq. Response Corresponding to Averaging Filters of Different Sizes

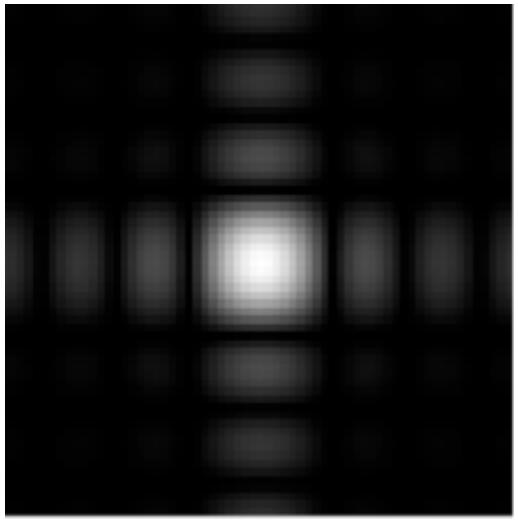




3x3 filter



7x7 filter



```
H=ones(3,3);
Hf=freqz2(H);
figure(1);
mesh(abs(Hf));
title('Freq. Response of 3x3
Averaging Filter');
figure(2);
imshow(abs(Hf),[])
```

Gaussian Filter in Continuous Space

- FT of Gaussian still a Gaussian Function!
- 1D Gaussian filter

$$\exp\left\{-\frac{x^2}{2\sigma^2}\right\} \Leftrightarrow \exp\left\{-\frac{u^2}{2\beta^2}\right\}, \beta = \frac{1}{2\pi\sigma}$$

- 2D Gaussian filter

$$\exp\left\{-\frac{x^2+y^2}{2\sigma^2}\right\} \Leftrightarrow \exp\left\{-\frac{u^2+v^2}{2\beta^2}\right\}, \beta = \frac{1}{2\pi\sigma}$$

- Note that STD in freq. β inversely related to STD σ in space

Gaussian Filter: from continuous to discrete space

- Analog form: STD σ controls the smoothing strength

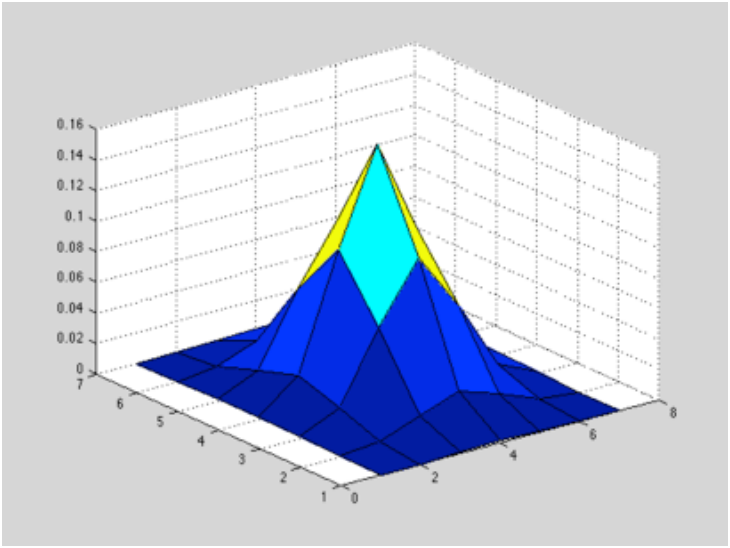
$$h(x,y) = \alpha \exp\left\{-\frac{x^2+y^2}{2\sigma^2}\right\},$$

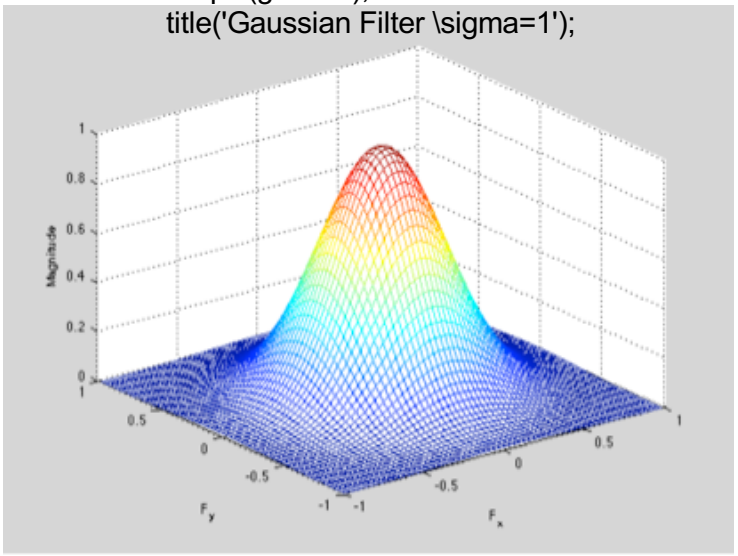
- Take samples, truncate after a few STD, normalize the sum to 1. Usually $\sigma >= 1$
- Size of mask nxn, n>=5σ, odd
  - Ex: $\sigma=1$, $n=7$.
  - Show filter mask,
  - Show frequency response
- Essentially a weighted average filter with decreasing weights away from the center
- A good filter for removing noise. σ should be chosen based on noise STD.

Ex: 7x7 Gaussian Filter Generation

0.0000	0.0002	0.0011	0.0018	0.0011	0.0002	0.0000
0.0002	0.0029	0.0131	0.0216	0.0131	0.0029	0.0002
0.0011	0.0131	0.0586	0.0966	0.0586	0.0131	0.0011
0.0018	0.0216	0.0966	0.1592	0.0966	0.0216	0.0018
0.0011	0.0131	0.0586	0.0966	0.0586	0.0131	0.0011
0.0002	0.0029	0.0131	0.0216	0.0131	0.0029	0.0002
0.0000	0.0002	0.0011	0.0018	0.0011	0.0002	0.0000

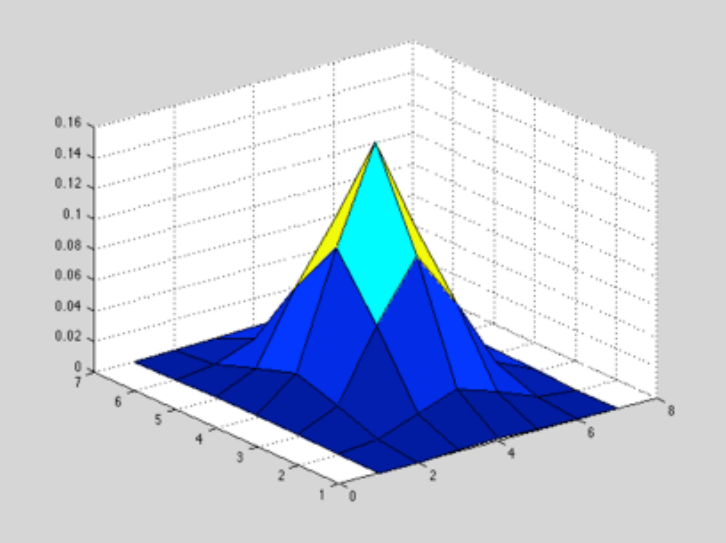
```
function gauss(s)

x=[-3.0:1.0:3.0];
gauss=exp(-x.^2/(2*s^2));
gauss2=gauss'*gauss;
gauss2=gauss2/(sum(sum(gauss2)));
H=gauss2;
disp(H);
figure(1);
surf(gauss2);
figure(2);
freqz2(gauss2);
title('Gaussian Filter \sigma=1');
```

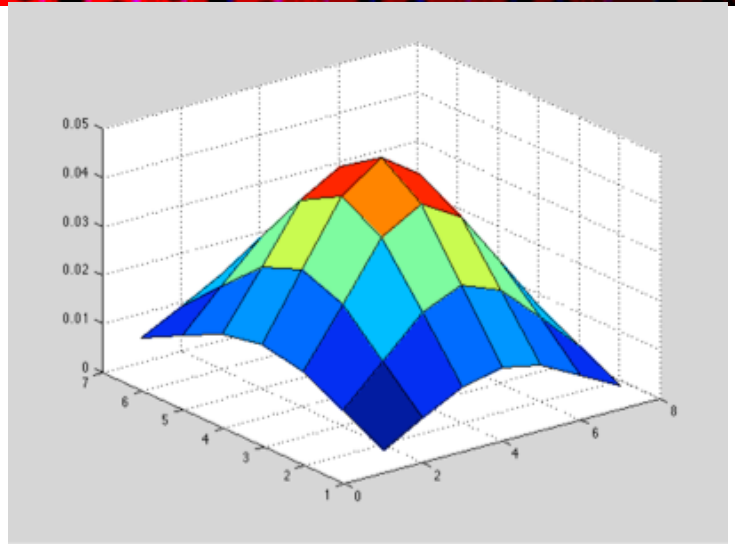




Gaussian Filter in Space and Freq.

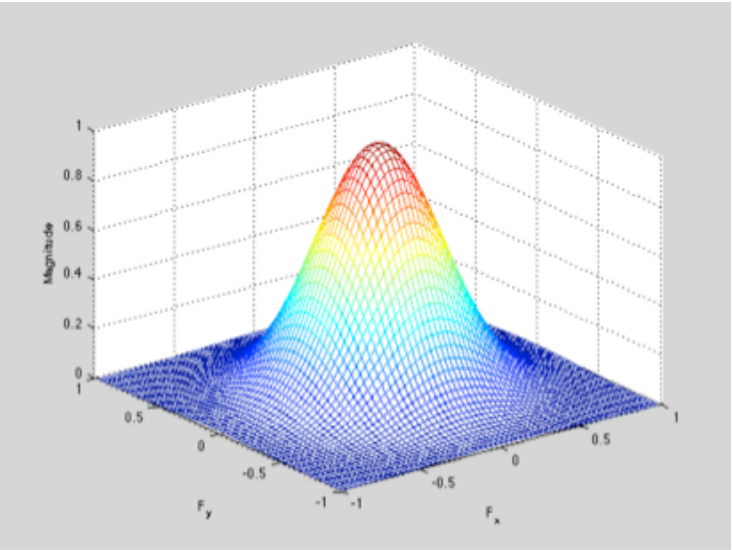
σ=1



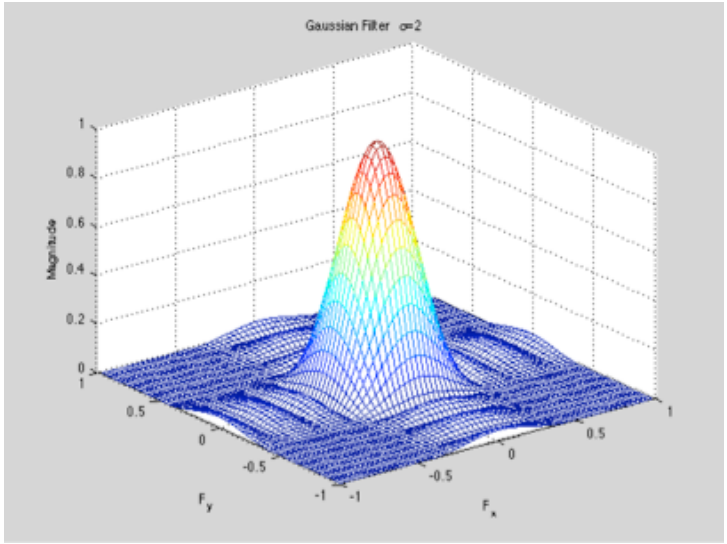


σ=2

β=0.16

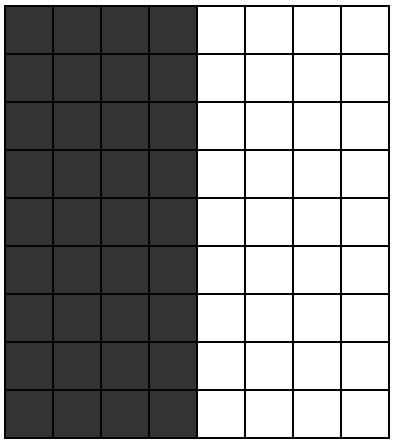


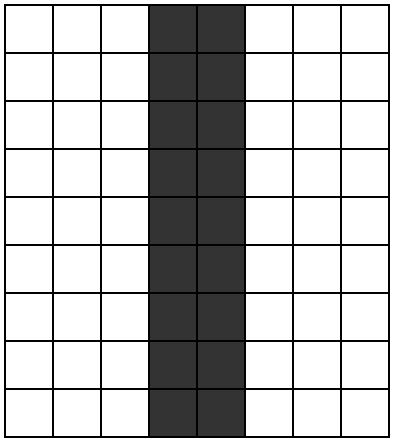
β=0.08

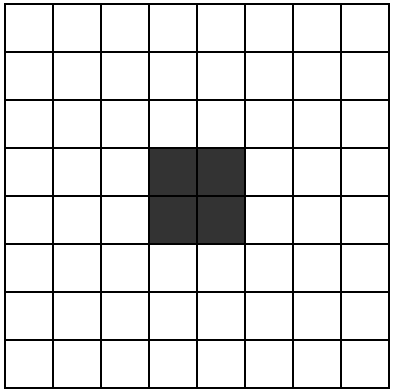


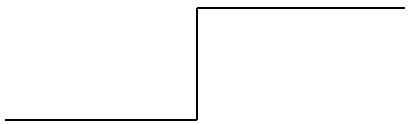
Edge Detection

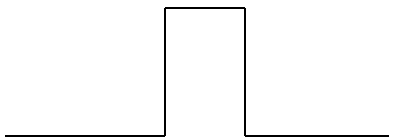
- What is an edge?
- Difference between edge and line and point
- With high resolution images, even a thin line will occupy multiple rows/columns, may have step edges on both sides

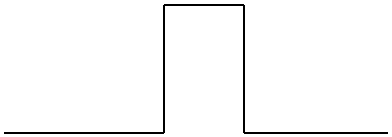












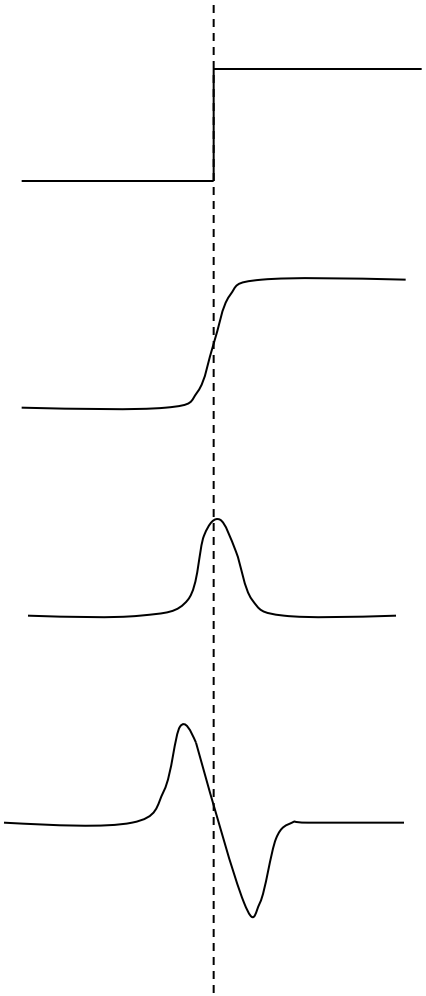
Characterization of Edges

Ideal step edge

Real edge has a slope

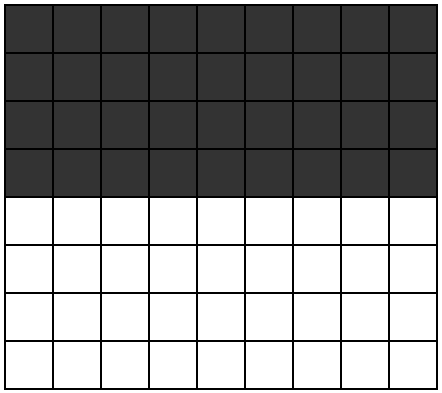
First order derivative:
Maximum at edge location

Second order derivative:
Zero crossing at edge location



Edge Detection Based on First Order Derivatives

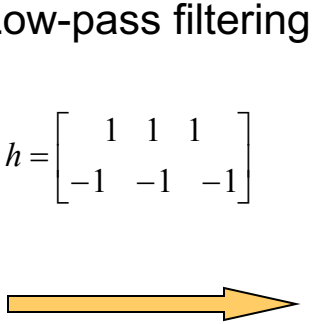
- Edge

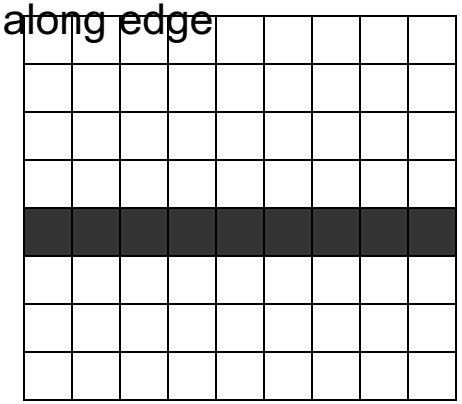


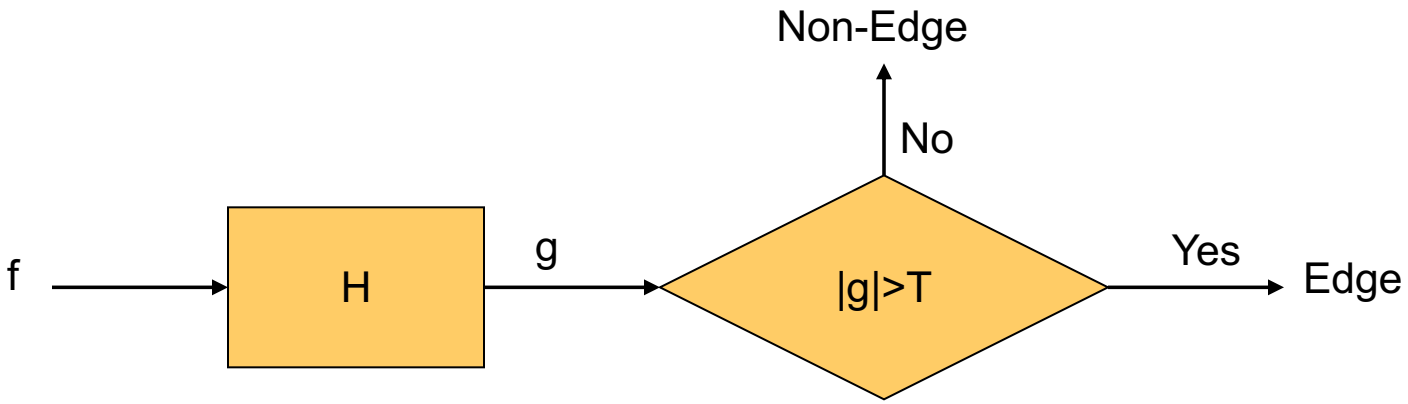
High-pass filtering across edge

Low-pass filtering along edge

$$h = \begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix}$$



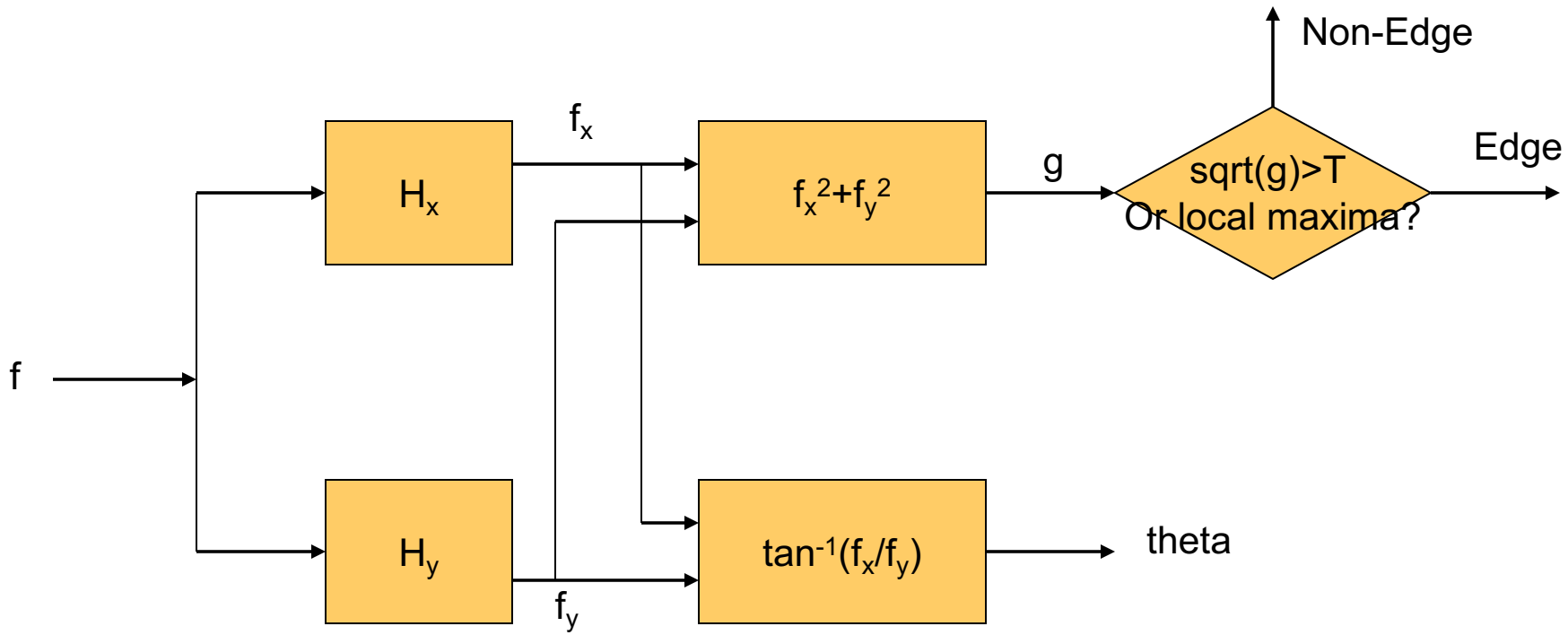




What if we don't know edge direction?

Edge Detection Based on Gradients in Two Orthogonal Directions

- Combine results from directional edge detectors in two orthogonal directions and determine the magnitude and direction of the edge.



Directional Edge Detector

- High-pass (or band-pass) in one direction (simulating first order derivative)
- Low pass in the orthogonal direction (smooth noise)
- Prewitt edge detector

LP BP

$$H_x = \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}; \quad H_y = \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

- Sobel edge detector

$$H_x = \frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}; \quad H_y = \frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

The sobel filter provides better smoothing along the edge

Freq. Response of Sobel Filter

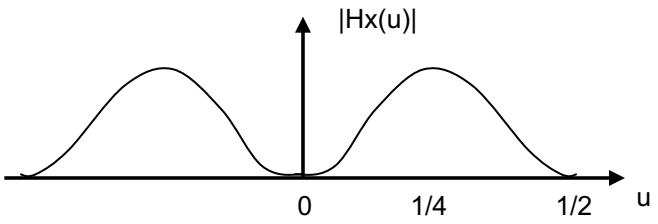
Sobel Filter for Horizontal Edges :

$$H_x = \frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

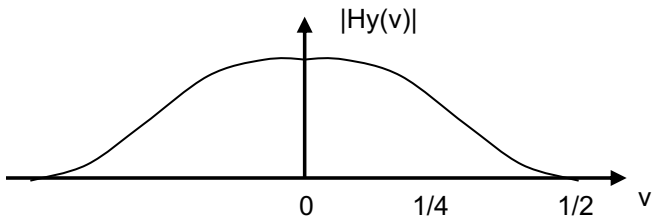
Frequency Response (DTFT) :

$$h_x = [-1\ 0\ 1] <-> H_x = \left(-e^{j2\pi u} + e^{-j2\pi u}\right) = -2j\sin 2\pi u$$

$$h_y = \frac{1}{4}[1\ 2\ 1] <-> H_y = \frac{1}{4}\left(e^{j2\pi v} + 2 + e^{-j2\pi v}\right) = \frac{1}{2}(1+\cos 2\pi v)$$



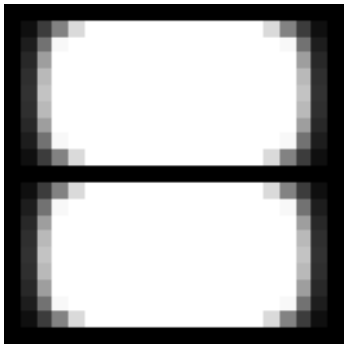
Band-pass

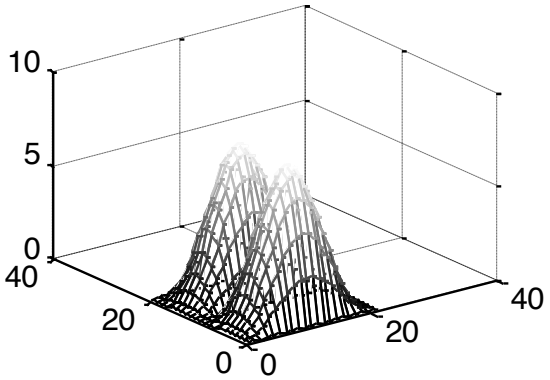


Low-pass

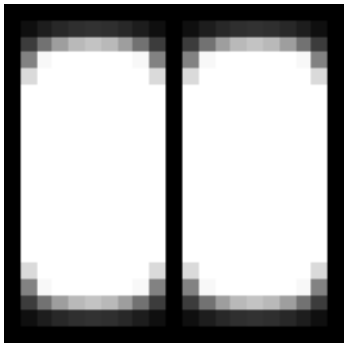
Spectrum of the Sobel Filter

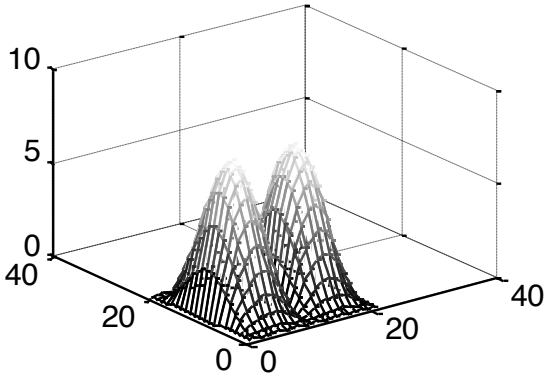
H_x





H_y





Low pass along the edge, band pass cross the edge

Example of Sobel Edge Detector

Original image

Filtered image by H_x

Filtered image by H_y

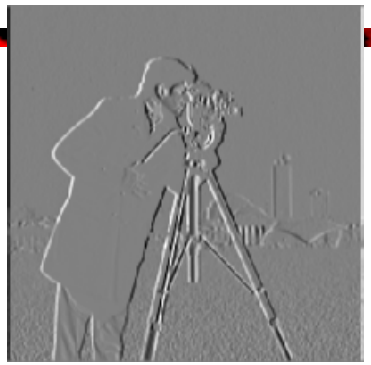
How to set threshold?

- Trial and error
- According to edge magnitude distribution
  - E.g assuming only 5% pixels should be edge pixels, then the threshold should be the 95% percentile of the edge magnitude
  - Illustrate on board

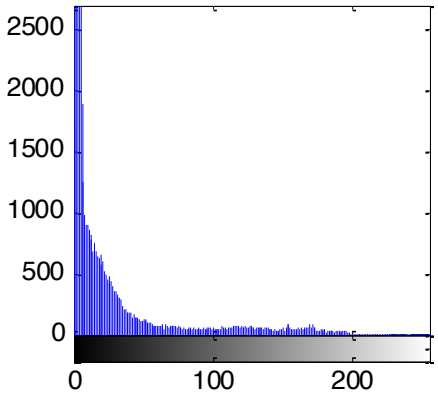
gx

gy

gm



Histogram of gm



T=100

T=50

T=20

DoG Filter for Taking Derivatives

- Apply Gaussian filtering first to smooth the image, STD depends on noise level or desired smoothing effect
- Then take derivative in horizontal and vertical directions
- = Convolve the image with a Derivative of Gaussian (DoG) filter

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

$$H_x(x,y)=\frac{\partial G}{\partial x}=-\frac{x}{\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$H_y(x,y)=\frac{\partial G}{\partial y}=-\frac{y}{\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Sample the above continuous filter to get digital filter. Hy is rotated version of Hx

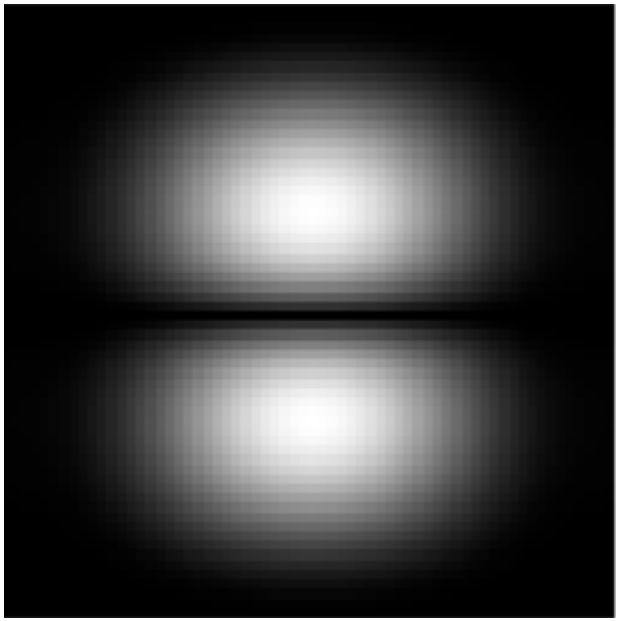
DoG Filter Examples

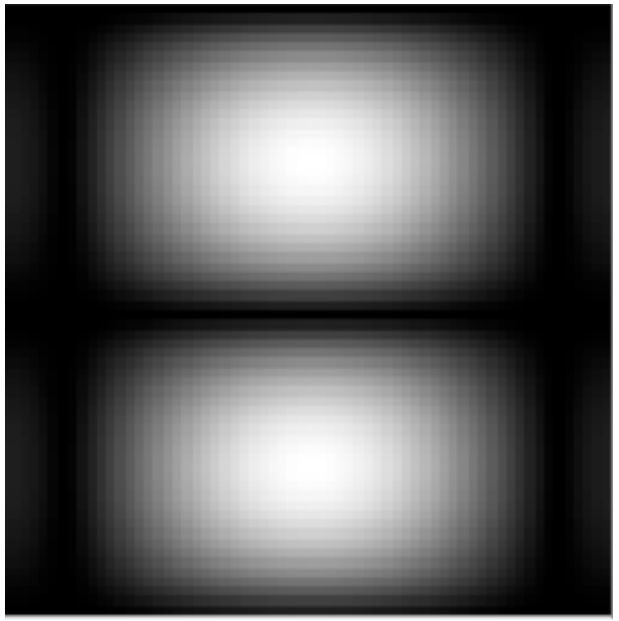
s=1, n=5

0.0366	0.1642	0.2707	0.1642	0.0366
0.0821	0.3679	0.6065	0.3679	0.0821
0	0	0	0	0
-0.0821	-0.3679	-0.6065	-0.3679	-0.0821
-0.0366	-0.1642	-0.2707	-0.1642	-0.0366

S=1, n=3 (similar to Sobel)

0.3679	0.6065	0.3679
0	0	0
-0.3679	-0.6065	-0.3679





DoG filters are low-pass along the edge, and band-pass in orthogonal direction (across edge)

Problems of previous approach

- Cannot locate edges precisely
- Ramp edges can lead to many edge pixels detected depending on the threshold T
  - T too high: may not detect weak edges
  - T too small: detected edges too think, noisy points falsely detected
- Remedy:
  - Detecting local maximum of |g| in the normal direction of the edge, or try all possible 8 direction in a 3x3 neighbor
  - Only consider pixels with |g| > T

Edge Detection with Many Directional Filters

- Instead of using two orthogonal directions, can design multiple directional filters
  - 0, 45, 90, 135
- See which one gives the highest response in the normal direction

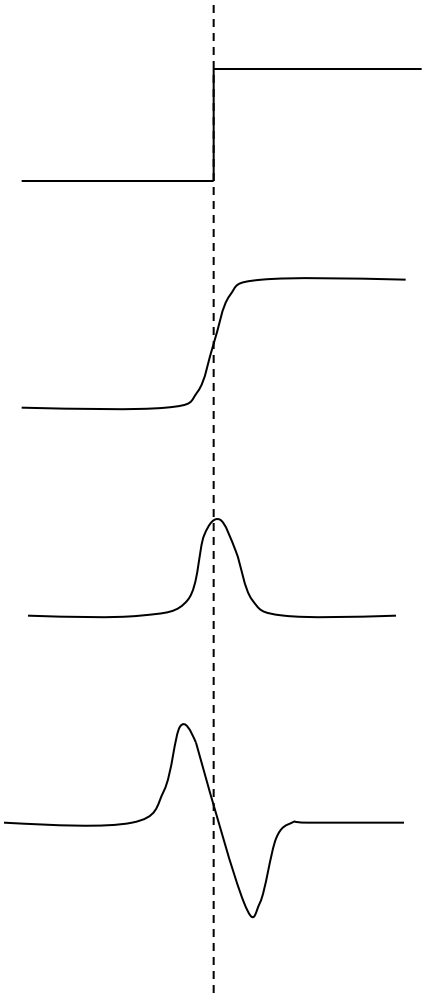
Characterization of Edges

Ideal step edge

Real edge has a slope

First order derivative:
Maximum at edge location

Second order derivative:
Zero crossing at edge location



Edge Detection Based on Second Order Derivative

- Convolve an image with a filter corresponding to taking second order derivative (e.g. Laplacian or LoG operator)
- Locate zero-crossing in the filtered image

Laplacian Operator

$$\nabla_f^2 = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\frac{\partial f}{\partial x} = f(x+1, y) - f(x, y)$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) - 2f(x, y) + f(x-1, y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) - 2f(x, y) + f(x, y-1)$$

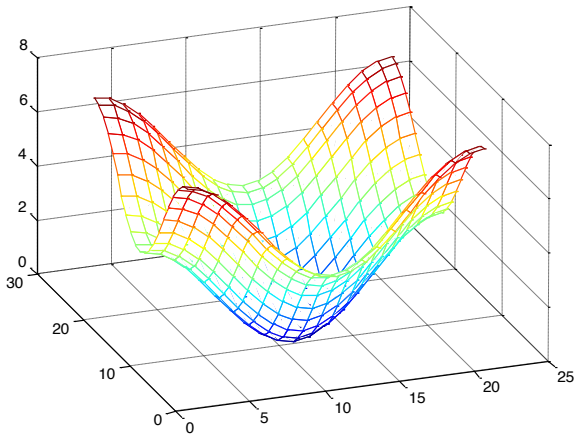
$$\nabla_f^2 = \left[f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)\right] - 4f(x, y)$$

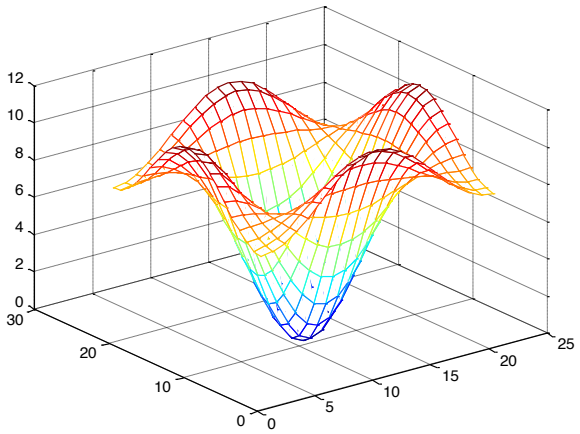
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}; \quad \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix};$$

Fourier Transform of Laplacian Operator

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$





- Laplacian operator are isotropic, can detect changes in all directions

Laplacian of Gaussian (LoG)

- To suppress noise, smooth the signal using a Gaussian filter first
  - F(x,y)* G(x,y)
- Then apply Laplacian filter
  - F(x,y)*G(x,y)*L(x,y) = F(x,y)* (L(x,y)*G(x,y))
- Equivalent filter: LoG
  - H(x,y)=L(x,y)*G(x,y)

Derivation of LoG Filter

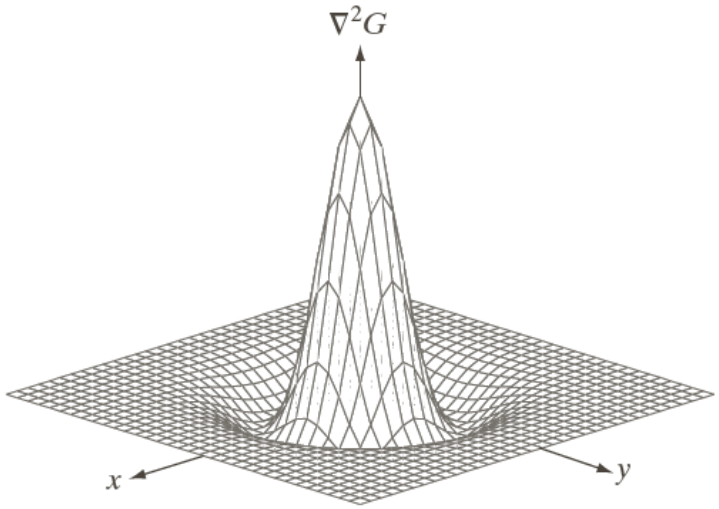
- Continuous form

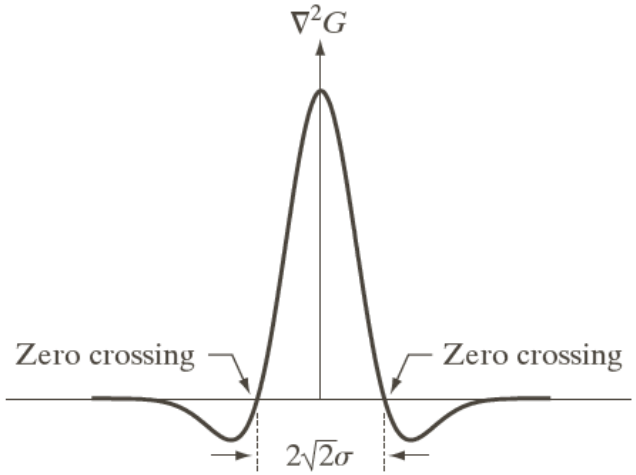
$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\nabla^2 G(x,y) = \frac{\partial^2 G}{\partial^2 x} + \frac{\partial^2 G}{\partial^2 y} = \frac{x^2+y^2-2\sigma^2}{\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Take samples to create filter mask
  - Size of mask nxn, n>=5σ, odd
  - Ex: σ=1, n=5.

LoG Filter





0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

a b
c d

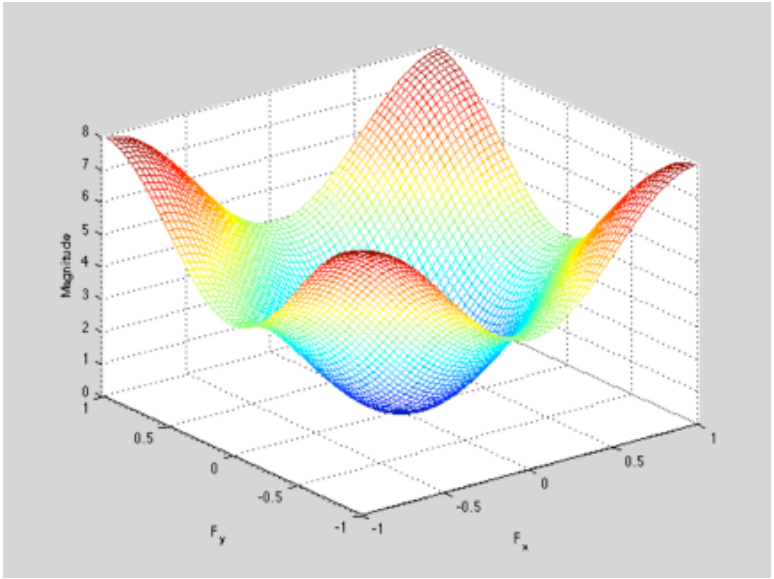
FIGURE 10.21 (a) Three-dimensional plot of the *negative* of the LoG. (b) Negative of the LoG displayed as an image. (c) Cross section of (a) showing zero crossings. (d) 5 × 5 mask approximation to the shape in (a). The negative of this mask would be used in practice.

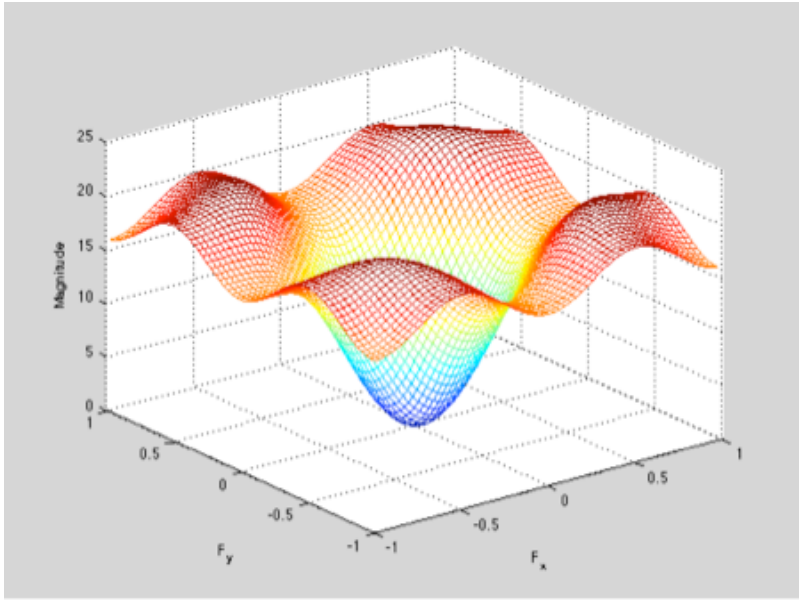
Laplacian vs. LoG in Freq. Domain

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

H =

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0





Laplacian filtered

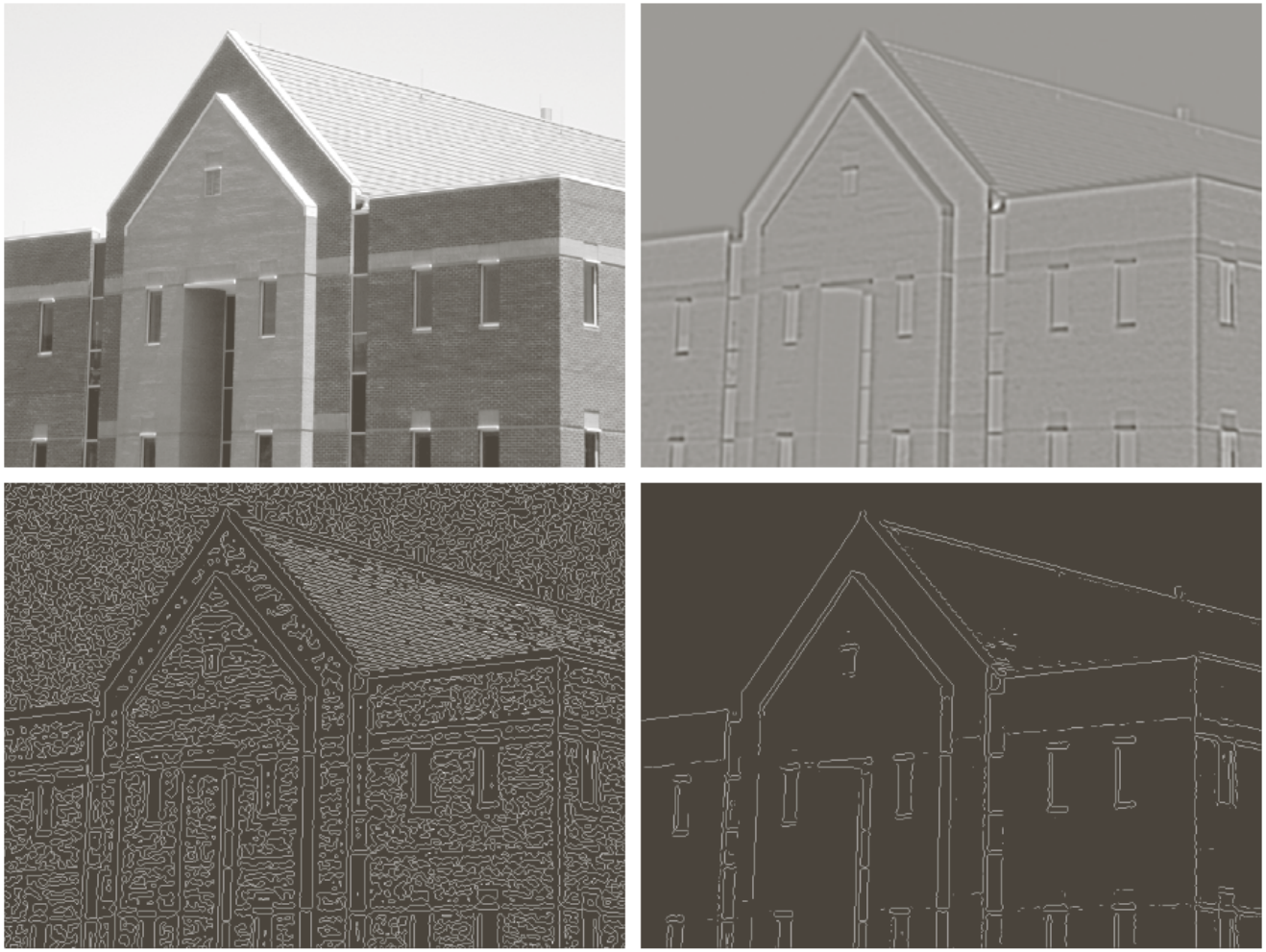
LOG filtered

Note that each strong edge in the original image corresponds to a thin stripe with high intensity in one side and low intensity in the other side.

How to detect zero crossing?

- For each pixel that has low filtered value, check a 3x3 neighbor, to see whether its two neighbors in opposite direction have opposite sign and their difference exceeds a threshold (Marr-Hildreth edge detection method)

Example



a b
c d

FIGURE 10.22
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1]. (b) Results of Steps 1 and 2 of the Marr-Hildreth algorithm using $\sigma = 4$ and $n = 25$. (c) Zero crossings of (b) using a threshold of 0 (note the closed-loop edges). (d) Zero crossings found using a threshold equal to 4% of the maximum value of the image in (b). Note the thin edges.

From: [Gonzalez]

Pros and Cons

- Can locate edges more accurately
- Can detect edges in various direction
- But more prone to noise
- Remedy:
  - Need to select the smoothing parameter σ properly

Summary of Edge Detection Method

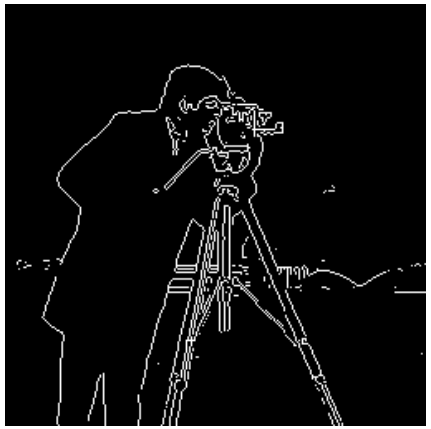
- First order gradient based:
  - Using edge detectors in two orthogonal directions
    - For each direction: low-pass along edge, band-pass across edge
    - Sobel, DoG filters
  - Using edge detectors in multiple (>2) directions
  - Use threshold or detect local maximum across the edge direction
- Second order gradient based
  - Laplacian is noise-prone, LoG is better
  - Detect zero crossing
  - Isotropic

More on Edge Detection

- Methods discussed so far generally cannot yield connected thin edge maps
- Need sophisticated post processing
  - Thinning
  - Connecting broken lines
- Noise can lead to many false edge points
- Even with many years of research, no perfect edge detectors exist!
  - Canny edge detector: Gaussian smoothing along edges, high pass in each possible edge direction
- For more on edge detection, See Gonzalez Sec. 10.1, 10.2

Results using MATLAB "edge( )" function

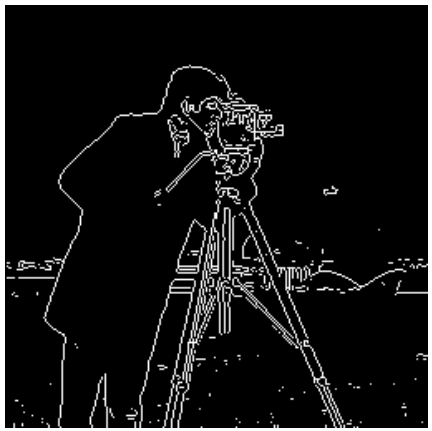
Sobel,T=0.14



LOG, T=0.0051

canny,T=[0.0313,0.0781]

Sobel,T=0.1



LOG, T=0.01

canny,T=[0.1,0.15]

Line Detection Using Hough Transform

- First do edge detection. Then detect edge pixels falling on a line through a “voting” mechanism
- Pixels (x,y) in a line are related by
  - $y = ax + b$ or $\cos\theta\, x + \sin\theta\, y = l$
  - Using a counter for all possible pairs of (θ, l): Each edgel contributes one count to all possible (θ, l). -> $H(\theta, l)$: represents the count for the possible line (θ, l)
  - If there is a line with actual parameter (θ^*, l^*), $H(\theta^*, l^*)$ will have a large value!
  - Line detection => peak detection in $H(\theta, l)$
- Generalizable to detecting any shape with analytical parameterization: circles, ovals, etc.
- Optional (Not required, but good to know!)

Image Sharpening (Debluring)

- Sharpening : to enhance line structures or other details in an image
- Enhanced image = original image + scaled version of the line structures and edges in the image
- Line structures and edges can be obtained by applying a high pass filter on the image
- In frequency domain, the filter has the "high-emphasis" character
  - Design a high-emphasis filter directly
  - Construct such filter from low pass or high pass filter

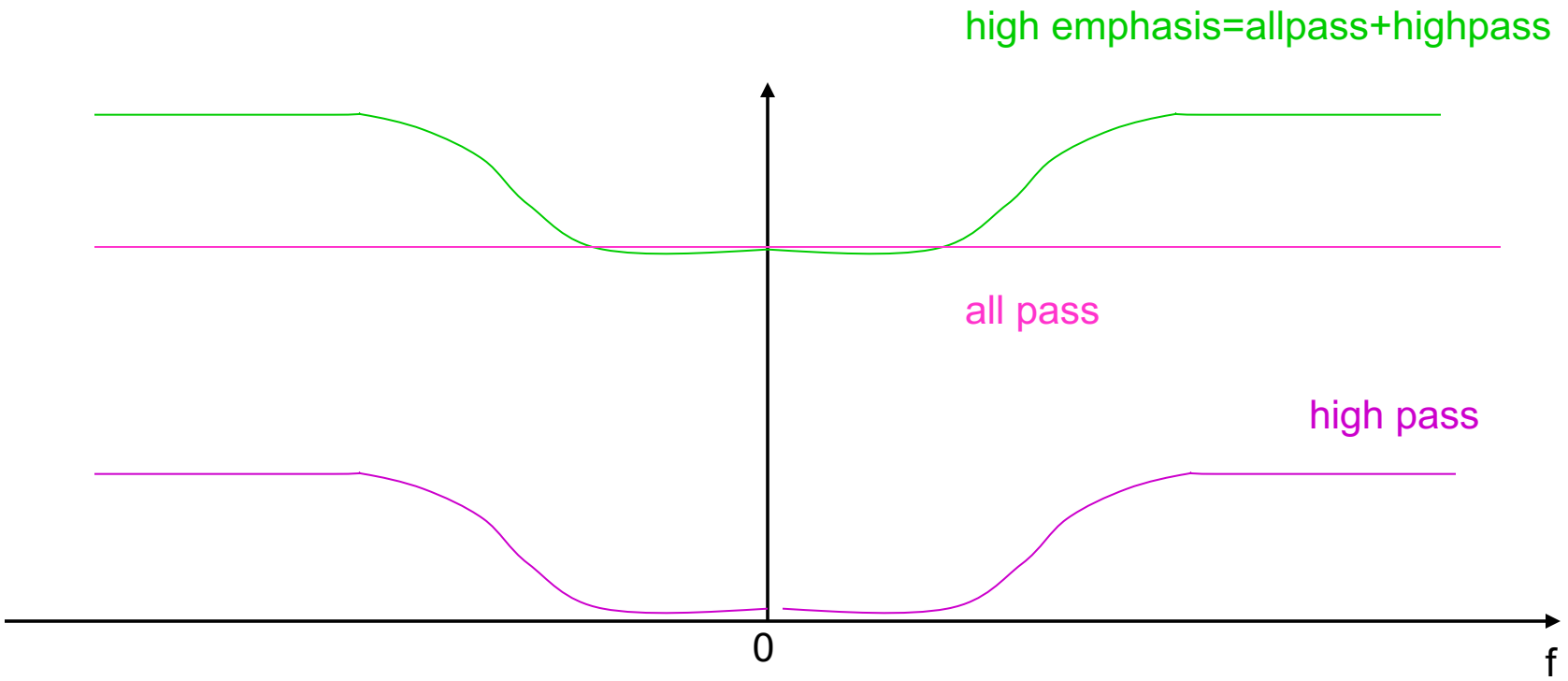
Designing Sharpening Filter Using High Pass Filters

- The desired image is the original plus an appropriately scaled high-passed image
- Sharpening filter

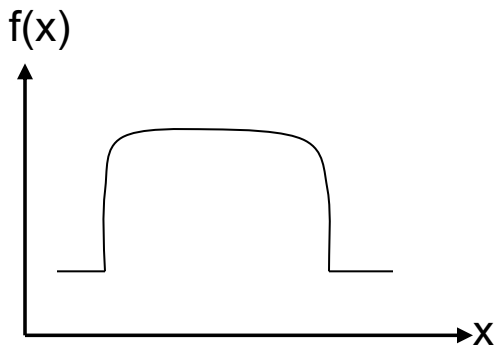
$$f_s = f + \lambda f_h$$

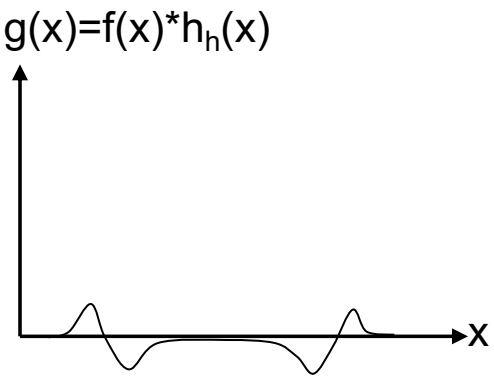
$$h_s(m,n) = \delta(m,n) + \lambda h_h(m,n)$$

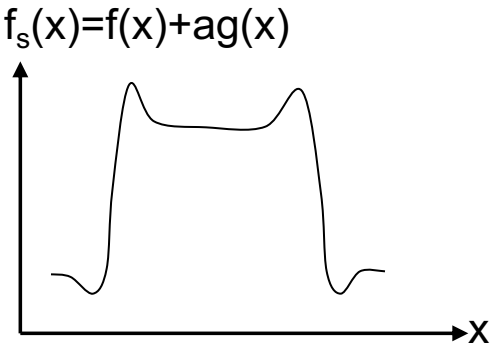
Interpretation in Freq Domain



High Emphasis Filter Using Laplacian Operator as Highpass Filter







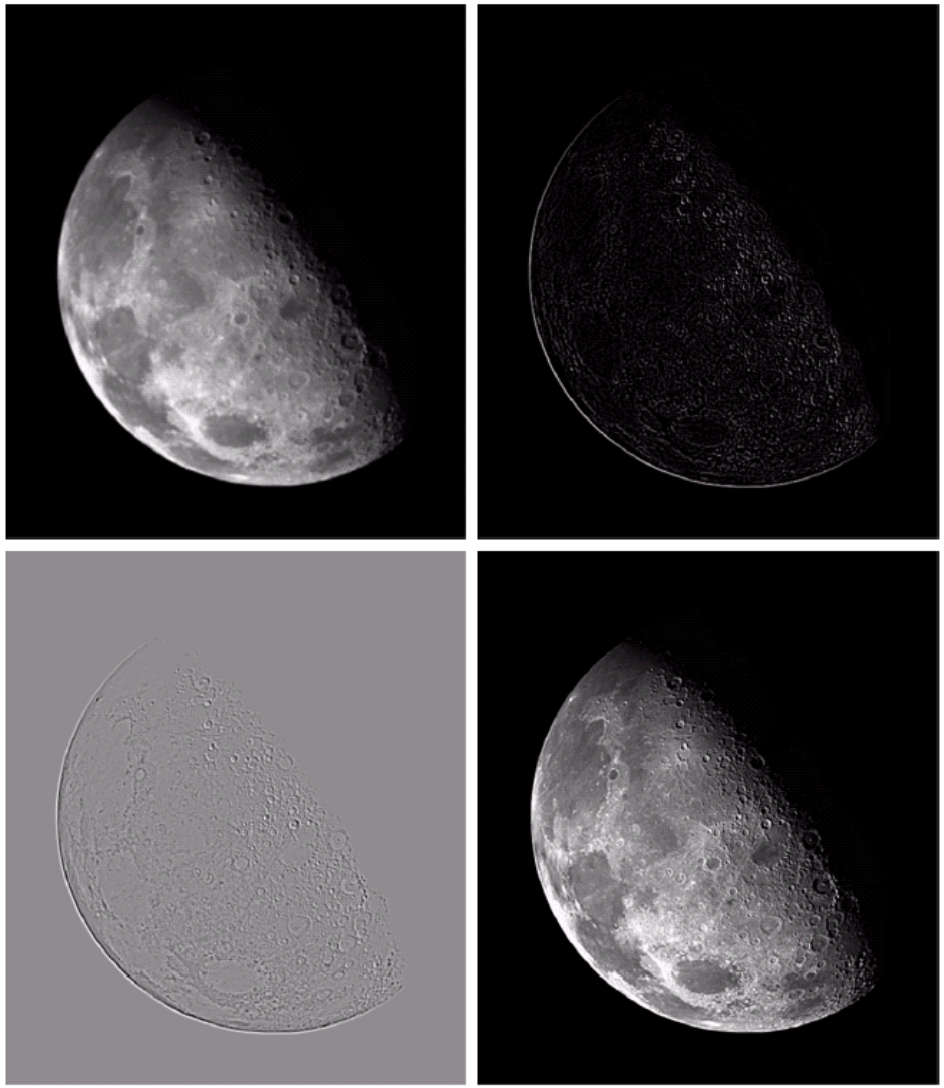
$$H_h = \frac{1}{4}\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \Rightarrow H_s = \frac{1}{4}\begin{bmatrix} 0 & -1 & 0 \\ -1 & 8 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad with \quad \lambda = 1.$$

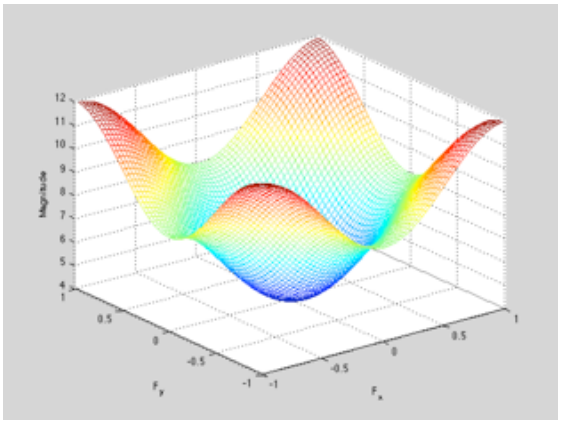
$$H_h = \frac{1}{8}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \Rightarrow H_s = \frac{1}{8}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 16 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad with \quad \lambda = 1.$$

Example of Sharpening Using Laplacian Operator

a b
c d

FIGURE 3.40
(a) Image of the North Pole of the moon.
(b) Laplacian-filtered image.
(c) Laplacian image scaled for display purposes.
(d) Image enhanced by using Eq. (3.7-5).
(Original image courtesy of NASA.)





$$H_h = \frac{1}{4}\begin{bmatrix} 0 & -1 & 0 \\ -1 & 8 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Summary (1)

- 2D CSFT and DSFT
  - Many properties of 1D CTFT and DTFT carry over, but there are a few things unique to 2D
  - Meaning of spatial frequency
  - 2D FT of separable signal = product of 1D FT
  - Rotation in space <-> rotation in frequency plane
- 2D linear convolution = weighted average of neighboring pixels
  - Filter=Point spread function (impulse response in 2D)
  - Any LSI (linear and shift invariant) operation can be represented by 2D convolution
  - DSFT of filter = frequency response = response to complex exponential input
- Computation of convolution:
  - boundary treatment, separable filtering
- Convolution theorem
  - Computation of convolution using DFT
- MATLAB function: conv2( ), freqz2( )
- Python: scipy.signal.freqz( )

Summary (2)

- Noise removal using low-pass filters (averaging, Gaussian).
- Edge detection
  - Based on 1st order derivatives: Sobel, DoG (difference of Gaussian) + finding local maxima
  - Based on 2nd order derivatives: Laplacian, LoG (Laplacian of Gaussian) + finding zero crossing
- Image sharpening by high emphasis filters
  - Filter = low pass+ a* high pass
- Given a filter, you should be able to tell what does it do (smoothing, edge detection, sharpening?) by looking at its coefficients, and also through its frequency response
  - Smoothing: coefficient sum =1, typically all positive
  - Edge detection / high/band pass: coefficient sum=0
  - Sharpening: coefficient sum=1, has negative coefficients

Reading Assignments

- Fourier transform and convolution:
  - [Szeliski2012] Richard Szeliski, Computer Vision: Algorithms and Applications. 2012. Ch. 3. (Sec. 3.4.1, 3.4.2)
  - Wang et al, Digital video processing and communications. Sec. 2.1, 2.2 (for general multi-dimensional case)
  - Gonzalez & Woods, "Digital Image Processing", Prentice Hall, 3rd Ed, Chap 4. (You can skip sections 4.3) Note: in Gonzalez & Woods, DTFT and DSFT are not introduced, the authors go directly to DFT. (with more in depth discussion)
- Image smoothing, edge detection and sharpening
  - Gonzalez & Woods, "Digital Image Processing", Prentice Hall, 2008, Section 3.5, 3.6, 10.2
  - [Szeliski2012] Richard Szeliski, Computer Vision: Algorithms and Applications. 2012. Sec. 4.2 (Edge detection)

Written Assignment (1)

1. Let $f(x,y)=\sin 2\pi f_0(x+y), \quad h(x,y)=\dfrac{\sin(2\pi f_c x)\sin(2\pi f_c y)}{\pi^2 xy}$

   Find the convolved signal $g(x, y) = f(x, y) * h(x, y)$ for the following two cases:

   a) $f_0/2 < f_c < f_0$; and b) $f_0 < f_c < 2f_0$.

   Hint: do the filtering in the frequency domain. Explain what happened by sketching the original signal, the filter, the convolution process and the convolved signal in the frequency domain.

2. Repeat the previous problem for

$$h(x,y)=\begin{cases} f_c^2, & -\dfrac{1}{2f_c} < \{x,y\} < \dfrac{1}{2f_c} \\ 0, & otherwise \end{cases}$$

3. Prove the rotation property of 2D CSFT
4. Show that the DSFT of f(m,n)=δ(m-n) is F(u,v)= δ (u+v)

Written Assignment (2)

5. For the three filters given below (assuming the origin is at the center):

a) find their Fourier transforms (2D DSFT);

b) sketch the magnitudes of the Fourier transforms. You should sketch by hand the DTFT as a function of u, when v=0 and when v=1/2; also as a function of v, when u=0 or ½. Also please plot the magnitude of DSFT as a function of both u and v, using Matlab plotting function.

c) Also use MATLAB freqz2( ) to compute and plot the magnitude of DSFT and compare to your answer.

c) Comment on the functions of the three filters.

In your calculation, you should make use of the separable property of the filter whenever appropriate. When the filter is not separable, you may be able to split the filter into several additive terms such that each term can be calculated more efficiently.

$$H_1 = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad H_2 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}; \qquad H_3 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$